GENE EXPRESSION INFERENCE USING ARTIFICIAL NEURAL NETWORKS

Mgr. Ing. Vladimír Kunc

Doctoral Thesis

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

Study Programme: **Electrical Engineering and Information Technology (P2612)**
Field of Study/Specialization: **Artificial Intelligence and Biocybernetics (3902V035)**
Supervisor: **doc. Ing. Jiří Kléma, Ph.D.**

Prague, February 2024

Vladimír Kunc: *Gene expression inference using artificial neural networks,* Doctoral Thesis
Study Programme: Electrical Engineering and Information Technology (P2612)
Field of Study/Specialization: Artificial Intelligence and Biocybernetics (3902V035)

© February 2024

SUPERVISOR:
doc. Ing. Jiří Kléma, Ph.D.

LOCATION:
Prague

ABSTRACT

Gene expression profiling is necessary for understanding cellular states in different experimental conditions, which is needed in various fields of biomedical research. Despite the significant progress in gene expression profiling, large-scale genome-wide profiling is still expensive and challenging. The introduction of the L1000 microarray platform made this analysis significantly cheaper by measuring the gene expression of only a few landmark genes and using computational models to infer the gene expression levels of the remaining genes. Initially, linear regression models were used but were soon replaced by neural network (NN) models such as the D–GEX as they are better suited for modeling the complex nonlinear relationships of the expressions of individual genes.

This thesis introduces significant enhancements to the original D–GEX model — primarily the introduction of transformative adaptive activation functions (TAAFs), a novel class of adaptive activation functions. The TAAFs introduce four adaptive parameters allowing for any horizontal and vertical scaling and translation of any inner activation function. The TAAFs improve the performance of the NNs for gene expression inference and also add some robustness to the choice of the activation function. The performance of NNs with TAAFs is shown on the task of gene expression inference from the expressions of landmark genes of the L1000 microarray platform and also using several artificially generated datasets to demonstrate their applicability outside the omics domain. Additionally, we also show that the improvements in the gene expression inference also translate to improvements in the subsequent analyses, thus validating the practical impact of the usage of the TAAFs.

A second important enhancement to the original NNs used for gene expression inference is the introduction of a tower and checkerboard architectures that further improve the NNs with TAAFs and reach even better performance. Notably, this improvement extends to subsequent analyses, demonstrating its statistical significance in enhancing inferred data quality.

Although these improvements were demonstrated mainly on the gene expression inference task, their scope is not confined to omics as they are transferable to broader NNs applications. The TAAFs generalize various activation functions already proposed in the literature and used for various tasks, proving their versatility in multiple settings beyond gene expression inference.

Additionally, this work provides an extensive list of activation functions, serving as a reference to streamline future research and prevent redundant proposals of activation functions already present in the literature.

Keywords: adaptive activation functions, deep learning, neural networks, gene expression inference, tower architecture, checkerboard architecture, transformative adaptive activation functions, L1000

ABSTRAKT

Měření genové exprese je nezbytné pro porozumění buněčným procesům a stavům v rozličných experimentálních podmínkách, což je potřeba v různých oblastech biomedicínského výzkumu. Navzdory významnému pokroku v měření genové exprese jsou velkorozsahové studie stále velmi drahé a náročné. Nástup měřící platformy L1000 výrazně zlevnil podobné studie díky měření jen vybraných klíčových genů a použití výpočetních modelů k rekonstrukci úrovní genové exprese zbylých genů. Původně byly použity modely využívající lineární regresi, ale brzy byly nahrazeny neuronovými sítěmi jako je D–GEX, které jsou vhodnější pro modelování složitých nelineárních vztahů mezi expresemi jednotlivých genů.

Tato disertační práce přináší významná vylepšení původního D–GEX modelu — zejména představuje transformativní adaptivní aktivační funkce (TAAF), novou třídu adaptivních aktivačních funkcí. TAAF zavádějí čtyři adaptivní parametry umožňující libovolné horizontální a vertikální škálování a translaci libovolné vnitřní aktivační funkce. TAAF zlepšují kvalitu inference genové exprese a také přidávají určitou robustnost vůči výběru aktivační funkce. Lepší modelovací schopnosti neuronových sítí s TAAF jsou ukázány na úloze inference genové exprese z exprese klíčových genů microarray platformy L1000 a také pomocí několika uměle vytvořených datasetů za účelem prokázání jejich aplikovatelnosti mimo oblast biomedicíny. Dále je ukázáno, že zpřesnění inference genové exprese se také promítá do zpřesnění následných analýz, což demonstruje, že TAAF jsou vhodné pro použití v praxi.

Druhým důležitým vylepšením původních neuronových sítí použitých pro inferenci genové exprese je představení věžových a šachovnicových architektur, které dále zlepšují neuronové sítě s TAAF a dosahují ještě lepší přesnosti inference. Tato vylepšení se projevují i v následných analýzách dopočtených dat, čímž je ukázáno, že TAAF mají statisticky význam dopad na zlepšení kvality dopočtených dat.

I když byla tato zlepšení předvedena hlavně na úloze inference genové exprese, jejich použití není omezeno na oblast biomedicíny, neboť tato zlepšení jsou použitelná v mnoha jiných aplikací neuronových sítí. Jelikož TAAF zobecňují různé aktivační funkce, které již byly navrženy v literatuře a použity na rozličných úlohách, tak i TAAF jsou vhodné nejen pro inferenci genové exprese.

Dále tato práce poskytuje rozsáhlý seznam aktivačních funkcí, který slouží jako reference k zjednodušení budoucího výzkumu a předcházení opakovaným návrhům aktivačních funkcí již přítomných v literatuře.

Klíčová slova: adaptivní aktivační funkce, hluboké učení, neuronové sítě, inference genové exprese, věžová architektura, šachovnicová architektura, transformativní adaptivní aktivační funkce

PUBLICATIONS

PUBLICATIONS RELATED TO THE TOPIC OF THIS THESIS

The presented work is published in the following papers:[1][2]

Journal papers

[1] **V. Kunc** and J. Kléma. "On transformative adaptive activation functions in neural networks for gene expression inference." In: *PLOS ONE* 16.1 (Jan. 2021). Ed. by H. Fröhlich, e0243915. DOI: 10.1371/journal.pone.0243915. URL: https://doi.org/10.1371/journal.pone.0243915.
Journal metrics: IF: 3.7 (Q2), JCI: 0.91 (Q1), CiteScore: 6.0 (Q1)
Citations: WoS: 4, Scopus: 6, Google: 19,
Author statement: V.K.: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing; J.K.: Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing.

[2] **V. Kunc** and J. Kléma. "On tower and checkerboard neural network architectures for gene expression inference." In: *BMC Genomics* 21.S5 (Dec. 2020). DOI: 10.1186/s12864-020-06821-6. URL: https://doi.org/10.1186/s12864-020-06821-6.
Journal metrics: IF: 4.4 (Q1), JCI: 1.1 (Q1), CiteScore: 7.5 (Q1)
Citations: WoS: 0, Scopus: 0, Google: 2,
Author statement: V.K. designed the model and the computational framework, carried out the implementation, performed the calculations. **V.K.** and J.K. conceived the study and analyzed the data and wrote the manuscript. J.K. was in charge of overall direction and planning.

Journal papers — submitted, under review or in press

[3] **V. Kunc** and J. Kléma. *Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks*. 2024. DOI: 10.48550/ARXIV.2402.09092. URL: https://arxiv.org/abs/2402.09092. *Submitted.*,
Author statement: V.K. conceived and carried out the survey and wrote the manuscript. J.K. was in charge of overall direction and planning.

1 Citation counts and journal metrics are as reported on February 16, 2024. Therefore, journal metrics are the 2022 values since the 2023 update was not yet available.
2 Authorship Statements present in the original articles have been reused verbatim.

Conference papers

[4] **V. Kunc** and J. Kléma. "On functional annotation with gene co-expression networks." In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Dec. 2022. DOI: 10.1109/bibm55620.2022.9995542. URL: https://doi.org/10.1109/bibm55620.2022.9995542.
Citations: WoS: ∅, Scopus: 0, Google: 1
Author statement: V.K.: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing; J.K.: Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing.

[5] **V. Kunc**. "On the importance of dropout for Checkerboard D-GEX architectures." In: *Proceedings of the International Student Scientific Conference Poster - 23/2019*. ČVUT FEL, Středisko vědecko-technických informací, May 2019. ISBN: 978-80-01-06581-5. URL: https://poster.fel.cvut.cz/poster2019/.
Citations: WoS: ∅, Scopus: ∅, Google: ∅.

[6] **V. Kunc**. "The development of DNA microarrays." In: *Proceedings of the International Student Scientific Conference Poster - 23/2019*. ČVUT FEL, Středisko vědecko-technických informací, May 2019. ISBN: 978-80-01-06581-5. URL: https://poster.fel.cvut.cz/poster2019/.
Citations: WoS: ∅, Scopus: ∅, Google: ∅.

Other

[7] **V. Kunc**. *Exploring the Relationship: Transformative Adaptive Activation Functions in Comparison to Other Activation Functions*. 2024. DOI: 10.48550/ARXIV.2402.09249. URL: https://arxiv.org/abs/2402.09249.

OTHER PUBLICATIONS OF THE AUTHOR

Journal papers

[8] M. Belbl, D. Kachlík, M. Beneš, **V. Kunc**, and V. Kunc. "Variations of the lumbrical muscles of the hand: Systematic review and meta-analysis." In: *Annals of Anatomy - Anatomischer Anzeiger* 247 (Apr. 2023), p. 152065. DOI: 10.1016/j.aanat.2023.152065. URL: https://doi.org/10.1016/j.aanat.2023.152065.
Journal metrics: IF: 2.2 (Q2), JCI: 1.39 (Q1), CiteScore: 4.6 (Q2)
Citations: WoS: 0, Scopus: 0, Google: 0,
Author statement: MB: Manuscript writing, Data extraction, DK: Manuscript editing, Project development, MB: Manuscript editing, Data extraction, **VlK**: Statistical analysis, Manuscript editing, VoK: Manuscript writing, Project development, Clinical correlations.

[9] M. Beneš, D. Kachlík, M. Belbl, Š. Havlíková, **V. Kunc**, A. Whitley, R. Kaiser, and V. Kunc. "A meta-analysis on the anatomical variability of the brachial plexus: Part III – Branching of the infraclavicular part." In: *Annals of Anatomy - Anatomischer Anzeiger* 244 (Oct. 2022), p. 151976. DOI: 10.1016/j.aanat.2022.151976. URL: https://doi.org/10.1016/j.aanat.2022.151976.
Journal metrics: IF: 2.2 (Q2), JCI: 1.39 (Q1), CiteScore: 4.6 (Q2)
Citations: WoS: 5, Scopus: 6, Google: 6
Author statement: Michal Benes: Conceptualization, Methodology, Investigation, Data curation, Figures, Writing – original draft. David Kachlik: Conceptualization, Resources, Data curation, Writing – review editing. Miroslav Belbl: Investigation, Data curation, Writing – review & editing. Sarlota Havlikova: Investigation, Data curation, Writing – review & editing. **Vladimir Kunc**: Software, Formal analysis, Data management, Writing – review & editing. Adam Whitley: Methodology, Writing – review & editing. Radek Kaiser: Visualization, Writing – review & editing. Vojtech Kunc: Writing, Conceptualization, Supervision, Project administration, Writing – review & editing.

[10] E. H. Bergou, Y. Diouane, **V. Kunc**, V. Kungurtsev, and C. W. Royer. "A Subsampling Line-Search Method with Second-Order Results." In: *INFORMS Journal on Optimization* 4.4 (Oct. 2022), pp. 403–425. DOI: 10.1287/ijoo.2022.0072. URL: https://doi.org/10.1287/ijoo.2022.0072.
Journal metrics: IF: ∅, JCI: ∅, CiteScore: ∅
Citations: WoS: ∅, Scopus: ∅, Google: 24,
Author statement: E.B.,Y.D., Vy.K., C.R.: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing; **Vl.K**: Implementation, running experiments, data analysis, visualization.

[11] V. Kunc, **V. Kunc**, K. Kuncová, D. Kachlík, and L. Kopp. "Ambiguity of the radiographs around the elbow joint: Anatomical variant versus degenerative changes." In: *Journal of the Anatomical Society of India* 71.4 (2022), p. 303. DOI: 10.4103/jasi.jasi_80_21. URL: https://doi.org/10.4103/jasi.jasi_80_21.
Journal metrics: IF: 0.4 (Q4), JCI: 0.11 (Q4), CiteScore: 0.3 (Q4)
Citations: WoS: 0, Scopus: 0, Google: 1,
Author statement: Vo.K. designed the study, collected the main data and wrote the main part of the manuscript, **Vl.K.** performed all the statistics, helped with analysis of the collected data and wrote a significant part of the manuscript. K.K. reviewed the manuscript and helped analyze the data. D.K and L.K. supervised the study, edited the manuscript and helped to design the study.

[12] M. Beneš, D. Kachlík, M. Belbl, **V. Kunc**, Š. Havlíková, A. Whitley, and V. Kunc. “A meta-analysis on the anatomical variability of the brachial plexus: Part I – Roots, trunks, divisions and cords.” In: *Annals of Anatomy - Anatomischer Anzeiger* 238 (Nov. 2021), p. 151751. DOI: 10.1016/j.aanat.2021.151751. URL: https://doi.org/10.1016/j.aanat.2021.151751.
Journal metrics: IF: 2.2 (Q2), JCI: 1.39 (Q1), CiteScore: 4.6 (Q2)
Citations: WoS: 14, Scopus: 16, Google: 31, **Author statement:** Michal Benes: Conceptualization, Methodology, Investigation, Data curation, Figures, Writing - original draft. David Kachlik: Conceptualization, Resources, Data curation, Writing - review & editing. Miroslav Belbl: Investigation, Data curation, Writing - review editing. **Vladimir Kunc**: Software, Formal analysis, Data management, Writing - review & editing. Sarlota Havlikova: Investigation, Data curation, Writing - review & editing. Adam Whitley: Writing - review & editing, Methodology. Vojtech Kunc: Conceptualization, Writing - review & editing, Supervision, Project administration.

[13] M. Beneš, D. Kachlík, M. Belbl, A. Whitley, Š. Havlíková, R. Kaiser, **V. Kunc**, and V. Kunc. “A meta-analysis on the anatomical variability of the brachial plexus: Part II — Branching of the supraclavicular part.” In: *Annals of Anatomy - Anatomischer Anzeiger* 238 (Nov. 2021), p. 151788. DOI: 10.1016/j.aanat.2021.151788. URL: https://doi.org/10.1016/j.aanat.2021.151788.
Journal metrics: IF: 2.2 (Q2), JCI: 1.39 (Q1), CiteScore: 4.6 (Q2)
Citations: WoS: 7, Scopus: 8, Google: 12
Author statement: Michal Benes: Conceptualization, Methodology, Investigation, Data curation, Figures, Writing – original draft. David Kachlik: Conceptualization, Resources, Data curation, Writing – review editing. Miroslav Belbl: Investigation, Data curation, Writing – review & editing. Sarlota Havlikova: Investigation, Data curation, Writing – review & editing. **Vladimir Kunc**: Software, Formal analysis, Data management, Writing – review & editing. Adam Whitley: Methodology, Writing – review & editing. Radek Kaiser: Visualization, Writing – review & editing. Vojtech Kunc: Writing, Conceptualization, Supervision, Project administration, Writing – review & editing.

[14] M. Beneš, D. Kachlík, **V. Kunc**, and V. Kunc. “The arcade of Frohse: a systematic review and meta-analysis.” In: *Surgical and Radiologic Anatomy* 43.5 (Mar. 2021), pp. 703–711. DOI: 10.1007/s00276-021-02718-5. URL: https://doi.org/10.1007/s00276-021-02718-5.
Journal metrics: IF: 1.4 , JCI: 0.54, CiteScore: 2.4 (Q2)
Citations: WoS: 7, Scopus: 7, Google: 14,
Author statement: MB: Project development, Data collection, Manuscript writing/editing. DK: Data collection, Manuscript writing/editing. VlK: Data analysis, Manuscript writing/editing. VK: Project development, Manuscript writing/editing.

[15] V. Kunc, K. Edelmann, V. Bába, M. Debnar, P. Kmeť, K. Kučera, **V. Kunc**, R. Mišičko, and L. Kopp. “Retrospektivní analýza komplikací po sutuře Achillovy šlachy metodou podle Kesslera.” In: *Rozhledy v chirurgii* 100 (Oct. 2021). DOI: https://doi.org/10.33699/PIS.2021.100.8. URL: https://perspinsurg.com/rvch/article/view/560.
Journal metrics: IF: ∅, JCI: ∅, CiteScore: 0.4 (Q4)
Citations: WoS: ∅, Scopus: 0, Google: 1.

[16] V. Kunc, **V. Kunc**, V. Černý, M. Polovinčák, and D. Kachlík. “Accessory bones of the elbow: Prevalence, localization and modified classification.” In: *Journal of Anatomy* 237.4 (Aug. 2020), pp. 618–622. DOI: 10.1111/joa.13233. URL: https://doi.org/10.1111/joa.13233.
Journal metrics: IF: 2.4 (Q2), JCI: 1.26 (Q1), CiteScore: 4.6 (Q2)
Citations: WoS: 9, Scopus: 11, Google: 12,
Author statement: Vo.K. designed the study, collected the main data and wrote the main part of the manuscript, **Vl.K.** performed all the statistics, helped with analysis of the collected data and wrote a significant part of the manuscript. V.C collected a significant part of the data, reviewed the manuscript and helped analyze the data. M.P helped analyze all images as well as wrote the radiological part of manuscript and reviewed the whole text. D.K supervised the study, edited the manuscript and helped to design the study.

[17] V. Kunc, M. Štulpa, G. Feigl, C. Neblett, **V. Kunc**, and D. Kachlík. “The superficial anatomical landmarks are not reliable for predicting the recurrent branch of the median nerve.” In: *Surgical and Radiologic Anatomy* 42.8 (Apr. 2020), pp. 939–943. DOI: 10.1007/s00276-020-02475-x. URL: https://doi.org/10.1007/s00276-020-02475-x.
Journal metrics: IF: 1.4 (Q3), JCI: 0.54 (Q3), CiteScore: 2.4 (Q2)
Citations: WoS: 1, Scopus: 1, Google: 3,
Author statement: VoK Project development; Dissection; Manuscript writing/editing. MS Photo documentation and its processing; Dissection; Manuscript writing/editing. GF Dissection supervision; Manuscript writing/editing. CN Manuscript writing/editing and data processing. **VlK** Statistical analysis of data, manuscript writing/editing. DK Manuscript writing/editing and supervision.

Conference papers

[18] **V. Kunc**. “A Novel Aerial Dataset for Scene Classification Annotated Using OSM for Learning Deep CNNs.” In: *Proceedings of the International Student Scientific Conference Poster - 22/2018*. ČVUT FEL, Středisko vědecko-technických informací, May 2018. ISBN: 978-80-01-06428-3. URL: https://poster.fel.cvut.cz/poster2018/.
Citations: WoS: ∅, Scopus: ∅, Google: ∅.

[19] **V. Kunc**, J. Kléma, and M. Anděl. "Increasing Weak Classifiers Diversity by Omics Networks." In: *Proceedings of 2nd Workshop on Machine Learning in Life Sciences*. Wroclaw: ENGINE - European Research Centre of Network Intelligence and Innovation, 2015, pp. 16–28. ISBN: 978-83-943803-0-4. URL: `http://ida.felk.cvut.cz/klema/publications/Biotex/kunc_mlls2015.pdf`.
Citations: WoS: ∅, Scopus: ∅, Google: 0
Author statement: V.K.: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft; J.K.: Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing; M.A.: Conceptualization, Software, Methodology, Writing – original draft, Writing – review & editing.

The list includes Journal Impact Factor (JIF) and Journal Citation Indicator (JCI) using items indexed in Web of Science (WoS) by Clarivate and CiteScore using items indexed in Scopus by Elsevier including relative ranks in the documents's category.[3]

Any parts of the original papers reused verbatim in this thesis have been included with the approval of the co-authors.

3 If a document is in more categories, highest rank is reported.

"Everyone you will ever meet knows something you don't."

— Bill Nye

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor, doc. Ing. Jiří Kléma, Ph.D., whose invaluable guidance, unwavering support, and immense patience have been instrumental throughout my PhD journey. Your mentorship and insightful advice have been pivotal in shaping the direction of my research and academic growth. I am profoundly grateful for your dedication and encouragement.

I would also like to extend my heartfelt thanks to my family and friends for their steadfast support, understanding, and encouragement during this pursuit. Their belief in me has been a constant source of motivation and strength.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

LIST OF ACRONYMS

ϵ-HRVFLN RVFLN with ϵ-insensitive Huber loss function 174
π-ESN probabilistic ESN 178

A adenine 29, 30, 34, *Glossary:* adenine
AAF adaptive activation function v, xxxiv, xxxvi, xli, xliii, xlv, xlvii, xlix–li, liii, 2–4, 105, 107, 114, 121–129, 132, 134, 135, 137, 138, 140, 143–145, 148, 151–154, 157, 158, 160, 168, 169, 193, 194, 197–199, 207, 229, 231, 233, 277–279, 287, 290, *Glossary:* adaptive activation function
ABReLU average biased ReLU, *see Section 4.2.6.42*; 81, 110, 196, 200, 289
AbsLU absolute linear unit, *see Section 4.2.6.27*; 76, 77
ABU adaptive blending unit, *see Section 4.3.47*; xxxv, 156–159, 210, 218
AC-GAN auxiliary classifier GAN 182
ACON activate or not activation function, *see Section 4.3.3.1*; xlv, xlix, 129, 130
ADA apical dendrite activation, *see Section 4.2.6.50*; xxxvi, 83, 84
AdaLU adaptive linear unit, *see Section 4.3.35*; 151, 209, 218
ADHKELM adaptive deep hybrid kernel extreme learning machine 176
AE autoencoder xxxiii, xli, xliv, 44–47, 181
AF activation function v, xxxi–xxxiii, xxxvi–xxxix, xli–liii, 2–6, 10, 14–17, 43, 48–68, 71–87, 89–103, 105–108, 110, 112–119, 121–124, 126–132, 135–138, 143–148, 150–154, 156–159, 161–166, 168–172, 179, 184, 189, 190, 193–199, 204–211, 220–222, 225, 226, 229–236, 239, 245, 246, 252, 256, 257, 260–262, 264, 265, 275–279, 287–290
AGumb adaptive Gumbel, *see Section 4.3.15.7*; 137
AHAF adaptive hybrid activation function, *see Section 4.3.3.2*; 84, 129, 144, 198, 203
ALiSA adaptive linearized sigmoidal activation, *see Section 4.3.32.2*; 150
All-ReLU alternated left rectified linear unit, *see Section 4.3.33*; 78, 150, 217
AM-GAN fused GAN with attention mechanism 182
ANN artificial neural network, *see Chapter 2*; xlix, 2, 4, 9, 10, 13, 14, 22, 41, 42, 51, 173, 278, *Glossary:* artificial neural network
AOAF adaptive offset activation function, *see Section 4.3.1.12*; 109, 196, 200
APAF average of a pool of activation functions, *see Section 4.3.47.2*; 157, 210
APLU adaptive piece-wise linear unit, *see Section 4.3.28*; 48, 111, 113, 124, 146–150, 209, 217
AppReLU approximate ReLU, *see Section 4.2.6.40*; 80
AQuLU adaptive quadratic linear unit, *see Section 4.3.3.10*; 131
ARBF parametric radial basis function, *see Section 4.3.7*; 134
arctan arctangent, *see Section 4.2.2.4*; 52, 98, *Glossary:* arctangent
AReLU Attention-based ReLU, *see Section 4.3.1.19*; 80, 111, 207, 214
ARiA Adaptive Richard's curve weighted activation, *see Section 4.3.37*; xlv, 151, 152
ARiA2 Adaptive Richard's curve weighted activation 2, *see Section 4.3.37*; 151, 152, *Glossary:* ARiA2

ASELM daptive semi-supervised ELM 176
ASSF adaptive slope sigmoidal function, *see Section* 4.3.2.3; 127, 198, 202, 279
ASU amplifying sine unit, *see Section* 4.2.34; 100
ATAC-seq Assay for Transposase-Accessible Chromatin using sequencing xli, 46
ATSiLU arctan sigmoid-weighted linear unit, *see Section* 4.2.3.25; 63
ATU adaptive transformative unit, *see Section* 5.4.1.1; 5, 225, 226, *Glossary:* adaptive transformative unit
AutoGAN GAN with neural architecture search 182
AVAE autoencoder VAE 181

BAF bipolar activation function, *see Section* 4.2.6.35; 78
bbfire bounded bi-firing activation function, *see Section* 4.2.22; 97
BELM bayesian extreme learning machine 176
bfire bi-firing activation function, *see Section* 4.2.21; 97
BiLSTM bidirectional long short-term memory xlv, 88
binary AF binary activation function, *see Section* 4.2.1; 50
BLReLU bounded leaky ReLU, *see Section* 4.2.6.24; 75, 97, 149, 206, 213, *Glossary:* bounded leaky ReLU
BLU bendable linear unit, *see Section* 4.3.1.37; xl, 94, 117, 208, 215
BP backpropagation 10, 11, 23–25, 50, 51, 111, 127, 174, *Glossary:* backpropagation
BReLU bounded ReLU, *see Section* 4.2.6.16; xlv–xlvii, 3, 73, 75, 97, 165, *Glossary:* bounded ReLU
BRNN bidirectional recurrent neural network 43, *Glossary:* bidirectional recurrent neural network
BSCN bidirectional SCN 175

C cytosine 29, 30, 34, *Glossary:* cytosine
CAF chaotic activation function, *see Section* 4.2.48; xxxii, xxxiv, xxxvi, 103
CaLU Cauchy linear unit, *see Section* 4.2.3.3; 60
CatAF catalytic activation function, *see Section* 4.3.41; 152
CCAF cascade chaotic activation function, *see Section* 4.2.48.3; 104
CDBN convolutional deep belief network xxxvii, 181
CDF cumulative distribution function 59, 60, 134, 135, 137, 142, 164
cDNA copy DNA xlvi, xlviii, 35, 36, 46, *Glossary:* cDNA
CELU continuously differentiable exponential linear unit, *see Section* 4.3.1.51; 59, 111, 121, 208, 216
ChPAF Chebyshev polynomial-based activation function, *see Section* 4.3.47.7; 159, 160, 210, 219
CI confidence interval 191
CI-ELM convex incremental ELM 176
cLogLog complementary LogLog, *see Section* 4.2.2.21; xxxii, 56
cLogLogm modified cLogLog, *see Section* 4.2.2.21; 56
CMAFGAN cross-modal attention gusion based GAN 182
CML-GAN contrastive meta-learning GAN 182
CNCI Category Normalized Citation Impact xlviii, *Glossary:* Category Normalized Citation Impact

CNN convolutional neural network 10–13, 17, 18, 23, 47, 78, 149, 152, 168, 179
CoLU Collapsing linear unit, *see Section* 4.2.3.5; 60
CosLU cosinu-sigmoidal linear unit, *see Section* 4.3.15.6; 137
CPN continuous piecewise nonlinear activation function, *see Section* 4.3.44; 153, 154
CPU central processing unit 171
CReLU concatenated ReLU, *see Section* 4.2.6.34; xxxviii, 78, 79, 113
cRNA copy RNA 35, *Glossary:* cRNA
CRRAGAN cascading residual–residual attention GAN 182
CRVFLN convolutional RVFLN 174
cscGAN conditional single-cell GAN 44
CSS combination of sine and logistic sigmoid, *see Section* 4.3.40; 152
CSSA-SCN chaotic sparrow search algorithm based SCN 175
CT computed tomography 182, 183
cWOB-ELM coiflet wavelet-based optimization method-based ELM 176
CycleGAN cycle-consistent GAN 39, 182

D-HELM densely connected D-HELM xxxiii, 176
D2GAN dual discriminator GAN 182
D2WMGAN dual discriminator weighted mixture GAN 182
DAE denoising autoencoder 44, 45
DAELM domain adaptation ELM 176
DBEN deep belief echo state network 178
DBM deep Boltzmann machine 44, 180
DBN deep belief network xxxii, 44, 157, 180, 181
DDPM denoising diffusion probabilistic model 183
DE differentially expressed 193, 238–240, 245–251, 285
DeepESN deep echo state network 177
DF-GAN deep fusion GAN 182
DGattGAN dual Generator attentional GAN 182
DGE analysis differential gene expression analysis 5–7, 45, 46, 192, 193, 235, 236, 239, 241, 245, 270, 281, 282, 285, *Glossary:* differential gene expression analysis
DiffELU difference exponential linear unit, *see Section* 4.2.7.3; 85
DisReLU displaced ReLU, *see Section* 4.2.6.44; 82, 110, 195, 201
DKNN Deep Kronecker neural network 158, *see* NN
DL deep learning v, 9, 10
DLReLU dynamic leaky ReLU, *see Section* 4.3.1.13; 109, 205, 212
DM diffusion model 183, 184
DNA deoxyribonucleic acid xlv–xlvii, xlix, liii, 1, 4, 29–34, 46, *Glossary:* DNA
DNN deep neural network 179
DOS-ELM dynamic forgetting factor based OS-ELM algorithm xxxvii, 176
DoubleSiLU double sigmoid-weighted linear unit, *see Section* 4.2.3.22; 63
DP dynamic programming 10
DPAF dual parametric activation function, *see Section* 4.3.1.23; xlvii, 112, 113, 207, 215, *Glossary:* dual parametric activation function
DPGAN dual-stream GAN with phase awareness 182

DPReLU dual parametric ReLU, *see Section 4.3.1.20*; xlvi, 3, 106, 111, 112, 207, 215
DReLU dynamic ReLU, *see Section 4.3.1.14*; 81, 82, 109, 110, 169, 195, 200
DRESN double-reservoir ESN 178
DRLU delay ReLU, *see Section 4.2.6.43*; 81, 82, 195, 200
DSCN deep SCN 175
dSiLU derivative of sigmoid-weighted linear unit, *see Section 4.2.3.21*; 63
DSU decaying sine unit, *see Section 4.2.36*; 100
DTAAF dual transformative adaptive activation function xxxv, 290
DY–ReLU dynamic parameter ReLU, *see Section 4.3.55.3*; 169, 210, 218

EA-GAN example attention GAN 182
EACU elastic adaptively parametric compounded unit, *see Section 4.2.7.9*; 86
EANAF efficient asymmetric nonlinear activation function, *see Section 4.2.3.33*; 65, 66
EBM energy-based model 181, 184
EDELU extended exponential linear unit, *see Section 4.3.1.44*; 119
EELU elastic exponential linear unit, *see Section 4.3.1.60*; 125, 208, 217
EGAN edge adversarial GAN 182
ELiSH exponential linear sigmoid squashing, *see Section 4.2.7.17*; 89
ELM extreme learning machine xxxi–xxxix, 47, 175, 176
ELU exponential linear unit, *see Section 4.2.6.48*; xxxii–xxxv, xxxix, xl, xlii, xliii, xlvi–xlviii, l, li, 54, 59, 61, 65, 82–89, 91–93, 108, 111, 114, 117–125, 140, 141, 145, 156, 159, 160, 167, 194, 197, 205, 208, 213, *Glossary:* exponential linear unit
EM-ELM error minimized extreme learning machine 176
EPLAF even power linear activation function, *see Section 4.2.6.41*; 81
EPReLU Elastic PReLU, *see Section 4.3.1.25*; 113, 207, 215
ERA enhanced rational activation, *see Section 4.3.50*; 161, 162, 210
ERC external RNA control 35, *Glossary:* external RNA control
EReLU elastic ReLU, *see Section 4.2.6.38*; 79, 113, 125, 205, 213
ErfReLU Erf-based ReLU, *see Section 4.3.1.52*; 122
ES-ELM evolutionary optimized ELM 176
ESGNN echo state graph neural networks 178
ESN echo state network xxxi, xxxiii–xl, xlii, xliv, 177, 178, 185
ESN-DE differential evolution based ESN 178
EVWCA-MKRVFLN MK-RVFLN with evaporation-based water cycle based parameter optimization 174

FAAF fractional adaptive activation function, *see Section 4.3.18*; 139
FAB flexible activation bag, *see Section 4.3.55.2*; 169
FALU fractional adaptive linear unit, *see Section 4.3.18.4*; 140
FCAF fusion of chaotic activation function, *see Section 4.2.48.2*; 104
FELU fast ELU, *see Section 4.3.1.46*; l, 120, 197, 202, 208, 216
FESN functional ESN 178
FFNN feed-forward neural network 11, 14, 16, 48, 158, 168, 174, 175, 177
FISH fluorescence in situ hybridization 30
FOS-ELM fuzziness-based OS-ELM algorithm 176

FPAF fully parameterized activation function, *see Section 4.3.1.24*; 113, 207, 215, *Glossary:* fully parameterized activation function
FPFLU faster power function linear unit, *see Section 4.2.7.8*; 86
FPGA field-programmable gate array 18, 73, 175, 178
FPLUS first power linear unit with sign, *see Section 4.2.7.4*; 85
FracELU fractional ELU, *see Section 4.3.18.7*; 141
FracGELU fractional GELU, *see Section 4.3.18.9*; xxxv, 142
FracGELU1 FracGELU variant 1, *see Section 4.3.18.9*; 142
FracGELU2 FracGELU variant 2, *see Section 4.3.18.9*; 142, 143
FracLReLU fractional LReLU, *see Section 4.3.18.5*; 140
FracPReLU fractional PReLU, *see Section 4.3.18.6*; 141
FracReLU fractional ReLU, *see Section 4.3.18.1*; 139
FracSiLU fractional SiLU, *see Section 4.3.18.8*; xxxv, 141, 142
FracSiLU1 FracSiLU variant 1, *see Section 4.3.18.8*; 141, 142
FracSiLU2 FracSiLU variant 2, *see Section 4.3.18.8*; 141, 142
FracSoftplus fractional softplus, *see Section 4.3.18.2*; 139
FracTanh fractional tanh, *see Section 4.3.18.3*; 139
FReLU flexible ReLU, *see Section 4.3.1.15*; 82, 106, 110, 112, 122, 139, 196, 201
FSA Fourier series activation, *see Section 4.3.47.11*; 160, 211, 220
FSCN fast SCN xxxvi, 175
FSDESN fast subspace decomposition echo state network 178
FTS flatted-T swish, *see Section 4.2.6.46*; xxxix, li, 82, 134, 198
FunPReLU funnel parametric rectified linear unit, *see Section 4.3.1.4*; 106
FunReLU funnel rectified linear unit, *see Section 4.3.1.4*; 106
Fuzzy-ELM fuzzy ELM 176

G guanine 29, 30, 34, *Glossary:* guanine
G-BAPSO-SCN SCN with hybrid bat-particle swarm optimization 175
GA-SCN SCN based on genetic algorithms 175
GABU gating adaptive blending unit, *see Section 4.3.47.3*; 158, 210, 219
GAGAN geometry-aware GAN 182
GaLU Gaussian ReLU, *see Section 4.3.31.2*; 148
GAN generative adversarial network xxxi–xxxvi, xxxix–xliv, 4, 38, 42–47, 180–184, 276, 285, 290
GARBM Gaussian RBM with binary auxiliary units 181
GCD-GAN gradient-guided dual-branch GAN 182
GCN-RW graph convolutional networks with random weights 179
GCU growing cosine unit, *see Section 4.2.33*; 99, 100
GDTAAF generalized dual transformative adaptive activation function 290
GE gene expression v, xlvi–xlviii, li, 1–7, 9, 17, 29, 30, 33, 35, 36, 38, 39, 41–48, 175, 179, 181, 183, 192, 229, 270, 271, 275–277, 280–282, 284, 285, 287, 288, 290, *Glossary:* gene expression
GEGLU gated GELU, *see Section 4.2.4.3*; 67
GELU Gaussian error linear unit, *see Section 4.2.3.1*; xxxv, xxxix, xlii, l, 59, 60, 62, 65, 67, 89, 91, 111, 131, 134, 140, 142, 143, *Glossary:* GELU
GeNN genetic neural network xlviii, 42, 43, *Glossary:* genetic neural network
GEO Gene Expression Omnibus 38, 41, *Glossary:* Gene Expression Omnibus
GEU Gaussian error unit, *see Section 4.3.11*; 134, 135

GGAN graph GAN 182
GLN global-local neuron, *see Section* 4.3.26; 145
glsoftmax generalized Lehmer softmax, *see Section* 4.3.5; 133
GLU gated linear unit, *see Section* 4.2.4; 66, 67
GMDH group method of data handling 9, 10
GNN graph neural networks xxxiv, 42, 43, 161, 179, *see* NN
GO gene ontology 45
gpsoftmax generalized power softmax, *see Section* 4.3.6; 133
GPU graphics processing unit 3, 12, 41, 190, 222, 289, *Glossary:* GPU
GRA generalized Riccati activation, *see Section* 4.3.2.9; 128
GReLU generalized ReLU, *see Section* 4.3.23; 144
GRN gene regulatory network 43
GRNN gene regulatoryneural network 43
GTU gated tanh unit, *see Section* 4.2.4.1; 66, 130

H-ELM hierarchical extreme learning machine 176
HardELiSH hard exponential linear sigmoid squashing, *see Section* 4.2.7.18; 89
HCAF hybrid chaotic activation function, *see Section* 4.2.48.1; 103, 104
HcLSH hyperbolic cosine linearized squashing function, *see Section* 4.2.37; 100
HCR-ESN hybrid circle reservoir ESN 178
HGAN hyperbolic GAN 182
HPAF Hermite polynomial-based activation function, *see Section* 4.3.47.9; 160

I-ELM incremental ELM xxxii, 176
ILSVRC ImageNet Large-Scale Visual Recognition Challenge 12
IpLU polynomial linear unit, *see Section* 4.2.7.5; 86
ISRLU inverse square root linear unit, *see Section* 4.2.8.9; 92, 93
ISRU inverse square root unit, *see Section* 4.2.8.10; 65, 93, 145
ISSA-FSCN FSCN with an improved sparrow search algorithm 175

JCI Journal Citation Indicator xii, *Glossary:* Journal Citation Indicator
JIF Journal Impact Factor xii, *Glossary:* Journal Impact Factor

K-ELM kernel based ELM 176
K-RVFLN kernel RVFLN 174
k-WTA k-winner-take-all, *see Section* 4.2.46; 103, *Glossary:* k-winner-take-all
KAF kernel activation function, *see Section* 4.3.55.5; 140, 170, 171, 210, 219
KDAC knowledge discovery activation function, *see Section* 4.2.45; 102
KNN k–nearest neighbor 41, 47

L–ReLU Lipschitz ReLU, *see Section* 4.2.7.10; 87
LAAF locally adaptive activation function, *see Section* 4.3.15; 129, 135, 136, 202, 278, 279
LADA leaky apical dendrite activation, *see Section* 4.2.6.51; 84
LAF logarithmic activation function, *see Section* 4.2.13; 93, 94
LaLU Laplace linear unit, *see Section* 4.2.3.4; 60
LAU logmoid activation unit, *see Section* 4.3.15.5; liii, 94, 137

LEAF learnable extended activation function, *see Section 4.3.22*; 144
LeLeLU leaky learnable ReLU, *see Section 4.3.1.8*; 107, 196, 201
LinGeNN linear GeNN 43, *Glossary:* linear GeNN
LinQ linear quadratic activation, *see Section 4.2.8.8*; 92, 209, 217
LiSA linearized sigmoidal activation, *see Section 4.3.32.2*; xxxi, 150
LiSHT linearly scaled hyperbolic tangent, *see Section 4.2.3.28*; 64, 77, 130
LogSQNL square logistic sigmoid, *see Section 4.2.8.6*; 92
LP linear programming 43
LPAF Legendre polynomial-based activation function, *see Section 4.3.47.8*; 159, 160, 210, 219
LPSCN locality preserving SCN 175
LPSELU leaky parametric scaled exponential linear unit, *see Section 4.3.1.54*; xxxvii, 122, 208, 216
LPSELU_RP leaky parametric scaled exponential linear unit with reposition parameter, *see Section 4.3.1.55*; 122, 208, 216
LR linear regression v, xlviii, 2, 41, 177, 179, 185, 287
LReLU leaky ReLU, *see Section 4.2.6.2*; xxxv, xxxviii, xxxix, xliv, xlv, xlix, l, 52, 54, 61, 65, 68–70, 75, 82–84, 87, 88, 97, 105, 107, 109, 112, 117, 124, 136, 140, 143, 149, 155, 160, 165, 195, 196, 205, 206, 212, 290, *Glossary:* leaky ReLU
LRTanh linearized hyperbolic tangent, *see Section 4.2.2*; 51
LRTLU leaky rectified triangle linear unit, *see Section 4.2.6.29*; 77
LSELU leaky scaled exponential linear unit, *see Section 4.2.7.12*; 88, 205, 214
LSPTLU leaky single-peaked triangle linear unit, *see Section 4.2.6.29*; 77
LSTM long short-term memory 18, 42, 64, 101, 103
LuTU look-up table unit, *see Section 4.3.45*; 154, 155, 160, 209

m-arcsinh modified arcsinh, *see Section 4.2.26*; 98
M-RVFLN M-estimation-based RVFLN 174
MAE mean absolute error xxxviii, 41, 190–192, 233–235, 238–240, 245–251, 261, 267–271, 275
MAF multiquadratic activation function, *see Section 4.3.24*; 144
MarReLU marginPReLU, *see Section 4.3.1.3*; 106
MBA multi-bias activation, *see Section 4.3.30*; 147, 197, 203
MCC Matthew's correlation coefficient 193, 237, 241, 270, 272, 273, 282, 285, 444–455
MCDNN multi-column deep neural network 171
MDAE mean difference of absolute errors, *see Eq.* (5.3); xxxviii, 5
MEF modified Elliott function, *see Section 4.2.8.11*; 93
MeLU Mexican ReLU, *see Section 4.3.31*; xxxviii, xlix, 148, 166, 209
MI-CDBN mode isolation convolutional deep belief network 181
MI-ESN mutual information optimized ESN 178
MIN maxout-in-network 168
miRNA microRNA 1, 47, 48, *Glossary:* microRNA
MK-RVFLN multi-kernel RVFLN xxxiv, 174
ML machine learning li, 3, 11–13, 48
ML-DOS-ELM DOS-ELM 176
ML-ELM multi-layer extreme learning machine 176

ML-OCELM multilayer neural network based one-class classification with ELM 176
MLP multi-layer perceptron 13, 43, 136
MLReLU modified LReLU, *see Section 4.2.6.45*; 82
MMAE mean mean absolute error 5, 191, 229, 235–237, 249, 254, 255, 257–259, 261, 266–268, 270, 275, 276, 280, 281, 284, 285, 456–459
MMDAE mean MDAE, *see Eq.* (5.4); 191, 229–235
MMeLU modified Mexican ReLU, *see Section 4.3.31.1*; 148
MNN mutual nearest neighbor 47
MoGL-SCN Bayesian robust SCN based on a mixture of the Gaussian and Laplace distributions 175
MoGU mixture of Gaussian unit, *see Section 4.3.47.10*; 156, 160, 210, 219
MP max-pooling 10–12
MPELU multiple parametric exponential linear unit, *see Section 4.3.1.48*; 120, 124, 125, 208, 216
MR master regulator 42
MR-ESN multiple reservoirs echo state network 178
mReLU mirrored rectified linear unit, *see Section 4.2.6.28*; 76, 106
MRI magnetic resonance imaging xlii, 182, 183
mRNA messenger RNA xlvi, 1, 35, 37, 48, *Glossary:* messenger RNA
MSAF multistate activation function, *see Section 4.2.2.24*; xliii, 57, 211, 219
MSE mean squared error 25
MSiLU modified sigmoid-weighted linear unit, *see Section 4.2.3.23*; 63
MSRF mollified square root function, *see Section 4.3.54*; 164, 165
MTLU multi-bin trainable linear unit, *see Section 4.3.43*; 153, 154, 209, 218
MWF modified Weibull function, *see Section 4.3.38*; 152

N-PWLU non-uniform piecewise linear unit, *see Section 4.3.43*; 153
NAF neuron-adaptive activation function, *see Section 4.3.27*; 146, 199, 203, 209, 217
NCBI National Center for Biotechnology Information xlvii
NCReLU negative CReLU, *see Section 4.2.6.35*; 78
NCU non-monotonic cubic unit, *see Section 4.2.42*; 101
NFT non-fungible token 99
NGRC next generation reservoir computing 178
NGS next-generation sequencing li, 33
NIN network in network 113, 120, 131, 149, 168, 169
NLP natural language processing 54
NLReLU natural-logarithm-ReLU, *see Section 4.2.6.12*; 71
NN neural network, *see Chapter 2*; v, xxxiii, xxxiv, xxxvi, xxxviii, xl, xli, xlv–l, 2–4, 6, 9–18, 24–27, 39, 41–50, 56, 60, 64, 66, 68, 73, 79, 95, 96, 98, 103, 105, 111, 117, 120, 130, 150, 153, 161, 163, 168, 169, 171, 173, 174, 176, 177, 179–181, 184, 186, 193, 221, 222, 225, 229, 230, 232, 262, 264, 265, 276–284, 287–289, *Glossary:* neural network
NReLU noisy ReLU, *see Section 4.2.6.6*; 70, 108, 206, 213
nt nucleotide xlix, 35, 39

O-ESN particle swarm optimized ESN 178

OAF Optimal Activation Functio, *see Section 4.2.6.47*; 83
OLReLU optimized leaky ReLU, *see Section 4.2.6.2*; 69, 205, 212
OPAU orthogonal Padé activation unit, *see Section 4.3.51*; 162, 163
OPLAF odd power linear activation function, *see Section 4.2.6.41*; 81
OPLU orthogonal permutation liner unit, *see Section 4.2.6.37*; 79
OS-ELM online sequential ELM xxxiii, xxxiv, xli, 176
OSCN orthogonal SCN 175

p-swish parametricswish, *see Section 4.3.1.42*; 118
parameterized softplus parametrized softplus, *see Section 4.3.20*; 143, 145, 199, 203
PAU Padé activation unit, *see Section 4.3.48*; xxxix, xli, 111, 161, 162, 210, 219
PC Parallel Circuit 171, 222
PCA principal component analysis 38, 44, *Glossary:* principal component analysis
PDELU parametric deformable exponential linear unit, *see Section 4.3.1.59*; 124, 208, 216
PELU parametric exponential linear unit, *see Section 4.3.1.43*; xxxviii, 118–120, 123, 197, 208, 216
PESN polynomial ESN xli, 178
PFLU power function linear unit, *see Section 4.2.7.7*; xxxv, 86, 132
PFPLUS parametric first power linear unit with sign, *see Section 4.3.1.61*; 125
PFPM parametric flatten-p mish, *see Section 4.3.10*; 134
PFTS parametric flatted-T swish, *see Section 4.3.9*; 134, 198, 203
PGELU parametric Gaussian error linear unit, *see Section 4.3.8*; 134, 135
PiLU piecewise linear unit, *see Section 4.3.1.22*; 106, 112, 207, 215
PLAF power linear activation function, *see Section 4.2.6.41*; xxxiv, xxxix, 81
PLGAN panoptic layout GAN 182
pLogish parametric logish, *see Section 4.2.3.15*; 62, 196, 200
PLU piecewise linear unit, *see Section 4.3.34*; 149, 150, 209, 217
PMAF piecewise Mexican-hat activation function, *see Section 4.2.23*; 97
PoLU power linear unit, *see Section 4.2.7.6*; 86
PolyLU polynomial linear unit, *see Section 4.2.7.4*; 85
positive PReLU positive parametric rectified linear unit (PReLU), *see Section 4.3.1.2*; 106, 196, 200
PPAF piecewise polynomial activation function, *see Section 4.3.52*; 163
PPGN Plug and Play generative networks 183
PRBF piecewise radial basis function, *see Section 4.2.24*; 98
PReLU parametric rectified linear unit, *see Section 4.3.1.1*; xxxiv, xxxv, xxxvii, xxxix, xli, 54, 72, 82, 99, 105–108, 111–114, 118, 119, 124, 141, 148, 149, 201, 206, 214, *Glossary:* PReLU
PReNU parametric rectified non-linear unit, *see Section 4.2.6.15*; 72
PREU parametric rectified exponential unit, *see Section 4.3.1.9*; 83, 108, 206, 214
ProbAct probabilistic activation, *see Section 4.3.1.11*; 108, 138
pRPPSG p-recursive piecewise polynomial sigmoid generator, *see Section 4.2.2*; 51
PRSCN pruning regularization SCN 175

pRVFLN parsimonious random vector functional link network 174
pSechSig parametric SechSig, *see Section 4.2.2.22*; 57
PSELU parametric scaled exponential linear unit, *see Section 4.3.1.53*; xxxvii, 122, 208, 216
pserf parametric serf, *see Section 4.3.3.13*; 132
PSF parametric sigmoid function, *see Section 4.3.2.7*; 128
PSGU Parameterized self-circulating gating unit, *see Section 4.3.3.7*; 130, 131
PShELU exponential linear unit with a trainable horizontal shift, *see Section 4.3.1.56*; li, 197, 201, 289
psigmoid parametric sigmoid, *see Section 4.3.2.6*; 128, 198, 203
PSiLU parametric SiLU, *see Section 4.3.3.1*; 129, *Glossary:* PSiLU
PSoftplus parametric softplus, *see Section 4.2.18*; 96, 196, 201
PSSiLU parametric shifted SiLU, *see Section 4.3.3.3*; 129
PSTanh parametric scaled hyperbolic tangent, *see Section 4.3.15.2*; 136, 198, 203, 289
PSvELU exponential linear unit with a trainable vertical shift, *see Section 4.3.1.56*; li, 197, 202
ptanh penalized hyperbolic tangent, *see Section 4.2.2.9*; 54
pTanhSig parametric TanhSig, *see Section 4.2.2.23*; 57
PTELU parametric tanh linear unit, *see Section 4.3.1.30*; 58, 114–116, 136, 207, 208, 215
PUAF polynomial universal activation function, *see Section 4.2.16*; 95
PuVAE purifying VAE 181
PVLU parametric variational linear unit, *see Section 4.3.1.62*; 126
PWLU piecewise linear unit, *see Section 4.3.43*; xxxviii, 153

QNN quadratic neural network 14

RBF radial basis function xxxi, xxxix, 176
RBM restricted Boltzmann machine xxxv, 44, 157, 180, 181
RC reservoir computing xxxviii, 177–179, 185
RCN random convolution node 176
ReBLU rectified BLU, *see Section 4.3.1.38*; 117
ReGLU gated ReLU, *see Section 4.2.4.2*; 67
ReLTanh rectified linear tanh, *see Section 4.3.1.36*; 71, 116
ReLU rectified linear unit, *see Section 4.2.6*; xxxi, xxxiii–xxxviii, xl–liii, 47–50, 52, 54, 59, 61, 62, 64–66, 68–73, 75–85, 87–91, 93–95, 97–99, 101, 105, 106, 108–114, 117, 118, 120–122, 124, 125, 131, 135, 136, 138, 140, 143, 144, 146–149, 151, 153, 155, 156, 158–161, 163, 164, 168, 169, 184, 195–198, 209, 279, *Glossary:* ReLU
RePGAN GAN with residual partial modules 182
RePSHU rectified parametric sigmoid stretchage unit, *see Section 4.3.1.58*; 124
RePSKU rectified parametric sigmoid shrinkage unit, *see Section 4.3.1.58*; 124
RePSU rectified parametric sigmoid unit, *see Section 4.3.1.58*; 124
RePU rectified power unit, *see Section 4.2.6.39*; l, 3, 72, 80, 110
RESN robust echo state network 178
ReSP rectified softplus, *see Section 4.2.6.14*; 72, 206, 213
REU rectified exponential unit, *see Section 4.2.6.49*; 83, 91, 108

RI-GAN GAN with residual inception modules 182
RMAF ReLU memristor-like activation function, *see Section 4.3.1.29*; 114, 196, 201
RMDL random multimodel deep learning 179
RMS root mean square 134, 135
RNA ribonucleic acid xlvi, xlix, li, 1, 30, 34–37, *Glossary:* RNA
RNA-Seq RNA sequencing xli, 1, 29, 33, 34, 36–39, 41–48, *Glossary:* RNA-Seq
RNN recurrent neural network xlv, 14, 17, 18, 43, 66, 126, 177, 179, *Glossary:* recurrent neural network
RoCGAN robust conditional GAN 182
RPAU randomized Padé activation unit, *see Section 4.3.49*; 161
RPReLU react-PReLU, *see Section 4.3.1.5*; 106, 201
RReLU randomized leaky ReLU, *see Section 4.2.6.3*; lii, 65, 68–70, 79, 111, 161, 205, 213, *Glossary:* randomized leaky ReLU
rRNA ribosomal RNA 34, 35, *Glossary:* ribosomal RNA
RS-SCN SCN with rough set based attribute reduction 175
RSign react-sign, *see Section 4.3.13*; 135, 196, 201
RSP rand softplus, *see Section 4.2.19*; 96
RT–PReLU randomly translational PReLU, *see Section 4.3.1.10*; 108, 206, 214
RT–ReLU randomly translational ReLU, *see Section 4.2.6.11*; 71, 81, 108, 206, 213
RVFLN random vector functional link network xxxi, xxxiii, xxxvi, xxxvii, xl, xlii, xliv, 174, 175

S-PESN simplified PESN 178
S-RReLU softsign randomized leaky ReLU, *see Section 4.2.6.4*; 70, *Glossary:* softsign randomized leaky ReLU
SAAAF shape autotuning adaptive activation function, *see Section 4.3.16*; 138
SAAF smooth adaptive activation function, *see Section 4.3.28*; 138, 146
SAE sparse autoencoder 48
SAF spline interpolating activation function, *see Section 4.3.52*; 162, 163
SAGAN self-attention GAN 182
SAO-ELM structure-adjustable OS-ELM 176
SaRa swish and ReLU activation, *see Section 4.2.6.53*; 84
SAU smooth activation unit, *see Section 4.3.1.6*; 107
SAVE structured activation of vertex entropy 171, 172
SBAF Saha-Bora activation function, *see Section 4.2.12*; 94
SC soft clipping, *see Section 4.2.2.11*; 54
SC-mish soft clipping mish, *see Section 4.3.1.40*; 118
SC-swish soft clipping swish, *see Section 4.3.1.41*; 118
SCAA spatial context-aware activation, *see Section 4.2.6.10*; 71
scATAC-Seq single-cell ATAC-seq 45, 46
SCBNN stochastic configured Bayesian neural network 175
SCIBER single-cell integrator and batch effect remover 47
SCL-mish soft clipping learnable mish, *see Section 4.3.1.40*; 118
SCN stochastic configuration network xxxii–xxxv, xxxvii–xxxix, xli, 175
scRNA-Seq single-cell RNA-Seq 43–47
scVI single-cell variational inference 46

SE-DCGAN squeeze-excitation network-deep convolution GAN 182
SELU scaled ELU, *see Section 4.2.7.11*; 54, 59, 65, 87, 88, 111, 122, 140, 155, 205, 208, 213
SERLU scaled exponentially-regularized linear unit, *see Section 4.2.7.13*; xlv, 88
SESN sinusoidal ESN 178
SExp scaled exponential function, *see Section 4.3.15.4*; 137, 198, 203
SG Sigmoid-Gumbel, *see Section 4.2.2.17*; 56
SGD stochastic gradient descent xlv, 26, 27, *Glossary:* stochastic gradient descent
SGELU symmetrical Gaussian error linear unit, *see Section 4.2.3.2*; 59, 60, 195, 201
SGT scaled-gamma-tanh, *see Section 4.3.12*; 135
SHAP Shapley additive explanations 47, *Glossary:* Shapley additive explanations
ShELU exponential linear unit with a fixed horizontal shift, *see Section 4.3.1.56*; li, 194, 197, 201
ShHardTanh HardTanh with a fixed horizontal shift, *see Section 4.2.6.19*; 74, 197, 202
ShiLU adaptive shifted ReLU, *see Section 4.3.1.16*; 110, 196, 201, *Glossary:* ShiLU
SiELU Gaussian error linear unit with sigmoid activation function, *see Section 4.2.3.35*; 66
SigLU sigmoid linear unit, *see Section 4.2.6.52*; 58, 84
SiLU sigmoid-weighted linear unit, *see Section 4.2.3*; xxxii–xxxv, xl, xliv, xlvii–liii, 53, 58–64, 66, 117, 129–132, 141, 142, 144, 198
SinLU sinu-sigmoidal linear unit, *see Section 4.3.3.11*; 132, 210, 219
SLAF self-learnable activation function, *see Section 4.3.47.6*; 111, 159, 210, 219
SlReLU Sloped ReLU, *see Section 4.2.6.5*; 70, 106, 112, 195, 196, 200
SLS-SS scaled logistic sigmoid with scaled sine, *see Section 4.3.27.1*; 146, 209, 217
SLU softplus linear unit, *see Section 4.2.6.13*; 72
sMRI structural MRI 183
SMU smooth maximum unit, *see Section 4.3.1.7*; lii, 107
snmC-Seq single-nucleus methylcytosine sequencing 46
SNN self-normalizing neural network 87
SP-RVFLN sparse pre-trained RVFLN 174
SPECT single-photon emission computed tomography 183
SPLASH simple piecewise linear and adaptive function with symmetric hinges, *see Section 4.3.29*; 147
SPOCU scaled polynomial constant unit, *see Section 4.2.15*; 95
SQLU square linear unit, *see Section 4.2.8.2*; 91
SQMAX square softmax, *see Section 4.2.8.7*; 92
SQRT square-root-based activation function, *see Section 4.2.9*; 93
SqSoftplus square softplus, *see Section 4.2.8.5*; 92
SQU shifted quadratic unit, *see Section 4.2.44*; 102
squish square swish, *see Section 4.2.8.3*; 91

SReLU S-shaped ReLU, *see Section 4.3.32*; lii, 65, 124, 149, 150, 153, 166, 209, 217

SRS soft-root-sign, *see Section 4.2.2.10*; 54

SSAF S-shaped activation function, *see Section 4.2.9*; 93

SSBS smooth sigmoid-based shrinkage, *see Section 4.3.1.58*; 124

SSE sum of squared errors 25

sSELU scaled scaled exponential linear unit, *see Section 4.2.7.14*; 88, 206, 214

SSinH scaled sine-hyperbolic function, *see Section 4.3.15.3*; 136, 137, 198, 203

SSS shifted and scaled sigmoid, *see Section 4.2.2.1*; 51, 195, 200

SSU shifted sine unit, *see Section 4.2.35*; 100

STAC-tanh slope and threshold adaptive activation function function with tanh function, *see Section 4.3.2.8*; 128, 205, 212

stanh scaled hyperbolic tangent, *see Section 4.2.2.3*; 52, *Glossary:* scaled hyperbolic tangent

STGAN selective transfer GAN 182

STM short-term memory 177

SVAF slope varying activation function, *see Section 4.3.2.4*; 127, 198, 202, 279

SvELU exponential linear unit with a fixed vertical shift, *see Section 4.3.1.56*; li, 123, 194, 197, 201

SVESM support vector echo-state vector machine 178

SvHardTanh HardTanh with a fixed vertical shift, *see Section 4.2.6.19*; 74, 197, 202

SVM support vector machine 47, 98, 177, 179

SwiGLU gated swish, *see Section 4.2.4.4*; 67

SymMSAF symmetrical MSAF, *see Section 4.2.2.24*; 57

T thymine 29, 30, 34, *Glossary:* thymine

T-swish arctan swish, *see Section 4.3.1.57*; 123, 208, 216

TAAF transformative adaptive activation function, *see Section 5.2*; v, vi, xxxiv, xlv, 2–7, 173, 187–189, 192–199, 201, 203–212, 217, 220–222, 225, 226, 229–235, 237–251, 254–258, 260–262, 266–268, 270–273, 275–285, 287–290, 426–455, *Glossary:* transformative adaptive activation function

TAF trainable activation function 105, 156, *Glossary:* trainable activation function

TaLU tanh linear unit, *see Section 4.3.1.31*; l, 115

tanh hyperbolic tangent, *see Section 4.2.2*; xxxv–xxxvii, xl, xlii–xliv, xlviii, xlix, li–liii, 5, 6, 49, 51, 52, 54, 55, 58, 59, 62–65, 68, 73, 77, 79, 85, 87, 91–93, 98, 101, 108, 111, 114–116, 127, 131, 136, 138, 139, 143, 144, 149, 155, 156, 158, 164, 168, 169, 229–231, 233, 235–237, 239, 245, 252, 260, 261, 267, 275, 276, 280, 281, *Glossary:* hyperbolic tangent

TBSReLU tangent-bipolar-sigmoid ReLU, *see Section 4.2.3.19*; xliii, 62, 131

TBSReLUl TBSReLU learnable, *see Section 4.3.3.8*; 131

TCA trained compound activation function, *see Section 4.3.47.1*; xliii, 157, 211, 218

TCAv2 trained compound activation function variant 2, *see Section 4.3.47.1*; 157, 211, 218

TeLU tanh exponential linear unit, *see Section 4.3.1.34*; 115

ThLU tanh linear unit, *see Section 4.2.7.1*; 85, 114

TransGAN transformer based GAN 182
TRec truncated rectified, *see Section 4.2.6.21*; 3, 75
TReLU tanh based ReLU, *see Section 4.3.1.35*; xliv, 85, 116, 207, 215
TReLU2 TReLU variant 2, *see Section 4.3.1.35*; 116
TRN transcriptional regulatory network 43
TruG truncated gaussian unit, *see Section 4.3.53*; 164
TS-sigmoid triple-state sigmoid unit, *see Section 4.2.2.6*; 53, 60
TS-swish triple-state swish unit, *see Section 4.2.3.6*; 53, 60, 129
TSAF trapezoid-shaped activation function, *see Section 4.3.36*; liii, 151, 166
TSiLU hyperbolic tangent sigmoid-weighted linear unit, *see Section 4.2.3.24*; 63
tsoftmax tuned softmax, *see Section 4.3.4*; 132
TSP travelling salesman problem 184
TSReLU tangent-sigmoid ReLU, *see Section 4.2.3.18*; xliv, 62, 131
TSReLUl TSReLU learnable, *see Section 4.3.3.7*; 131
TWIESN time warp invariant echo state network 178

UAF universal activation function, *see Section 4.3.21*; xl, 95, 143, 144

VAE variational autoencoder xxxii, xl, 4, 43–46, 181–184
VAF variable activation function, *see Section 4.3.55.1*; 168, 169, 211, 220
VARGAN variance enforcing GAN 182
VBAF volatility-based activation function, *see Section 4.2.47*; 103
VLReLU very leaky ReLU, *see Section 4.2.6.2*; 68, 69, 155, 205, 212
VLU variational linear unit, *see Section 4.2.6.9*; xl, 71, 126
VML-ESN variable memory length ESN 178
vReLU V-shaped ReLU, *see Section 4.2.6.25*; lii, liii, 76, 77, 97, 165
VSF variant sigmoid function, *see Section 4.2.2.2*; 52, 195, 200

WCRVFLN wavelet-coupled RVFLN 174
WGAN-GP Wasserstein GAN with gradient penalty 44, 45, 182
WHE wide hidden expansion 168, 169
WiG weighted sigmoid gate unit, *see Section 4.2.3*; 59
WoS Web of Science xii, xlviii
WRN wide residual network 149, 177
WTA winner-take-all 10

GLOSSARY

β-softmax an extension of the softmax activation function (AF), *see Section 4.2.5.1*; 67

1Dmeta-ACON a MetaACON variant, *see Section 4.3.3.6*; 130

ACON-A an adaptive activation function from the ACON family; another name for the swish, *see Section 4.3.3.1*; xlv, 129

ACON-B an adaptive activation function from the ACON family; extension of the ACON-A, *see Section 4.3.3.5*; xlv, 130

ACON-C an adaptive activation function from the ACON family; extension of the ACON-B, *see Section 4.3.3.6*; xlix, 130

Adam a popular variant of the SGD optimization algorithm; more in [291] 26, 27

adaptive activation function an activation function that can adapt to the data; often, it has a parameter whose value is data-dependent v, xxxiv, xxxvi, xli, xliii, xlv, xlvii, xlix–li, liii, 2–4, 105, 107, 114, 121–129, 132, 134, 135, 137–140, 143–146, 148, 151–154, 157, 158, 160, 168, 169, 193, 194, 197–199, 207, 229, 231, 233, 277–279, 287, 290

adaptive transformative unit a proposed unit, used for the implementation of the proposed TAAF, *see Section 5.4.1.1*; 5, 225, 226

adenine a purine nucleobase. One of the four bases in the DNA. 29, 30, 34

Aranda-Ordaz an activation function, *see Section 4.2.20*; 96

arctangent the inverse of the tangent function, also used as an activation function, *see Section 4.2.2.4*; 52, 98

arctid an activation function, *see Section 4.2.28*; 98

ARiA2 an activation function; special case of ARiA, *see Section 4.3.37*; 151, 152

artificial neural network a biologically inspired computational model, interchangeably used with the term neural network, *see Chapter 2*; xlix, 2, 4, 9, 10, 13, 14, 22, 41, 42, 51, 173, 278

ASERLU an activation function; an extension of the SERLU for BiLSTM architectures, *see Section 4.2.7.13*; 88

backpropagation a method for calculation of the gradient of the loss function w.r.t. the network's weight 10, 11, 23–25, 50, 51, 111, 127, 174

bent identity an activation function, *see Section 4.2.10*; 94

bidirectional recurrent neural network a type of recurrent neural network where the outputs can use information from both past and future states, more in [447] 43

BipolarPlus an activation function proposed in [904], *see Section 4.3.54.2*; 164, 167

bounded leaky ReLU an activation function; ReLU variant combining BReLU and LReLU, *see Section 4.2.6.24*; 75, 97, 149, 206, 213

bounded ReLU an activation function; ReLU variant with bounds, *see Section 4.2.6.16*; xlv–xlvii, 3, 73, 75, 97, 165

BReLUPlus an activation function; a smoothed variant of BReLU, *see Section 4.3.54.7*; 165

Category Normalized Citation Impact a document citation metric calculated by Clarivate; it is equal to the number of a document's citations normalized by the expected number of citations for the same document type, publication year, and subject area [4] xlviii

cDNA copy DNA, also called complementary DNA; synthetic DNA transcribed from mRNA [381] using reverse transcriptase [379, p. 19] xlvi, xlviii, 35, 36, 46

CiteScore a journal citation metric calculated by Elsevier using items indexed in Scopus; it is equal to the average number of citations per document published in the last four years [5] xii

comb-H-sine an activation function, *see Section 4.2.25*; 98, 195, 201

cosid an activation function, *see Section 4.2.31*; 99

cRNA copy RNA, transcribed from cDNA during amplification phase 35

cytosine a pyrimidine nucleobase. One of the four bases in the DNA. 29, 30, 34

D–GEX a neural network for gene expression inference [2], *see Section 4.1.1*; v, 2–6, 38, 39, 41, 42, 47, 171, 187–190, 193, 221–225, 229–251, 266–268, 270–273, 275–277, 279–285, 287, 426–455

DELU an activation function proposed in [889]; not an abbreviation, *see Section 4.3.1.39*; 85, 117, 119

differential gene expression analysis a commonly used computational approach for identifying genes whose expressions are significantly different between two phenotypes [2112] 5, 192

DLU different name for SignReLU used in [904, 912], *see Section 4.2.6.32*; 77, 167

DNA an extended molecular structure for storing hereditary information xlv–xlvii, xlix, liii, 1, 4, 29–34, 46

DNA microarray used for measuring DNA levels [384]; mainly for gene expression analysis xlix, 3, 4, 29, 30, 32–34, 36, 37, 45

Dual Line an activation function; extension of DPReLU, *see Section 4.3.1.21*; 106, 112, 207, 215

dual parametric activation function an activation function inspired by DPReLU, *see Section 4.3.1.23*; xlvii, 112, 113, 207, 215

DualELU an activation function; an ELU variant similar to DualReLU, *see Section 4.2.7.2*; 85

DualReLU an activation function; a two-dimensional ReLU variant, *see Section 4.2.6.36*; xlvi, 79, 85

E-swish an activation function based on the swish, *see Section 4.3.3.4*; 123, 129, 130

E-Tanh an activation function, *see Section 4.2.40*; 101, 195, 200

4 see https://incites.help.clarivate.com/Content/Indicators-Handbook/ih-normalized-indicators.htm

5 see https://service.elsevier.com/app/answers/detail/a_id/14880/supporthub/scopus/

EIS a family of activation functions, not an abbreviation, *see Section 4.3.25*; 144, 145

Elliott an activation function, *see Section 4.2.2.15*; xxxvii, 55, 93

ELUPlus an activation function; a mollified variant of ELU, *see Section 4.3.54.13*; 167

ErfAct an adaptive activation function based on the swish, *see Section 4.3.3.12*; 132

expcos an activation function, *see Section 4.3.42*; 153

ExpExpish an activation function based on the SiLU, *see Section 4.2.3.13*; 61

exponential linear unit a popular activation function extending ReLU, *see Section 4.2.6.48*; xxxii–xxxv, xxxvii–xl, xlii, xliii, xlvi–xlviii, l, li, 54, 59, 61, 65, 74, 82–89, 91–93, 108, 111, 114, 115, 117–125, 140, 141, 145, 156, 159, 160, 167, 194, 197, 205, 208, 213

exponential swish an activation function related to the sigmoid-weighted linear unit (SiLU), *see Section 4.2.3.8*; 61, 129

external RNA control an approach for *microarray performance assessment*; see [391] for more details 35

fully parameterized activation function an activation function similar to DPAF, *see Section 4.3.1.24*; 113, 207, 215

GELU a popular activation function based on the cumulative distribution function of the normal distribution, *see Section 4.2.3.1*; xxxv, xxxix, xlii, l, 59, 60, 62, 65, 67, 89, 91, 111, 131, 134, 140, 142, 143

gene expression process of synthesizing gene product (e.g., a protein) using information encoded in a gene v, xlvi–xlviii, li, 1–7, 9, 17, 29, 30, 33, 35, 36, 38, 39, 41–48, 175, 179, 181, 183, 192, 229, 270, 271, 275–277, 280–282, 284, 285, 287, 288, 290

Gene Expression Omnibus a central and public repository of high-throughput gene expression data by NCBI, more in [430] 38, 41

generalized hyperbolic tangent an adaptive activation function, *see Section 4.3.2.1*; 126, 128, 146, 198, 202

generalized swish an activation function related to the SiLU, *see Section 4.2.3.7*; 60, 129

genetic neural network a neural network architecture for gene expression tasks presented in [439] xlviii, 42, 43

gish an activation function based on the SiLU, *see Section 4.2.3.10*; 61

GPU a specialized piece of hardware initially designed to accelerate image processing and computer graphics in general; often used for acceleration of training and inference of neural networks 3, 12, 41, 190, 222, 289

guanine a purine nucleobase. One of the four bases in the DNA. 29, 30, 34

Hard sigmoid an activation function similar to BReLU with sigmoid-like shape, *see Section 4.2.6.17*; xlvii, xlviii, lii, 73, 75, 89, 148

Hard-Swish an activation function similar to Hard sigmoid related to the swish function, *see Section 4.3.31.3*; 148, 149

Hardshrink an activation function similar to Hard sigmoid, *see Section 4.2.6.22*; xlviii, 75, 166

HardshrinkPlus an activation function; a smoothed variant of Hardshrink, *see Section 4.3.54.10*; 166
HardSReLUE an activation function; a parametric ELU, *see Section 4.2.7.16*; 89
HardTanh an activation function similar to Hard sigmoid with tanh-like shape, *see Section 4.2.6.18*; xlii, xliii, xlviii, 73, 74, 111, 166, 197, 202
HardTanhPlus an activation function; a smoothed variant of HardTanh function, *see Section 4.3.54.9*; 166
hat an ReLU-based activation function, *see Section 4.3.1.28*; 114
Hexpo an activation function, *see Section 4.2.2.12*; 54, 55, 89, 205, 212
Hi-C high-throughput method for detection of chromatin interactions 46
hybridization a process in which a cDNA binds to the probes on the microarray surface 29–31, 33–35
hyper-sinh an activation function, *see Section 4.2.27*; 98
hyperbolic tangent one of most common activation functions used in NNs, *see Section 4.2.2*; xxxv–xxxvii, xl, xlii–xliv, xlviii, xlix, li–liii, 5, 6, 49, 51, 52, 54, 55, 58, 59, 62–66, 68, 73, 77, 79, 85, 87, 91–93, 98, 101, 108, 111, 114–116, 127, 128, 131, 135, 136, 138, 139, 143, 144, 149, 155, 156, 158, 164, 168, 169, 229–231, 233, 235–237, 239, 245, 252, 260, 261, 267, 275, 276, 280, 281

improved logistic sigmoid an activation function based on the logistic sigmoid, *see Section 4.2.2.7*; 53, 128, 205, 206, 212

Journal Citation Indicator a journal citation metric calculated by Clarivate using items indexed in WoS; it is equal to the average CNCI published in the last three years xii
Journal Impact Factor a journal citation metric calculated by Clarivate using items indexed in WoS; it is equal to the mean number of citations for articles published in the last two years[6] xii

k-winner-take-all an activation function based on the winner-take-all principle, *see Section 4.2.46*; 103
k-means a greedy clustering algorithm 39

L1000 a cost-efficient gene expression assay, see [1] v, 1, 2, 29, 37–39, 41, 42, 44, 47, 275, 281, 282, 287, 288, 290
lasso a linear regression variant with l_1 regularization; more in [448] 43
leaky ReLU a popular ReLU based activation function; allows information flow for negative inputs unlike ReLU, *see Section 4.2.6.2*; xxxv, xxxviii, xxxix, xliv, xlv, xlix, l, 52, 54, 61, 65, 68–70, 75, 82–84, 87, 88, 97, 105, 107, 109, 112, 117, 124, 136, 140, 143, 149, 155, 160, 165, 195, 196, 205, 206, 212, 290
Li-ReLU an activation function; a combination of a linear function and ReLU , *see Section 4.2.6.33*; 78
linear GeNN a variant of GeNN with linear activation function [439] 43
logish an activation function based on the SiLU, *see Section 4.2.3.11*; xxxix, xlix, 61, 62, 167

6 see https://incites.help.clarivate.com/Content/Indicators-Handbook/ih-journal-citation-reports.htm

LogishPlus an activation function; a mollified variant of logish, *see Section 4.3.54.16*; 167

logistic sigmoid one of most common activation functions used in NNs, *see Section 4.2.2*; xlviii, l–liii, 5, 6, 49–53, 55, 57, 58, 60–63, 65, 66, 73, 82, 84, 92, 95, 98, 103, 111, 117–119, 126–132, 135–140, 142–144, 146, 149, 151, 154, 156, 158, 164, 167, 169, 184, 195, 198, 205, 207, 209, 211, 221, 230–235, 237, 239, 245, 246, 252, 267, 268, 276, 277, 280, 281

LogLog an activation function, *see Section 4.2.2.20*; xxxii, xlix, 56, 61

LogLogish an activation function based on the SiLU and the LogLog, *see Section 4.2.3.12*; 61

LReLUPlus an activation function; a smoothed variant of LReLU, *see Section 4.3.54.3*; 165, 166

LS–ReLU an activation function inspired by rectified linear unit (ReLU); not an abbreviation, *see Section 4.2.7.20*; 90

maxout unit an activation function returning the maximum of several linear functions, *see Section 4.3.46*; 69, 113, 149, 155, 168, 210, 218

maxsig an activation function, *see Section 4.2.7*; 84

maxtanh an activation function, *see Section 4.2.7.1*; 85

MeLUPlus an activation function; a smoothed variant of MeLU, *see Section 4.3.54.11*; 166

messenger RNA copied from DNA during transcription; used for protein synthesis during translation xlvi, 1, 35, 37, 48

MetaACON an extension of the ACON family where the parameter a_i is determined by a small NN, *see Section 4.3.3.6*; xlv, 130

MetaACON-C an ACON-C variant where the parameter a_i is determined by a small NN, *see Section 4.3.3.6*; 130

microarray used for measuring, usually DNA or RNA levels [384]; called DNA microarray when measuring DNA levels and RNA microarray when measuring RNA levels [384]; see [376] for other types v, vi, xlviii, 1, 4, 5, 9, 29, 32–37, 39, 42, 43, 45–47, 193, 282, 287

microRNA a small, non-coding RNAs containing 21 – 28 nucleotides 1, 47, 48

minsin an activation function, *see Section 4.2.6.8*; 71

mish an activation function; combination of tanh and softplus, *see Section 4.2.3.29*; xxxix, xli, xlix, lii, 61, 62, 64–66, 99, 115, 118, 131, 134, 159

MishPlus an activation function; a mollified variant of mish, *see Section 4.3.54.15*; 167

Mishra an activation function; unnamed in the original paper, *see Section 4.2.11*; 94

N-activation an adaptive activation function resembling the letter N, *see Section 4.3.32.1*; 149

n-sigmoid a sigmoid-based activation function, *see Section 4.2.2*; 51

neural network a biologically inspired computational model, interchangeably used with the term artificial neural network, *see Chapter 2*; v, xxxiii, xxxiv, xxxvi, xxxviii, xl, xli, xlv–l, 2–4, 6, 9–18, 24–27, 39, 41–50, 56, 60, 64, 66, 68, 73, 79, 95, 96, 98, 103, 105, 111, 117, 120, 130, 150, 153, 158, 161, 163, 168, 169, 171, 173–177, 179–181, 184, 186, 193, 221, 222, 225, 229, 230, 232, 262, 264, 265, 276–284, 287–289

NewSigmoid an activation function similar to logistic sigmoid, *see Section 4.2.2.18*; 56

NPF an adaptive activation function proposed in [1237]; based on Nonparametric Fourier Basis Expansion, *see Section 4.3.55*; 168

P+FELU trainable FELU based AF, *see Section 4.3.1.47*; 120, 202

P-E2-Id an adaptive activation function based on P-E2-ReLU, *see Section 4.3.1.49*; 121

P-E2-ReLU an adaptive activation function combining two ELUs and ReLU, *see Section 4.3.1.49*; l, 121

P-E2-ReLU-1 an adaptive activation function based on P-E2-ReLU, *see Section 4.3.1.49*; 121

P-E2-XU family of adaptive activation functions, *see Section 4.3.1.49*; 121

P-SIG-RAMP an adaptive activation function combining logistic sigmoid and ReLU, *see Section 4.3.14*; 135

paired ReLU paired ReLU, *see Section 4.3.1.26*; 113, 147, 196, 197, 201

pan a piecewise linear activation function, *see Section 4.2.6.26*; l, 76, 165

PanPlus an activation function; a smoothed variant of pan function, *see Section 4.3.54.6*; 165

PATS an activation function; not an abbreviation, *see Section 4.3.3.9*; 131

phish an activation function based on the SiLU and GELU, *see Section 4.2.3.16*; 62

polyexp an activation function, *see Section 4.2.38*; 100, 101

power activation function an activation function; also known as RePU, *see Section 4.2.6.39*; 80

PReLU a popular adaptive activation function; a LReLU variant with trainable leakiness, *see Section 4.3.1.1*; xxxiv, xxxv, xxxvii, xxxix, xli, 54, 72, 82, 99, 105–108, 111–114, 118, 119, 124, 141, 148, 149, 201, 206, 214

principal component analysis a dimensionality reduction method by linear transformation into a new coordinate system that respects the variance in the data 38, 44

probit another name for the cumulative standard distribution function when used as activation function, *see Section 4.2.2*; 51

PSiLU another name for the swish AF, *see Section 4.3.3.1*; 129

PTaLU an AAF, TaLU variant with another parameter, *see Section 4.3.1.32*; 115

randomized leaky ReLU a LReLU based activation function with stochastic leakiness during training, *see Section 4.2.6.3*; lii, 65, 68–70, 79, 111, 161, 205, 213

rectified hyperbolic secant an activation function based on the hyperbolic secant, *see Section 4.2.3.27*; 59, 64

recurrent neural network a type of neural network that contains loop in the information flow xlv, 14, 17, 18, 43, 66, 126, 177, 179

ReLU one of the most popular activation functions, *see Section 4.2.6*; xxxi, xxxiii–xxxviii, xl–liii, 47–50, 52, 54, 59, 61, 62, 64–66, 68–73, 75–85, 87–91, 93–95, 97–99, 101, 105–114, 117, 118, 120–122, 124, 125, 131, 135, 136, 138, 140, 143, 144, 146–151, 153, 155, 156, 158–161, 163, 164, 168, 169, 184, 195–198, 209, 279

ReLU-Swish an activation function; special case of FTS, *see Section 4.2.6.46*; 3, 82

ribosomal RNA a primary component of ribosomes; non-coding RNA 34, 35

RNA an extended molecular structure; stores hereditary information but also can catalyze biological reactions and control gene expression among other things xlvi, xlix, li, 1, 30, 34–37

RNA microarray used for measuring RNA levels [384] xlix

RNA-Seq a NGS RNA sequencing method; allows for measuring gene expression levels xli, 1, 29, 33, 34, 36–39, 41–48

root2sigmoid an activation function similar to logistic sigmoid, *see Section 4.2.2.19*; 56

rootsig an activation function, *see Section 4.2.2.25*; 57, 58

RSigELU an activation function; a parametric ELU, *see Section 4.2.7.15*; li, 88, 89, 171, 205, 212

RSigELUD an activation function; a variant of RSigELU with two parameters, *see Section 4.2.7.19*; 89, 171, 206, 214

scaled hyperbolic tangent a scaled variant of hyperbolic tangent, *see Section 4.2.2.3*; 52

scaled logistic sigmoid an activation function based on the logistic sigmoid, *see Section 4.3.27.1*; 146, 199, 203

scaled softsign an adaptive activation function; an adaptive variant of , *see Section 4.3.19*; 143, 209, 217

SechSig an activation function, *see Section 4.2.2.22*; xl, 57

self arctan an activation function based on the SiLU, *see Section 4.2.3.14*; 61

serf an activation function; combination of the Gauss error function and softplus, *see Section 4.2.3.32*; xl, 65, 132

Shapley additive explanations a game theoretic approach for explaining the output of a machine learning model 47

shifted exponential linear unit an activation function; an ELU with a vertical shift (SvELU, PSvELU) or an ELU with ahorizontal shift (ShELU, PShELU), *see Section 4.3.1.56*; 74, 123

Shifted ReLU an activation function; translated ReLU, *see Section 4.2.6.1*; li, 68, 82, 106, 110

ShiLU a ReLU based activation function; not to be confused with Shifted ReLU, *see Section 4.3.1.16*; 110, 196, 201

SigLin an activation function; a combination of the logistic sigmoid and a linear function, *see Section 4.2.2.8*; 53

sigmoid a mathematical function having *S-shaped* curve, logistic sigmoid is the most known example xxxix, xl, xlii–xliv, xlvii, xlix, 6, 48, 50, 52, 53, 55, 56, 58, 59, 66, 87, 124, 128, 169, 279

Sigmoid-Algebraic an activation function similar to logistic sigmoid, *see Section 4.2.2.5*; 53

SignReLU an activation function; a combination of ReLU and softsign, *see Section 4.2.6.32*; xlvi, lii, 77, 167, 205, 213

SignReLUPlus an activation function; a smoothed variant of , *see Section 4.3.54.18*; 168

sinc an activation function, *see Section 4.2.35*; 100

Sinc-Sigmoid an activation function, *see Section 4.2.2.16*; 55

sincos an activation function, *see Section 4.3.39*; 152

SineReLU an activation function; extension of ReLU, *see Section 4.2.6.7*; 70

sinp an activation function based on the sine function, *see Section 4.2.32*; 99

SinSig an activation function; uses logistic sigmoid and is similar to mish and swish, *see Section 4.2.3.34*; 66, 152

smish an activation function; combination of tanh, logarithm, and logistic sigmoid, *see Section 4.2.3.30*; 61, 65

smooth step an activation function, *see Section 4.2.2.14*; 55

SMU-1 an activation function; a variant of the SMU using a different smoothing approach, *see Section 4.3.54.3*; 165

soft exponential an activation function interpolating between logarithmic, linear, and exponential functions, *see Section 4.3.1.50*; 48, 121

softmax a popular activation function for classification problems; outputs a soft argmax of outputs of a given layer, *see Section 4.2.5*; xxxvi, xliv, xlv, 67, 92, 132, 133, 156, 169

SoftModulusQ an activation function; a quadratic approximation of the vReLU, *see Section 4.2.6.30*; 77

SoftModulusT an activation function; a tanh based approximation of the vReLU, *see Section 4.2.6.31*; 64, 77, 206, 213

softplus an activation function, *see Section 4.2.17*; xxxv, xxxix, xlix, li, 54, 61, 62, 64, 65, 68, 72, 89, 92, 95, 96, 111, 127, 139, 143, 145, 160, 164, 194, 196, 206

Softshrink an activation function similar to Hard sigmoid, *see Section 4.2.6.23*; lii, 75, 76, 165

SoftshrinkPlus an activation function; a smoothed variant of Softshrink, *see Section 4.3.54.5*; 165

softsign an activation function, *see Section 4.2.2.13*; li, lii, 55, 70, 77, 88, 143, 156, 167

softsign randomized leaky ReLU a RReLU based activation function combined with softsign, *see Section 4.2.6.4*; 70

SoftsignPlus an activation function; a mollified variant of softsign, *see Section 4.3.54.17*; 167

SQNL an activation function; not an abbreviation, *see Section 4.2.8.1*; 90–92

SquarePlus an activation function proposed in [1234], *see Section 4.3.54.1*; 164, 167

SReLUPlus an activation function; a smoothed variant of SReLU, *see Section 4.3.54.8*; 166

StarReLU an activation function; extension of ReLU, *see Section 4.3.1.17*; 110

StepPlus an activation function proposed in [904], *see Section 4.3.54.2*; 164

stochastic gradient descent a method for optimization of an objective function; it is a variant of gradient descent that uses stochastic batches of data instead of the entire dataset to calculate the gradient xlv, 26, 27

suish an activation function; proposed as the alternative to the SiLU and swish, *see Section 4.2.3.17*; 62

SwAT an activation function combining logistic sigmoid and arctan; not an abbreviation, *see Section 4.2.3.26*; 64

swim an adaptive activation function similar to the swish, *see Section 4.3.3.14*; 132

swish an adaptive activation function; an adaptive variant of SiLU, *see Section 4.3.3.1*; xxxix, xli, xliii–xlvii, l, lii, liii, 52, 54, 58, 60–62, 64–67, 74, 82–84, 88, 89, 91, 95, 108, 111, 114, 123, 129–131, 136, 140, 144, 145, 149, 151, 152, 156, 159, 167, 198, 203, 245, 246, 252, 257, 260, 261, 279

SwishPlus an activation function; a mollified variant of swish, *see Section 4.3.54.14*; 167

symexp an activation function inverse of the LAU, *see Section 4.2.14*; 94

symlog an alternative name of the LAU, *see Section 4.3.15.5*; 94

TanhExp an activation function; combination of tanh and exponential function, *see Section 4.2.3.29*; 62, 65, 115

tanhLU an AAF, combination of tanh and a linear function, *see Section 4.3.1.33*; 115, 207, 214

TanhSig an activation function, *see Section 4.2.2.23*; xl, 57

TanhSoft a family of adaptive activation functions proposed in [1095], *see Section 4.3.2.5*; 127

tent an ReLU-based activation function, *see Section 4.3.1.27*; 77, 113, 114

thymine a pyrimidine nucleobase. One of the four bases in the DNA. 29, 30, 34

trainable activation function an activation function; another name for the adaptive activation function (AAF) 105, 156

transformative adaptive activation function a class of adaptive activation functions allowing for translation and scaling of an activation function; proposed in this work, *see Section 5.2*; v, vi, xxxiv, xxxv, xlv, 2–7, 173, 187–189, 192–199, 201, 203–212, 217, 220–222, 225, 226, 229–235, 237–251, 254–258, 260–262, 266–268, 270–273, 275–285, 287–290, 426–455

triple an adaptive activation function, *see Section 4.2.43*; 101

TSAFPlus an activation function; a smoothed variant of TSAF, *see Section 4.3.54.12*; 166

vReLUPlus an activation function; a smoothed variant of vReLU, *see Section 4.3.54.4*; 165

wave an activation function, *see Section 4.2.41*; 101

1 INTRODUCTION

Deciphering the intricate mechanisms that govern gene expression (GE) is a challenging task in computational biology, as this fundamental process plays a crucial role in the production of a variety of proteins and ribonucleic acids (RNAs), each with distinct roles in cellular functions. Gene expression is the process through which the genetic information encoded in deoxyribonucleic acid (DNA) is transcribed into functional RNA molecules and then translated into proteins within a cell. The regulation of gene expression is a complex and dynamic process that involves various stages, ranging from transcriptional initiation to RNA processing, splicing, transport, and translation. The regulation of gene expression is tightly controlled by internal and external signals, environmental stimuli, and developmental cues.

Understanding the mechanisms underlying gene expression is essential to unraveling the fundamental principles that govern cellular function, contribute to disease pathology, and drive organismal development. The study of gene expression is of paramount importance in various scientific disciplines, from basic biological research to cutting-edge clinical applications. Gene expression patterns provide valuable insights into cellular processes, developmental biology, evolutionary dynamics, and cellular responses to diverse environmental cues.

The emergence of high-throughput technologies such as microarray analysis and RNA sequencing (RNA-Seq) has revolutionized the collection of transcriptomic datasets, enabling comprehensive exploration of gene expression profiles under different conditions, tissues, developmental stages, and organisms. These technologies have also led to the discovery of various RNAs that play crucial roles in regulating gene expression. For instance, microRNAs (miRNAs) are small non-coding RNAs that can bind to complementary sequences in messenger RNA (mRNA) molecules and inhibit their translation into proteins, thereby regulating gene expression. Similarly, long non-coding RNAs have also been shown to regulate gene expression by interacting with chromatin-modifying complexes and regulating the transcription of target genes.

Despite a significant drop in price in recent years, gene expression profiling is still relatively expensive for large-scale experiments, limiting its widespread adoption. To facilitate such experiments, the LINCS program developed the L1000 Luminex bead technology [1]. This technology measures the expression profile of approximately 1,000 selected *landmark genes* and then reconstructs the full gene profile of about 10,000 *target genes* [1] (see Section 3.3.3 for more details). The L1000 assay is a cost-effective alternative to traditional gene expression profiling methods, allowing researchers to study gene expression in large-scale experiments.

1.1 PROBLEM STATEMENT

The emergence of the cost-effective L1000 platform (see Section 3.3.3) represents a milestone in the field of transcriptomics, as it enables researchers to collect large datasets that capture diverse gene expression profiles encompassing numerous biological conditions — the total size of the collected dataset is over 1,300,000 gene expression profiles. To facilitate such a cost-effective method of measuring GE, the L1000 relies on computational methods to infer the full GE profile [1]. While the first approaches for inferring the full GE profile relied on the linear regression (LR) [2], more advanced methods based on an artificial neural network (ANN) emerged [2]. Artificial neural network — or, for the purposes of this work, NN in short — is a biologically inspired computational model that is able to model complex behavior through a composition of many simple operations (see Chapter 2 for a brief introduction into NNs). NNs are comprised of interconnected nodes that are (usually) organized into layers; these nodes are called neurons, and they aggregate a number of input signals to produce a single output that might be an input of a neuron in the following layer. Most commonly, the aggregation is an application of a non-linear function, called AF, to a linear combination of the input signals.

This work focuses on improving the full gene expression profile inference from the GE profiles of the landmark genes of the L1000 platform via neural networks. Specifically, it focuses on improving the performance inference of the NN based model called D–GEX introduced in [2] primarily in two ways — adaptive modification of AFs and modification of the original D–GEX architecture to allow for more expressive models given the same resource constraints. The full GE profile inference is a supervised task — it is a non-linear multivariate regression task with many *independent variables* and even more *dependent variables*. It is known that there are non-linear relationships between expression of individual genes [3–8].

1.2 MAIN CONTRIBUTIONS

The thesis makes significant contributions to the gene expression inference for profile reconstruction for the L1000 platform by exploring novel architectural and activation function improvements of the D–GEX neural network model. It introduces two key enhancements to the D–GEX architecture, significantly improving its performance.

The first innovation is a novel class of adaptive activation functions in Section 5.2, which dynamically adjusts activations of individual neurons based on input data during the training process, making the model more adaptable to diverse gene expression patterns. The activation function is called transformative adaptive activation function (TAAF) and introduces four parameters that allow for any horizontal and vertical scaling and translation of any inner activation function. Together with the introduction of TAAFs, an empirical evaluation of several activation functions in the D–GEX architecture was carried out and it was shown that the original D–GEX benefits from usage of different AFs even without the TAAFs — albeit the TAAFs partially alleviate the need for search of ideal activation function and improve the performance even further. The performance of TAAFs is shown using both DNA microarray and artificial data.

Furthermore, the thesis presents architectural refinements, specifically two novel parallel architectures within the D–GEX framework in Section 5.3, which enables more efficient use of resources, thereby significantly enhancing the predictive power of the NN given the same resource constraints. These two architectural improvements are the tower and checkerboard architectures [9] and markedly enhance the capacity of the NN while keeping the number of parameters unchanged — this leads to the identical memory profile as the original D–GEX and, therefore, the model can be accelerated using the same graphics processing unit s (GPUs) or other accelerators as the original NN. Another architectural improvement to the original D–GEX architecture is the introduction of skip connections in a ResNet manner.

While the focus of this thesis is on the GE profile reconstruction, both of the aforementioned improvements of D–GEX architecture apply to a broad class of NN models. The TAAFs [10] are a general class of activation functions that can be used for various tasks as we believe that this formulation is especially useful for a wider class of regression problems and show this using several artificial datasets. Similarly, the architectural improvements are not limited to the original D–GEX but can be used for any NN based model utilizing fully connected or similar layers. Therefore, the presented findings are of interest to a wider audience of NN researchers and machine learning (ML) engineers as they might be utilized in various fields and applications.

In addition to the aforementioned architectural advancements and the introduction of a new class of adaptive activation functions, the thesis provides a comprehensive list of activation functions in neural networks. This list is important, as even extensive surveys and reviews such as [11, 12] often omit many activation functions that are present in the literature. This can lead to cases where an activation function is redundantly proposed a few years later as a novel activation function, even though the same or a very similar activation function has already been introduced in the literature — e.g., rectified power unit (RePU) (Section 4.2.6.39), dual parametric ReLU (DPReLU) (Section 4.3.1.20), truncated rectified (TRec) (Section 4.2.6.21), ReLU-Swish (Section 4.2.6.46), and bounded ReLU (BReLU) (Section 4.2.6.16). By providing a more extensive list of available activation functions, the thesis aims to avoid such redundancy and promote faster advances of the research of activation functions in neural networks.

1.3 THESIS ORGANIZATION

After this introductory Chapter 1, where the task was introduced in Section 1.1 and the main contributions of this work summarized in Section 1.2, we continue with two background chapters — Chapters 2 and 3.

First, an introduction to the realm of neural networks is provided in Chapter 2, where we start with a brief overview of the development and history of NNs in Section 2.1 and then continue with basics of NNs in Section 2.2 describing building blocks of NNs: the basic unit — neuron — (Section 2.2.1) with an example of a simple NN (Section 2.2.2), various layer types (Section 2.2.3), and the training of NNs (Section 2.2.4).

Second, an introduction to the used biological terms and principles is provided in Chapter 3 — starting with a very brief overview of DNA and genetics in Section 3.1, continuing with the history of development of microarrays in Section 3.2, and slightly more detailed description of measuring gene expression with focus on DNA microarray in Section 3.3 (as a more detailed description would be out of the scope of this work, therefore many links to more detailed literature are provided).

After these two introductory chapters, we can fully immerse into the available literature in Chapter 4. We first review ANNs for GE inference and classification in Section 4.1 with focus on the D–GEX NN architecture in Section 4.1.1 as this architecture is the basis for our work. Moreover, other tasks related to the GE inference are reviewed — most notably clustering, analysis, and generation of GE data in Section 4.1.2.2 and classification of GE data in Section 4.1.2.3.

As already mentioned in Section 1.2, one of the contributions of this work is a literature review of available activation functions. First, regular activation functions that are not adaptive or trainable, as these are still most commonly used, are reviewed in Section 4.2. Then we focus on the adaptive activation functions in Section 4.3 as they have been getting more attention recently and become popular.

In Section 4.4, we briefly overview selected architectures of NNs with parallel connections as the proposed tower and checkerboard architectures use parallel blocks of neurons to increase a network's capacity without increasing the number of weights.

The last part of literature review, Section 4.5, focuses on neural networks with random weights (Section 4.5.1) and neural networks for data generation (Section 4.5.2) to provide context for Section 5.1.2 where we use a NN with random weights to generate several artificial datasets to show the performance of TAAFs on other regression tasks besides the GE inference. The overview of neural networks with random weights in Section 4.5.1 is general and is not limited to NNs for data generation and other applications, including various supervised classification and regression tasks such as load forecasting and object recognition, are reviewed. Then we review NNs for artificial data generation such as GANs and VAE in general in Section 4.5.2 — including listing applications that can be solved using these approaches that are not limited to data generation such as anomaly detection and object tracking. Finally, we review more in-depth the much narrower intersection of the two previous sections — neural networks with random weights for data generation — in Section 4.5.3 as this is most relevant for the artificial datasets generated in Section 5.1.2.

After the two background Chapters 2 and 3 and the literature review in Chapter 4 setting context to this work, we can finally dive into the practical research in Chapter 5 Neural networks with random weights for data generation where we describe the proposed transformative adaptive activation function, the tower and checkerboard architectures, and also the data, training procedure, and the performance evaluation. First, we start with preliminaries that are common for experiments of both concepts in Section 5.1. The datasets used for experiments are described in Section 5.1.1 including the *GEO*- series aware variant in Section 5.1.1.1 and the normalization in Section 5.1.1.2. While most of the conducted experiments used real microarray data described in Section 5.1.1, we have also done a few experiments with artificial data to show the usability of the TAAFs outside the omics field — the creation of the artificial datasets is described in Section 5.1.2.

The training procedure and the baseline architecture (reimplementation of the D–GEX from [2]) are then described in Section 5.1.3. Then we describe the metrics we have used for performance evaluation including the MMAE and MDAE, describe the statistical tests used for comparison of two models and their predictions in Section 5.1.4.1, and also focus on the evaluation of the practical impact on a subsequent analysis — we opted for differential gene expression (DGE) analysis — in Section 5.1.5 to show that the lowered inference error has also practical benefits.

One of the main contributions of this work, the transformative adaptive activation functions are described next in Section 5.2 where we first start with the formulation of TAAFs (Eq. (5.5)) and continue with the motivation for the used formula and the proposed parameters in Section 5.2.1. The motivation of the parameters includes a discussion of the activation functions that the TAAFs generalize in Section 5.2.1.1 Activations as special cases of TAAFs and also the discussion of AFs that employ similar concepts as TAAFs in Section 5.2.1.2 Activations as special cases of TAAFs. The TAAFs are also useful for regression tasks as they can be used in the output layer instead of the commonly used linear function, which we describe in Section 5.2.1.3. Last, we briefly described the used ensembling approach in Section 5.2.2.

Another main contribution, besides the TAAFs from Section 5.2 and the overview of activation functions in Sections 4.2 and 4.3, are the tower and checkerboard architectures that are described in Section 5.3. We start the description with the simpler tower architecture in Section 5.3.1, including the motivation for such architecture, and subsequently we describe its extension called checkerboard architecture in Section 5.3.2. Both of these architectures can be extended using skip connections in a ResNet-like manner [13]; these tower and checkerboard architecture variants are described in Section 5.3.3.

Then we move onto practical matters in Section 5.4 Implementation. We describe the used libraries for the practical part of the work and also the adaptive transformative unit (ATU) — the basic building block of a TAAF that simplifies the implementation — in Section 5.4.1.1.

The Chapter 6 TAAF as the application of ATUs empirically evaluates the methods introduced in Chapter 5. We start by evaluation of the TAAFs using the microarray GE data provided by the authors of D–GEX [2] for empirical evaluation in Section 6.1. We establish that the D–GEX equipped with TAAFs leads to improved performance over the plain D–GEX in Section 6.1.1, then we show that the performance can be further improved by using logistic sigmoid AF instead of hyperbolic tangent (tanh) in the TAAFs and that this can be seen only as a different initialization of parameters of a TAAF with tanh in Section 6.1.2; we also show that even the original, plain D–GEX benefits from using the logistic sigmoid AFs instead of the originally used tanh AFs.

After establishing that the TAAFs improve the performance of the GE inference in Sections 6.1.1 and 6.1.2, we proceed with the analysis of the improvements. In Section 6.1.3, we show that the performance improvements are not just due to the increased number of parameters of a NN but rather to the TAAFs themselves. The TAAF formula is empirically analyzed in Section 6.1.4 where we show that all four TAAF parameters are improving the performance and that any subset of the trainable parameters leads to a strictly worse performance using Wilcoxon sign rank test for evaluation.

The GE inference task is a regression problem and NNs commonly use a linear activation function in the last layer to allow the output range reach any value even if sigmoid AFs were used; however, this is no longer necessary with TAAFs and we show that using TAAFs even the last layer leads to further performance improvements in Section 6.1.5.

The previously described sections focus on establishing the individual improvements by comparing identical models with a single change; however, the Section 6.1.7 revisits the models from the previously described section with a focus on the overall performance and provides the best-performing models.

Unlike the works of [2, 14, 15], we also show that the established performance improvements from Section 6.1 also have a practical impact on the subsequent analyses of the inferred GE data in Section 6.2. We ran repeated differential gene expression analyses using either artificial phenotypes (Section 6.2.1) or real phenotypes using a subset of the data (Section 6.2.2).

To show the application of TAAFs outside the field of omics, we proceed with experiments using artificially generated data in Section 6.3. This includes again establishing the performance of TAAFs in Section 6.3.1 but also various analyses — an analysis of the impact of noise to the targets on the performance in Section 6.3.2, an analysis of the performance impact of the inference network's layer configurations in Section 6.3.3, a demonstration that the results are consistent over various initialization of the data generation networks in Section 6.3.4, and analyses of the impact of the width (Section 6.3.5) and depth (Section 6.3.6) of the data generation network.

After establishing that TAAFs indeed improve the performance of NNs in the tested tasks, we show that another main contribution of this work — tower and checkerboard architectures — improves the performance even further in Section 6.4. The Section 6.4.1 contains statistical testing of the significance of the observed improvements in performance. The Section 6.4.2 analyses the impact of dropout rates on the best-performing checkerboard architecture.

Similarly as for TAAFs, once we establish that the architectural improvements lower the inference error, we show that the architectural modifications have a statistically significant practical impact on the subsequent analyses with the inferred data in Section 6.5.

The results from the experiments from Chapter 6 are then discussed in Chapter 7 which loosely follows the structure of previous chapter: we discuss

TAAFs for GE inference task in Section 7.1, where we first discuss the performance improvements in Section 7.1.1, the TAAFs parameters in Section 7.1.2, the conceptual simplification for regression tasks in Section 7.1.3, general performance from the perspective of the GE inference task in Section 7.1.4, and the practical impact of TAAFs in Section 7.1.5, then we continue with discussion of the results on artificial data in Section 7.2, and finally we discuss the tower and checkerboard architectures in Section 7.3 including their practical impact on the subsequent DGE analysis in Section 7.3.1.

At last, we conclude the work in Chapter 8 including possible future research directions extending this work in Section 8.1. Additionally, supplementary figures are provided in Appendix A.

2 NEURAL NETWORKS — A BRIEF OVERVIEW

This work focuses on improving the gene expression inference using deep learning models introduced in [2]. Since this work is about an artificial neural network model for gene expression inference, there might be readers that are unfamiliar with one of the fields — the goal of the following two chapters is to provide a basic introduction to both areas so the presented work is somewhat understandable even by a reader unfamiliar with either of fields.

The basic concepts of deep learning are presented in this chapter, whereas basic concepts of gene expression measuring and inference using microarrays are shown in the following Chapter 3. The notation utilized in this work may not always strictly conform to mathematical conventions; however, it is prevalent within the field, and we have retained it for consistency and clarity.

2.1 BRIEF HISTORY OF NEURAL NETWORKS AND DEEP LEARNING

The realm of neural networks (NNs) encompasses a vast expanse, having existed for several decades, with a notable surge in the past two decades. Consequently, this chapter offers a cursory exposition, focusing exclusively on a sub-domain known as deep learning (DL) within neural networks. DL gained prominence in recent years for its remarkable aptitude in resolving intricate problems that were hitherto formidable to address. The goal of this section is to provide a brief history of neural networks to provide context for current trends; a more detailed history of deep learning is available in [16–18].

2.1.1 *Early neural networks*

While it is difficult to say when the neural networks first appeared as they gradually evolved from the linear regression methods that have been around for a long time as the first works using linear regression appeared in the early 19th century [16]. The nascent NNs could not glean knowledge from data [16, 19]; McCulloch initially introduced neural networks as a form of logical calculus [19]. More details about the history of neuron models are available in [17]. The seminal learnable network emerged in 1949 in [20] (reference from [16]), where Hebb introduced the concept of unsupervised learning for NNs. Subsequently, approaches to supervised learning in NNs emerged, exemplified by the perceptron algorithm in 1958 [21], other examples in [16].

In the realm of deep learning history, the year 1965 saw the advent of deep networks founded on the group method of data handling (GMDH) [22–24]. These GMDH based networks represented a pioneering endeavor in the development of Feedforward Multilayer Perceptron-type deep learning systems [16]. While earlier neural networks with a single hidden layer existed (e.g., [25, 26]), GMDH based networks stood out by employing polynomial activation functions, specifically implementing Kolmogorov–Gabor polynomials, which offered greater generality than other contemporary activation functions [16]. Furthermore, GMDH based networks were the first deeper networks described in literature — an eight-layer deep GMDH network appeared in 1971 in [22]. The GMDH network employed a hierarchical structure where only elements whose performance exceeded the given threshold were allowed to pass to the next layer [22]; it involved incremental growth and training of layers through regression analysis with subsequent pruning facilitated by a validation set, akin to modern decision regularization techniques [16]. The number of layers and units per layer were problem-dependent and adaptable. This pioneering approach exemplified open-ended, hierarchical representation learning in neural networks [16].

In 1979, Fukushima introduced the Neocognitron, a groundbreaking neural network architecture that incorporated neurophysiological insights and is often credited as one of the earliest deep artificial neural networks [16, 27]. The Neocognitron introduced convolutional neural networks (CNNs or convnets), featuring receptive fields that systematically traversed 2D arrays of input data, such as image pixels. This mechanism, characterized by significant weight replication, reduced the number of parameters required to describe the convolutional layer's behavior [16].

Despite its resemblance to modern supervised feedforward deep learning architectures with alternating convolutional and downsampling layers, Fukushima's Neocognitron employed local, winner-take-all (WTA) based unsupervised learning rules [27] or pre-wired weights, rather than supervised backpropagation [16]. This difference from contemporary deep learning practices shows that Fukushima did not address the problem of deep learning despite his architecture's considerable depth [16]. Notably, spatial averaging was used for downsampling, in contrast to the now popular max-pooling (MP) mechanism [16].

2.1.2 *The ascent of backpropagation*

In the realm of gradient-based error minimization within complex, nonlinear, and differentiable multi-stage neural network related systems, the historical trajectory dates back to the early 1960s [16]. According to Schmidhuber, the methodology of steepest descent in the weight space was introduced in [28–30], leveraging iterative applications of the chain rule using dynamic programming (DP) concepts. A simplified derivation of this approach, termed backpropagation (BP), relied solely on the chain rule, as elucidated in [31].

These approaches were efficient from the point of view of DP already in the 1960s [16]. However, they propagated derivative information through standard Jacobian matrix calculations between adjacent "layers" without explicitly considering direct interlayer connections or potential efficiency enhancements related to network sparsity. Surprisingly, despite the prior body of work on learning in multilayer NN-like systems, including deep nonlinear networks since 1965, a seminal book [32] in 1969 dampened enthusiasm for further NN research, particularly in the context of simple linear perceptrons [16].

The emergence of explicit, efficient error backpropagation (BP) in arbitrary, discrete, and possibly sparsely connected NN-like networks can be traced back to Linnainmaa's 1970 master's thesis [33], though it lacked reference to neural networks at that time [16]. BP, also known as the reverse mode of automatic differentiation, involved costs of forward activation spreading nearly equivalent to the costs associated with backward derivative calculation [16].

Efficient BP was swiftly employed to minimize cost functions by adapting control parameters (weights), as demonstrated in [34] by Dreyfus in 1973 [16]. The use of BP in neural networks first appeared in 1981 [16] in [35]. The BP became quite popular during the 1980s, e.g., [36–38], though it seemed that BP was suitable only for shallow networks [16].

By the late 1980s, it became evident that backpropagation alone did not provide a universal solution [16]. Most applications of feed-forward neural networks predominantly utilized networks with few hidden layers, with additional hidden layers often not yielding discernible empirical advantages [16]. A theorem, as put forth in [39–41], offered solace for many researchers by asserting that a single-layer neural network with a sufficient number of hidden units could accurately approximate any multivariate continuous function [16].

In summary, while backpropagation theoretically allows for deep problem-solving, it appeared to excel primarily in shallow problem domains. The late 1980s and early 1990s witnessed the emergence of a few promising ideas aimed at addressing this challenge [16].

Multiple optimization techniques have been proposed to improve the efficiency of BP for neural networks in following years [16]. These methods include least-squares [42, 43] and quasi-Newton approaches, along with strategies like momentum [38] and sign-only error derivatives BP variants such as R-prop [44] event though the least-squares and quasi-Newton approaches have been recognized as computationally too expensive for large neural networks [16]. Gradient normalization techniques and dynamic learning rate adaptations have also been explored. Numerous additional enhancements and tricks exist to further boost neural network performance, as documented in the literature — see [16] for a general overview.

The first application of backpropagation to convolutional neural networks occurred in 1989 when LeCun applied BP to a network *LeNet* similar to *Neocognitron* to recognize handwritten digits of the MNIST dataset [16, 37]. This integration, combined with max-pooling and graphics card optimization, has become a fundamental component of contemporary, high-performing, feedforward visual Deep Learners [16]. These advancements have been instrumental in various competitions and benchmark records, including superhuman vision performance, object detection, segmentation, and more. Additionally, this work introduced the widely renowned MNIST dataset [45] for handwritten digit recognition, which has become a prominent benchmark in machine learning [16].

However, during that period, the majority of utilized neural networks remained shallow due to a prevalent issue known as the *Fundamental Deep Learning Problem* [16, 46], characterized by vanishing or exploding gradients, which was initially documented in [47] (ref. from [16]).

2.1.3 *Winning competitions*

ML competitions play a pivotal role in the discovery of superior algorithms and approaches, with neural networks garnering increased attention as they consistently achieve victories, particularly in the domain of pattern recognition.

Although neural networks had secured victories in several competitions during the 1990s and beyond (see [16] for details), it was the advent of the deep network known as *AlexNet* in 2012 that marked a significant breakthrough. AlexNet triumphed in the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [48], achieving a top-5 test error rate of 15.3%, a remarkable improvement over the second-best entry, which had an error rate of 26.2% [49]. This achievement was especially noteworthy as *AlexNet* had 60 million parameters and 650,000 neurons distributed across eight layers, pushing the limits of contemporary hardware and requiring training on 2 GPUs.

The success of convolutional neural networks (CNNs) was further affirmed in the ILSVRC 2013, where *ZF Net*, a CNN based on the *AlexNet* architecture, achieved an even lower error rate of 11.2%. While ZF Net retained the core structure of AlexNet, the primary contribution of the paper introducing it was a novel technique for visualizing feature maps and analyzing the network's responses to various input data transformations [50].

Deep CNNs have demonstrated a key advantage in their ability to deliver exceptional performance while maintaining relative simplicity, provided they possess sufficient depth. This was exemplified by the VGG networks introduced in a study by Simonyan and Zisserman. VGG networks employed compact 3×3 filters, the smallest size that captures positional information, with a stride and padding of 1 [51]. Additionally, these networks incorporated 2×2 max-pooling layers after some of the convolutional layers, and the last three layers were fully connected. The authors proposed multiple network architectures within this framework, ranging from 11 to 19 layers [51]. This approach represented a departure from previous networks that employed larger convolutional layers, such as the 7×7 layers used in [50] or the 11×11 layers in [49]. Simonyan and Zisserman demonstrated that comparable performance could be achieved with 3×3 convolutional layers, offering the added advantage of reduced parameter count.

Since then, neural networks started to dominate many ML competitions and became one of the most used ML tools. A transformative idea was introduced by Google's deep learning group, featuring the split, transform, and merge paradigm, embodied in the *Inception unit* in their architecture *GoogLeNet* [52]. The *GoogLeNet* network contains parallel connections instead of simply chaining all layers. Each *Inception unit* comprises multiple parallel streams, and dimensionality reduction is applied to these streams; see [52] for details. This dimensionality reduction is essential to manage a single unit's depth effectively. A notable aspect of this architecture is its emphasis on computational efficiency. To achieve this, traditional dense layers are replaced with average pooling layers. This modification results in a significant reduction in the number of parameters, with the entire network having 12× fewer parameters compared to the *AlexNet* [52]. This innovation pioneered branching within layers, enabling the abstraction of features at varying spatial scales [53].

In 2015, the concept of skip connections, initially proposed by ResNet, gained widespread adoption for training deep CNNs. This concept has been incorporated into subsequent networks such as Inception-ResNet [54], Wide ResNet [55], [56], and others.

Architectural designs such as Wide ResNet [55], ResNeXt [56], Pyramidal Net [57], Xception [58], PolyNet [59], and more have explored the impact of multilevel transformations on CNNs' learning capacity [53]. This exploration has involved introducing cardinality or increasing network width [53].

Consequently, the research focus has shifted from parameter optimization and connection adjustments to the enhancement of network architecture [53]. This shift has given rise to many innovative architectural ideas, including channel boosting, spatial and feature-map-wise exploitation, and attention-based information processing [53].

Nowadays, the neural networks dominates many ML tasks, and there are various specialized architectures for individual tasks. It would be greatly out of the scope of this work to list all notable recent achievements and applications in the realm of neural networks. Books [46, 60] and reviews about neural networks and deep learning in general [16, 53, 61–64] are recommended to a curious reader. More specialized and narrowly focused reviews of applications of neural network are available, for example, in [65–118].

2.2 BUILDING BLOCKS OF NEURAL NETWORKS

Artificial neural networks (ANNs), often referred to as simply neural networks (NNs) are a class of ML models inspired by the structure and function of biological neural systems. They have garnered significant attention and proven to be highly effective in various fields, including computer vision, natural language processing, speech recognition, and data analysis. Most neural networks are structured architectures that consist of well-defined blocks that consist of individual neurons. Their complexity can vary from very simple architectures such as the multi-layer perceptron (MLP) [38] to very complicated architectures such as GoogLeNet [52], Inception-v4 and Inception-ResNet [54], and transformer [119] based models such as BERT [120] or LLaMMA [121]. This section aims to provide a cursory introduction to the field of neural networks; a more detailed introduction to the field is available in [46, 122–125].

2.2.1 *Basic unit — neuron*

An artificial neural network is a biologically inspired computational model consisting of individual neurons that are connected. It can be represented as a weighted graph, where individual neurons are nodes and paths through which signals flow are depicted as edges. Each neuron takes multiple inputs, combining them into a single output signal, which is then distributed to all of its output connections. Typically, these inputs are aggregated using weighted summation, with the weights assigned to connections through which the signals arrive at the neuron. There are two main types of neural networks — feed-forward neural networkswith unidirectional flow of information (the information flows in one direction from input to output layer) and recurrent neural networks where the information flow forms a cycle [64]. Other common types of neural networks include *radial basis function neural networks*, *Kohonen neural networks* (also called *self-organizing maps*) [64]. Since this work focuses on real-valued feed-forward neural networks, recurrent neural networks and other types of networks are out of the scope of this brief overview; for more about recurrent neural networks (RNNs) see, for example, reviews [126–128].

More precisely in an usual real-valued feed-forward neural network, the output y_i of a single neuron i with inputs $x_0, x_1, \ldots, x_n$ with weights $w_0, w_1, \ldots, w_n$, and a bias term b is defined as

$$y_i = f\left(b_i + \sum_{j=0}^{n} w_{i,j} x_j\right), \tag{2.1}$$

where f is an activation function[1] [122, 130–132]. The Eq. (2.1) is simplified for the case when the time indices are not important for clarity. If the processing time is an issue and the time indices are needed, then the output the output y_i of a single neuron i with inputs $x_0, x_1, \ldots, x_n$ with weights $w_0, w_1, \ldots, w_n$, a bias term b, and activation function f at time t is defined as

$$y_i[t] = f\left(b_i + \sum_{j=0}^{n} w_{i,j} x_j[t-1]\right) \tag{2.2}$$

based on [133, p. 7]. However, this work focuses on feed-forward networks with uni-directional flow of information where the specific processing time is not important; therefore, the time indices will be dropped for clarity as in Eq. (2.1).

The Eq. 2.1 is often written in a matrix form:[2]

$$x_i = f\left(\boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{x}\right), \tag{2.3}$$

1 There are also element-wise activation functions that are applied directly to the x_j and can be thought of as nonlinear gating functions [129].

2 This type of neuron is used in the most common type of NNs; there are other types such as the quadratic s (QNNs) [134–142] but these are not discussed in this work.

where

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_0 \\ \vdots \\ x_n \end{bmatrix}, \tag{2.4}$$

and

$$\boldsymbol{w}_i = \begin{bmatrix} b_i \\ w_{i,0} \\ \vdots \\ w_{i,n} \end{bmatrix}. \tag{2.5}$$

The activation function f is what differentiates neural networks from a simple linear classifier and allows for meaningful stacking of neurons if the activation function is non-linear as the output of arbitrarily deep neural network with linear activation functions can still be expressed as a linear combination of inputs which is precisely what a single neuron does [11]. Historically, the activation functions modeled the activation of the neuron in the range between 0 and 1 [122] — the most straightforward activation is a *signum*-like function which is used in perceptron:

$$y = \begin{cases} 1 & b + \sum_{j=0}^{n} w_j x_j > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where b is the bias term, $x_0, \ldots, x_n$ are real valued inputs and $w_0, \ldots, w_n$ are weights. However, historically, the most common activation function is the logistic sigmoid (see Section 4.2.2), which produces a continuous output in the range of 0 to 1 [122]. Activation functions are discused in depth in Sections 4.2 and 4.3.

2.2.2 *Simple neural network*

While a single neuron can be considered as the simplest neural network, typical neural network is a hierarchical model consisting of interconnected neurons. A simple neural network is shown in Fig. 2.1.

The output y of the example network from Fig. 2.1 for given inputs $x_0, \ldots, x_3$ is:

$$y = \sigma\left(b_7 + w_{4,7}\sigma\left(b_4 + \sum_{j=0}^{3} w_{j,4}x_j\right) + w_{5,7}\sigma\left(b_5 + \sum_{j=0}^{3} w_{j,5}x_j\right) + w_{6,7}\sigma\left(b_6 + \sum_{j=0}^{3} w_{j,6}x_j\right)\right) \tag{2.7}$$

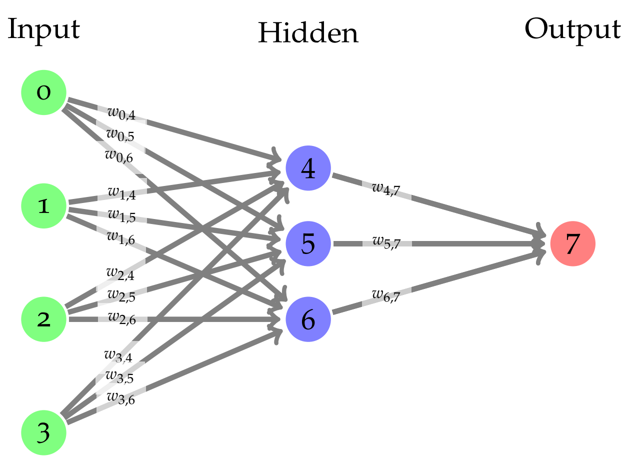


Figure 2.1: An example of a simple feed-forward neural network with 4 input neurons, 3 neurons in the hidden layer, and 1 output neuron.

where σ is the sigmoid activation function and $w_{i,j}$ describes the weight of the connection between neurons i and j. Even though any feed-forward neural network can be expressed as a single expression, usually the intermediary results from individual blocks are reused to avoid unnecessary computation — while only the inputs $x_0, \ldots, x_3$ occur repeatedly in the example in Fig. 2.1 if the network had more layers or more outputs, there would be more elements in the formula that occur several times. These repeated elements are one of the main reasons why neural network are usually organized into separate layers, as it allows for simple and efficient evaluation of neural networks using matrix-vector operations [122].

2.2.3 *Layers*

Neural networks are hierarchical models where the neurons are separated into individual interconnected layers [122]. The neurons within a single layer are usually considered to be of the same type with the same activation function — a layer is a basic building block (this is the traditional terminology, however, as there might be blocks that are connected in parallel, the term *layer* might be slightly confusing as it usually represents only a layer within a single block and not the set of all neurons that are in certain depth). This section briefly overviews the most common layers utilized in neural networks.

2.2.3.1 *Fully connected layer*

Fully connected layers (FC) (also called *dense layers*) are the most basic type of layers and historically the first used. Each neuron in a fully connected layer is connected to every neuron in the preceding layer — hence the name. Fully connected layers are used in the example in Fig. 2.1. Most of the early NNs used this kind of layer. The main disadvantage of fully connected layers is that they have a vast number of parameters even for layers with relatively few neurons; this makes them unsuitable for many problems, such as the image classification where a neuron is needed for each pixel (this issue is addressed by weight replication in convolutional layers, see Section 2.2.3.3).

Additionally, training such networks can be challenging due to their susceptibility to overfitting. Various techniques have been proposed to address this issue, such as dropout [143], L1 and L2 regularization, or soft weight sharing [144]. In modern image pattern recognition, dense layers are often employed only as top layers, utilizing features extracted by other types of layers, e.g., [13, 51, 52]. However, fully connected layers are still very commonly used in other domains; for example, the D–GEX for gene expression inference uses only fully connected layers [2].

2.2.3.2 *Dropout layer*

As mentioned earlier in Section 2.2.3.1, dropout is a technique designed to combat overfitting [145]. It efficiently approximates the simultaneous training of numerous neural network architectures [143, 146]. This method randomly turns off neurons during the training phase, including all incoming and outgoing connections. Each neuron has a probability p of remaining active; otherwise, it is dropped out during training. During the testing phase of the original dropout from [143], all neurons are active; however, that requires scaling down the outputs from the neurons to average the outputs and reach similar values as in the training phase [143]. In most applications, however, *inverted dropout* is used, which scales the outputs during training to avoid the need for scaling during testing [122].

Regularization is a technique for addressing overfitting by enforcing a preference for certain weight types over others [46, 147, 148]; dropout is a form of regularization of the optimization [143, 149–152] as it tries to force neurons to be robust and rely on population behavior [153]. Interestingly, it was also shown that dropout reduces underfitting as it helps to reduce the directional variance of gradients across mini-batches and, therefore, reduces the influence of a single batch [154]. Dropout can also be seen as a form of data augmentation [155] especially in the case of *cutout* [156] that drops only input units and only in contiguous sections, which effectively modifies the input data. Another view of dropout is to consider it as an approximate Bayesian inference in deep Gaussian processes; see [157] for details.

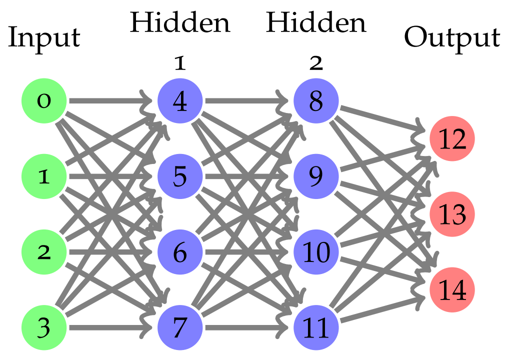
A similar concept is *DropConnect*, which drops individual connections instead of entire neurons. *DropConnect* has been found to achieve slightly better results than classical dropout [158], and its modified version, Sparse DropConnect [159], has demonstrated even better performance. Visual demonstration of these approaches is shown in Fig. 2.2. There are other dropout variants such as dropout modifications for RNNs [149] which takes into account the time frames of RNNs instead of just randomly dropping neurons; *max-drop* [160] aimed for CNNs that selectively drops activations with the maximum value within each feature map, *DropFilterR* [161] also for CNNs that randomly drops elements in convolution filters; *Drop-Activation* [162] which randomly drops activation functions by replacing them with identity function; *cutout* [156] that drops only input units and only in contiguous sections; *stochastic depth* [163] which randomly drops subsets of layers; *stochastic residual network*

[164] that randomly remove skip connections; *standout* [165] that overlies a binary belief network on a NN to selectivelly setting the NN's activations to zero; *jumpout* [166] adapting the dropout rate; *DropAll* [167] combining dropout and DropConnect; *curriculum dropout* [168] with adaptive scheduled dropout rates; *late dropout* [154] that is active only in later epochs; *DropMaps* [169] applying dropout on feature maps; *LayerOut* [170] which freezes random layers; *DropIn* [171] that instead of setting activations to zero uses activations from previous layer; *fast dropout* [172] improving the computation speed of standard dropout; *annealed dropout* [173] with decreasing dropout rate; *variational dropout* [174] with learned dropout rates; *rnnDrop* [175] for RNNs; another variants of dropout for RNNs and long short-term memorys (LSTMs) in particular in [176–178], *max-pooling dropout* [179]; *SpatialDropout* [180, 181] for CNNs; *evolutional dropout* [182] computing sampling probabilities from mini-batches; *swapout* [183] which is a generalization of standard dropout and stochastic depth; dropout for network compression and acceleration through sparsity [184–187]; *concrete dropout* [188] with automatic tuning of dropout probabilities; dropout for Bayesian NNs [189]; adversarial dropouts [190, 191] and its RNN variant [192]; *fraternal dropout* [193] that uses a pair of identical RNNs with different dropout masks; *information dropout* [194] automatically adapting to the data using information theory; *spectral dropout* [152] using a decorrelation transform with fixed basis functions; hardware-oriented dropout for field-programmable gate arrays (FPGAs) [195]; *ranked dropout* [196] which masks active neurons; *surrogate dropout* [197] that drops neurons based on their importance; *tabu dropouts* [198, 199] that give temporary protection against dropping to a neuron that has been just dropped; *supervision dropout* [200] using genetic algorithms for selection of dropped neurons; *iDropout* [201] that gives higher probability of dropping to neurons with lower relevance; *Wasserstein dropout* [202] for uncertainty estimation; using dropout with diversity sampling for uncertainty estimation [203]; *adaptive infinite dropout* [204] for streams; *Icing-dropout* [205, 206] that adaptively selects neurons to be dropped; and many others — see reviews [150, 207, 208] for more dropout and regularization variants.

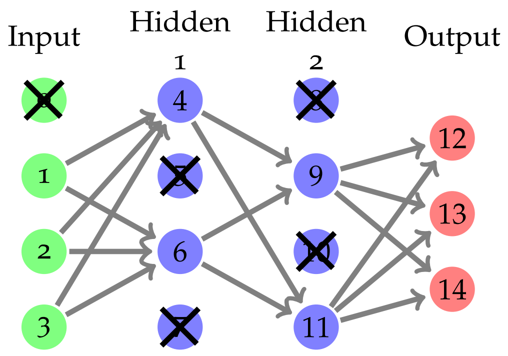
While dropout is a regularization technique, it is called *dropout layer* throughout this work as it is often implemented using a special layer masking individual neurons, which is the case of the used frameworks Keras [209] and Tensorflow [210].

2.2.3.3 *Convolutional layer*

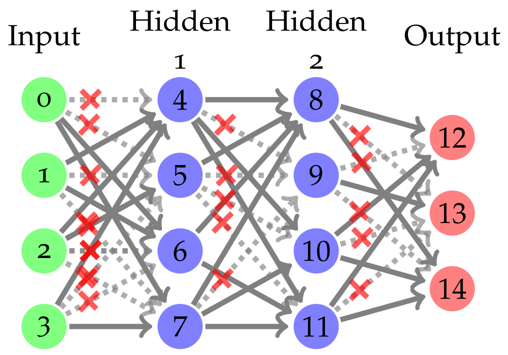
While not used in this work, convolutional layers are one of the most used layers nowadays as they are crucial for image pattern recognition as they allow the extraction of spatial features in images without incurring significant computational costs. These layers are based on the principles of parameter sharing, sparse interactions, and equivariant representations [46]. Convolutional layers are widely employed in applications where the arrangement of inputs encodes spatial or temporal information, such as images, videos, audio, and time series. In this section, we focus on using 2D convolution



(a) without dropout



(b) with dropout



(c) with DropConnect

Figure 2.2: Visual comparison of dropout and DropConnect.

for image processing as the 2D convolution is easy to visualize; 1D and 3D convolution layers work similarly.

Convolutional layers apply filters to input data, which introduces an additional dimension to the data. Multiple convolutions are usually employed, each resulting in a different feature map as shown in Fig. 2.3. The Fig. 2.3 shows a convolution layer with 9 filters being applied to an input image. Note that the Fig. 2.3 also shows *pooling* layer (see Section 2.2.3.4) and also another convolutional layer, this time with 36 filters.

The convolutional layer's parameters include the kernel's width w and height h, the number of filters d, padding p, and stride s. These parameters determine the size of the output volume. The width w and height h define the *receptive field* [122] of the layer, whereas the parameter d determines the number of filters that will be applied to each input and thus, it represents the depth of the output volume, a single feature map at a certain depth is called *depth slice*. The kernel dimensions determine the size of the matrix that defines the kernel, with square kernels being common in image pattern recognition. However, some architectures, such as Inception v3 [52], deviate from the norm by using non-square kernels, such as 3×1 and 1×3 convolutions in two layers instead of a single 3×3 convolution.

The parameter p controls the padding of the input volume with zeros, an aspect that affects the dimensions of the output. The absence of padding is often termed *valid* padding (e.g., [46, 209, 211]), which leads to output dimensions that are smaller than the input's due to the convolution operation; such convolution is sometimes called the *narrow* convolution (e.g., [212]). In contrast, padding is typically employed to maintain the output spatial dimensions as those of the input. Common terms for this padding strategy are *same* (e.g., [46, 209, 211, 213]) or *half* padding (e.g., [213]). Another padding variant is called *full* [211], where the filters are applied to the input end-to-end — every input pixel is used by every part of the filter.

The stride s parameter controls the frequency at which the filter is applied to input pixels. For example, $s = 1$ applies the filter to every pixel, while $s = 3$ implies application to every third pixel. Convolution with $s > 1$ is called *strided convolution* [123]. The stride also determines spatial downsampling in the convolution process. It essentially combines regular convolution with $s = 1$, followed by subsequent downsampling. However, this approach is less computationally efficient compared to convolution with a stride [46, 213, 214]. An alternative to strided convolution is the *space-to-depth* convolution [215].

For example, if the layer had stride $s = 1$ and $h = w$, then the *same* padding keeping the spatial dimensions would be $p = \frac{w-1}{2}$. An extreme case is the *full* padding, which pads the input with enough zeros such that every input pixel is visited equally — the border pixels in the case of the *same* padding are underrepresented in the model [46, 213].

Weight sharing is a critical feature of convolutional layers, as the same filter is applied to different pixels — this is in contrast to the *fully connected layer* (see Section 2.2.3.1) where each input has its own weight. This results in a significant reduction in the number of parameters when the kernel size is considerably smaller than the spatial dimensions of the input. Within the convolutional layer, a single set of weights is assigned to each depth slice, meaning that all neurons within that slice share the same weights [122].

The output volume of a convolutional layer $H_o \times W_o \times D_o$ is determined by the parameters in following way [122]:

$$H_o = \frac{H_i - h + 2p}{s+1} \tag{2.8}$$

$$W_o = \frac{W_i - w + 2p}{s+1} \tag{2.9}$$

$$D_o = d \tag{2.10}$$

where h, w, p and s are defined above and H_i and W_i is the spatial dimension of the input volume.

The concept of *sparse interactions* or *sparse connectivity* plays a crucial role in convolutional layers. This idea is grounded in the size of the convolutional layer's kernel, such as when the kernel size is denoted as $n \times n$. In this scenario, each neuron within the layer possesses a *receptive field* of $n \times n$, meaning it is not linked to every neuron in the preceding layer but only to $n^2 \times d_i$ neurons, where d_i represents the depth of the input. Sparse connectivity contributes to a significant reduction in the number of parameters for each neuron. For instance, if the input volume measures $224 \times 224 \times 3$ and the filter size is 5×5, each neuron within the convolutional layer will have $5 \times 5 \times 3 + 1 = 76$ parameters. In contrast, a neuron in a dense layer would have $224 \times 224 \times 3 + 1 = 150,529$ parameters.

Another important concept for convolutional layers is the *equivariance to a translation* [46] — the property that shifting the output of a convolutional layer is essentially the same as shifting the input first and then applying the convolution operation, with minor exceptions at the border regions. This property is particularly desirable when our objective is to identify features or patterns that may appear at various positions within the input data, a common scenario in typical image applications (though not always, as certain images may be perfectly centered) [46]. However, in cases where *equivariance* to translation is not the desired characteristic, the convolutional layer can be substituted with a *locally-connected layer*. This alternative layer functions like a convolutional layer but doesn't adhere to strict weight sharing [122, 211]. The *tiled convolution* [216] is in between the standard convolutional layer and the locally-connected layer; instead of having all weight shared as in the standard convolutional layers, it uses a tiled pattern of tied weights, therefore immediately neighboring cells have different filters but cells k step aways will share weights [216].

Besides the *strided convolution* and *tiled convolution* described above, there are also other convolution variants. One such variant is *dilated convolution* (also called *atrous convolution* [217]), which increases the filter's receptive field size by inserting zeros between filter elements [123]. According to [217], it first appeared in [218]. It is often applied to problems that require longer sequence information dependencies [219]. The *dilated convolution* gave rise to popular *dilated residual networks* [220]. Dilated networks were used for, e.g., image sharpening and denoising [221–225], semantic segmentation [226], learning optical flows [227], speech separation [228], image classification [229–231], and speech emotion recognition [232].

Another variant is the *transposed convolution* [123, 213] (also called *deconvolution* [123, 233, 234], *fractionally strided convolution* [123, 235], and *convolutional transpose* [236]) proposed in [233, 234]. Whereas standard convolution connects a single output activation to multiple input activations, transposed convolution has multiple output activations for single input activation by upsampling the input by a factor of the stride value with padding [123].

Examples of usage of transposed convolution include super-resolution [237], semantic segmentation [238], feature visualization [239], deblurring [240], and image generation [236].

2.2.3.4 *Pooling layer*

Even though the *pooling layer* is also not used in this work, it is one of the most commonly used layers in ANNs. A pooling layer plays a crucial role in decreasing the spatial dimensions of the input data and is frequently employed alongside convolutional layers [46, 122]. Pooling layers downsample the feature maps by aggregating features from local regions [241]. The fundamental concept behind pooling shares some common ground with convolutional layers. However, in contrast to convolution, which involves the convolution of neighboring pixels with a kernel, pooling employs a specific pooling function to process nearby pixels.

Pooling increases the size of the receptive field of convolutional kernels over layers, but it also reduces computational complexity and memory requirements by reducing the resolution of the feature maps [241]. All the while, it retains important features essential for processing by subsequent layers [241]. Furthermore, as it reduces the number of parameters, it can also serve to reduce overfitting. Not only that, it also introduces invariance to small translations of the input [46].

Commonly used pooling layers share similar parametric attributes with convolutional layers, including width w, height h, stride s, and the choice of pooling function f. Width and height determine the neighborhood size

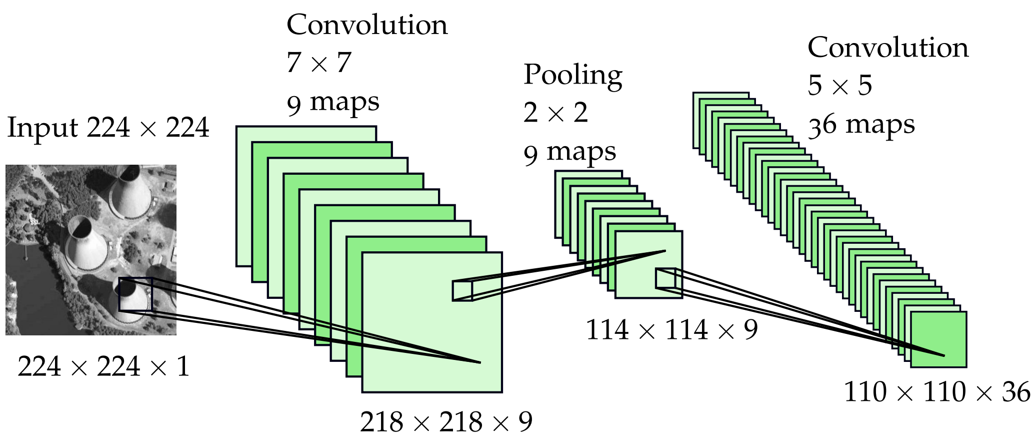


Figure 2.3: Example of usage of convolutional and pooling layers. Note that this example has no padding, and thus, the convolutions also reduce dimension. More about pooling layers in Section 2.2.3.4.

considered for pooling, while stride regulates how frequently pooling is applied. The pooling is usually applied to non-overlapping blocks; therefore, the stride is used more often than in convolutional layers, e.g., a ubiquitous pooling layer has $w = 2$ $h = 2$ (filter 2×2) together with $s = 2$ and max pooling function [122] which results in downsampling and getting rid of 75 % outputs [122] as every max operation takes a maximum over four numbers (patches $w \times h = 2 \times 2$ in a depth slice). Larger pooling receptive fields are usually too destructive [122].

The output volume of pooling layer $H_o \times W_o \times D_o$ is determined by the parameters in a following way [122]:

$$H_o = \frac{H_i - h}{s + 1} \tag{2.11}$$

$$W_o = \frac{W_i - w}{s + 1} \tag{2.12}$$

$$D_o = D_i \tag{2.13}$$

where h, w, and s are defined above and H_i, W_i, D_i are the dimensions of the input volume.

While pooling layers are widely used, they are not strictly necessary and can be replaced efficiently with convolutional layers using appropriate stride values [122, 242]. This substitution can be advantageous in cases where pooling may not align with specific architectural or computational requirements. For example, certain visualization tasks rely on the use of *switches* from max pooling layers during the forward pass — this is actually also common during the backward pass of the backpropagation. Substituting *max pooling layers* with convolutional layers allows for unconditional visualization, not reliant on the forward pass [242]. In fact, an architecture referred to as an *all convolutional network* [242] that excludes pooling layers has shown strong performance in image classification on datasets like CIFAR-10 [243], CIFAR-100 [243], and ImageNet [48].

There are many variants of pooling layers. There are two leading groups regarding the type of pooling used — *local pooling* and *global pooling* [241]. The *local pooling* is described above; it performs pooling from small local regions determined by the width and height. The *global pooling* is done over the whole feature maps to get a single value for each of the features [241].

Historically, the two most common pooling layers are the max pooling layer described above and the average pooling layer that produces the mean value over the pooled region [244]. These two pooling approaches are used due to their simplicity in many CNNs [241]. The max pooling layer selects the highest value from the pooled region, and therefore, it does not degrade found features [245]; however, the max operation complicates the backward pass in backpropagation for optimization or visualization purposes [242]. The *average pooling* is also simple; however, it might reduce feature contrast if there are small values in the considered region [245].

There are also many other pooling methods such as *learned-norm pooling* [246], *fractional max pooling* [247–249], *rank-based pooling* [250], *gated max average pooling* [251], *mixed max average pooling* [252, 253], *dynamic correlation pooling*

[254], *Log-Sum-Exp pooling* [255], *dynamic pooling* [256], *smooth-maximum pooling* [257], *soft pooling* [258], *polynomial pooling* [259], *maxfun pooling* [260], *ordinal pooling* [261], *regularized pooling* [262], *Root-Mean-Square pooling* (termed *sqrt* in the original paper) [263], *global feature guided local pooling* [264], *stochastic pooling* [265], *stochastic spatial sampling pooling* (*S3 pooling*) [266], *spatial pyramid pooling* [267], *concentric circle pooling* [268], *polycentric circle pooling* [269], *multi-pooling* [270], *second-order pooling* [271], improved *bilinear pooling* [272, 273], *detail-preserving pooling* [274], *local importance based pooling* [275], *generalized max pooling* [276], *transformation invariant pooling* [277], *kernelized subspace pooling* [278], *region pooling learning* [279], and *random crop pooling* [280]. More about these and many other pooling approaches is available in reviews [241, 245, 281, 282].

2.2.4 *Optimization*

Optimizing a neural network typically involves three key steps: *forward propagation*, *loss optimization*, and *error backpropagation* with parameter update [46, 122],. During forward propagation, the neural network computes its output for a given input. Loss optimization measures how well the network's output matches the ground truth as defined by a loss function, and this loss function is what the training process aims to minimize. Finally, error backpropagation with parameter update is where gradients are computed using the chain rule, and the network's parameters are updated using gradient-descent-based approaches [46, 122, 283] as analytical solutions for parameter estimation are often unattainable [284]. There are also other methods that do not involve gradient-descent-based approaches; however, those are out of the scope of this work as only gradient-descent-based approaches were used throughout this work.

2.2.4.1 *Loss function*

The term *loss function*, often referred to as an *objective function*, is a mathematical construct that associates an event or one or more variables with a real number, which intuitively represents a quantification of the "cost" or "error" associated with that event [284]. In optimization problems, the objective is to minimize this loss function [284].

In a neural network, loss functions assess the quality of a parameter assignment after the forward pass [122]. In the forward pass, the neural network generates scores for input data, and the loss function quantifies how closely these scores align with the ground truth [122, 132]. Essentially, it quantifies the quality of the network's predictions by computing a numerical score that reflects the degree of dissimilarity between the observed and predicted values [284]. This process involves assessing the error between true and predicted values and aggregating these errors across the entire dataset to produce a singular metric that assesses the network's performance relative to the desired outcome [284].

Common loss functions include cross-entropy, mean squared error (MSE) (or the unaveraged sum of squared errors (SSE)), hinge loss (and its squared and cubed variants [285]), and others [209, 284, 285].

2.2.4.2 *Backpropagation*

Backpropagation (BP) is a method for computing gradients based on iterative use of the *chain rule of differentiation* [46]. While it is often attributed to Rumelhart, Hinton, and Williams [38], several research teams published the algorithm independently around the same time [284].

For $x \in \mathbb{R}$, $f(x) : \mathbb{R} \to \mathbb{R}$, $g(x) : \mathbb{R} \to \mathbb{R}$, $y = g(x)$, and $z = f(y) = f(g(x))$, then the chain rule is [46]:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.14}$$

In vector notation, let $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^n$, $z \in \mathbb{R}$, $g(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}^n$, $f(\boldsymbol{y}) : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{y} = g(\boldsymbol{x})$, and $z = f(\boldsymbol{y})$, then the chain rule is [46] is:

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i} \tag{2.15}$$

Since the backpropagation in NNs is usually done with tensors, the tensor notation might be more relevant — the tensor notation of the chain rule is [46]:

$$\boldsymbol{\nabla}_{X} z = \sum_{j} (\boldsymbol{\nabla}_{X} Y_j) \frac{\partial z}{\partial Y_j} \tag{2.16}$$

where $\boldsymbol{X}$, $\boldsymbol{Y}$ are tensors, $\boldsymbol{Y} = g(\boldsymbol{X})$ and $z = f(\boldsymbol{Y})$ and j is a tuple of indices [46, p. 203].

For a comprehensive and numeric illustration of the backpropagation process, a detailed example can be found in the work by López, López, and Crossa, which offers a practical and specific insight into this fundamental neural network training technique [284].

Backpropagation, a fundamental process in neural networks, systematically applies the chain rule in an iterative manner [46]. This approach stems from the conceptualization of neural networks as complex compound functions. The practical implementation of backpropagation typically involves the use of a computational graph, which serves as a descriptive framework for this compound function. Within this graph, each node corresponds to an operation that applies a function to a set of arguments, which are derived from the values of preceding nodes [46]. For a more comprehensive and detailed exploration of backpropagation and computational graphs, one can refer to the in-depth discussions available in resources like [46, 122].

2.2.4.3 *Gradient descent*

Gradient descent stands as the prevailing approach for optimizing deep neural networks [122, 132, 284]. This algorithm's primary objective is to navigate the landscape of the loss function, systematically moving towards local optima. The fundamental vanilla version of gradient descent computes the gradient at the current point and follows a trajectory of fixed length, known as the *learning rate*, in the direction of the steepest descent.

However, in the realm of neural networks, the traditional gradient descent is frequently supplanted by alternative variants as neural networks often have a large number of parameters, and they are trained using large datasets and calculating the gradient with respect to all training samples might be infeasible. These include *online gradient descent* [122], which factors in a single example per iteration, or *minibatch gradient descent*, where small batches of examples are considered for each weight update. These variants are collectively referred to as stochastic gradient descent (SGD) even though SGD is technically only a different name for the *online gradient descent* [122].

Moreover, enhancements to the gradient descent approach have been introduced to address specific challenges. One such enhancement is *momentum*, a technique designed to surmount local optima and expedite convergence, particularly when navigating prolonged and narrow valleys in the loss function landscape [122, 286]. Momentum methods incorporate the history of descent, whereby the negative gradient influences the particle's velocity in relation to its momentum. Subsequently, the new position is determined by taking into account the current position and the velocity. A related concept, known as *Nesterov momentum*, functions similarly to classical momentum but computes the gradient after the momentum update.

The optimization of the learning rate, a critical component in gradient descent, has also been a subject of exploration. Single formulas for dynamically setting the learning rate have been proposed, including *step decay*, *exponential decay*, or $\frac{1}{t}$ *decay*. Alternatively, tuning the learning rate on an individual parameter basis, informed by the training history, has led to methods such as *adagrad* [287], *Adadelta* [288], *RMSprop* [289], *ESGD* [290], *Adam* and *AdaMax*[291], and *Nadam* (Adam with Nesterov momentum)[292]. In-depth comparison of SGD and Adam is available in [293]; analysis of the Adam in [294–299]. The SGD and its variants are first-order optimization methods as they use a first derivative; higher-order methods such as [300–303] might converge at a faster speed as they use the information about curvature; however, these methods are more difficult to utilize compared to first-order methods in NNs — usually due to the operation and storage of the inverse of the Hessian matrix [286, 304, 305].

More recent improvements of these techniques include, for example, a hybrid combination of Adam and AMSGrad called *HN Adam* [306], SGD with warm restarts and regularization called SGDRE [307], SGD with fractional-order momentum [308], projection Adam *AdamP* [309], Adam with quasi-hyperbolic momentum *QHAdam* [310], Adam with partially adaptive momentum *PAdam* [311–313], *adaptive inertia* optimizers [293], Adam with adaptive variance reduction *Adam*$^{+}$ [314], Adam with adaptive bilevel optimization *BiAdam* [315], *AdaGDA* [316], *AMSGrad* [297], *AdaBound* and *AMSBound* with learning rate clipping [317, 318], Adam with second-order momentum *AdaXod* [319], *AdaMod* [320], *WSAGrad* [321], *AdaLip* estimating the Lipschitz constant [322], *AdaBelief* [323], and *AdaCB* [324] limiting the learning rates, *super-adam*

[325] providing a generalizing framework, Adam with the Kalman filter *KAdam* [326] ans its extension *sKAdam* [327], *iAdam* [328], *diffMoment* [329], *Nadax* [330], *AdaDrift* [331], *AdaSecant* [332, 333], *AdaHessian* [334], *NRMSProp* [335], and SGD with random learning rate *mSGD* [336]; these are only few examples demonstrating how vast is the field of the optimizers, see reviews mentioned below.[3] Nevertheless, the vanilla Adam remains popular as it converges well even in the vanilla form when tuning the Adam hyperparameters [299].

More about optimization methods for NNs is available in reviews and comparative studies [286, 304, 305, 337–345].

3 Also, an extensive list of optimizers is available in the supplementary material of [337] at `https://proceedings.mlr.press/v139/schmidt21a/schmidt21a-supp.pdf`.

3 DNA MICROARRAYS AND GENE EXPRESSION MEASUREMENT — ANOTHER BRIEF OVERVIEW

This chapter serves as a brief overview of DNA microarrays, an important technology in genomics [346]. Microarrays entail the immobilization of thousands of nucleic acids on a surface and are employed to gauge the relative concentrations of nucleic acid sequences within a mixture to get an estimation of gene expression levels [347] of thousands of genes simultaneously [348]. This quantification is achieved through the process of hybridization, followed by the detection of the resultant hybridization events [347].

The last two decades in genomics were characterized by the massive use of oligonucleotide and DNA microarrays, which allows for obtaining genome-wide mRNA expression data [349]. While the very predecessor of DNA microarrays was used already in 1975 [347, 349], the DNA microarrays are still actively used for research even today (e.g., [350–357]). Other contemporary uses include DNA fingerprinting for forensic uses [358] and clinical applications such as [359].

The Section 3.2 briefly describes the historical development of DNA microarrays; a brief overview of the function and types of DNA microarrays is then provided in Section 3.3 together with a comparison to another popular approach for measuring gene expression — RNA-Seq — in Section 3.3.2. Finally, an L1000 microarray platform is introduced in Section 3.3.3 due to its importance for the motivation of this work.

3.1 DNA AND GENETICS

Genes are the basic hereditary units through which living organisms inherit characteristics and attributes from their progenitors [360]. For instance, children often exhibit physical resemblances to their parents due to the inheritance of their parents' genes. Genetics delves into the intricate study of genes, aiming to explain their composition and functionality.

Genes are stored in an extended molecular structure termed DNA, which undergoes replication and is passed down through successive generations [360, 361]. DNA comprises basic building blocks arranged in a specific sequence, holding genetic information. This genetic code, inherent to DNA, provides the language that enables organisms to interpret the data contained within genes. This information serves as the blueprint for the construction and operation of a living organism [360, 361].

The genetic information encoded in DNA is stored as a code consisting of four distinct chemical bases — adenine (A), guanine (G), cytosine (C), and thymine (T) [361, 362]. The arrangement, or sequence, of these bases dictates the data required for the construction and maintenance of an organism, similarly to the manner in which the alphabet's letters are ordered to form words and sentences [361].

DNA consists of two strands forming a double helix; the two strands of DNA are polynucleotides, and they are constructed from simpler monomeric units known as nucleotides [361]. Each nucleotide comprises one of four nitrogen-containing bases (A, G, C, and T), a sugar molecule deoxyribose, and a phosphate group [361]. A chain is formed by the linkage of deoxyribose and a phosphate group, serving as the scaffold upon which nucleotides are attached as shown in Fig. 3.1. Each DNA molecule is composed of two complementary strands, and the orientation of the deoxyribose within each strand dictates their direction. These strands are identified as the 3′ end and 5′ end based on the bonds of the deoxyribose [361]. The complementary strands align in reverse directions and are linked by peptide bonds connecting the nucleotides. The nucleotides within these strands are not connected at random; usually, adenine pairs with thymine, while cytosine forms a peptide bond with guanine [361]; nevertheless, there are exceptions and other pairs are formed, e.g. [363].

However, the DNA does not directly partake in protein synthesis. Instead, an intermediary molecule, ribonucleic acid (RNA), serves as the courier, shuttling the information encoded in DNA to ribosomes [361, 364]. At the ribosomes, proteins are manufactured in alignment with the nucleotide sequence. These nucleotides are read in triplets, termed codons, where each of the 64 potential triplets codes for one of the 21 amino acids, a start codon, or a stop codon [361]. This intricate process comprises two primary phases: *transcription*, which is the initial step where a gene acts as a template for RNA synthesis, and *translation*, the subsequent step involving ribosomal protein synthesis [361]. This process of the flow of genetic information called GE is often summarized in the so-called *central dogma of molecular biology* [365–369].

3.2 BRIEF HISTORY OF MICROARRAYS

The first description of the DNA in 1953 by Watson and Crick [370] led to the emergency of genomics. The first steps towards the creation of the DNA microarrays were presented in the late 60s when the way for locating the position of specific sequences (*in situ* hybridization) was discovered [371]. The method became known as fluorescence in situ hybridization (FISH) after the introduction of fluorescent probes [371]. The FISH uses fluorescent probes that bind only sequences with a high degree of complementarity, which can be observed using fluorescence microscopy.

3.2.1 *First arrays*

The development of the colony hybridization method led to the creation of the first microarrays; the colony hybridization method randomly cloned the probe DNA into *E. coli* plasmids, and then grown the colonies, thus replicating the probe DNA. This was used for the first larger scale experiments where it was used to screen thousands of colonies to identify clones with the DNA

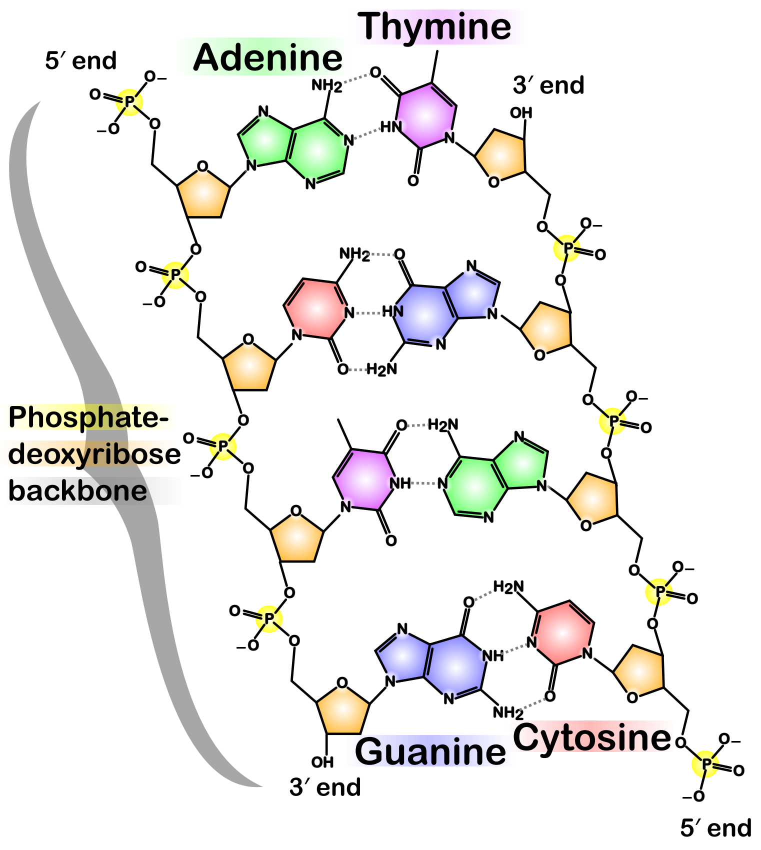


Figure 3.1: Structure of a DNA molecule. By Madprime [CC0], via Wikimedia Commons, https://commons.wikimedia.org/wiki/File:DNA_chemical_structure.svg.

complement to the probe DNA [347]. The approach was later extended in 1979 by creating an array of 1728 colonies in a 26×38 cm region [347]. The extension was done by the creation of mechanical pin devices that allowed it to operate simultaneously on 144 well microplates. The first and simplest arrays were called the *dot blots* with simplified processing and better reproducibility — they allowed for parallel hybridization and also for parallel image processing [371]. The density of the first array was further increased by replacing the manual work with robotic systems, which also removed the human errors that inevitably occurred [371].

3.2.2 *Increasing the density*

The *dot blot* procedure used a porous support as it provided a larger surface for binding, and also, the nucleic acid could be applied in relatively large volumes because it soaked into the porous material, thus preventing excessive lateral spreading [371]. The porous support came, however, with several disadvantages — the boundaries and shapes of spots were poorly defined, and it was difficult to control the amount of oligonucleotide deposited. The porous support was an obstacle to increasing the density of spots, which was necessary to increase the number of spots. Furthermore, the permeable membranes tended to swell in solvent and to shrink and distort when dried, and also the non-rigidness made spotting and reading their position more difficult [371, 372].

The solution was to replace the porous support with an impermeable material such as glass or silicon, which allowed the use of very small sample volumes and high density of spots [372]. Furthermore, since the nucleic acids form a monolayer that saturates the surfaces, the impermeable support allowed for the consistency of attached amounts between regions of the array [371]. Also, such supports increased the reaction speeds of the solution phase as the molecules did not have to diffuse into and out of the spores [371]. The impermeable supports allowed the technology to reach the high accuracy, reliability, and reproducibility needed for larger-scale experiments.

3.2.3 *Mature microarrays*

After the introduction of impermeable supports, three major directions of microarrays emerged — spotted arrays, in-situ synthesized arrays, and self-assembled arrays [347].

3.2.3.1 *Spotted arrays*

The first method allowing high-density arrays on glass substrates was published in 1996; the process used poly-lysine coated glass microscope slides that provided good binding of DNA and also a robotic spotter that was spotting multiple glass slide arrays from DNA stored in microtiter dishes [347].

3.2.3.2 *In-situ synthesised arrays*

Another direction represented the *in-situ* synthesis of nucleic acid on the surfaces, which brought multiple advantages over the deposition of presynthesized probes [371]. One of the possible approaches was an ink-jet fabrication that used ink-jet printers to fire a solution of nucleotide reagents at a glass surface. Due to the similarity of ink-jet printing on paper, most of the engineering work was already researched, which allowed a quick introduction of the method as it required only the modification from printing four colors to delivering precursors for four different bases [371]. This increased the flexibility of DNA microarrays as it could synthesize any set of oligonucleotides and place it at any position in the array [371].

A different approach is the *light-directed* fabrication (manufactured by Affymetrix) that directed the synthesis of oligonucleotides by using patterned photolithographic masks [347, 371]. A single mask was required for each base addition; thus, to create a probe of 20 nucleotides, 80 masks were required [371]. The biggest advantage of such an approach is the high density — there are DNA microarrays that have 65,536 probes in 1.28×1.28 cm area [371].

3.2.3.3 *Self assembled arrays*

Yet another method was introduced in 1998; it synthesizes DNA on small polystyrene beads and then deposits the beads on the end of a fiber optic array[371]. A randomly assembled array is then created by applying a mixture of such beads to the optic fiber. The first versions of the technology used optically encoded beads with different fluorophore combinations to determine the position of the nucleotides. However, this rather limited the number of unique beads that could be distinguished [371]. This was solved in 2004 by decoding the beads using hybridization and detection of several short and fluorescently labeled oligonucleotides, which allowed for a large number of types of beads on a single array and for functionality tests of such array prior to its use in a biological assay [371].

3.2.4 *Future of microarrays*

The DNA microarrays characterized a whole phase in the genomics research. Even though they are still actively used in various research, a new competitor has arisen recently — it is called the RNA-Seq [373]. It has one great advantage compared to DNA microarrays as the microarrays need to know the analyzed sequences *a priori* to the experiment [373]. The RNA-Seq represents a whole transcriptome shotgun sequencing and, as such, produces reads of sequences in the analyzed sample without the need to know the sequences before the experiment — the measurement of the gene expression is done by counting the reads during the transcriptome assembly. For more details, see Section 3.3.2. Coppée predicted in 2008 that microarrays would have been made obsolete by the sequencing platforms as over time more and more applications would have migrated from microarrays to the sequencing approaches [374]; while this prediction is partly true as, for example, the RNA-Seq is more suitable for certain applications, the microarrays are still very popular with new designs appearing in contemporary literature (e.g., [359, 375]) — furthermore, Aparna and Tetala stated in 2023 they expect that biomolecule-based microarrays will flourish over the next five years [376]. One of the reasons why microarrays might flourish in the near future might be *nanoarrays*, which are part of next-generation sequencing (NGS) approaches [377].

Furthermore, the microarrays are often preferred in certain applications as they are often cheaper [378] (depending on the goals and scope of an experiment). Also, there are microarray-based tests with proven clinical utility, and they are easier to use for diagnosis. The economic and technical aspects allow the use of microarrays outside the basic research directly in clinical practice [379, p. 17] such as [359].

3.3 DNA MICROARRAYS AND MEASURING GENE EXPRESSION

DNA microarrays, sometimes also called nucleic acid arrays [347], are used mainly for measurement of gene expression levels [347, 364] (there are also other popular approaches such as RNA-Seq, see Section 3.3.2). They are used to assess, for example, DNA mutations, DNA methylation, single nucleotide polymorphism, chromosomal fragments, microRNAs, and long noncoding RNAs [379, p. 17]. A typical microarray is constructed by immobilizing oligonucleotides [380], each comprising several dozen nucleotides, onto a glass slide [381], a specialized cassette [379, p. 17] or other materials [382]. Employing photolithographic techniques [383] (there are also other approaches, see [376]), one nucleotide (A, G, C, and T) is added at a time, enabling the creation of a microarray containing hundreds of thousands of distinct oligonucleotide sequences [381]. These sequences are designed to be complementary to characteristic fragments of known DNA or RNA sequences [384] and are organized into sets referred to as probes [381, 385].

When a sample containing DNA or RNA molecules is applied to the microarray's surface, these components specifically hybridize with their corresponding probes, which are present in multiple copies throughout the microarray [381] as shown in Fig. 3.2. A fluorescence-based method is then employed to determine the amount of material hybridized to a particular probe [381]. Although the relationship between fluorescence intensity and the quantity of DNA or RNA is not linear, the fluorescence intensity serves as an indicator of the amount of a specific gene's DNA or RNA in the sample [381]. This methodology facilitates the quantification of transcript levels for a multitude of genes within a relatively short timeframe [381].

3.3.1 *Microrrary experiments*

The whole microarray experiment consists of several steps. The first step is the RNA isolation, where the RNA is isolated from the cells. During this step, the degradation of the sample is also measured; a high-quality RNA sample should contain over 80 % of ribosomal RNA (rRNA) as its concentration is

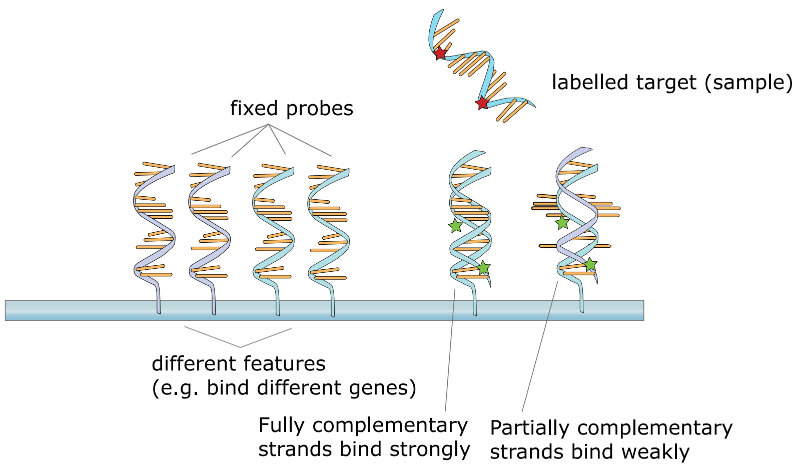


Figure 3.2: Working of DNA microarrays by hybridization of the target to the probe. By Squidonius [public domain], via Wikimedia Commons, https://commons.wikimedia.org/wiki/File:NA_hybrid.svg.

a good indicator of the overall RNA quality [381]. Nevertheless, the quality of RNA can also be measured after the completion of the experiment by evaluating results from a control probe-sets [381], often with the help of RNA degradation plots [386] or mixed effect modeling [387].

The second step is the synthesis of copy DNA (cDNA) [381]; the RNA is reverse transcripted into the first strand of cDNA using either oligo-dT primer such as or random primers [381]. As oligo-dT primer works only on mRNA and not on rRNA and therefore no cleansing of rRNA is needed [381]. The second strand of cDNA is synthesized using the first strand as a template.

The third step is the amplification and labeling [381]. The synthesized cDNA is amplified (replicated); however, this step is crucial for the GE experiment quality [381] and can introduce systematic errors [388] — which is one of the reasons why amplification is optional and why there are microarray protocols that omit it [381, 389]. The amplification process uses *in vitro* transcription to produce copy RNA (cRNA) [381, 390]. To enable control of the overall reaction yield and the sample's purity, this step also includes cleanup and quantification of the cRNA [381].

The fourth step is fragmentation when the cRNA targets are cut into fragments 50–100 nucleotide (nt) long [381]. Before the next step, bacterial RNAs are added to enable evaluation of the consistency of hybridization condition and the overall performance [381]. These bacterial RNAs are called *bacterial spikes* [381] and are one of the type of external RNA controls (ERCs) [381, 391].

The fifth step, hybridization, is the most time-consuming; during hybridization, the cRNA binds to the specific probes on the microarray chip [381].

The next, sixth, step is washing of all cRNAs that are non-specifically bound to the microarray surface [381]; as the non-specifically bound cRNA is washed with varying efficiency, the sensitivity and background level of the entire microarray are affected [381].

The seventh step is the staining of the hybridized cRNA with fluorescent dye [381, 392] so that probe-target hybridization might be detected during the scanning process [392]. There are also approaches for labeling the targets pre-hybridization [392]. There are also *two-channel* microarrays for comparing two samples labeled with two different fluorescent dyes.

The eighth step is the scanning, where the bound fluorescent dye is excited using a laser [381, 392]; the scanners measure the level of fluorescence, which is assumed to be proportional to the amount of cRNA bound to the corresponding probe [381]. An example of scanned *two-channel* microarray is shown in Fig. 3.3.

The final step is the data pre-processing from the microarray image obtained in the previous step [381]. Several pixels are tied to each probe in the image; thus, the pre-processing of the image has to convert these pixels into a single fluorescence intensity for each probe [381]. As there are many factors influencing the measurement of the intensities, such as experimental conditions and cRNA concentrations [381], a normalization is performed after the scanning [381, 393]; however, the used normalization approach can have a

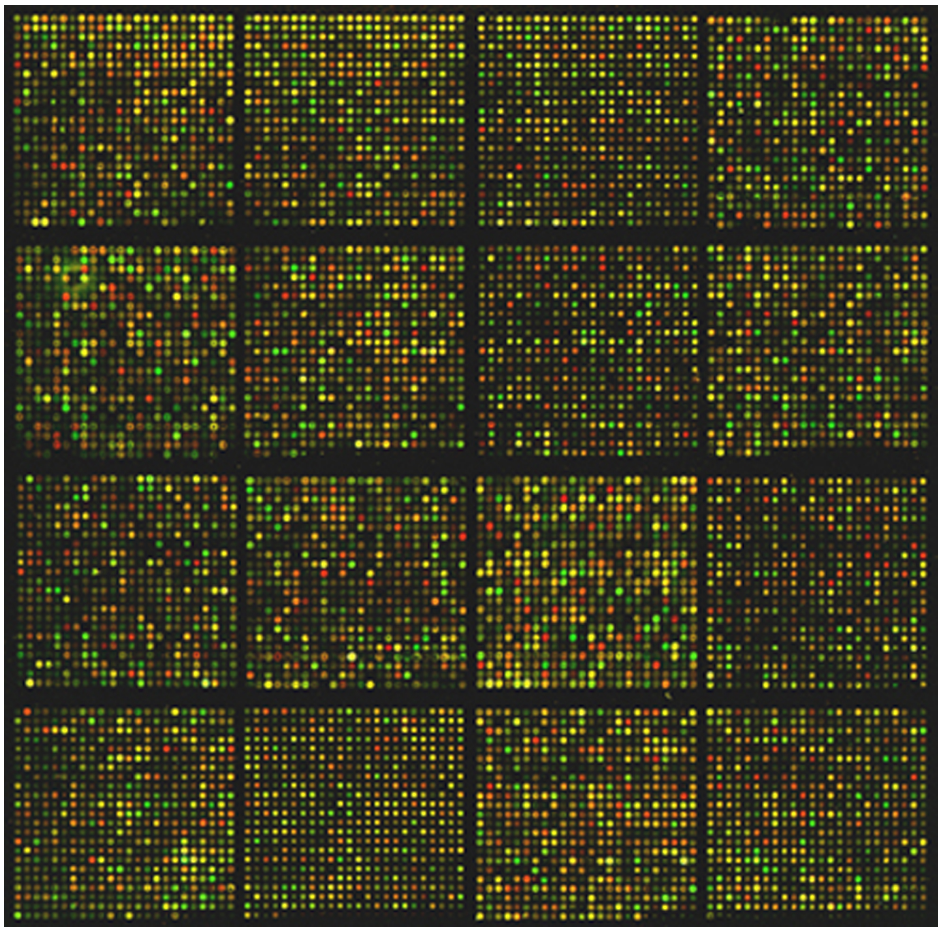


Figure 3.3: An example of scanned *two-channel* mouse cDNA microarray. It shows the gene expression differences of approx. 8,7000 genes between two different mouse tissues. By Louis M. Staudt [public domain], via Wikimedia Commons, `https://commons.wikimedia.org/wiki/File:Mouse_cdna_microarray.jpg`.

huge impact on the results of the experiment — more detailed discussion and examples of normalization approaches are available in [346, 348, 393–410].

More detailed description of microarray structure and the whole process of measurement gene expression using microarrays from first step of RNA isolation through amplification to scanning is available in [364, 376, 380–383, 411]; detailed protocols are available in [379, pp. 18–32] and [382]. A discussion of approaches for analyzing DNA microarray data is available in [348, 412], and a tutorial is available in [413].

3.3.2 *RNA-Seq*

The RNA sequencing (RNA-Seq) [373, 414–416], a competitor DNA microarrays is a more recent method for measuring gene expression leves. RNA-Seq boasts many advantages over the DNA microarrays.

First, RNA-Seq allows for the discovery of new genes and exons while DNA microarrays require to know *a priori* the measured sequence [414, 417]. While there is a microarray variant called *genome tiling array* (or just *tiling* *array*) [418–421] that can discover new genes (e.g., [421]) and exons, it needs a lot of input RNA and has several limitations affecting sensitivity, specificity, and direct splice detection [414].

Second, DNA microarrays have a limited measurement range as they cannot reliably detect transcripts with low abundance or alternative isoforms [417]; this is not the case for the RNA-Seq [414, 417]. On the other hand, RNA-Seq approaches also have several disadvantages compared to microarrays; they are costlier [417] and have higher data analysis requirements (see Fig. 3.5 for individual steps of the data analysis) as there are no "gold standard" pipelines at the moment — unlike the DNA microarrays which have standardized workflows and available user-friendly software for data processing and analysis [417]. Also, microarray experiments have faster turnaround times, especially for small studies, as RNA-Seq that requires a flow cell with multiple lanes or a chip with multiple samples for cost-efficient operation [417]. Also, RNA-Seq requires mRNA selection or removal of abundant transcripts in order to avoid high sequencing costs [417].

An in-depth comparison of microarray and RNA-Seq technologies, together with guidelines for the selection of the appropriate approach for an experiment, is given in [417]. Further description of RNA-Seq is out of the scope of this work as mainly DNA microarray data were used for the experiments and, moreover, the task is motivated by inferring the full expression profile out of the very cost-effective L1000 microarray platform (see Section 3.3.3). More details about RNA-Seq is available in [422–427] and illustrative overview of the RNA-Seq process is shown in Fig. 3.4 just to provide reader an idea about the whole process; full description of the figure is available in [422].

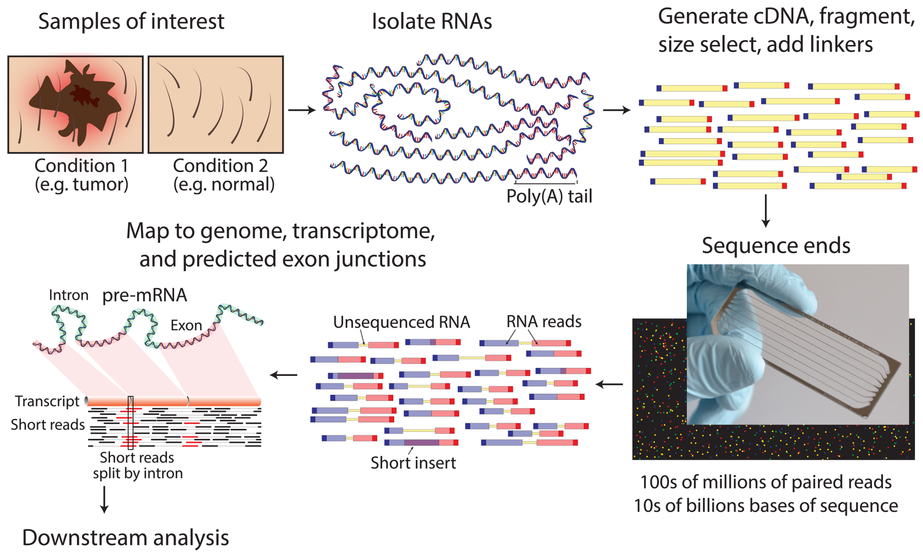

Samples of interest
Condition 1 (e.g. tumor)
Condition 2 (e.g. normal)
Isolate RNAs
Poly(A) tail
Generate cDNA, fragment, size select, add linkers
Sequence ends
100s of millions of paired reads
10s of billions bases of sequence
Unsequenced RNA
RNA reads
Short insert
Map to genome, transcriptome, and predicted exon junctions
Intron
pre-mRNA
Exon
Transcript
Short reads
Short reads split by intron
Downstream analysis

Figure 3.4: General overview of RNA-Seq. By Griffith et al. [422] [CC BY 4.0], taken from `https://doi.org/10.1371/journal.pcbi.1004393.g002`.

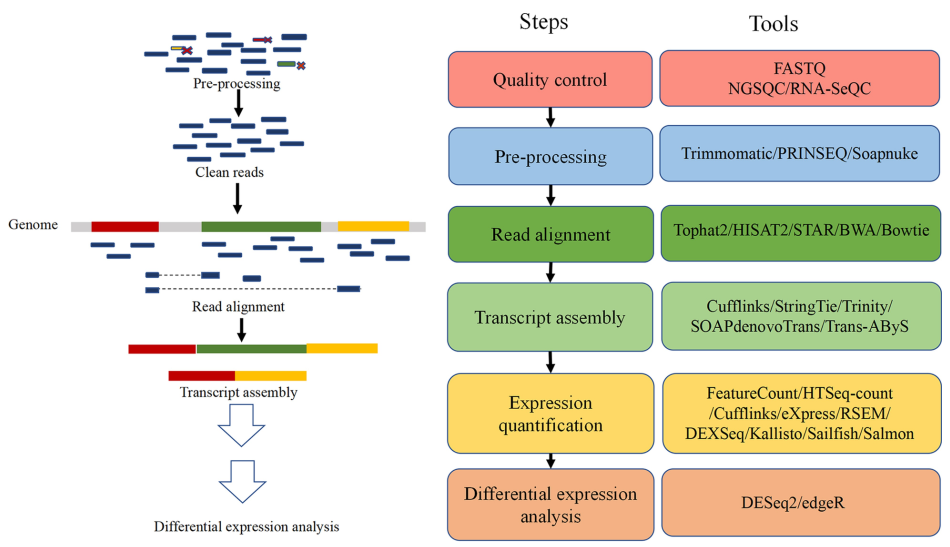

Pre-processing
Clean reads
Genome
Read alignment
Transcript assembly
Differential expression analysis
Steps
Tools
Quality control
FASTQ
NGSQC/RNA-SeQC
Pre-processing
Trimmomatic/PRINSEQ/Soapnuke
Read alignment
Tophat2/HISAT2/STAR/BWA/Bowtie
Transcript assembly
Cufflinks/StringTie/Trinity/
SOAPdenovoTrans/Trans-ABySS
Expression quantification
FeatureCount/HTSeq-count
/Cufflinks/eXpress/RSEM/
DEXSeq/Kallisto/Sailfish/Salmon
Differential expression analysis
DESeq2/edgeR

Figure 3.5: Structure of an analysis of RNA-Seq experiment together with commonly used tools. By Hong et al. [425] [CC BY 4.0], taken from https://jhoonline.biomedcentral.com/articles/10.1186/s13045-020-01005-x.

3.3.3 *L1000 gene expression profiling assay*

The L1000 is a new low-cost and high-throughput gene expression profiling assay introduced in [1] and a motivation for the task solved in this thesis. The assay reaches its cost-effectiveness by directly measuring only 978 carefully selected landmark genes, and the rest is computationally inferred [1]. Originally, the inference involved a linear regression model where the expression of the inferred gene is computed as a linear combination of the expressions of the measured landmark genes [2]. Later on, more advanced, non-linear models such as the D–GEX were introduced [2]. The main advantage of the L1000 profiling platform is its low cost while still being comparable to RNA-Seq methods [1, 428], which allowed to create a huge dataset of over 1,300,000 gene expression profiles — this scale is unprecedented and would be very costly to reach with other methods such as the RNA-Seq. The L1000 data can be used, for example, for drug discovery using GANs [429].

3.3.3.1 *Selection of landmark genes*

The L1000 is measuring selected landmark genes that are suitable foundations for inference of the rest of the target genes [1]. The landmark genes were selected using a large (in terms of usual sizes of GE datasets) and diverse collection of 12,063 gene expression samples that were sampled using Affymetrix HG-U133A microarrays from the Gene Expression Omnibus (GEO) repository [430].

As this dataset contained a non-uniform representation of various biological aspects (e.g., disproportional representation of certain tumor types was present), a principal component analysis (PCA) was applied in order to reduce dimension and minimize bias towards any specific lineage or cellular state [1]. This reduced the dimension to 386 components that were able to explain 90 % of the variance. To identify commonly co-regulated transcripts, Subramanian et al. applied a cluster analysis using an iterative peel-of procedure for centroid selection [431] that repeatedly uses k-means algorithm on 100 independent subsamples, each covering 75% of the data. Based on the clustering analysis, Subramanian et al. created a sets of genes that co-clustered in more than 80% of the trials [1]. Stable clusters were then excluded from the data, and the procedure was repeated. This yielded potential landmark candidates whose transcripts were then empirically tested in order to evaluate their ability to measure the GE levels accurately [1]. For each gene, a 40 nt long sequence was selected and then split into two 20-mers that were then each coded on a probe in the L1000 assay [1].

3.3.3.2 *L1000 comparison to RNA-Seq*

As RNA-Seq is an emerging platform with many benefits (See Section 3.3.2), Subramanian et al. also compared the gene expression profiles generated using the L1000 assay with those generated using Affymetrix microarrays and RNA-Seq. Subramanian et al. profiled 3,176 samples using both L1000 and Illumina TrueSeq RNA sequencing data on the same samples. After normalization and batch corrections, Spearman rank correlations (sample self-correlations) were calculated for the 970 landmark genes[1] and the median sample self-correlation was 0.84 [1]; furthermore, sample recall defined as the fraction of reference similarity values that are lower than the similarity between the designated samples was calculated in order to provide an assessment of how well a particular pair of samples or genes match each other relative to an appropriate null (see [1] for details on the used definition of sample recall) [1]. The results were 98% of samples with sample recall > 0.99 and 99.84% samples with sample recall > 0.95 [1]. Moreover, a small subset of these samples was also profiled using the Affymetrix platform to provide a comparison to a microarray based gene expression profiling platform. The results from the analysis by Subramanian et al. is that the L1000 measured and inferred GE values are "*as similar with RNA-Seq as RNA-Seq is with Affymetrix*" [1].

Furthermore, the GE profiles obtained from the L1000 assay can be converted to RNA-Seq-like profiles using the deep learning approaches presented by Jeon et al. in [428]. Jeon et al. first use a modified cycle-consistent GAN (CycleGAN) to map the microarray GE profiles of the 978 landmark genes into a RNA-Seq-like profiles; these 978 landmark RNA-Seq-like GE profiles are then extrapolated using a a fully-connected D–GEX-like neural network into the full genome space [428]. For certain applications, this extrapolation allows biologists to use usual RNA-Seq data processing pipelines on the vast GE dataset obtained using the L1000 assay.

1 8 landmark genes from the L1000 were not included in the RNA-Seq data

4 LITERATURE REVIEW

4.1 ARTIFICIAL NEURAL NETWORKS FOR GENE EXPRESSION INFERENCE AND CLASSIFICATION

ANNs represent a state-of-the-art approach in many fields (e. g. image classification, segmentation or reconstruction, natural language processing, and time-series forecasts), and biology is no exception (review e. g. [66, 67, 432–438]). The ANNs were used, for example, to analyze gene expression relationships [439], for gene expression inference [2, 14, 15, 440], or for gene classification [440]. While NN model can be used for various tasks in biology, this work focuses mainly on NNs working with GE data; other modalities are out of the scope of this work. One such application particularly important for this thesis is D–GEX [2], which infers a full gene profile using only $\sim 1,000$ selected *landmark genes* measured by the L1000 microarray assay (see Section 3.3.3).

4.1.1 *D–GEX*

The D–GEX family is made up of 9 different architectures. For technical reasons, D–GEX consists of two separate feedforward NNs having from one to three hidden layers — each having either 3,000, 6,000, or 9,000 neurons. Each network predicts only half of the target genes ($\sim 4,760$ genes) and is trained on a separate GPU. The neural networks were trained using a standard back-propagation algorithm with mini-batch gradient descent with momentum and learning rate decay [2]. The initial weights were initialized using normalized initialization [441]. The error metric used was mean absolute error (MAE).

The original D–GEX was evaluated using data from three different sources — *GEO expression data* curated by the Broad Institute, *GTEx expression data* consisting of 2,921 gene expression profiles obtained using the Illumina RNA-Seq platform [442] and *1000 Genomes expression data* consisting of 462 gene expression profiles also obtained using the Illumina RNA-Seq platform [443]. The *GEO expression data* contained biological or technical replicates; the final dataset contained $\sim 110,000$ samples after removing these replicates. All three datasets were jointly quantile normalized and then standardized for each gene individually.

The D–GEX neural networks were compared with linear regression and k–nearest neighbor (KNN) regression. The linear regression builds a linear model for each target gene, while the KNN regression finds k closest expression profiles in the available data and returns the mean of the appropriate targets. The D–GEX neural networks were found to perform superiorly on all three datasets. The L_1 and L_2 regularized linear regression performed similarly to non-regularized linear regression.

Another approach for gene expression inference using the same data as D–GEX appeared concurrently with our research [10, 444]— this approach uses generative adversarial network (GAN) for estimating the joint distribution of landmark and target genes [14, 15]. This approach resembles a two-player minimax game between two neural networks – generative and discriminative models. Another approach based on the D–GEX, called L–GEPM, was presented in [445], where LSTM units were used. Yet another approach, albeit not based on neural networks, was presented in [446], where authors used the XGBoost algorithms for gene expression inference.

As briefly discussed in Section 3.3.3, a D–GEX-like network similar to the ones used throughout this work was used to obtain RNA-Seq-like gene expression profiles from the microarray gene expression profiles from the L1000 assay in [428]. The D–GEX-like network is very similar to the original D–GEX both in terms of depth and width — it has three hidden layers and 2,048, 4,096, and 8,162 neurons in the hidden layers [428]. This is unlike the original D–GEX, which has uniform width across the three hidden layers; nevertheless, the dimensions are similar as the widest D–GEX had 9,000 neurons in each layer [2]. Furthermore, the original D–GEX predicted GE only for 9,520 target genes [2] whereas this approach extrapolates the GE profiles for 23,614 genes [428].

4.1.2 *Usage of neural networks for other gene expression data tasks besides profile reconstruction from the L1000 assay*

While the GE profile reconstruction from the L1000 assay is a task that is of particular interest for this work, there are other tasks in biology where NNs and other machine learning methods can be used. For example, Eetemadi and Tagkopoulos used an ANN to capture GE relationships [439]. They used a NN architecture they called genetic neural network (GeNN)[1] for prediction of the genome-wide GE utilizing gene knockouts and master regulator perturbations [439] — more in Section 4.1.2.1. Neural networks can also be used for clustering and dimensionality reduction of the gene expression data but also for analysis of functional patterns of the GE data and even their generation (see Section 4.1.2.2). While the Section 4.1.2.2 mostly focuses on unsupervised approaches deepening the understanding of the GE data, there are also many applications of NNs for supervised tasks such as the classification of GE microarray, RNA-Seq, and other kinds of data; these tasks are briefly discussed in Section 4.1.2.3.

4.1.2.1 *Genetic neural network*

The GeNN incorporates existing gene regulatory information in its architecture — the inputs are the expression levels of master regulator (MR) genes and gene knockout information, the intermediary layers compute the expression levels of individual genes — every single layer computes the expression level of a single gene — and the outputs are the predicted gene expression

1 Eetemadi and Tagkopoulos abbreviated genetic neural network as GNN but this abbreviation is more commonly used for graph neural networks (GNN).

levels [439]. This architectural framework is founded on the assumption that the expression of a gene, regulated by d regulatory genes, can be estimated using a nonlinear transformation of the weighted sum of expression levels of the regulatory genes, i.e., by $f_\theta(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d_{\geq 0}$ is the expression level of the d regulatory genes, f is the activation function of the given node and θ is the set of function parameters including $\boldsymbol{p}$, $\boldsymbol{t}$, $\boldsymbol{b}$, and t_0 (see Eq. (4.1) for details) [439].

The activation f is defined as

$$f_\theta(\boldsymbol{x}) = \frac{t_0 + \sum_{k=1}^{d} t_k \exp(p_k x_k)}{1 + \sum_{k=1}^{d} b_k \exp(p_k x_k)}, \tag{4.1}$$

where $\boldsymbol{p} \in \mathbb{R}^d$ is the input weight vector, $\boldsymbol{t} \in \mathbb{R}^d_{\geq 0}$ is the numerator weight vector, $\boldsymbol{b} \in \mathbb{R}^d_{\geq 0}$ is the denominator weight vector, and $t_0 \in \mathbb{R}_{\geq 0}$ is the bias [439].

Since the layers are tied with individual genes whose expression they predict, these layers are in a topological order of the nodes of the regulatory graph [439]. If there is a cycle detected, the feedback edges are moved before the topological ordering of genes [439]. With such architecture, the expression levels of individual genes can be calculated using a single forward pass if the weights θ are known [439]. To obtain the weights θ for each layer, Eetemadi and Tagkopoulos used a layer-wise training algorithm (more details in the original work [439]) iteratively employing linear programming (LP).

The GeNN was empirically tested using the subset of transcriptional regulatory network (TRN) of *Escherichia coli* and gene expression levels from *in vivo* microarray data and simulated data using a real biological network [439]. While the GeNN outperformed its competitors such as MLP, RNN, bidirectional recurrent neural network (BRNN) [447], lasso [448], and another GeNN variant called linear GeNN (LinGeNN)[2] that used a linear function as the activation function [439], its main limitation is that the topological ordering of nodes is made in a way that ignore cycles in the TRN and as such cannot take into account feedback loops [439]. Similarly, Somathilaka et al. used a NN approach where gene-to-gene interaction dynamics is embedded in gene regulatory network (GRN) was used to create gene regulatory (GRNN) based on graph neural networks (GNN) in [449].

4.1.2.2 *Clustering, analysis, and generation of gene expression data*

Neural networks are often used in bioinformatics for the generation or augmentation of data (see [450] for an overview); one such approach represent GANs (see Section 4.5.2 for more details about GANs in general). GANs are often used for the generation of gene expression data [451], and there are various examples in the literature (more in reviews in [438, 451–453]).

A combination of variational autoencoder (VAE) and GAN was for data generation of RNA-Seq data in [454] where Yu and Welch used three large single-cell RNA-Seq (scRNA-Seq) (more about single-cell omics in [455]) datasets first to learn disentangled representations of RNA-Seq data using

2 Eetemadi and Tagkopoulos used abbreviation *LinGNN*, see Footnote 1.

a VAE and then used the representations to train a conditional GAN [454] as Yu and Welch observe that GANs generate better samples than VAEs. Similarly, a conditional single-cell GAN (cscGAN) was used to generate realistic cell samples in [456] where Marouf et al. showed that augmenting sparse cell populations with cscGAN improves robustness and reliability of classifiers and downstream analyses including detection of marker genes, thus possibly reducing the number of animal experiments required and in turn reducing costs of research [456]. Another variant of GAN model was used by Lall, Ray, and Bandyopadhyay in [457] for the generation of new scRNA-Seq cell samples. A conditional GAN was used to generate GE data of *Escherichia coli* and humans in [458]. Yet another GAN based model was used by Park et al. for a generation of good and bad prognosis samples in order to improve the prognosis classification [459]. Chaudhari, Agrawal, and Kotecha used GAN based model for augmentation of GE data to improve cancer classificaion in [460]. Similarly to the work by Park et al., Chaudhari, Agrawal, and Kotecha, Xiao, Wu, and Lin used a GAN to generate samples from a minority class to improve cancer diagnosis from RNA-Seq data in [461]. A Wasserstein GAN with gradient penalty (WGAN-GP) was used for GE data generation to improve the classification performance in [462]. A GAN with a semi-interpretable generator was used for a generation of synthetic RNA-Seq dataset in [463]. Yet another GAN based model was used for scRNA-Seq data augmentation in [464] and in [465], other examples of NN based GE and other omics data augmentation or generation are [466–474]. However, the GANs and other NN based models are not the only successful approaches for generation of GE data — e.g., Sun et al. used probabilistic models based on copulas to generate scRNA-Seq GE data capturing gene correlations in [475] and Dibaeinia and Sinha used models guided by gene regulatory networks to generate single-cell gene expressions in [476]. The NNs are also not limited to GE data; for example, Wan and Jones used a GAN based method by generating synthetic feature samples to improve protein function prediction [477], Méndez-Lucio et al. used NN to generate molecules from L1000 GE profiles [478], and Yelmen et al. used GANs and restricted Boltzmann machines (RBMs) to generate novel high-quality genomes [479]. An approach based on adversarial autoencoder was used in [480] for feature extraction from high dimensional RNA-Seq GE data. The autoencoder (AE) can also be used for denoising scRNA-Seq data as shown in [481, 482]. VAE based model was used for inference of cellular dynamics from RNA-Seq data in [483]. A NN model combining both unsupervised AE and supervised classification layer was used for scRNA-Seq data clustering and annotation in [484]. A reviews of the usage of NNs for analysis of RNA-Seq data are available in [433, 452, 453, 485–489].

A VAE has been used for learning the infection responses that are cell-type and species-specific from a single-cell gene expression data in [490]; this was improved in a later work also by [491] in [491]. An approach similar to VAEs based on denoising autoencoders (DAEs) and deep belief networks (DBNs) was used in [492] for a generation of gene expression data and for gene clustering. VAEs and deep Boltzmann machines (DBMs) were used for genartion of synthetic scRNA-Seq data in [493]. Comparison of PCA and AE for learning latent feature representation for human gene expression data is available in [494]. An approach combining VAEs with Bayesian Gaussian-mixture models was used to analyze single-cell ATAC-seq (scATAC-Seq) data (while not GE by themselves, they are also high dimensional and noisy) in [495]. An autoencoder based method was used for clustering of scRNA-Seq data in [496] where Tran et al. used 28 real scRNA-Seq datasets with more than three million cells in total for the empirical evaluation of their approach. A VAE based model was used for feature-level clustering in [497]. A pre-trained AE was also used for clustering of scRNA-Seq data in [498]. VAE using directly raw data from scRNA-Seq to avoid data preprocessing was used in [499] for clustering of GE data. A NN based methods combining gene ontology (GO) information with an AE model and fully-connected NN were used for dimensionality reduction in [500]. Another approach using external information in a NN model was proposed in [501], where He, Fan, and Yu used a gene-interaction graph to guide the clustering by encouraging adjacent genes to have similar weights in a NN model. A VAE aiming for better interpretability used a decoder whose wiring mirrors user-provided gene modules to provide direct interpretability in [502]. Walbech et al. used an interpretable AEs to show that biological concepts can be associated with specific nodes and can be interpreted in relation to biological pathways [503] and AEs can be used to assist in the interpretation of new unseen data [503]. A variant of DAE was used for semi-supervised clustering of scRNA-Seq data in [504]. Pati et al. used GAN for imputation of microarray data in [505]. An AE based approach was also used on scRNA-Seq data for data imputation and dimensionality reduction in [506] and [507]; other examples of imputation of scRNA-Seq and similar GE data using AEs, GANs, and other NN architectures include [508–527]; a review of NN based approach for omics data imputation is available in [528] and a review of GANs for data imputation in general in [529]. Pandey and Onkara used GAN to impute GE data with a focus on downstream functional analysis and showed that the GAN based approach outperforms other baseline methods in clustering, visualization, classification, and DGE analysis using the imputed data. Another data imputation method was provided in [530], where Hausmann et al. used a GAN model to reconstruct missing single-cell gene expressions. A comparison of a standard GAN and WGAN-GP for augmentation of DNA microarray and RNA-Seq data is available in [531]. Another example of DNA microarray gene expression data augmentation using a GAN based model is provided in [532] where Jahanyar, Tabatabaee, and Rowhanimanesh showed that their approach is able to generate artificial samples that are close to the original samples. A comparison of several GAN models on RNA-Seq data is available in [533]. He et al. used a NN based model for prediction of tissue gene expression profiles in [534].

Kinalis et al. used the deconvolution of AEs to learn biological regulatory modules from scRNA-Seq data [535]; similarly, [536] used sparsely connected AEs for functional-feature-based data reduction that could provide better links among cell clusters [536]. An AE based model was used for inference of a gene regulatory network in [537] and in [538]. A variant of a deep AE was used for cell-type-specific gene analysis by constructing an interpretable decoder in [539]. An interpretable decoder was also used in an AE model for analysis of cell-free DNA in [540]. An ensemble of AEs was used for clustering of scRNA-Seq data in [541]. A VAE extension using mutual information was used to learn an efficient low-dimensional representation of several RNA-Seq datasets leading to high clustering performance in [542]. Additional examples of other studies using AEs for clustering and dimensionality reduction of scRNA-Seq data are [543–572]. A graph-based AE model for integrating spatial transcriptomic data with *chromatin imaging* data to identify molecular and functional alterations in tissues in [573].

Neural network based model was used for assessing the importance of genes as possible biomarkers of Hepatocellular carcinoma in [574]. Tasaki et al. used a NN to predict differential expression in [575]. Fakhry, Khafagy, and Ludl used a NN model with two parallel branches to detect gene–gene interactions from GE data in [6].

The NN models can also be used for translation between individual domains of data; for example, Yang et al. used AEs for translation between several modalities including single-cell imagining, RNA-Seq, ATAC-seq, and Hi-C data [576]. Another VAE model was used for unpaired multi-omics integration of data sources such as scRNA-Seq, scATAC-Seq, and single-nucleus methylcytosine sequencing (snmC-Seq) data using graph-guided embeddings in [577]. Similarly, a deep generative model was used for integration of multimodal biological data in [578] allowing data imputation if some modality is missing [578]; this model was built upon previously published deep generative models for single-cell chromatin accessibility analysis [579], single-cell variational inference (scVI) of gene expression data [580], and single-cell multi-omic model called *totalVI* [581]. A comparison of various scRNA-Seq imputation methods, including scVI, is available in [582]. A GAN based model for the reconstruction of genome-wide gene expression profiles from DNA methylation data was proposed in [583]. A heterogeneous graph transformer NN model was used in [584] for biological network inference from multiple modalities. An adversarial model was used for integration of single-cell chromatin accessibility and gene expression data in an unsupervised manner in [585] while a VAE based model was used for similar task in [586]; additional examples of NN based data integration approaches are available in [587–595]. Note, however, that there are successful approaches for integration of multiple modalities of single-cell data that do not use NN based models, for example [596]. Another model for dimensionality reduction of ATAC-seq was introduced in [597] where Kopp, Akalin, and Ohler used a VAE based model using batch adversarial training strategy. Liu et al. used a pair of GANs to simultaneously learn the latent representation and infer cell labels using scATAC-Seq data in [598].

Neural networks can also be used for data corrections; for example, Shaham et al. and Wang, Liu, and Zhao used residual neural networks for removal of batch effects in scRNA-Seq data in [599] and [600] while Lotfollahi, Wolf, and Theis used model based on VAEs for the same task in [490] (performance comparison of both approaches with several other methods is available in [601]). Tarca and Cooke used a robust NN approach for spatial and intensity-dependent normalization of cDNA microarray data in [408]. An approach combining and mutual nearest neighbor (MNN) and a neural network with residual blocks was used for batch correction of scRNA-Seq data in [602]; other examples of usage of NNs for batch effect removal are [603–611]. The removal of batch effects can also be solved by other approaches besides NNs; e.g., SCIBER [612] and *scBatch* [613].

4.1.2.3 *Classification of gene expression data*

NNs are also often used for classification of GE data. One such example is [614] where Lahmer, Oueslati, and Lachiri used a fully-connected network similar to D–GEX (albeit with ReLU activations instead) for binary classification of microarray data — the goal was the identification of cell cycle-regulated genes. Interestingly, Lahmer, Oueslati, and Lachiri worked directly with the scanned images of the microarray data directly (similarly as in [615] where the authors used such data with support vector machine (SVM) and KNN algorithms). Since the inputs were images, they also used a CNN besides the D–GEX-like architecture; the CNN model indeed proved advantageous over the D–GEX-like architecture when working with the image inputs. Purba et al. used NNs to classify liver cancer using miRNA data in [616].

Quite an interesting approach was proposed in [617], where Schmauch et al. created a NN model to predict RNA-Seq gene expression profiles from whole-slide images. Similarly, Levy-Jurgenson et al. used a CNN based model to predict gene expressions from whole-slide images in [618], while Alsaafin et al. used a NN with attention-based topology predicting the RNA-Seq profiles from images in [619].

Yuan et al. used the gene expression profiles of the L1000 landmark genes (see Section 3.3.3) to train a deep AE for feature extraction from the human transcriptome and a second deep NN for cancer classification [620]. The AE part was designed such that in each layer, there was a 30 – 50% reduction in the number of neurons, leading to 30 output neurons that produced the feature vector; the second NN then used these 30 features to for cancer classification [620]. A NN based model for disease state classification using RNA-Seq data was proposed in [621]; this model consisted of two parts; first was a multitasking model classifying the disease state and tissue origin and second model was for subtype classification [621]. A similar approach was used in [622], where Azarkhalili et al. used an even smaller latent vector of size 8 compared to the 30 elements in [620]. Yap et al. analyzed a NN for tissue classification from RNA-Seq data using Shapley additive explanations (SHAP) values to test the reliability of model explainability in [623] (more about model explainability with respect to genomics, in particular, is available in [624, 625]). Bayesian NN models were used for cancer classification from RNA-Seq data in [626, 627].

As already mentioned in Section 4.1.2.2, Park et al. used a GAN-based model to improve the accuracy of prediction of cancer outcomes in [459]. Zhang et al. used an extreme learning machine (ELM) for cancer classification from microarray GE data in [628]. Another GAN based model together with a deep multilayer NN was used to improve the prediction of the prognostic outcome of cancer from multimodal data containing miRNA and mRNA expressions and histopathological image data [629]; similarly, Duroux et al. used both RNA-Seq data and histopathology whole-slide images to predict cancer subtypes and severity in [630] where they compared their proposed approach to a neural network model. Other examples of NN based model for cell and tissue type and cancer and other disease classifications are [328, 594, 595, 631–639]. A review of NN based models using GE data for cancer diagnosis is available in [640] and more about general challenges in incorporating ML models for in oncology and other medical fields is detailed in [641–645]. The classification of cancer patients is not limited to the GE data — for example, classification of cancer patients from the sequences from gut microbiome of cancer patients in [646], or decoding mutational signatures in human pan-cancers using a sparse autoencoder (SAE) in [647]. However, these and other modalities are not the focus of this work.

4.2 ACTIVATION FUNCTIONS

The activation function is employed to regulate the output behavior (firing) of neurons. The activation function is a mathematical function applied to the output of a neuron or a layer of neurons. It introduces non-linearity to the network, enabling it to model complex relationships and make non-linear transformations of the input data. The presence of non-linear functions is what confers the usefulness of deep networks in addressing complex problems. It can be demonstrated that a classical multilayer neural network with a linear activation function is equivalent to a single-layer perceptron, merely performing a linear combination of its input signals [46, p. 192]. By applying a non-linear activation function, the network becomes capable of representing complex patterns and relationships in the data.

The non-linearity exhibited by the activation function, previously referred to as a "squashing function," underlies the theoretical representational power of neural networks. It has been proven that any continuous function defined on compact subsets of n-dimensional real space ($\mathbb{R}^n$) can be approximated by a feed-forward neural network with a single hidden layer. Initially, this was established solely for the sigmoid function and subsequently for any continuous, bounded, and non-constant activation function. However, this theoretical representation power solely describes the network's potential and does not address its practical usability and trainability. Moreover, it does not render deep learning obsolete, despite some researchers in the 1990s presenting this theorem as an argument against the necessity of deep networks. It is noteworthy that not all activation functions are static in terms of learning; certain functions may possess parameters that are learned during network training, such as the soft exponential or the adaptive piece-wise linear unit.

Activation functions can have different mathematical forms [11, 69], such as sigmoid functions (e.g., logistic function), hyperbolic tangent, or rectified linear unit. The selection of an activation function significantly impacts the modeling capabilities of the network and the level of difficulty in training the network. Each activation function has its own characteristics and properties, influencing the network's behavior and performance [11]. Activation function can be smooth (e.g., logistic sigmoid or tanh) or they may be non-differentiable at specific points (e.g., ReLU); it was shown that smooth activation functions provide deeper information propagation [648].

The choice of activation function depends on the specific task and the network architecture. Different activation functions have different properties, such as differentiability, smoothness, or sparsity. Additionally, certain activation functions may be more suitable for specific problems, such as binary classification or regression tasks. Some authors use optimization approaches to select the suitable activation function for a particular problem; e.g. an evolutionary approach was used to evolve the optimal activation function in [649–666] and grid search using artificial data was used in [667]. Another search for the optimal activation functions was presented in [668] where several simple activation functions were found to perform remarkably well. These automatic approaches might be used for evolving the activation functions (e.g., [649, 655]) or for selecting the optimal activation function for a given neuron (e.g., [658, 669]). While evolved activation function may perform well for a given problem, they also might be very complex — e.g., evolved activation functions in [655]. The complexity of an activation function is also important characteristic as it significantly influences the computational efficiency of a neural network; however, this might be mitigated by efficient implementations (including hardware implementations) of such activation functions (e.g., [670–680]). An empirical analysis of computational efficiency and power consumption of various AFs is available in [681]. Empirical comparison of various activation functions is available in [11, 616, 668, 682–735].[3] Furthermore, mixing multiple activation functions might improve the performance of a NN [720]. Multiple AFs can also be used in activation ensembles where the used activation is selected randomly [719]; a different activation ensemble was used in [736] where Nandi, Jana, and Das trained identical NNs with different AFs and used majority voting to produce the final classificaiton. It can be even beneficial to swap the activation functions during the training as in [737]. An analysis of the initialization method with respect to activation functions is available in [738]. Saha et al. developed a framework where an AF is arising from a solution of differential equations in [739]; this framework can be used to generate more AFs. More details about activation functions available in reviews — e.g., [11, 12, 650, 682, 700, 713, 740–749].

The overview is limited to real-valued activation functions; complex-valued neural networks (e.g., [750–763], brief overview available in [700, 764]), bicomplex-valued neural networks (e.g., [765]), quaternion-valued neural networks (e.g., [766–770]), photonic neural networks (e.g., [771]), fuzzy neural networks (e.g., [772–777]), AFs for probabilistic boolean logic (e.g., [778]), quantum AFs (e.g., [779]) and others are out of the scope of this work.[4]

3 Unfortunately, most works compare mainly only very small subsets of available AFs.

4 While these kinds of NNs are not discussed throughout this work, some of these approaches will use AFs presented in this work.

4.2.1 *Binary activation function*

The binary activation function (binary AF) — also called a step function — is a simple yet important activation function used in neural networks [780]. It assigns an output value of 1 if the input is positive or zero and an output value of 0 if the input is negative [12]. Mathematically, it can be defined as follows:

$$f(z) = \begin{cases} 1, & z \geq 0, \\ 0, & z < 0. \end{cases} \tag{4.2}$$

Similar to binary activation function is the sign function, which produces an output value of -1 if the input is negative and 1 if it is positive (and 0 for outputs that are exactly zero) [12]. Since the sign and the binary activation functions have nearly exact properties from the point of view of neural networks, only the binary activation function is mentioned, but the points hold similarly for the sign activation function.

The main advantage of the binary activation function is that it is straightforward and computationally efficient to implement. It does not involve complex mathematical operations, making it suitable for networks with low computational resources or for hardware implementations [781, 782]. However, the binary activation function has one glaring disadvantage - the lack of differentiability. The binary activation function is not differentiable at the point of discontinuity (x = 0) and is zero elsewhere. This poses challenges for optimization algorithms that rely on gradients, such as BP, since the gradient is noninformative [684, 780, 783]. Since the gradient-based methods are used predominantly, the binary activation function is used very rarely and is important mainly for historical reasons as it was used in the original perceptron [21, 684].

4.2.2 *Sigmoid family of activation functions*

Various smoothed variants of the binary activation functions (sigmoids) are commonly used; the most common is the logistic function — the standard logistic sigmoid function was dominant in the field prior the introduction of ReLU (see Section 4.2.6) [46], the logistic function is often called just sigmoid in the literature which is also used throughout this work for brevity (unless specified otherwise, *sigmoid* is equivalent to standard logistic function in the text). Standard logistic function is defined as

$$f(z) = \sigma(z) = \frac{1}{1 + \exp(-z)}. \tag{4.3}$$

The logistic sigmoid was a popular choice since its output values can interpreted as the probability that a binary variable is 1 [46] since it squashes the input to the interval $(0, 1)$ [11]. The problem of sigmoid activation functions is that they saturate — they saturate when their input z is either a large positive number or a large negative number, which makes gradient-based learning difficult [11, 46]; therefore their use in feedforward networks is usually discouraged [46]. Another option, albeit significantly less popular in ANNs, is the probit AF [784], which is just the cumulative standard normal distribution function used as an AF [784].

Another popular sigmod function is the tanh activation function which is just scaled and shifted logistic sigmoid

$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} = 2\sigma(2z) - 1. \tag{4.4}$$

Similarly as the logistic sigmoid, the tanh also squashes the inputs; however, it squashes them to the interval $(-1, 1)$. The tanh function is often advantageous over the logistic sigmoid function because it is centered around zero and it is similar to the identity function near zero, which makes training of a network easier if the activations are kept small [46]. Nevertheless, the tanh function saturates similarly as does the logistic sigmoid and therefore similarly suffers from the vanishing gradients [11]. Computationally efficient approximation of the tanh activation functions based on splines were proposed in [785] – *tanh36* based on approximation relying on 36 equidistant points and *tanh3* using only 3 points. Scaled variant $\tanh\left(\frac{z}{2}\right)$ was used in [786]. The linearized unit (LRTanh) is a tanh variant used together with modified BP that substitutes a different activation function derivative proposed in [787]. There are also approximations of the logistic sigmoid and tanh that are meant to speed up the computations; e.g., pRPPSG [788] and other similar piecewise approximations [789, 790].

A scaled version of the logistic sigmoid function was proposed in [791] with the motivation to have the same linear regimes as the tanh and relu activation functions when initialized with the popular normalized initialized method proposed in [441]. The scaled version used fixed parameters

$$f(z) = 4\sigma(z) - 2. \tag{4.5}$$

A more complicated variant named n-sigmoid was proposed in [792]; however, it seems that the formula presented in the paper is not as the authors intended and, therefore, we omit this AF from the list.

4.2.2.1 *Shifted and scaled sigmoid (SSS)*

The shifted and scaled sigmoid (SSS) was used in [793]; it is the logistic sigmoid with horizontal scaling and translation defined as

$$f(z) = \sigma\left(a\left(z - b\right)\right) = \frac{1}{1 + \exp\left(-a\left(z - b\right)\right)}, \tag{4.6}$$

where a and b are predetermined parameters; Arai and Imamura used $a = 0.02$ and $b = 600$.

4.2.2.2 *Variant sigmoid function (VSF)*

The variant sigmoid function (VSF) is an older parametric variant of the logistic sigmoid proposed in [794]. It is defined as

$$f(z) = a\sigma(bz) - c = \frac{a}{1 + \exp(-bz)} - c, \tag{4.7}$$

where a, b, and c are predetermined parameters [794].

4.2.2.3 *Scaled hyperbolic tangent*

A parametric version called scaled hyperbolic tangent (stanh) was used in [795]:

$$f(z) = a \tanh(b \cdot z), \tag{4.8}$$

where a and b are fixed hyperparameters that control the scaling of the function. Lecun et al. proposed using $a = 1.7159$ and $b = \frac{2}{3}$

A similar concept was analyzed in [796] where sigmoids with bi-modal derivatives were used as activation functions. An example of such a function is

$$f(z) = \frac{1}{2}\left(\frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z - b)}\right), \tag{4.9}$$

where b is a hyperparameter [796]; similarly, additional three activation functions with bi-modal derivates were proposed in [796].

4.2.2.4 *Arctan*

The arctangent (arctan) function and its variation were used as activation functions in [797]:

$$f(z) = \tan^{-1}(z). \tag{4.10}$$

The arctan resembles a logistic sigmoid activation, however, it covers wider range $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ [797]. The arctan and several its variation were compared with the tanh, ReLU, leaky ReLU (LReLU), logistic sigmoid activation, and swish in [797]; the best-performing functions in the presented experiments were the arctan and its variation arctanGR [797]. Interestingly, the arctan was used as an AF twenty years earlier in [798]. The arctanGR is a scaled version of the arctan and is defined as

$$f(z) = \frac{\tan^{-1}(z)}{\frac{1+\sqrt{2}}{2}}. \tag{4.11}$$

Other scaling variants such as division by the π, $\frac{1+\sqrt{5}}{2}$, or the Euler number are presented in [799].

4.2.2.5 *Sigmoid-Algebraic activation function*

The Sigmoid-Algebraic is a sigmoid variant defined in [800]. It is defined as

$$f(z) = \frac{1}{1 + \exp\left(-\frac{z(1+a|z|)}{1+|z|(1+a|z|)}\right)}, \tag{4.12}$$

where $a \geq 0$ is a parameter [800].

4.2.2.6 *Triple-state sigmoid*

The triple-state sigmoid unit (TS-sigmoid) is a cascaded AF similar to TS-swish (see Section 4.2.3.6) [800]; it is defined as

$$f(z) = \frac{1}{1+\exp(-z)} \left(\frac{1}{1+\exp(-z)} + \frac{1}{1+\exp(-z+a)} + \frac{1}{1+\exp(-z+b)} \right), \tag{4.13}$$

where a and b are fixed parameters [800].

4.2.2.7 *Improved logistic sigmoid*

The improved logistic sigmoid is yet another sigmoid based activation function designed to deal with the vanishing gradient problem

$$f(z) = \begin{cases} a(z-b) + \sigma(b), & z \geq b, \\ \sigma(z), & -b < z < b, \\ a(z+b) + \sigma(b), & z \leq -b, \end{cases} \tag{4.14}$$

where a and b are fixed parameters [801]; a controls the slope and b is a thresholding parameter. The authors recommend a bound on the slope parameter a:

$$a > a_{\min} = \frac{\exp(-b)}{(1+\exp(-b))^2}. \tag{4.15}$$

Even though the parameters are fixed during the training of a network, a procedure for preseting them based on the network and data was proposed in [801]. The output range of the SiLU is $(-\infty, \infty)$ [11]. The authors Qin, Wang, and Zou also showed that the improved logistic sigmoid AF has a higher convergence speed than the logistic sigmoid AF [801].

4.2.2.8 *Combination of the sigmoid and linear activation (SigLin)*

A SigLin[5] was used as an AF in [802]. The SigLin is defined as

$$f(z) = \sigma(z) + az, \tag{4.16}$$

where $\sigma(z)$ is the logistic sigmoid AF and a is a fixed parameter [802]; however, this AF was used only in a modified optimization procedure [802]. Roodschild, Gotay Sardiñas, and Will experimented with $a \in \{0, 0.05, 0.1, 0.15\}$ [802].

5 This abbreviation is used only in this work; Roodschild, Gotay Sardiñas, and Will did not name the function in [802].

4.2.2.9 *Penalized hyperbolic tangent*

A penalized hyperbolic tangent (ptanh) the LReLU (see Section 4.2.6) but uses the tanh function instead of the linear function [791]:

$$f(z) = \begin{cases} \tanh(z), & z \geq 0, \\ \dfrac{\tanh(z)}{a}, & z < 0, \end{cases} \tag{4.17}$$

where $a \in (1, \infty)$. This function has similar values near 0 as the LReLU with identical parameter a as they both share the same Taylor expansion up to the first order [791]; however this function saturates to $-\frac{1}{a}$ for $z \to -\infty$ and to 1 for $z \to \infty$ [791]. The ptanh AF was found to perform consistently well for various natural language processing (NLP) tasks compared to ReLU, LReLU and several other activation functions [685].

4.2.2.10 *Soft-root-sign (SRS)*

A soft-root-sign (SRS) activation function is a parametric, smooth, non-monotonic, and bounded activation function [803]. It is defined as

$$f(z) = \frac{z}{\frac{z}{a} + \exp\left(-\frac{z}{b}\right)}, \tag{4.18}$$

where a and b are predetermined parameters [803]; the authors Li and Zhou propose using $a = 2$ and $b = 3$ whereas the parameters are said to be learnable in [11]. The output range of SRS is $\left[\frac{ab}{b-ae}, a\right]$ [11, 803]. The performance of the SRS was demonstrated using hte CIFAR-10 and CIFAR-100 [243] task in comparison with the ReLU (see Section 4.2.6 for the description of the ReLU family of AFs), LReLU, PReLU, softplus, exponential linear unit (ELU), scaled ELU (SELU), and swish [803].

4.2.2.11 *Soft clipping (SC)*

The soft clipping (SC) [804, 805] AF is another bounded AF; it is approximatelly piecewise linear in the range $z \in (0, 1)$ and it is defined as

$$f(z) = \frac{1}{a} \ln\left(\frac{1 + \exp(az)}{1 + \exp(a(z-1))}\right), \tag{4.19}$$

where a is a fixed parameter [805].

4.2.2.12 *Hexpo*

The Hexpo activation function [806] was proposed in order to minimize the problem of vanishing gradient [11]; it resembles a tanh activation function with scaled gradients [11]:

$$f(z) = \begin{cases} -a\left(\exp\left(-\frac{z}{b}\right) - 1\right), & z \geq 0, \\ c\left(\exp\left(-\frac{z}{d}\right) - 1\right), & z < 0, \end{cases} \tag{4.20}$$

where a, b, c, and d are fixed parameters. While the parameters could be trainable in theory, it is not recommended as it would lead to the vanishing gradient problem [806]. The Hexpo functions allow for control over the gradient by tunning the parameters a, b, c, and d and the ratios $\frac{a}{b}$ and $\frac{c}{d}$ — with increasing the ratios $\frac{a}{b}$ or $\frac{c}{d}$, the rate of gradient decay to zero decreases; increasing only a and c scales the gradient around the origin up [806].

4.2.2.13 *Softsign*

A softsign activation function is a smooth activation function similar to the tanh activation; however, it is less prone to vanishing gradients [683]. It is defined as

$$f(z) = \frac{z}{1+|z|}, \tag{4.21}$$

where $|z|$ denotes the absolute value of z [683].

4.2.2.14 *Smooth step*

The smooth step is a sigmoid AF; it is defined as

$$f(z) = \begin{cases} 1 & z \geq \frac{a}{2}, \\ -\frac{2}{a^3}z^3 + \frac{3}{2a}z + \frac{1}{2}, & -\frac{a}{2} \leq z \leq \frac{a}{2}, \\ 0 & z \leq -\frac{a}{2}, \end{cases} \tag{4.22}$$

where a is a fixed hyperparameter [807].

4.2.2.15 *Elliott activation function*

Elliott activation function is one of the earliest proposed activation functions to replace to replace the logistic sigmoid or tanh activation functions [808]; the Elliott AF is a scaled and translated softsign AF. It is defined as [11, 740]

$$f(z) = \frac{0.5z}{1+|z|} + 0.5. \tag{4.23}$$

The output of the Elliott activation functions is in range $[0, 1]$ [11, 740]. The main advantage of the Elliott AF is that it can be calculated much faster than the logistic sigmoid [809].

4.2.2.16 *Sinc-Sigmoid*

The Sinc-Sigmoid is a sigmoid-based AF proposed in [800]. It is defined as

$$f(z) = \text{sinc}\left(\sigma\left(z\right)\right), \tag{4.24}$$

where $\text{sinc}\left(x\right)$ is the unnormalized[6] sinc function [800].

6 Koçak and Üstündağ Şiray did not specify whether it is the normalized or unnormalized variant. Still, they provided the derivative of the Sinc-Sigmoid, which suggests that the unnormalized variant was used.

4.2.2.17 *Sigmoid-Gumbel activation function*

The Sigmoid-Gumbel (SG) is a non-adaptive AF proposed recently in [810]; it is defined as

$$f(z) = \frac{1}{1+\exp(-z)} \exp(-\exp(-z)). \tag{4.25}$$

4.2.2.18 *NewSigmoid*

The NewSigmoid is a sigmoid variant proposed in [811]. It is defined as

$$f(z) = \frac{\exp(z) - \exp(-z)}{\sqrt{2(\exp(2z) + \exp(-2z))}}. \tag{4.26}$$

4.2.2.19 *Root2sigmoid*

The root2sigmoid is another sigmoid variant proposed in [811]. It is defined[7] as

$$f(z) = \frac{\sqrt{2}^{z} - \sqrt{2}^{-z}}{2\sqrt{2}\sqrt{2\left(\sqrt{2}^{2z} + \sqrt{2}^{-2z}\right)}}. \tag{4.27}$$

4.2.2.20 *LogLog*

The LogLog is a simple AF proposed in [784]; it is defined as

$$f(z) = \exp(-\exp(-z)). \tag{4.28}$$

The LogLog, cLogLog (see Section 4.2.2.21) were used in NNs for forecasting financial time-series in [784].

4.2.2.21 *Complementary Log-Log (cLogLog)*

The complementary LogLog (cLogLog) is another simple AF proposed in [784] complementing the LogLog (see Section 4.2.2.20); it is defined as

$$f(z) = 1 - \exp(-\exp(-z)). \tag{4.29}$$

The variant called modified cLogLog (cLogLogm) [784] was also proposed:

$$f(z) = 1 - 2\exp(-0.7\exp(-z)). \tag{4.30}$$

7 The author had probably a typo in the definition in the original paper [811]; we present the formula we think Kumar and Sodhi intended to write — it resembles the NewSigmoid and fits the numerical values given in the paper.

4.2.2.22 *SechSig*

The SechSig [812] is another AF utilizing the logistic sigmoid in its definition; it is defined as

$$f(z) = (z + \operatorname{sech}(z))\,\sigma(z)\,. \tag{4.31}$$

Közkurt et al. also proposed a parametric version which we will call parametric SechSig (pSechSig):

$$f(z) = (z + a \cdot \operatorname{sech}(z + a))\,\sigma(z)\,, \tag{4.32}$$

where a is a fixed parameter [812].

4.2.2.23 *TanhSig*

The TanhSig [812] is an AF similar to SechSig; it is defined as

$$f(z) = (z + \tanh(z))\,\sigma(z)\,. \tag{4.33}$$

Közkurt et al. also proposed a parametric version which we will call parametric TanhSig (pTanhSig):

$$f(z) = (z + a \cdot \tanh(z + a))\,\sigma(z)\,, \tag{4.34}$$

where a is a fixed parameter [812].

4.2.2.24 *Multistate activation function (MSAF)*

The multistate activation function (MSAF) is a logistic sigmoid based AF proposed in [813]. The general MSAF is defined as

$$f(z) = a + \sum_{k=1}^{N} \frac{1}{1 + \exp(-z + b_k)}, \tag{4.35}$$

where a and b_k, $k = 1, \ldots, N$ are fixed parameters; $a \in \mathbb{R}$, $N \in \mathbb{N}^+$, $b_k \in \mathbb{R}^+$, and $b_1 < b_2 < \ldots < b_N$ [813]. If $a = 0$, it is named as N-order[8] MSAF.

There is also a special case called symmetrical MSAF (SymMSAF) defined as

$$f(z) = -1 + \frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z - a)}, \tag{4.36}$$

where a is required to be significantly smaller than 0 [813]

4.2.2.25 *Rootsig and others*

The rootsig is one of the activations listed in [814]. It is defined as

$$f(z) = \frac{az}{1 + \sqrt{1 + a^2 z^2}}, \tag{4.37}$$

8 This does not exactly fit into the exemplar MSAF of order two presented in [813]; it is possible that authors intended another constraint $b_1 = 0$ for such case.

where a is a parameter [814]. This function is called rootsig in [784] where the authors list a variant with $a = 1$.

There are also several other unnamed sigmoids in [814]:

$$f(z) = z\frac{\operatorname{sgn}(z)\, z - a}{z^2 - a^2}, \tag{4.38}$$

$$f(z) = \frac{az}{1 + |az|}, \tag{4.39}$$

and

$$f(z) = \frac{az}{\sqrt{1 + a^2 z^2}}. \tag{4.40}$$

4.2.2.26 *Sigmoid and tanh combinations*

Guevraa et al. proposed several activations mostly combining the logistic sigmoid, tanh, and linear function in [815]. The general approach is

$$f(z) = \begin{cases} g(z), & z \geq 0, \\ h(z), & z < 0, \end{cases}, \tag{4.41}$$

where $g(z)$ and $h(z)$ are two different AFs [815]. The authors used the following pairs $\{g(z), h(z)\}$: $\{\sigma_2(z), \tanh(z)\}$, $\{\sigma_2(z), \tanh(z)\}$, $\{\sigma_2(z), 0\}$, $\{\tanh(z), 0\}$, $\{\sigma_2(z), az\}$, and $\{\tanh(z), az\}$, where $a > 0$ is a fixed parameter and

$$\sigma_2(z) = \frac{2}{1 + \exp(-z)} - 1. \tag{4.42}$$

Guevraa et al. also proposed an AF we termed SigLU (see Section 4.2.6.52) and nonadaptive variant of PTELU.

4.2.3 *Class of sigmoid-weighted linear units*

The SiLU is the most common example of a larger class of sigmoidal units defined as

$$f(z) = z \cdot s(z), \tag{4.43}$$

where $s(z)$ is any sigmoidal function; it becomes the SiLU if the logistic sigmoid function is used. The SiLU is thus defined as

$$f(z) = z \cdot \sigma(z), \tag{4.44}$$

where $\sigma(z)$ is the logistic sigmoid [816]. The SiLU has the output range of $(-0.5, \infty)$ [11] and was first used [816] for reinforcement learning tasks such as SZ-Tetris and Tetris. The SiLU was also found to work well for the CIFAR-10/100 [243] and ImageNet [48, 817] tasks in [668]. The adaptive variant of the SiLU is called swish (see Section 4.3.3.1) [668].

For the purposes of this work, we also consider any squashing functions $s(z)$ and not necessarily only sigmoids — for example, we classify rectified hyperbolic secant (see Section 4.2.3.27) as a member of this class. We also list functions that are closely based on the SiLU and its variants.

A similar approach named weighted sigmoid gate unit (WiG) was proposed in [818], where the AF was used only for gating each of the raw inputs:

$$f(\boldsymbol{x})_i = x_i \cdot \sigma(z) = x_i \cdot \sigma(\boldsymbol{w}_i^T \boldsymbol{x} + b_i), \tag{4.45}$$

where $\boldsymbol{x}$ denotes the vector of raw inputs, $\boldsymbol{w}_i$ the weights of neuron i and b_i its bias [818]

4.2.3.1 *Gaussian error linear unit (GELU)*

Gaussian error linear unit (GELU) [819] is an activation function based on the standard Gaussian cumulative distribution function, and it weights inputs by their value rather than gating them as ReLUs do [819]. It is defined as

$$f(z) = z \cdot \Phi(z) = z \cdot \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right)\right), \tag{4.46}$$

where $\Phi(z)$ is the standard Gaussian cumulative distribution function (CDF) and $\operatorname{erf}(x)$ is the Gauss error function [819]. It is similar to the SiLU but it uses $\Phi(z)$ instead of the $\sigma(z)$. However, due to the complicated formula, the GELU can be approximated as

$$f(z) = \frac{1}{2} z \left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(z + 0.044715 z^3\right)\right)\right) \tag{4.47}$$

or

$$f(z) = z \cdot \sigma(1.702 z), \tag{4.48}$$

if the performance gains are worth the loss of exactness [819]. The function is similar to SiLU (see Section 4.2.3), it only uses Gaussian CDF $\Phi(z)$ instead of the logistic distribution CDF $\sigma(z)$ [819]. GELU was found to outperform many competitors (e.g., ReLU, ELU, SELU, continuously differentiable exponential linear unit (CELU), sigmoid, tanh) in [820]. Hendrycks and Gimpel also proposed to parameterize the GELU by μ and σ^2 — the parameters defining mean and variance of the Gaussian distribution whose CDF is used in the GELU [819], however, only the standard Gaussian distribution was used in experiments in [819]. Replacing ReLUs with GELUs led to better performance in [821]. More details about GELU are available in [820].

4.2.3.2 *Symmetrical Gaussian error linear unit (SGELU)*

A symmetric variant of GELU called symmetrical Gaussian error linear unit (SGELU) was proposed in [822]. It is defined as

$$f(z) = a \cdot z \cdot \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right), \tag{4.49}$$

where a is a fixed hyperparameter [822]. The symmetrical nature of the SGELU also leads to more symmetrically distributed weights of the neural network compared to SGELU [822]; it is believed that normal distribution of the weights can make the network more rational, accurate, and robust [822].

4.2.3.3 *Cauchy linear unit (CaLU)*

Another function related to the GELU and SiLU is the Cauchy linear unit (CaLU) [823] which uses the CDF of the standard Cauchy distribution instead of the Gaussian CDF in GELU and logistic sigmoid in SiLU. It is defined as

$$f(z) = z \cdot \Phi_{\text{Cauchy}}(z) = z \cdot \left(\frac{\tan^{-1}(z)}{\pi} + \frac{1}{2} \right), \tag{4.50}$$

where $\Phi_{\text{Cauchy}}(z)$ is the CDF of the standard Cauchy distribution [823].

4.2.3.4 *Laplace linear unit (LaLU)*

Another function related to the GELU and SiLU is the Laplace linear unit (LaLU) [823] which uses the CDF of the Laplace distribution; it is defined as

$$f(z) = z \cdot \Phi_{\text{Laplace}}(z) = z \cdot \begin{cases} 1 - \frac{1}{2}\exp(-z), & z \geq 0, \\ \frac{1}{2}\exp(z), & z < 0, \end{cases} \tag{4.51}$$

where $\Phi_{\text{Laplace}}(z)$ is the CDF of the Laplace distribution [823].

4.2.3.5 *Collapsing linear unit (LaLU)*

The Collapsing linear unit (CoLU) is an AF similar to the SiLU proposed in [824]. It is defined as

$$f(z) = z \cdot \frac{1}{1 - z\exp(-(z + \exp(z)))}. \tag{4.52}$$

4.2.3.6 *Triple-state swish*

The triple-state swish unit (TS-swish)[9] is a cascaded AF similar to TS-sigmoid (see Section 4.2.2.6) [800]; it is defined as

$$f(z) = \frac{z}{1 + \exp(-z)} \left(\frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z + a)} + \frac{1}{1 + \exp(-z + b)} \right), \tag{4.53}$$

where a and b are fixed parameters [800].

4.2.3.7 *Generalized swish*

A SiLU variant called generalized swish[10] was proposed in [800]. It is defined as

$$f(z) = z \cdot \sigma(\exp(-z)). \tag{4.54}$$

9 Koçak and Üstündağ Şiray called the function swish but it is actually based on the SiLU.
10 Also based on the SiLU instead of its adaptive variant swish.

4.2.3.8 *Exponential swish*

Another SiLU variant called exponential swish[11] was proposed in [800]. It is defined as

$$f(z) = \exp(-z)\,\sigma(z)\,. \tag{4.55}$$

4.2.3.9 *Derivative of sigmoid function*

The derivative of logistic sigmoid was used as an AF in [800]. Koçak and Üstündağ Şiray formulate the AF using the following form

$$f(z) = \exp(-z)\,(\sigma(z))^2\,. \tag{4.56}$$

4.2.3.10 *Gish*

Gish is another SiLU variant [825]; the gish is defined as

$$f(z) = z \cdot \ln(2 - \exp(-\exp(z)))\,. \tag{4.57}$$

Kaytan, Aydilek, and Yeroğlu found that gish outperformed logistic sigmoid, softplus, ReLU, LReLU, ELU, swish, mish, logish, and smish on the MNIST [45] and CIFAR-10 [243] datasets [825].

4.2.3.11 *Logish*

Logish is yet another SiLU variant [826]; it is defined as

$$f(z) = z \cdot \ln(1 + \sigma(z))\,. \tag{4.58}$$

4.2.3.12 *LogLogish*

LogLogish is a SiLU variant based on the LogLog (see Section 4.2.2.20) [823]; it is defined as

$$f(z) = z \cdot (1 - \exp(-\exp(z)))\,. \tag{4.59}$$

4.2.3.13 *ExpExpish*

ExpExpish is a SiLU variant [823]; it is defined as

$$f(z) = z \cdot \exp(-\exp(-z))\,. \tag{4.60}$$

4.2.3.14 *Self arctan*

The self arctan is an AF proposed in [799] whose formula resembles the SiLU. The self arctan is defined as

$$f(z) = z \cdot \tan^{-1}(z)\,, \tag{4.61}$$

where $\tan^{-1}(z)$ is the arctangent function [799].

11 Again, based on the SiLU instead of its adaptive variant swish.

4.2.3.15 *Parametric logish*

Zhu et al. also proposed a parametric variant of logish — we will call it parametric logish (pLogish) in this work. It is defined as

$$f(z_i) = a_i z_i \cdot \ln\left(1 + \sigma\left(b_i z_i\right)\right), \tag{4.62}$$

where a and b are fixed parameters [826]; Zhu et al. used $a = 1$ and $b = 10$ in [826].

4.2.3.16 *Phish*

Phish is a SiLU variant combining GELU and tanh [827]; it is defined as

$$f(z) = z \cdot \tanh\left(\mathrm{GELU}\left(z\right)\right). \tag{4.63}$$

The phish was found to outperform GELU, tanh, logistic sigmoid, and ReLU; it performed similarly as the mish and swish in the experiments in [827].

4.2.3.17 *Suish*

The suish [733] was proposed as an alternative to the swish AF in [828]. It is defined as

$$f(z) = \max\left(z, z \cdot \exp\left(-\left|z\right|\right)\right). \tag{4.64}$$

4.2.3.18 *Tangent-sigmoid ReLU (TSReLU)*

The tangent-sigmoid ReLU (TSReLU) [829] is an AF very similar to phish, mish, and TanhExp — it just uses the logistic sigmoid instead of the GELU in phish, softplus in mish, and the exponential in TanhExp. It is defined as

$$f(z) = z \cdot \tanh\left(\sigma\left(z\right)\right). \tag{4.65}$$

4.2.3.19 *Tangent-bipolar-sigmoid ReLU (TBSReLU)*

The tangent-bipolar-sigmoid ReLU (TBSReLU) is a variant of TSReLU proposed in [829]. It is defined as

$$f(z) = z \cdot \tanh\left(\frac{1 - \exp\left(-z\right)}{1 + \exp\left(-z\right)}\right). \tag{4.66}$$

4.2.3.20 *Log-sigmoid*

A logarithm of the logistic sigmoid is sometimes used as an activation function [738]. It is defined as

$$f(z) = \ln\left(\sigma(z)\right) = \ln\left(\frac{1}{1 + \exp(-z)}\right). \tag{4.67}$$

4.2.3.21 *Derivative of sigmoid-weighted linear unit (dSiLU)*

The derivative of sigmoid-weighted linear unit (dSiLU) can also be used as an activation function resembling a sigmoid [816]. It is defined as

$$f(z) = \sigma(z)\left(1 + z\left(1 - \sigma(z)\right)\right), \tag{4.68}$$

where $\sigma(z)$ is the logistic sigmoid [816]. The dSiLU has a maximum value of around 1.1, and the minimum is approximately -0.1 [816].

4.2.3.22 *Double sigmoid-weighted linear unit (DoubleSiLU)*

The double sigmoid-weighted linear unit (DoubleSiLU)[12] is an AF proposed in [830]. It is defined as

$$f(z) = z \cdot \frac{1}{1 + \exp\left(-z \cdot \frac{1}{1+\exp(-z)}\right)}, \tag{4.69}$$

where $\sigma(z)$ is the logistic sigmoid [830].

4.2.3.23 *Modified sigmoid-weighted linear unit (MSiLU)*

A modified sigmoid-weighted linear unit (MSiLU) is a variant of the SiLU that has faster convergence than the SiLU [831]. It is defined as

$$f(z) = z \cdot \sigma(z) + \frac{\exp\left(-z^2 - 1\right)}{4}, \tag{4.70}$$

where $\sigma(z)$ is the logistic sigmoid [831].

4.2.3.24 *Hyperbolic tangent sigmoid-weighted linear unit (TSiLU)*

Another SiLU variant is the hyperbolic tangent sigmoid-weighted linear unit (TSiLU) [830], which combines the tanh and SiLU. It is defined[13] as

$$f(z) = \frac{\exp\left(\frac{z}{1+\exp(-z)}\right) - \exp\left(-\frac{z}{1+\exp(-z)}\right)}{\exp\left(\frac{z}{1+\exp(-z)}\right) + \exp\left(\frac{z}{1+\exp(-z)}\right)}. \tag{4.71}$$

4.2.3.25 *Arctan sigmoid-weighted linear unit (ASiLU)*

Arctan sigmoid-weighted linear unit (ATSiLU) is yet another SiLU variant proposed in [830]; it is defined as

$$f(z) = \tan^{-1}\left(z \cdot \frac{1}{1 + \exp\left(-z\right)}\right). \tag{4.72}$$

12 Verma, Chug, and Singh termed the unit as DSiLU but that would collide with the dSiLU (see Section 4.2.3.21) proposed earlier by Elfwing, Uchibe, and Doya.

13 The formula in [830] was wrong as it evaluated to $\frac{2x}{0}$, we present the formula we think authors intented.

4.2.3.26 *SwAT*

Verma, Chug, and Singh proposed an AF named SwAT combining the SiLU and arctan in[830]. This function is defined as

$$f(z) = z \cdot \frac{1}{1 + \exp\left(-\tan^{-1} |left(z)\right)}. \tag{4.73}$$

4.2.3.27 *Rectified hyperbolic secant*

A rectified hyperbolic secant activation function was proposed in [832]. This function is totally differentiable, symmetric about the origin, and is approaching zero for inputs going to positive or negative infinity:

$$f(z) = z \cdot \operatorname{sech}(z), \tag{4.74}$$

where $\operatorname{sech}(z)$ is the hyperbolic secant function [832].

4.2.3.28 *Linearly scaled hyperbolic tangent (LiSHT)*

A linearly scaled hyperbolic tangent (LiSHT) activation function was proposed in [833] to address the problem of vanishing gradients and the non-utilization of large negative input values. The LiSHT function is defined as

$$f(z) = z \cdot \tanh(z). \tag{4.75}$$

The output range of LiSHT function is $[0, \infty]$ [11].The output of LiSHT is close to the ReLU (see Section 4.2.6) and swish for large positive values [833]; however, unlike the aforementioned AFs, the output is symmetric, and, therefore, it behaves identically for large negative values. While the LiSHT is symmetric, the fact that its output is unbounded and non-negative could be considered a disadvantage [11]. The effectiveness of the LiSHT activation function was tested on several different architectures ranging from multilayer perceptron (MLP) and residual neural networks to LSTM-based networks and on various tasks — the Iris dataset, the MNIST [45], CIFAR-10 and CIFAR-100 [243] and the *sentiment140* dataset from Twitter [834, 835] for sentiment analysis [833].

A parametric version of LiSHT named SoftModulusT (see Section 4.2.6.31) was proposed in [836].

4.2.3.29 *Mish*

A popular activation function mish[837] is a combination of the tanh and softplus activation function; the function resembles swish activation (see Section 4.3.3.1). It is defined as

$$f(z) = z \cdot \tanh\left(\operatorname{softplus}(z)\right) = z \cdot \tanh\left(\ln\left(1 + \exp\left(z\right)\right)\right). \tag{4.76}$$

Mish was found to outperform swish; it performed similarly to $f(z) = z \cdot \ln\left(1 + \tanh\left(\exp\left(z\right)\right)\right)$ but this activation function was found to often lead to unstable training [837]. The mish was found to outperform swish and ReLU for many architectures such as various ResNet architectures [13], Inception v3 [52], DenseNet-121 [838], and others [837]. Detailed comparison with other activation functions was run using the Squeeze Net [839] where it outperformed swish, GELU, ReLU, ELU, LReLU, SELU, softplus, S-shaped ReLU (SReLU) , inverse square root unit (ISRU), tanh, and randomized leaky ReLU (RReLU) [837]. The mish activation function was, for example, used in the YOLOv4 [840] and its variant Scaled-YOLOv4 [841].

4.2.3.30 *Smish*

The smish [842] is a variant of the mish where the exponential function is replaced by the logistic sigmoid. It is, therefore, defined as

$$f(z) = az \cdot \tanh\left(\ln\left(1+\sigma\left(bz\right)\right)\right), \tag{4.77}$$

where a and b are parameters [842]; however, Wang, Ren, and Wang recommend $a = 1$ and $b = 1$ based on a small parameter search in [842].

4.2.3.31 *TanhExp*

Similarly as the mish is the combination of tanh and softplus, the TanhExp [843] is a combination of tanh and the exponential function [843, 844]. It is defined as

$$f(z) = z \cdot \tanh\left(\exp(z)\right). \tag{4.78}$$

4.2.3.32 *Serf*

The serf is an AF similar to the mish; however, it uses the error function instead of the tanh [845]. It is defined as

$$f(z) = z\,\mathrm{erf}\left(\ln\left(1+\exp(z)\right)\right), \tag{4.79}$$

where erf is the Gauss error function [845]. It was found to outperform mish, GELU, and ReLU for various architectures on Multi30K [846], ImageNet [48, 817], the CIFAR-10, and CIFAR-100 [243] datasets; see [845] for details.

4.2.3.33 *Efficient asymmetric nonlinear activation function (EANAF)*

An activation function combining tanh and softplus called efficient asymmetric nonlinear activation function (EANAF) was proposed in [847]. The function is defined as

$$f(z) = z \cdot \mathrm{g}\left(\mathrm{h}\left(z\right)\right), \tag{4.80}$$

where $\mathrm{h}(z)$ is the softplus function and $\mathrm{g}(z) = \tanh\left(\frac{z}{2}\right)$, which can be simplified to

$$f(z) = \frac{z \cdot \exp\left(z\right)}{\exp\left(z\right)+2}. \tag{4.81}$$

The EANAF is continuously differentiable. The EANAF is very similar to swish with similar amount of computation but Chai et al. found that it performs better than swish and several other activation functions in RetinaNet [848] and YOLOv4 [841] architectures on object detection tasks [847].

4.2.3.34 *SinSig*

SinSig [849] is a self-gated non-monotonic activation function defined as

$$f(z) = z \cdot \sin\left(\frac{\pi}{2}\sigma(z)\right), \tag{4.82}$$

where $\sigma(z)$ is the logistic sigmoid function [849]. While SinSig is similar to swish and mish, it outperformed them in experiments in [849] as the number of layers in a neural network increased. It was also shown that the SinSig converges faster. The SinSig outperformed ReLU and mish on several deep architectures including ResNet 20 v2 [850], ResNet 110 v2 [850], SqueezeNet [851], and ShuffleNet [852] among others on the CIFAR-100 task [243] in experiments in [849].

4.2.3.35 *Gaussian error linear unit with sigmoid activation function (SiELU)*

The with sigmoid activation function (SiELU) was proposed in [853]; it is defined as

$$f(z) = z\sigma\left(2\sqrt{\frac{2}{\pi}}\left(z + 0.044715z^3\right)\right). \tag{4.83}$$

4.2.4 *Gated linear unit (GLU)*

A gated activation called gated linear unit (GLU) similar to SiLU (see Section 4.2.3) for use in RNNs was proposed in [854]. The GLU is defined as

$$f(z, z') = z \otimes \sigma(z'), \tag{4.84}$$

where $\otimes$ is the element-wise product and z and z' are two learned linear transformations of input vector $\boldsymbol{x}$ [855, 856].

4.2.4.1 *Gated tanh unit (GTU)*

A gated activation called gated tanh unit (GTU) similar to GLU (see Section 4.2.4) for use in RNNs was proposed in [857]. The GTU is defined as

$$f(z, z') = \tanh(z) \otimes \sigma(z'), \tag{4.85}$$

where $\otimes$ is the element-wise product and z and z' are two learned linear transformations of input vector $\boldsymbol{x}$ [855].

4.2.4.2 *Gated ReLU (ReGLU)*

Another GLU extension is the gated (ReGLU) [854, 855]. The ReGLU is defined as

$$f(z, z') = z \otimes \text{ReLU}(z'), \tag{4.86}$$

where $\otimes$ is the element-wise product and z and z' are two learned linear transformations of input vector $\boldsymbol{x}$ [855].

4.2.4.3 *Gated GELU (GEGLU)*

A GELU-based GLU extension is the gated (GEGLU) [855]; it is defined as

$$f(z, z') = z \otimes \text{GELU}(z'), \tag{4.87}$$

where $\otimes$ is the element-wise product and z and z' are two learned linear transformations of input vector $\boldsymbol{x}$ [855].

4.2.4.4 *Swish GELU (SwiGLU)*

A swish-based GLU extension is the gated swish (SwiGLU) [855]; it is defined as

$$f(z, z') = z \otimes \text{swish}(z'), \tag{4.88}$$

where $\otimes$ is the element-wise product, z and z' are two learned linear transformations of input vector $\boldsymbol{x}$, and swish is the swish with its own trainable parameter [855].

4.2.5 *Softmax*

The softmax is not a usual type of AF taking in a single value, but it takes all the output value of the unit i and, also, the output values of other units in order to compute a soft argmax of the values. It is defined as

$$f(z_j) = \frac{\exp(z_j)}{\sum_{k=1}^{N} \exp(z_k)}, \tag{4.89}$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons [858, 859].

4.2.5.1 *β-softmax*

The β-softmax is a softmax extension proposed in [860]; it is defined as

$$f(z_j) = \frac{\int \exp(bz_j)}{\sum_{k=1}^{N} \int \exp(bz_k)}, \tag{4.90}$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons and b takes random value from $\mathbb{N}^{+}$[14][860].

14 No further specification was provided in [860].

4.2.6 *Rectified linear function (ReLU)*

The rectified linear unit (ReLU) [861] is widely regarded as the most popular activation function in modern feedforward networks [46, 122, 862] due to its simplicity and improved performance [11]. It has been observed that ReLUs can significantly expedite the convergence of stochastic gradient descent [49]. Additionally, traditional ReLUs are computationally less expensive compared to activation functions like the logistic or tanh functions [122]. ReLUs often outperform sigmoidal activation functions [862]. However, a drawback of ReLUs is the potential for neurons to become "dead" or "disabled" during training. This means that they may never activate again for any input, resulting in a permanently zero output gradient [122]. This issue can occur after a weight update when a large gradient flows through the unit [122]. This might happen after a weight update after a large gradient flows through the unit [122]. However, ReLUs often lead to faster convergence than for sigmoid activation, as shown in [863]. It can also be shown that ReLUs and rational function efficiently approximate each other [864]. The ReLU was used as an example of the more general class of piecewise affine AFs for neural network verification[15] using theorem provers in [865].

A ReLU is mathematically defined as the maximum of zero and the input value [862, 866]:

$$f(z) = \max(0, z) . \tag{4.91}$$

ReLU is commonly recommended as the default choice for feedforward networks due to its usually superior performance compared to sigmoidal functions and its computational efficiency [122]; furthermore, it works comparably to its modifications [866]. Many popular NN models utilize ReLU as the activation function of choice, e.g., [49, 867].

Many ReLU modification and derivations were proposed [866, 868] — e.g. leaky ReLU (LReLU) [869], very leaky ReLU (VLReLU) [870], parametric ReLU [871], randomized leaky ReLU (RReLU) [872] or S-shaped ReLU [873]. Smoothed modifications are, for example, exponential linear unit [874] and softplus [862]. Most of the modifications solve the problem of dying out neurons as they allow for gradient flows for any input.

4.2.6.1 *Shifted ReLU*

A Shifted ReLU [874] is a simple translation of a ReLU and is defined as

$$f(z) = \max(-1, z) . \tag{4.92}$$

4.2.6.2 *Leaky ReLU (LReLU)*

Leaky ReLU (LReLU) [869] is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{z}{a}, & z < 0, \end{cases} \tag{4.93}$$

15 More details are out of the scope of this work, see [865] for more details.

where $a \in (1, \infty)$ is set to large number;[16] the recommended setting from [869] is $a = 100$.

LReLU solves the problem of dying neurons when neurons have permanently zero output gradient in classical ReLU by "leaking" the information for $z < 0$ instead of outputting exact zero. Both ReLU and LReLU can be considered to be a special case of the maxout unit (see Section 4.3.46) [11]. A theoretical analysis of the ReLU and LReLU is available in [875].

Very leaky ReLU (VLReLU) [870] is almost identical to the LReLU but has much higher slope when the z is negative for faster training [870] by setting $a_i = 3$. While it can be considered as a special case of LReLU, some researchers consider it as a separate case, e.g., [868].

The so-called optimized leaky ReLU (OLReLU) [876] propose another reformulation of LReLU and calculation of the slope parameter a that is inspired by the RReLU (see Section 4.2.6.3):

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \exp(-a), & z < 0, \end{cases} \tag{4.94}$$

where

$$a = \frac{u + l}{u - l}, \tag{4.95}$$

where u and l are hyperparameters of the bounds of the RReLU [876].

4.2.6.3 *Randomized leaky ReLU (RReLU)*

RReLU is a leaky ReLU where the leakiness is stochastic during the training [872], i.e.:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \frac{z}{a_i}, & z_i < 0, \end{cases} \tag{4.96}$$

where a_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(l, u)$ where $l < u$ and $l, u \in (0, \infty)$ [872]. Similarly as in the dropout approach [143], an average over all a_i over is taken during inference phase — the a_i is set to $\frac{l+u}{2}$:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \frac{z}{\frac{l+u}{2}}, & z_i < 0. \end{cases} \tag{4.97}$$

Recommended distribution is $U(3, 8)$ for sampling the a_i [872].

16 Depending on the source, researchers use either this form $\frac{z}{a}$ or the inverted form az for the negative inputs.

4.2.6.4 *Softsign randomized leaky ReLU (S-RReLU)*

The softsign randomized leaky ReLU (S-RReLU)[17] is a RReLU combined with the softsign proposed in [877, 878]. It is defined as

$$f(z_i) = \begin{cases} \frac{1}{(1+z_i)^2} + z_i, & z_i \geq 0, \\ \frac{1}{(1+z_i)^2} + a_i z_i, & z_i < 0, \end{cases} \tag{4.98}$$

where a_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(l, u)$ where $l < u$ and $l, u \in (0, \infty)$ [877]. Elakkiya and Dejey used $l = \frac{1}{8}$ and $u = \frac{1}{3}$ [877].

4.2.6.5 *Sloped ReLU (SlReLU)*

A Sloped ReLU (SlReLU) [879] is similar to the LReLU — whereas the LReLU parameterizes the slope for negative inputs, the SlReLU parameterizes the slope of ReLU for positive inputs. It is, therefore, defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ 0, & z < 0, \end{cases} \tag{4.99}$$

where a is a fixed, predetermined parameter [879]. Seo, Lee, and Kim recommended $a \in [1, 10]$ based on their experiments in [879].

4.2.6.6 *Noisy ReLU (NReLU)*

A stochastic variant of the ReLU called noisy ReLU (NReLU) was proposed in [861]:

$$f(z) = \max(0, z + a), \tag{4.100}$$

where a is a stochastic parameter $a \sim \mathrm{N}(0, \sigma(z))$, $N(0, \sigma^2)$ is the Gaussian distribution with zero mean and variance σ^2 and $\sigma(z)$ is the standard deviation of the inputs z. The NReLU was designed for use with Restricted Boltzmann machines [861]. More details about the NReLU is available in [861].

4.2.6.7 *SineReLU*

The SineReLU [880, 881] is a ReLU based activation that uses trigonometric functions for negative inputs. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a(\sin(z) - \cos(z)), & z < 0, \end{cases} \tag{4.101}$$

where a is a fixed parameter [880, 881].

17 Elakkiya and Dejey used S-RReLU as a name and not an abbreviation; however, since S-RReLU is a combination of the softsign and RReLU, we feel that using it as an abbreviation is appropriate.

4.2.6.8 *Minsin*

The minsin is a ReLU-based AF used in [733]. It is defined as

$$f(z) = \min\left(z, \sin\left(z\right)\right) = \begin{cases} \sin(z), & z \geq 0, \\ z, & z < 0. \end{cases} \tag{4.102}$$

4.2.6.9 *Variational linear unit (VLU)*

The variational linear unit (VLU) is an AF combining the ReLU and sine functions proposed in [880]. It is defined as

$$f(z) = \mathrm{ReLU}\left(z\right) + a\sin\left(bz\right) = \max\left(0, z\right) + a\sin\left(bz\right), \tag{4.103}$$

where a and b are fixed parameters [880].

4.2.6.10 *Spatial context-aware activation (SCAA)*

The spatial context-aware activation (SCAA) is a ReLU extension proposed in [882]. The ReLU performs an element-wise max operation on the feature map $\mathbf{X}$:

$$\mathrm{ReLU}\left(\mathbf{X}\right) = \max\left(\mathbf{X}, \mathbf{0}\right), \tag{4.104}$$

where $\mathrm{ReLU}\left(\mathbf{X}\right)$ is the ReLU in the matrix notation and $\mathbf{0}$ is a matrix of zeroes with the same shape as $\mathbf{X}$ [882]. The SCAA first applies a depth-wise convolution on $\mathbf{X}$ to produce spatial context aggregated feature map denoted $\mathbf{f}_{\mathrm{DW}}\left(\mathbf{X}\right)$ and then proceeds with the elementwise max operation [882]; the SCAA is, therefore, defined as

$$\mathbf{f}\left(\mathbf{X}\right) = \max\left(\mathbf{X}, \mathbf{f}_{\mathrm{DW}}\left(\mathbf{X}\right)\right). \tag{4.105}$$

4.2.6.11 *Randomly translational ReLU (RT-ReLU)*

A randomly translational ReLU (RT–ReLU) is a ReLU with a randomly added jitter during each iteration of the training process [883]. It is defined as

$$f(z_i) = \begin{cases} z_i + a_i, & z_i + a_i \geq 0, \\ 0, & z_i + a_i < 0, \end{cases} \tag{4.106}$$

where a_i is stochastic parameter for each neuron i randomly sampled from the Gaussian distribution at each iteration, $a_i \sim \mathrm{N}\left(0, \sigma^2\right)$, where σ^2 is the variance of the Gaussian distribution. The authors Cao et al. set the $\sigma^2 = 0.75^2$ for their experiments [883]. The a_i is set to 0 during the test phase [11].

4.2.6.12 *Natural-Logarithm-ReLU (NLReLU)*

The natural-logarithm-ReLU (NLReLU) introduces non-linearity to ReLU similarly as rectified linear tanh (ReLTanh) (see Section 4.3.1.36) but only for positive part of the activation function [11]:

$$f(z) = \ln\left(a \cdot \max\left(0, z\right) + 1\right), \tag{4.107}$$

where a is a predefined constant [884].

4.2.6.13 *Softplus linear unit (SLU)*

An activation function softplus linear unit (SLU) combining the ReLU with the softplus activation function was proposed in [885]; the function is based around the assumption that zero mean activations improve learning performance [885]. The SLU is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ b \ln(\exp(z) + 1) - c, & z < 0, \end{cases} \tag{4.108}$$

where a_i, b_i, and c_i are predefined parameters; however, to ensure that the function is continuous, differentiable at zero and to avoid vanishing or exploding gradients, its parameters are set to $a = 1$, $b = 2$, and $c = 2\ln(2)$ [885]. The SLU is therefore equal to

$$f(z) = \begin{cases} z, & z \geq 0, \\ 2\ln\frac{\exp(z)+1}{2}, & z < 0. \end{cases} \tag{4.109}$$

4.2.6.14 *Rectified softplus (ReSP)*

Another activation function combining ReLU and softplus called rectified softplus (ReSP) [11] was proposed in [886]. The function is defined as

$$f(z) = \begin{cases} az + \ln(2), & z \geq 0, \\ \ln(1 + \exp(z)), & z < 0, \end{cases} \tag{4.110}$$

where a is a fixed hyperparameter controlling the slope [886]. Larger values of a between 1.4 and 2.0 were found to work well [886].

4.2.6.15 *Parametric rectified non-linear unit (PReNU)*

A ReLU variant called parametric rectified non-linear unit (PReNU) [887] replaces the linear part of the ReLU for positive inputs by a non-linear function similarly to RePU (see Section 4.2.6.39). It is defined as

$$f(z) = \begin{cases} z - a \cdot \ln(z + 1), & z \geq 0, \\ 0, & z < 0, \end{cases} \tag{4.111}$$

where a is a fixed hyperparameter [887] — however, this parameter could be adaptive similarly as in PReLU (see Section 4.3.1.1) that PReNU extends since Jaafari, Ellahyani, and Charfi thought of the PReLU as non-adaptive function for some reason [887].

4.2.6.16 *Bounded ReLU (BReLU)*

A BReLU [888] is a variant of ReLU that limits the output as the unlimited output of the original ReLU might lead to an instability [11]. It is defined as

$$f(z) = \min\left(\max\left(0, z\right), a\right) = \begin{cases} 0, & z \leq 0, \\ z, & 0 < z < a, \\ a, & z > a, \end{cases} \tag{4.112}$$

where a is a predefined parameter [888]. The BReLU appeared later in the literature under the name *ReLUN* in [889], where it seems that it was independently proposed.

4.2.6.17 *Hard sigmoid*

A Hard sigmoid is very similar to BReLU; it is a very crude approximation of the logistic sigmoid and is commonly defined [651, 890] as

$$f(z) = \max\left(0, \min\left(\frac{z+1}{2}, 1\right)\right). \tag{4.113}$$

Other definitions are sometimes used; e.g. variant from [891] is defined as

$$f(z) = \max\left(0, \min\left(0.2z + 0.5, 1\right)\right). \tag{4.114}$$

While the Hard sigmoid is not as commonly used as the logistic sigmoid, it can be used, for example, in binarized neural network with stochastic activation functions [890] — the binaryized neural networks can lead to much faster inference than regular neural networks, e.g., Courbariaux et al. reached up to $7\times$ speed up without any loss in classification accuracy [890] (however, even better speed-ups can be obtained using, for example, FPGA implementations as in [892]).

4.2.6.18 *HardTanh*

The HardTanh is another piecewise linear function; it is very similar to Hard sigmoid, but it approximates the tanh instead of the logistic sigmoid. It is defined as

$$f(z) = \begin{cases} a, & z < a, \\ z, & a \leq z \leq b, \\ b, & z > b, \end{cases} \tag{4.115}$$

where a and b are fixed parameters [893]; Liu et al. used $a = -1$ and $b = 11$ in [894]. NNs with HardTanhs are more suitable for linear predictive control than NNs with ReLUs as they usually require less hidden layers and neurons for representing identical min-max maps [893].

4.2.6.19 *Shifted HardTanh*

Kim et al. proposed HardTanh variants with vertical and horizontal shifts in [895]. The SvHardTanh[18] is defined as

$$f(z) = \begin{cases} -1 + a, & z < -1, \\ z + a, & -1 \leq z \leq 1, \\ 1 + a, & z > 1, \end{cases} \tag{4.116}$$

where a is a fixed parameter [895]. Kim et al. used HardTanh variant with thresholds -1 and 1; a more general variant with parametric thresholds from Eq. (4.115) could be defined similarly.

The SvHardTanh is defined as

$$f(z) = \begin{cases} -1 + a, & z < -1, \\ z + a, & -1 \leq z \leq 1, \\ 1 + a, & z > 1, \end{cases} \tag{4.117}$$

where a is a fixed parameter [895].

The ShHardTanh is defined as

$$f(z) = \begin{cases} -1, & z < -1 - a, \\ z, & -1 - a \leq z \leq 1 - a, \\ 1, & z > 1 - a, \end{cases} \tag{4.118}$$

where a is a fixed parameter [895].

Kim et al. used HardTanh variant with thresholds -1 and 1; more general variants of SvHardTanh and ShHardTanh with parametric thresholds from Eq. (4.115) could be defined similarly.

4.2.6.20 *Hard swish*

A linearized variant of the swish AF (see Section 4.3.3.1) was proposed in [896]. It is defined as

$$f(z) = z \cdot \begin{cases} 0, & z \leq -3, \\ 1, & z \geq 3, \\ \frac{z}{6} + \frac{1}{2}, & -3 < z < 3. \end{cases} \tag{4.119}$$

The linearization allows for more efficient computation [896].

18 Both SvHardTanh and ShHardTanh are named using the same convention as shifted ELUs (see Section 4.3.1.56) for the purposes of this work.

4.2.6.21 *Truncated rectified (TRec) activation function*

The truncated rectified (TRec) AF is a truncated variant of the ReLU [897]. It resembles onesided variant of the Hardshrink (see Section 4.2.6.22) — it is defined as

$$f(z) = \begin{cases} z, & z > a, \\ 0, & z \leq a, \end{cases} \tag{4.120}$$

where a is a fixed parameter. Konda, Memisevic, and Krueger used $a = 1$ for most of their experiments [897].

4.2.6.22 *Hardshrink*

The Hardshrink [703, 897–899] (named *thresholded linear* AF in [897][19]) is very similar to Hard sigmoid, TRec, and other piecewise linear functions; it is defined as

$$f(z) = \begin{cases} z, & z > a, \\ 0, & -a \leq z \leq a, \\ z, & z < -a, \end{cases} \tag{4.121}$$

where $a > 0$ is a fixed parameter.

4.2.6.23 *Softshrink*

The Softshrink is an AF similar to the Hardshrink used in [661, 900]. It is defined as

$$f(z) = \begin{cases} z - a, & z > a, \\ 0, & -a \leq z \leq a, \\ z + a, & z < -a, \end{cases} \tag{4.122}$$

where $a > 0$ is a fixed thresholding parameter [900].

4.2.6.24 *Bounded leaky ReLU (BLReLU)*

Similarly as the BReLU is a bounded variant of the ReLU, the bounded leaky ReLU (BLReLU) is a bounded variant of LReLU (see Section 4.2.6.2) [888]. It is defined as

$$f(z) = \begin{cases} az, & z \leq 0, \\ z, & 0 < z < b, \\ az + c, & z > b, \end{cases} \tag{4.123}$$

19 Konda, Memisevic, and Krueger proposed it as a novel AF but it was already proposed in [898].

where a and b are predefined parameters and c is computed such that $b = ab + c$ [888], i.e. $c = (1-a)b$. The parameter a controls the leakiness, the parameter b is the threshold of saturation, and c is computed such that the function is continuous.

4.2.6.25 *V-shaped ReLU (vReLU)*

A V-shaped variant of ReLU called V-shaped ReLU (vReLU) is proposed in [901, 902] and tackles the problem of dying neurons that is present with ReLUs [901]. The vReLU is identical to the absolute value function and is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ -z, & z < 0. \end{cases} \tag{4.124}$$

The output range of vReLU is $[0, \infty)$ [11]. The *modulus* activation function later proposed in the literature by Vallés-Pérez et al. in [836] is identical to the vReLU. The absolute value function was used as an AF also in [903].

4.2.6.26 *Pan function*

The pan function is an AF similar to the vReLU and Softshrink [904, 905]. It is defined as

$$f(z) = \begin{cases} z - a, & z \geq a, \\ 0, & -a < z < a, \\ -z - a, & z \leq -a, \end{cases} \tag{4.125}$$

where a is a fixed boundary parameter [904].

4.2.6.27 *Absolute linear unit (AbsLU)*

The absolute linear unit (AbsLU) [906] is a ReLU-based AF similar to the vReLU. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a \cdot |z|, & z < 0, \end{cases} \tag{4.126}$$

where $a \in [0, 1]$ is a fixed hyperparameter [906].

4.2.6.28 *Mirrorer rectified linear unit (mReLU)*

The mirrored rectified linear unit (mReLU) is a bounded AF that suppresses the output for unusual inputs [907]. It is defined as

$$f(z) = \min\left(\mathrm{ReLU}\left(1 - z\right), \mathrm{ReLU}\left(1 + z\right)\right) = \begin{cases} 1 + z, & -1 \leq z \leq 0, \\ 1 - z, & 0 < z \leq 1, \\ 0, & \text{otherwise}\,. \end{cases} \tag{4.127}$$

4.2.6.29 *Leaky single-peaked triangle linear unit (LSPTLU)*

An AF similar to vReLU, AbsLU, and tent activation named leaky single-peaked triangle linear unit (LSPTLU) was proposed in [908]. It is defined as

$$f(z) = \begin{cases} 0.2z, & z < 0, \\ z, & 0 \leq z \leq a, \\ 2a - z, & a < z \leq 2a, \\ 0, & z \geq 2a, \end{cases} \tag{4.128}$$

where a is a fixed parameter [908]. An identical AF was proposed under the name leaky rectified triangle linear unit (LRTLU) in [909].

4.2.6.30 *SoftModulusQ*

The SoftModulusQ is a quadratic approximation of the vReLU proposed in [836]. The SoftModulusQ is defined as

$$f(z) = \begin{cases} z^2 \left(2 - |z|\right), & |z| \geq 1, \\ |z|, & |z| > 1. \end{cases} \tag{4.129}$$

4.2.6.31 *SoftModulusT*

While the SoftModulusQ (see Section 4.2.6.30) is a quadratic approximation of the vReLU (see Section 4.2.6.25), the SoftModulusT [836] is a tanh based approximation of the vReLU. It is basically a parametric version of the LiSHT activation function (see Section 4.2.3.28):

$$f(z) = z \cdot \tanh\left(\frac{z}{a}\right), \tag{4.130}$$

where a is a predetermined parameter; the authors Vallés-Pérez et al. used $a = 0.01$ in their experiments [836]. When $a = 1$, the SoftModulusT becames the LiSHT activation function.

4.2.6.32 *SignReLU*

The combination of ReLU and softsign resulted in SignReLU [910] that improves the convergence rate and alleviates the vanishing gradient problem [910]. The SignReLU is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a\frac{z}{|z|+1}, & z < 0, \end{cases} \tag{4.131}$$

where a is a fixed parameter [910, 911]; the SignReLU becomes ReLU for $a = 0$. The SignReLU was independently proposed under the name DLU in [912];[20] this name is sometimes used in the literature — e.g., [904].

20 [912] is a preprint of [911].

4.2.6.33 *Li-ReLU*

Elakkiya and Dejey proposed a combination of a linear function and the ReLU in [877]; they named the function Li-ReLU[21] and it is defined as

$$f(z) = \begin{cases} az + z, & z_i \geq 0, \\ az, & z_i < 0, \end{cases} \tag{4.132}$$

where a_i is a fixed parameter [877].

4.2.6.34 *Concatenated ReLU (CReLU)*

A concatenated ReLU (CReLU) is an adaptation of the ReLU function proposed based on the observation that filters in CNNs in the lower layers form pairs consisting of filters with opposite phase [913]. The CReLU conserves both negative and positive linear responses after convolution by concatenating the output of two ReLUs (hence the name) [913]. The CReLU is a function $\mathbb{R} \to \mathbb{R}^2$ and is defined as [913]

$$f(z) = \begin{bmatrix} \mathrm{ReLU(z)} \\ \mathrm{ReLU(-z)} \end{bmatrix}, \tag{4.133}$$

with the output range of $[0, \infty)$ for both output elements [11].

4.2.6.35 *Negative CReLU (NCReLU)*

A CReLU extension named negative CReLU (NCReLU) was proposed in [914]; while it is very similar to CReLU, it multiplies the second element by -1:

$$f(z) = \begin{bmatrix} \mathrm{ReLU(z)} \\ -\mathrm{ReLU(-z)} \end{bmatrix}. \tag{4.134}$$

Very similar AF was proposed concurrently in [915] under the name bipolar activation function (BAF). Unlike the NCReLU, it does not produce a vector output but is applied in an alternating manner similar to All-ReLU (see Section 4.3.33) but for neurons instead of layers. It is defined for the i-th neuron as

$$f(z_i) = \begin{cases} g\left(z_i\right), & i\%2 = 0, \\ -g\left(-z_i\right), & i\%2 = 1, \end{cases} \tag{4.135}$$

where $g\left(z_i\right)$ is any ReLU family AF and % is the modulo operation.

21 Not an abbreviation.

4.2.6.36 *DualReLU*

Where CReLU activation functions takes a single value and outputs a vector of two values, the DualReLU [916] takes two values as an input and outputs a single value. The DualReLU is a two-dimensional activation function meant as a replacement of the tanh activation function for Quasi-Recurrent neural networks [916]. It is defined as

$$f(z,z') = \max(0,z) - \max(0,z') = \begin{cases} 0, & z \leq 0 \wedge z' \leq 0, \\ z, & z > 0 \wedge z' \geq 0, \\ -b, & z \leq 0 \wedge z' > 0, \\ a-b, & z > 0 \wedge z' > 0. \end{cases} \quad (4.136)$$

4.2.6.37 *Orthogonal permutation liner unit*

The orthogonal permutation liner unit (OPLU) is not applied to a single neuron but always to a pair of neurons [917]. First, the neurons are grouped into pairs of neurons $\{i,j\}$ and the OPLU takes two inputs z_i and z_j of neurons i and j and produces the output

$$f(z_i,z_j) = \max(z_i,z_j) \quad (4.137)$$

for neuron i and

$$f(z_i,z_j) = \min(z_i,z_j) \quad (4.138)$$

for neuron j [917].

4.2.6.38 *Elastic ReLU (EReLU)*

Another extension is the elastic ReLU (EReLU), which slightly randomly changes the slope of the positive part of the ReLU during the training [918]. The EReLU is defined as

$$f(z_i) = \begin{cases} k_i z_i, & z_i \geq 0, \\ 0, & z_i < 0, \end{cases} \quad (4.139)$$

where k_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(1-\alpha, 1+\alpha)$ where $\alpha \in (0,1)$ is a parameter controlling the degree of response fluctuations [918]. The EReLU thus complements the principle of RReLU, which randomly changes the leakiness during the training while keeping the positive part fixed, while the EReLU changes the positive part and keeps the output constantly zero for negative inputs. The EReLU sets the k_i its expected value $\mathrm{E}(k_i)$ which is equal to one — the EReLU becomes the ReLU during the test phase [918].

4.2.6.39 *Power activation functions & rectified power units (RePU)*

A power activation function extending ReLU together with a training scheme for better generalization was proposed in [919]. This activation function was later independently proposed under the name RePU in [920]. The RePU is defined as

$$f(z) = \begin{cases} z^a, & z \geq 0, \\ 0, & z < 0, \end{cases} \tag{4.140}$$

where a is a fixed parameter [919, 921]. The RePU is a generalization of several activation functions — it becomes the Heaviside step function for $a = 0$ and ReLU for $a = 1$; the case $a = 2$ is called *rectified quadratic unit* (ReQU) in [920] and *squared ReLU* in [922]; finally, the case $a = 3$ is called *rectified cubic unit* (ReCU) [920]. The disadvantage of RePU is its unbounded and asymmetric nature and that it is prone to vanishing gradient [11]. Theoretical analysis of the RePU is available in [921].

However, Berradi recommends alternating using $a = b$ and $a = \frac{1}{b}$ each epoch; i.e.:

$$f_1(z) = \begin{cases} z^b, & z \geq 0, \\ 0, & z < 0, \end{cases} \tag{4.141}$$

and

$$f_2(z) = \begin{cases} z^{\frac{1}{b}}, & z \geq 0, \\ 0, & z < 0. \end{cases} \tag{4.142}$$

Then the activation function $f_1(z)$ is used during odd epochs and $f_2(z)$ during even epochs; their mean is used during the test phase [919]. The value $b > 1$ was used in the experiments in [919] - $b \in \{1.05, 1.1, 1.15, 1.20, 1.25\}$.

4.2.6.40 *Approximate ReLU (AppReLU)*

The approximate ReLU (AppReLU)[22] [923, 924] is the RePU with aditional scaling parameter; it is defined as

$$f(z) = \begin{cases} az^b, & z \geq 0, \\ 0, & z < 0. \end{cases} \tag{4.143}$$

22 Saha et al. used the abbreviation AReLU but this is already used for the Attention-based ReLU in this work.

4.2.6.41 *Power linear activation function (PLAF)*

The power linear activation function (PLAF)[23] is a class of two similar AFs proposed in [925]. The first, even power linear activation function (EPLAF), is defined as

$$f(z) = \begin{cases} z - \left(1 - \frac{1}{d}\right), & z \geq 1, \\ \frac{1}{d} \left|z\right|^{d}, & -1 \leq z < 1, \\ -z - \left(1 - \frac{1}{d}\right), & z < -1, \end{cases} \tag{4.144}$$

where d is a fixed parameter [925]. Similarly, the second AF — odd power linear activation function (OPLAF) — is defined as

$$f(z) = \begin{cases} z - \left(1 - \frac{1}{d}\right), & z \geq 1, \\ \frac{1}{d} \left|z\right|^{d}, & 0 \leq z < 1, \\ -\frac{1}{d} \left|z\right|^{d}, & -1 \leq z < 0, \\ -z - \left(1 - \frac{1}{d}\right), & z < -1, \end{cases} \tag{4.145}$$

where d is a fixed parameter [925]. Nasiri and Ghiasi-Shirazi focused on the EPLAF in their work [925] and showed that EPLAF with $d = 2$ performed similarly as the ReLU for some of the tasks but it performed significantly better for other tasks; the OPLAF was not experimentally validated in [925].

4.2.6.42 *Average biased ReLU (ABReLU)*

Similarly as the RT–ReLU (see Section 4.2.6.11), the average biased ReLU (ABReLU) [926] uses horizontal shifting in order to handle negative values [11]. It is defined as

$$f(z_i) = \begin{cases} z_i - a_i, & z_i - a_i \geq 0, \\ 0, & z_i - a_i < 0, \end{cases} \tag{4.146}$$

where a_i is the average of input activation map to the neuron/filter i [11, 926], which makes the function data dependent and adjusts the threshold based on the positive and negative data dominance [926]. The output range is $[0, \infty)$ [11].

4.2.6.43 *Delay ReLU (DRLU)*

The delay ReLU (DRLU)[24] is a function that also adds a horizontal shift to the ReLU [927]; however, the DRLU uses a fixed, predetermined shift whereas RT–ReLU uses stochastic shifts (see Section 4.2.6.11) and ABReLU computes

23 Originally, Nasiri and Ghiasi-Shirazi named PLAF as *PowerLinear* AF. Also, its variants EPLAF and OPLAF were named as *EvenPowLin* and *OddPowLin* in [925].

24 Authors termed the function DRLU; however, the usual notation in this work would be DReLU. Since such notation would collide with the dynamic ReLU, we will use the original notation from [927] despite the inconsistency.

the shift as the average of input activation map (see Section 4.2.6.42). The DRLU is defined as

$$f(z) = \begin{cases} z - a, & z - a \geq 0, \\ 0, & z - a < 0, \end{cases} \tag{4.147}$$

where a is a fixed, predetermined parameter [927]. Shan, Li, and Chen also add a constraint $a > 0$ [927] and they used $a \in \{0.06, 0.08, 0.10\}$ in their experiments [927].

4.2.6.44 *Displaced ReLU (DisReLU)*

Very similar to the flexible ReLU (FReLU) (see Section 4.3.1.15) and dynamic ReLU (DReLU) (see Section 4.3.1.14) is the displaced ReLU (DisReLU)[25] as it also shifts the ReLU [928]:

$$f(z) = \begin{cases} z, & z + a \geq 0, \\ -a, & z + a < 0, \end{cases} \tag{4.148}$$

where a is a predefined hyperparameter [11, 928]. A Shifted ReLU (see Section 4.2.6.1) is a special case of DisReLU with $a = 1$ [928]. The VGG-19 [51] with DisReLUs outperform the ReLU, LReLU, PReLU, and ELU activation functions with a statistically significant difference in performance on the CIFAR-10 and CIFAR-100 datasets [243] as shown in [928].

4.2.6.45 *Modified LReLU*

Inspired by the DisReLU [928], Yang et al. proposed the modified LReLU (MLReLU) in [929]. The MLReLU is a translated LReLU and is defined as

$$f(z) = \begin{cases} z, & z + a > 0, \\ -az, & z + a \leq 0, \end{cases} \tag{4.149}$$

where a is a fixed parameter controlling both the slope and the threshold [929].

4.2.6.46 *Flatted-T swish*

An activation function flatted-T swish (FTS) [930] combines ReLU and the logistic sigmoid activation function; it is defined as

$$f(z) = \mathrm{ReLU}(z) \cdot \sigma(z) + T = \begin{cases} \frac{z}{1+\exp(-z)} + T, & z \geq 0, \\ T, & z < 0, \end{cases} \tag{4.150}$$

where T is a predefined hyperparameter [930], the recommended value is $T = -0.20$ [930]. The FTS is identical to a shifted swish for the positive z. The FTS was shown to outperform ReLU, LReLU, swish, ELU, and FReLU activation functions [930]. The special case with $T = 0$ was proposed independently under the name of ReLU-Swish in [800].

25 Macêdo et al. originally abbreviated the displaced ReLU as DReLU but that is already taken by dynamic ReLU from Section 4.3.1.14.

4.2.6.47 *Optimal activation function (OAF)*

The so-called Optimal Activation Functio (OAF) is a combination of ReLU and swish activations proposed in [931]. It is defined as

$$f(z) = \text{ReLU}(z) + z \cdot \sigma(z) = \begin{cases} z + z \cdot \sigma(z), & z \geq 0, \\ z \cdot \sigma(z), & z < 0. \end{cases} \tag{4.151}$$

4.2.6.48 *Exponential linear unit (ELU)*

An ELU is an extension of LReLU where the function employs an exponential function for the negative inputs, which speeds up the learning process [874]:

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{\exp(z)-1}{a}, & z < 0, \end{cases} \tag{4.152}$$

where a is a hyperparameter; the authors Clevert, Unterthiner, and Hochreiter used $a = 1$ in their work [874]. The a determines the value to which an ELU saturates for inputs going to negative infinity [874].

4.2.6.49 *Rectified exponential unit (REU)*

A rectified exponential unit (REU) [932] is an activation function inspired by the ELU and swish (see Sections 4.2.6.48 and 4.3.3.1) and is based on the assumption that the success of the swish activation functions is due to the non-monotonic property in the negative quadrant [932]. The REU is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \exp(z), & z < 0. \end{cases} \tag{4.153}$$

A parametric version called parametric rectified exponential unit (PREU) was also proposed in [932]; see Section 4.3.1.9 for details.

4.2.6.50 *Apical dendrite activation (ADA)*

A biologically inspired AF named apical dendrite activation (ADA) was proposed in [933]. It is similar to the ELU, but it applies an exponential function for positive inputs. It is defined as

$$f(z) = \begin{cases} \exp\left(-az + b\right), & z \geq 0, \\ 0, & z < 0, \end{cases} \tag{4.154}$$

where a and b are fixed parameters [933].

4.2.6.51 *Leaky apical dendrite activation (LADA)*

As LReLU extends the ReLU, the leaky apical dendrite activation (LADA) [933] extends the ADA.

$$f(z) = \begin{cases} \exp\left(-az+b\right), & z \geq 0, \\ cz, & z < 0, \end{cases} \tag{4.155}$$

where a, b, and $c \in [0,1]$ are fixed parameters [933]. Georgescu et al. used $c = 0.01$ in their experiments in [933].

4.2.6.52 *Sigmoid linear unit (SigLU)*

The sigmoid linear unit (SigLU)[26] is an ELU alternative that uses a modified logistic sigmoid instead of the exponential [815]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{1-\exp(-2z)}{1+\exp(-2z)}, & z < 0. \end{cases} \tag{4.156}$$

4.2.6.53 *Swish and ReLU activation (SaRa)*

The swish and ReLU activation (SaRa) is an AF combining the swish and ReLU AFs proposed in [934]. It is defined[27] as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{z}{1+a\cdot\exp(-bz)}, & z < 0, \end{cases} \tag{4.157}$$

where a and b are fixed parameters; Qureshi and Sarosh Umar recommend $a = 0.5$ and $b = 0.7$ [934].

4.2.7 *Maxsig*

The maxsig is one of the AFs listed in [733]. The maxsig is similar to the SigLU (see Section 4.2.6.52) and is defined as

$$f(z) = \max\left(z, \sigma(z)\right), \tag{4.158}$$

where $\sigma(z)$ is the logistic sigmoid [733].

26 The AF is unnamed in the original work [815].

27 The formula in [934] is malformed; we believe that this is the intended case. It is possible that authors intended that the SaRa is actually only the part that is defined for the negative inputs in Eq. (4.157) — however, we think that it is less likely as that would be only a swish (see Section 4.3.3.1) AF with some fixed scaling of the output or the AHAF (see Section 4.3.3.2) with fixed parameters.

4.2.7.1 *Tanh linear unit (ThLU)*

The tanh linear unit (ThLU) [935][28] is an AF combining tanh and ReLU. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{2}{1+\exp(-z)} - 1, & z < 0, \end{cases} = \begin{cases} z, & z \geq 0, \\ \tanh\left(\frac{z}{2}\right), & z < 0. \end{cases} \tag{4.159}$$

The ThLU is a special case of the tanh based ReLU (TReLU) with $b_i = \frac{1}{2}$. Similar AF was used under the name maxtanh in [733] — it just omitted the scaling factor. The maxtanh can also be written as $f(z) = \max(z, \tanh(z))$ [733].

4.2.7.2 *DualELU*

The DualELU [916] is equivalent of DualReLU (see Section 4.2.6.36) for ELUs and are defined as

$$f(z, z') = f_{\mathrm{EL}}(z) - f_{\mathrm{EL}}(z'), \tag{4.160}$$

where $f_{\mathrm{EL}}(z)$ is the ELU activation function applied to an input z.

4.2.7.3 *Difference ELU (DiffELU)*

An ELU variant named difference exponential linear unit (DiffELU)[29] was proposed in [936]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a\left(z \exp(z) - b \exp(bz)\right), & z < 0, \end{cases} \tag{4.161}$$

where a and $b \in (0, 1)$ are fixed parameters [936]. Hu et al. also tested setting the parameters to be trainable but that led to worse performance [936]. The recommended setting is $a = 0.3$ and $b = 0.1$ [936].

4.2.7.4 *Polynomial linear unit (PolyLU)*

The polynomial linear unit (PolyLU) is an AF similar to the ELU proposed in [937]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{1}{1-z} - 1, & z < 0. \end{cases} \tag{4.162}$$

Despite the similarity with the ELU, Feng and Yang have shown that the PolyLU outperformed the ELU on the CIFAR-10/100 [243] and Dogs vs. Cats [938, 939] datasets [937]. The PolyLU was also proposed under the name first power linear unit with sign (FPLUS)[30] in [940].

28 The ref [935] is not the original work with ThLUs; it references another work but that uses pure tanh as the AFs.

29 Hu et al. used the abbreviation DELU but this name is used for the AF proposed by Pishchik in [889] throughout this work.

30 Duan, Yang, and Dai used the equivalent definition $f(z) = (\operatorname{sgn}(z) \cdot z + 1)^{\operatorname{sgn}(z)} - 1$ in [940], hence the name.

4.2.7.5 *Inverse polynomial linear unit (IpLU)*

The polynomial linear unit (IpLU) was proposed in [906]; it is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{1}{1+|z|^a}, & z < 0, \end{cases} \tag{4.163}$$

where $a > 0$ is a fixed hyperparameter guaranteeing a small slope for negative inputs [906].

4.2.7.6 *Power linear unit (PoLU)*

The power linear unit (PoLU) [941] is an AF similar to the ELU. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ (1-z)^{-a} - 1, & z < 0, \end{cases} \tag{4.164}$$

where a is a fixed parameter [941]. Li, Ding, and Li used $a \in \{1, 1.5, 2\}$ in their experiments in [941].

4.2.7.7 *Power function linear unit (PFLU)*

The power function linear unit (PFLU) is an AF proposed in [942]; it is defined as

$$f(z) = z \cdot \frac{1}{2}\left(1 + \frac{z}{\sqrt{1+z^2}}\right). \tag{4.165}$$

4.2.7.8 *Faster power function linear unit (FPFLU)*

The faster power function linear unit (FPFLU) is an AF proposed in [942] that resembles the IpLU (see Section 4.2.7.5) It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ z + \frac{z^2}{\sqrt{1+z^2}}, & z < 0. \end{cases} \tag{4.166}$$

4.2.7.9 *Elastic adaptively parametric compounded unit (EACU)*

The elastic adaptively parametric compounded unit (EACU) [943] is a stochastic AF. It is defined as

$$f(z_i) = \begin{cases} b_i z_i, & z_i \geq 0, \\ a_i z_i \cdot \tanh\left(\ln\left(1 + \exp\left(a_i, z_i\right)\right)\right), & z_i < 0, \end{cases} \tag{4.167}$$

where b_i is stochastically sampled during training as

$$b_i = \begin{cases} s_i, & 0.5 < s_i < 1.5, \\ 1, \text{ otherwise}, \end{cases} \tag{4.168}$$

$$s_i \sim mathrmN\left(0, 0.01\right), \tag{4.169}$$

and a_i is an adaptive parameter for each neuron or channel i [943].

4.2.7.10 *Lipschitz ReLU (L–ReLU)*

A L–ReLU [944] is a piecewise linear activation function. The slope of the negative part is selected with respect to a data-dependent Lipschitz constant [944]. It builds on a proposed piecewise function that treats the positive $z > 0$ and negative values ($z \leq 0$) separately:

$$f(z) = p(z|z > 0) + n(z|z \leq 0), \tag{4.170}$$

where

$$p(z) = \max\left(\phi(z), 0\right), \tag{4.171}$$

and

$$n(z) = \min(\mu(z), 0), \tag{4.172}$$

where $\phi(z)$ and $\mu(z)$ can be any function $f : \mathbb{R} \rightarrow \mathbb{R}$ [944]. This makes the positive part of the piecewise lay in the first quadrant of the Cartesian coordinate system and the negative part in the third quadrant [944].

4.2.7.11 *Scaled exponential linear unit (SELU)*

A SELU [945] was proposed in order to make the network self-normalize by automatically converging towards zero mean and unit variance [11]. The ELU was chosen as the basis for self-normalizing neural networks (SNNs) because these cannot be derived with ReLUs, sigmoid, and tanh units or even LReLUs [945] — the activation function has to have negative and positive values for controlling the mean, saturation region where derivatives approach zero in order to dampen the variance if it is too large, a slope larger than one in order to increase the variance if it is too small, and a continuous curve to ensure a fixed point where the variance dampening is balanced out by the variance increasing [945]. The SELU is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ ab\left(\exp\left(z\right) - 1\right), & z < 0, \end{cases} \tag{4.173}$$

where $a > 1$ and b are predefined parameters [11, 945]; the recommended values are $a \approx 1.05078$ and $b \approx 1.6733$ [945].

4.2.7.12 *Leaky scaled exponential linear unit (LSELU)*

A leaky variant of SELU called leaky scaled exponential linear unit (LSELU) was proposed in [946] and is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ ab\left(\exp\left(z\right) - 1\right) + acz, & z < 0, \end{cases} \tag{4.174}$$

where $a > 1$ and b are predefined parameters of the original SELU (see Section 4.2.7.11), and c is a new, predefined parameter controlling the leakiness of the unit [946].

4.2.7.13 *Scaled exponentially-regularized linear unit (SERLU)*

The scaled exponentially-regularized linear unit (SERLU) is a modification of the SELU proposed in [947]; it is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ abz \exp\left(z\right), & z < 0, \end{cases} \tag{4.175}$$

where $a > 0$ and $b > 0$ are predefined parameters [947]. An extension of this approach named ASERLU for bidirectional long short-term memory (BiLSTM) architectures was proposed in [948].

4.2.7.14 *Scaled scaled exponential linear unit (sSELU)*

Additional scaling of the negative pre-activations was introduced in the scaled scaled exponential linear unit (sSELU) [946]:

$$f(z) = \begin{cases} az, & z \geq 0, \\ ab\left(\exp\left(cz\right) - 1\right), & z < 0, \end{cases} \tag{4.176}$$

where $a > 1$ and b are predefined parameters of the original SELU (see Section 4.2.7.11), and c is a new, predefined parameter controlling the scaling of the negative inputs to the unit [946].

4.2.7.15 *RSigELU*

A parametric ELU variant called RSigELU [949] is defined as

$$f(z) = \begin{cases} z\left(\frac{1}{1+\exp(-z)}\right)a + z, & 1 < z < \infty, \\ z, & 0 \geq z \geq 1, \\ a\left(\exp(z) - 1\right), & -\infty < z < 0, \end{cases} \tag{4.177}$$

where a is a predefined parameter, Kiliçarslan and Celik used $0 < a < 1$ in their work [949]. For $a = 0$, the RSigELU becomes ReLU [949]. The RSigELU was shown to outperform ReLU, LReLU, softsign, swish, ELU, SEU, GELU, LISA, Hexpo and softplus on the MNIST dataset [45], Fashion MNIST [950] and the IMDB Movie dataset; it still outperformed these activation functions on the CIFAR-10 dataset [243] but it was outperformed by its variant RSigELUD [949].

4.2.7.16 *HardSReLUE*

Another AF proposed by Kiliçarslan is the HardSReLUE [951]. Kiliçarslan defined the AF as

$$f(z) = \begin{cases} az\left(\max\left(0, \min\left(1, \frac{z+1}{2}\right)\right)\right) + z, & z \geq 0, \\ a\left(\exp(z) - 1\right), & z < 0, \end{cases} \tag{4.178}$$

where a is a fixed slope parameter [951].

4.2.7.17 *Exponential linear sigmoid squashing (ELiSH)*

An activation function exponential linear sigmoid squashing (ELiSH) [651] combines the swish (see Section 4.3.3.1) and the ELU function [11]. It is defined as

$$f(z) = \begin{cases} \frac{z}{1+\exp(-z)}, & z \geq 0, \\ \frac{\exp(z)-1}{1+\exp(-z)}, & z < 0. \end{cases} \tag{4.179}$$

4.2.7.18 *Hard exponential linear sigmoid squashing (HardELiSH)*

As ELiSH (see Section 4.2.7.17) combines swish with ELU and linear function, the hard exponential linear sigmoid squashing (HardELiSH) combines the Hard sigmoid [890] with ELU and linear function [651]. It is defined as

$$f(z) = \begin{cases} z \cdot \max\left(0, \min\left(\frac{z+1}{2}, 1\right)\right), & z \geq 0, \\ \left(1 + \exp\left(-z\right)\right) \cdot \max\left(0, \min\left(\frac{z+1}{2}, 1\right)\right), & z < 0. \end{cases} \tag{4.180}$$

4.2.7.19 *RSigELUD*

The RSigELUD is a double parameter variant of the RSigELU (see Section 4.2.7.15) [949] that is defined as

$$f(z) = \begin{cases} z\left(\frac{1}{1+\exp(-z)}\right)a + z, & 1 < z < \infty, \\ z, & 0 \leq z \leq 1, \\ b\left(\exp(z) - 1\right), & -\infty < z < 0, \end{cases} \tag{4.181}$$

where a and b are predefined parameters, Kiliçarslan and Celik used $0 < a < 1$ and $0 < b < 1$ in their work [949]. For $a = b = 0$, the RSigELUD becomes the ReLU the same as the RSigELU; however, for $a = 0$ and positive b, the function resembles the vanilla ELU [949].

4.2.7.20 *LS–ReLU*

The LS–ReLU[31] is a ReLU-inspired AF proposed in [952]. It is defined as

$$f(z) = \begin{cases} \frac{z}{1+|z|}, & z \le 0, \\ z, & 0 \le z \le b, \\ \log(az+1) + |\log(ab+1) - b|, & z \ge b, \end{cases} \tag{4.182}$$

where a and b are fixed[32] parameters [952].

4.2.8 *Square-based activation functions*

Several square-based activation functions were proposed in [953–955] for better computational efficiency, especially on low-power devices [953]. The approach uses the square function to replace the potentially costly exponential function. These function leads to significantly more efficient computation when there is no hardware implementation of the exponential function [953]. The efficiency gains can be further improved with a custom hardware operator

$$f_h(x) = -|x| \cdot x, \tag{4.183}$$

which can be used for efficient hardware implementation of all of the activation functions of the square-based family [953]. The usage of the AFs from the family can lead to performance gains of one order of magnitude compared to traditional AFs [953] for both forward and backward passes (depends on the particular activation function and the usage of fixed or floating point representations) [953].

4.2.8.1 *SQNL*

A computationally efficient activation function was proposed in [954]; unlike many other sigmoidal functions, it uses the square operator instead of the exponential function in order to achieve better computational efficiency. The derivative of the function is linear, which leads to a less computationally costly computation of the gradient. The function is defined in [953] (the original paper [954] had several mistakes in the definition) as

$$f(z) = \begin{cases} 1, & z > 2, \\ z - \frac{z^2}{4}, & 0 \le z \le 2, \\ z + \frac{z^2}{4}, & -2 \le z < 0, \\ -1, & z < -2. \end{cases} \tag{4.184}$$

The SQNL[33] has bounded range $[-1, 1]$ [953]. The performance of the SQNL was verified on several datasets from the UCI Machine Learning

31 Not an abbreviation.
32 Wang et al. do not specify whether the parameters are trainable or fixed.
33 SQNL is not an abbreviation but rather a name given by Wuraola and Patel.

Repository [956] and on the MNIST dataset [45]; more details available in [954].

4.2.8.2 *Square linear unit (SQLU)*

Similarly as the SQNL (see Section 4.2.8.1) uses square function to form a sigmoidal function to approximate tanh, the square linear unit (SQLU) [953] uses square function to form a ELU-like activation function that is computationally efficient:

$$f(z) = \begin{cases} z, & z > 0, \\ z + \frac{z^2}{4}, & -2 \leq z \leq 0, \\ -1, & z < -2. \end{cases} \tag{4.185}$$

The SQLU basically uses the negative part of the SQNL and replaces the positive part with a linear function.

4.2.8.3 *Square swish (squish)*

Another example of the family of activation functions based on the square operator is the square swish (squish) [953], which is an AF inspired by the swish and GELU (see Section 4.2.3.1). It uses the square non-linearity in order to achieve good computational efficiency:

$$f(z) = \begin{cases} z + \frac{z^2}{32}, & z > 0, \\ z + \frac{z^2}{2}, & -2 \leq z \leq 0, \\ 0, & z < -2. \end{cases} \tag{4.186}$$

While the squish was inspired by the swish and GELU activation functions, it is an approximation of neither [953].

4.2.8.4 *Square REU (SqREU)*

Similarly as REU (see Section 4.2.6.49) is a combination of the ReLU and swish activation functions, the *square REU* (SqREU) [953] is a combination of ReLU and squish:

$$f(z) = \begin{cases} z, & z > 0, \\ z + \frac{z^2}{2}, & -2 \leq z \leq 0, \\ 0, & z < -2. \end{cases} \tag{4.187}$$

4.2.8.5 *Square softplus (SqSoftplus)*

A square softplus (SqSoftplus) is another square-based computationally efficient replacement of an activation function — softplus [953]:

$$f(z) = \begin{cases} z, & z > \frac{1}{2}, \\ z + \frac{\left(z+\frac{1}{2}\right)^2}{2}, & -\frac{1}{2} \leq z \leq \frac{1}{2}, \\ 0, & z < \frac{1}{2}. \end{cases} \tag{4.188}$$

4.2.8.6 *Square logistic sigmoid (LogSQNL)*

While the SQNL [954] replaces the tanh AF, the square logistic sigmoid (LogSQNL) [953] is a square-based replacement for the logistic sigmoid:

$$f(z) = \begin{cases} 1, & z > 2, \\ \frac{1}{2}\left(z - \frac{z^2}{4}\right) + \frac{1}{2}, & 0 \leq z \leq 2, \\ \frac{1}{2}\left(z + \frac{z^2}{4}\right) + \frac{1}{2}, & -2 \leq z < 0, \\ 0, & z < -2. \end{cases} \tag{4.189}$$

4.2.8.7 *Square softmax (SQMAX)*

The square softmax (SQMAX) is a square-based replacement for the softmax, which is exponential-based. It is defined as

$$f(z_j) = \frac{\left(z_j + c\right)^2}{\sum_{k=1}^{N} \left(z_k + c\right)^2}, \tag{4.190}$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons and $c = 4$ is a predefined constant [953].

4.2.8.8 *Linear quadratic activation*

Another square-based AF called linear quadratic activation (LinQ) was proposed in [955].

$$f(z) = \begin{cases} az + \left(1 - 2z + z^2\right), & z \geq 2 - 2a, \\ \frac{1}{4}z\left(4 - |z|\right), & -2 + 2a < z < 2 - 2a, \\ az - \left(1 - 2z + z^2\right), & z \leq -2 + 2a, \end{cases} \tag{4.191}$$

where a is a fixed parameter controlling the slope of the function's linear parts [955].

4.2.8.9 *Inverse square root linear unit (ISRLU)*

Inverse square root linear unit (ISRLU) [957] is an activation function similar to the ELU (see Section 4.2.6.48). It has similar properties and a shape as

ELU; however, it is faster to compute, leading to more efficient training and inference [957]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \frac{1}{\sqrt{1+az^2}}, & z < 0, \end{cases} \tag{4.192}$$

where a is a hyperparameter controlling the value to which the ISRLU saturates for negative inputs [957]. While the authors state that the hyperparameter a could be trainable for each neuron i, only the non-trainable variant was analyzed [957]. Carlile et al. analysed ISRLU with $a = 1$ and $a = 3$ [957].

4.2.8.10 *Inverse square root unit (ISRU)*

ISRU [957] is an activation function meant to replace sigmoidal activation functions. It is defined as

$$f(z) = z \cdot \frac{1}{\sqrt{1+az^2}}, \tag{4.193}$$

where a si a fixed hyperparameter controlling the saturation values; the parameter could be trainable similarly as in the ISRLU (see Section 4.2.8.9) but only the nonadaptive variant was used [957].

4.2.8.11 *Modified Elliott function (MEF)*

The modified Elliott function (MEF) [809] is an AF inspired by the Elliott function (see Section 4.2.2.15); it can also be considered to be a translated special case of the ISRU (see Section 4.2.8.10) with $a = 1$. It is defined as

$$f(z) = z \cdot \frac{1}{\sqrt{1+z^2}} + \frac{1}{2}. \tag{4.194}$$

4.2.9 *Square-root-based activation function (SQRT)*

A square-root-based activation function (SQRT) is a monotonically increasing, unbounded activation function proposed in [958] with a similar structure as the earlier proposed logarithmic activation function (LAF) but with the square root function instead of the natural logarithm used in LAF. It is defined as

$$f(z) = \begin{cases} \sqrt{z}, & z \geq 0, \\ -\sqrt{-z}, & z < 0. \end{cases} \tag{4.195}$$

The SQRT activation function was found to outperform both tanh and ReLU activation functions on the CIFAR-10 dataset [243] in experiments in [958].

A parametric variant of the SQRT called S-shaped activation function (SSAF) was proposed independently in [959]. It is defined as

$$f(z) = \begin{cases} \sqrt{2az}, & z \geq 0, \\ -\sqrt{-2az}, & z < 0, \end{cases} \tag{4.196}$$

where a is a fixed parameter [959].

4.2.10 *Bent identity*

The bent identity [960] is an AF approximating the ReLU; it can be seen as a fixed variant of the bendable linear unit (BLU) (see Section 4.3.1.37) with $a_i = \frac{1}{2}$. It is defined as

$$f(z) = \frac{\sqrt{z^2+1}-1}{2} + z. \tag{4.197}$$

4.2.11 *Mishra activation function*

The Mishra[34] AF is defined as

$$f(z) = \frac{1}{2}\left(\frac{z}{1+|z|}\right)^2 + \frac{1}{2}\frac{z}{1+|z|}. \tag{4.198}$$

4.2.12 *Saha-Bora activation function (SBAF)*

A Saha-Bora activation function (SBAF) was proposed in [923, 962] to be used for the habitability classification of exoplanets. It employs two non-trainable parameters α and k, which were set to $k = 0.98$ and $\alpha = 0.5$, where authors determined a stable fixed point. It is defined as:

$$f(z) = \frac{1}{1+kz^{\alpha}(1-z)^{(1-\alpha)}}. \tag{4.199}$$

4.2.13 *Logarithmic activation function*

The logarithmic activation function (LAF) was proposed in [963] (ref. from [798]). According to [798], it is defined as

$$f(z) = \begin{cases} \ln z + 1, & z \geq 0, \\ -\ln -z + 1, & z < 0. \end{cases} \tag{4.200}$$

The LAF was independently proposed under the name symlog in [964].

4.2.14 *Symexp*

The symexp [964] is an activation function that is inverse of the logmoid activation unit (LAU). It is defined as

$$f(z) = \operatorname{sgn}(z)\left(\exp(|z|) - 1\right). \tag{4.201}$$

34 The AF was unnamed in the original papers [709, 961]; however, the work [707] named it using the name of the original author. We keep the naming in this work.

4.2.15 *Scaled polynomial constant unit (SPOCU)*

The scaled polynomial constant unit (SPOCU) is a polynomial-based AF proposed in [965, 966]. It is defined as

$$f(z) = ah\left(\frac{z}{c} + b\right) - ah\left(b\right), \tag{4.202}$$

where

$$h(x) = \begin{cases} r(d), & x \geq d, \\ r(x), & 0 \leq x < d, \\ x, & x < 0, \end{cases} \tag{4.203}$$

$$r(x) = x^3\left(x^5 - 2x^4 + 2\right), \tag{4.204}$$

and $a > 0$, $b \in (0, 1)$, $c > 0$, and $d \in [1, \infty)$ are fixed parameters satisfying additional conditions listed in [965, 966].

4.2.16 *Polynomial universal activation function (PUAF)*

Similarly as the universal activation function (UAF) (see Section 4.3.21), the polynomial (PUAF)[35] is able to approximate popular AFs such as the logistic sigmoid, ReLU, and swish [967]. It is defined as

$$f(z) = \begin{cases} z^a, & z > c, \\ z^a \frac{(c+z)^b}{(c+z)^b + (c-z)^b}, & |z| \leq c, \\ 0, & z < -c, \end{cases} \tag{4.205}$$

where a, b and c are fixed parameters [967]. The PUAF becomes the ReLU with $a = 1$, $b = 0$, and $c = 0$; the logistic sigmoid is approximated with $a = 0$, $b = 5$, and $c = 10$; finally, the swish is approximated using $a = 1$, $b = 5$, and $c = 10$ [967].

4.2.17 *Softplus*

The softplus function was proposed in [969] and is defined as

$$f(z) = \ln\left(\exp\left(z\right) + 1\right). \tag{4.206}$$

The softplus was used as an activation function in [862] where it was used alongside with a ReLU. The advantage of softplus over ReLU is that it is smooth and it has a non-zero gradient for negative inputs; thus, it does not suffer from the phenomenon of dying out neurons that is common in networks with ReLU activations [970]. The softplus was found to outperform ReLU for certain applications and architectures [970]. A noisy variant was used for spiking neural networks in [971].

35 Hwang and Kim named the function only as the *universal activation function* but this name is already taken by the UAF by Yuen et al. from [968].

4.2.18 *Parametric softplus (PSoftplus)*

Parametric softplus (PSoftplus) [972] is a softplus variant that allows for scaling and shifting using two additional parameters. The PSoftplus is defined as

$$f(z) = a\left(\ln\left(\exp\left(z\right)+1\right)-b\right), \tag{4.207}$$

where a and b are fixed predetermined hyperparameters [972]. The creation of the softplus was motivated by the assumption that activations with mean outputs close to zero can improve the performance of a neural network; since the output of the softplus is always positive, a shift parameter b was introduced to shift the mean output closer to zero [972]. The slope controlling parameter a is used to adjust the function and the gradient disappearance or overflow during training [972]. The recommended values are $a = 1.5$ and $b = \ln(2)$ [972].

4.2.18.1 *Soft++*

Another softplus extension *Soft++* is a multiparametric nonsaturating nonlinear activation function proposed in [973]. It is defined as

$$f(z) = \ln\left(1+\exp\left(az\right)\right)+\frac{z}{b}-\ln(2), \tag{4.208}$$

where a and b are fixed predetermined hyperparameters [973]; however, Ciuparu, Nagy-Dăbâcan, and Mureşan proposed they could be adaptable in future works. Multiple values of the parameters were used in the experiments in [973], but $a = 1$ and $b = 2$ were found to work well [973]; nevertheless, a hyperparameter optimization is recommended [973].

4.2.19 *Rand softplus (RSP)*

A softplus variant rand softplus (RSP) [974] introduces a stochastic parameter a_l that is determined by the noise level of the input data [974]. The RSP is defined as

$$f(z_l) = \left(1-a_l\right)\max\left(0,z_l\right)+a_l\cdot\ln\left(1+\exp\left(z_l\right)\right), \tag{4.209}$$

where a_l is adapting to the input noise levels of each layer l — the exact procedure is described in [974].

4.2.20 *Aranda-Ordaz*

The Aranda-Ordaz AF[975, 976] was used in NNs in [975]. It is defined as

$$f(z) = 1-\left(1+a\exp\left(z\right)\right)^{-\frac{1}{a}}, \tag{4.210}$$

where $a > 0$ is a fixed parameter [975]. Essai Ali, Abdel-Raman, and Badry used $a = 2$ in their work [710].

4.2.21 *Bi-firing activation function (bfire)*

A bi-firing activation function (bfire) was proposed in [977] and is defined as

$$f(z) = \begin{cases} z - \frac{a}{2}, & z > a, \\ \frac{z^2}{2a}, & -a \geq z, \geq a \\ -z - \frac{a}{2}, & z < -a, \end{cases} \tag{4.211}$$

where a is a predefined smoothing hyperparameter [977]. The bfire is basically a smoothed variant of the later proposed vReLU (see Section 4.2.6.25) as it becomes vReLU as $a \to 0$.

4.2.22 *Bounded bi-firing activation function (bbfire)*

A bounded variant of the bi-firing (bfire) activation function (see Section 4.2.21) called bbfire was proposed in [888]; similarly as BReLU and BLReLU bounds ReLU and LReLU respectively (see Sections 4.2.6.16 and 4.2.6.24), the bounded bi-firing function (bbfire) is defined as

$$f(z) = \begin{cases} b, & z < -b - \frac{a}{2}, \\ -z - \frac{a}{2}, & -b - \frac{a}{2} \geq z < -a, \\ \frac{z^2}{2a}, & -a \geq z \geq a, \\ z - \frac{a}{2}, & a < z \geq b + \frac{a}{2}, \\ b, & z > b + \frac{a}{2}, \end{cases} \tag{4.212}$$

where a and b are predefined hyperparameters [888]. The is symmetrical about the origin and has a near inverse-bell-shaped activation curve [888]. While authors of the original bfire [977] solved potential numerical instabilities caused by the unboundedness by imposing a small L_1 penalty on the hidden activation values [977], the bbfire alleviates this problem explicitly without any need for such penalty.

4.2.23 *Piecewise Mexican-hat activation function (PMAF)*

The piecewise Mexican-hat activation function (PMAF) was used in [978]; it is defined as

$$f(z) = \begin{cases} \left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\right)\left(1 - (z+a)^2\right)\exp\left(-\frac{(z+a)^2}{2}\right), & z < 0, \\ \left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\right)\left(1 - (z-a)^2\right)\exp\left(-\frac{(z-a)^2}{2}\right), & z \geq 0, \end{cases} \tag{4.213}$$

where a is a fixed parameter — Liu, Zeng, and Wang used $a = 4$ [978].

4.2.24 *Piecewise radial basis function (PRBF)*

The piecewise (PRBF) was used in [978]; it is defined as

$$f(z) = \begin{cases} \exp\left(-\frac{(z-2a)^2}{b^2}\right), & z \geq a \\ \exp\left(-\frac{z^2}{b^2}\right), & -a < z < a \\ \exp\left(-\frac{(z+2a)^2}{b^2}\right), & z \leq -a \end{cases} \tag{4.214}$$

where a and b are fixed parameters [978] — Liu, Zeng, and Wang used $a = 3$ and $b = 1$ [978].

4.2.25 *Comb-H-sine*

A comb-H-sine is an activation function that was found using an evolutionary approach in [659]. It is defined as

$$f(z) = \sinh(az) + \sinh^{-1}(az), \tag{4.215}$$

where $\sinh(x)$ is the hyperbolic sine, $\sinh^{-1}(x)$ is its inverse, and a is a predefined hyperparameter [659]. This function was found to outperform ReLU, tanh, logistic sigmoid, and several other activation functions in LSTM models in [659].

4.2.26 *Modified arcsinh*

The modified arcsinh (m-arcsinh) AF was proposed in [979] and is defined as

$$f(z) = \frac{1}{12}\sinh^{-1}(z)\sqrt{|z|}. \tag{4.216}$$

Interestingly, the m-arcsinh can be used either as an AF in a NN or as a kernel function in the SVM [979].

4.2.27 *hyper-sinh*

The hyper-sinh is an AF that uses the sinh and cubic functions [980, 981]; it is defined as

$$f(z) = \begin{cases} \frac{\sinh(z)}{3}, & z > 0, \\ \frac{z^3}{4}, & z \leq 0. \end{cases} \tag{4.217}$$

4.2.28 *Arctid*

The arctid is an arctan-based AF used in [733]; it is defined as

$$f(z) = \left(\tan^{-1}(z)\right)^2 - z. \tag{4.218}$$

4.2.29 *Sine*

The sine with inputs scaled by π was used as an activation in [982]:

$$f(z) = \sin(\pi z). \tag{4.219}$$

It was, for example, used recently with a data-driven determination of a network's biases in [786]. Just the sine function without any scaling was used as an activation in [983–988].

Scaled sine with vertical shift was used in [667]; the used AF is defined as

$$f(z) = 0.5\sin(az) + 0.3, \tag{4.220}$$

where a is a fixed parameter; $a \in \{0.2, 0.8, 1.2, 1.8, 4\}$ [667].

4.2.30 *Cosine*

A cosine activation was used in simulations in [989]; it was defined as

$$f(z) = 1 - \cos(z). \tag{4.221}$$

4.2.31 *Cosid*

The cosid is one of the AFs listed in [733]. It is defined as

$$f(z) = \cos(z) - z. \tag{4.222}$$

4.2.32 *Sinp*

A parametric AF similar to the cosid was proposed in [990] under the name sinp.[36] It is defined as

$$f(z) = \sin(z) - az, \tag{4.223}$$

where a is a fixed parameter [990]. Chan et al. used $a \in \{1, 1.5, 2\}$ [990].

4.2.33 *Growing cosine unit (GCU)*

Another cosine-based AF is the growing cosine unit (GCU) proposed in [991]. It is defined as

$$f(z) = z\cos(z). \tag{4.224}$$

Empirical evaluation of the performance of GCU compared to ReLU, PReLU, and mish is available in [992]; its brief evaluation with respect to the generation of NFTs is available in [993].

36 Technically, the full name used by Chan et al. is SinP$[N]$ but we ommited the parameter from the name of the AF.

4.2.34 *Amplifying sine unit (ASU)*

The amplifying sine unit (ASU) is the sine equivalent of the GCU [994, 995]

$$f(z) = z \sin(z). \tag{4.225}$$

4.2.35 *Sinc*

The sinc is an older AF proposed in [996]. It is defined as

$$f(z) = \begin{cases} \frac{\sin(\pi z)}{\pi z}, & z \neq 0, \\ 1, & z = 1. \end{cases} \tag{4.226}$$

A shifted variant was proposed under the name shifted sine unit (SSU) in [997]. It is defined as

$$f(z) = \pi \text{sinc}(z - \pi). \tag{4.227}$$

4.2.36 *Decaying sine unit (DSU)*

The decaying sine unit (DSU) is a sinc based AF proposed in [997]. It is defined as

$$f(z) = \frac{\pi}{2}\left(\text{sinc}(z - \pi) - \text{sinc}(z + \pi)\right). \tag{4.228}$$

4.2.37 *Hyperbolic cosine linearized squashing function (HcLSH)*

The hyperbolic cosine linearized squashing function (HcLSH) is an AF proposed in [998]; it is defined as

$$f(z) = \begin{cases} \ln\left(\cosh(z) + z \cdot \cosh\left(\frac{z}{2}\right)\right), & z \geq 0, \\ \ln(\cosh(z)) + z, & z < 0. \end{cases} \tag{4.229}$$

4.2.38 *Polyexp*

The polyexp is an AF combining quadratic function and an exponential function [996];[37] it is defined as

$$f(z) = \left(az^2 + bz + c\right) \exp\left(-dz^2\right), \tag{4.230}$$

where a, b, c, and d are fixed parameters [996].

4.2.39 *Exponential*

The exponential was used as an AF in [667]. The AF was defined as

$$f(z) = \exp(-z). \tag{4.231}$$

37 The [982] is referenced as the origin of polyexp in [996] but we have not seen the definition there.

4.2.40 *E-Tanh*

An AF named E-Tanh combining the exponential and tanh functions was proposed in [999]. It is defined as

$$f(z) = a \cdot \exp(z) \tanh(z), \tag{4.232}$$

where a is a fixed scaling parameter [999, 1000].

4.2.40.1 *Evolved combination of tanh and ReLU*

The combination of tanh and ReLU was found using neuroevolution in [666] — while Vijayaprabakaran and Sathiyamurthy also mentioned other AFs, this combination led to the best performance on the HAR dataset using the LSTM units. The best-performing recurrent AF was

$$f(z) = a\left(\tanh\left(z^2\right) + \text{ReLU}(z)\right) \tag{4.233}$$

and the regular AF was

$$f(z) = \max\left(\tanh(\log(z)), \text{ReLU}(z)\right). \tag{4.234}$$

See [666] for evaluation details and for other top AFs.

4.2.41 *Wave*

The wave is an AF combining quadratic function and an exponential function [996];[38] similarly as the polyexp but only with a single parameter; it is defined as

$$f(z) = \left(1 - z^2\right) \exp\left(-az^2\right), \tag{4.235}$$

where a is a fixed parameter [996].

4.2.42 *Non-monotonic cubic unit (NCU)*

A simple AF based on a third-degree polynomial was proposed in [997]. It is named non-monotonic cubic unit (NCU) and is defined as

$$f(z) = z - z^3. \tag{4.236}$$

4.2.43 *Triple*

Another AF based on a third-degree polynomial called triple was proposed in [1001]. It is defined as

$$f(z) = a \cdot z^3, \tag{4.237}$$

where a is a fixed parameter [1001]. Chen et al. tested values of $a \in \{0.1, 0.5, 1, 2\}$ and observed that $a = 1$ reaches the best results [1001].

38 The [982] is referenced as the origin of wave in [996] but we have not seen the definition there.

4.2.44 *Shifted quadratic unit (SQU)*

The shifted quadratic unit (SQU) [997] is a simple non-monotonic AF defined as

$$f(z) = z^2 + z. \tag{4.238}$$

4.2.45 *Knowledge discovery activation function (KDAC)*

Wang et al. proposed a special AF for knowledge discovery in [1002]. This function named knowledge discovery activation function (KDAC) has two adaptive parameters $a > 0$ and $b > 0$ and one fixed parameter c. It is defined[39] as

$$f(z) = p \cdot (1 - \mathrm{h}_{\max}(p,r)) + r \cdot \mathrm{h}(p,r) + k\mathrm{h}_{\max}(p,r)(1 - \mathrm{h}_{\max}(p,r)), \tag{4.239}$$

where

$$\mathrm{h}_{\max}(x,y) = \mathrm{clip}\left(\frac{1}{2} - \frac{1}{2}\frac{x-y}{c}\right), \tag{4.240}$$

$$\mathrm{clip}(x) = \begin{cases} 0, & x \leq 0, \\ x, & 0 < x < 1, \\ 1, & x \geq 0, \end{cases} \tag{4.241}$$

$$p = az, \tag{4.242}$$

$$q = \mathrm{h}_{\min}(bz, s), \tag{4.243}$$

$$r = \begin{cases} p, & z > 0, \\ bz \cdot (1-q) + s \cdot \mathrm{h}(q,s) + kq(1-q), & z \leq 0, \end{cases} \tag{4.244}$$

$$s = \tanh(z), \tag{4.245}$$

and

$$\mathrm{h}_{\min}(x,y) = \mathrm{clip}\left(\frac{1}{2} + \frac{1}{2}\frac{x-y}{c}\right). \tag{4.246}$$

Wang et al. used fixed $c = 0.01$ [1002].

39 The original code by Wang et al. is available at https://github.com/pyy-copyto/KDAC/blob/main/KDAC.py.

4.2.46 *K-winner-takes-all activation function (k-WTA)*

The k-winner-take-all (k-WTA) AF was used to improve adversarial robustness in [1003]. It is defined as

$$f(\mathbf{z})_j = \begin{cases} z_j, & z_j \in \{k \text{ largest elements of } \mathbf{z}\}, \\ 0, & \text{otherwise}, \end{cases} \tag{4.247}$$

where $f(\mathbf{z}) : \mathbb{R}^N \to \mathbb{R}^N$ is the k-WTA AF and $f(\mathbf{z})_j$ its j-th element, $\mathbf{z}$ is the input to the AF, and k a fixed parameter [1003].

4.2.47 *Volatility-based activation function (VBAF)*

The volatility-based activation function (VBAF)[40] is an AF with multiple intputs proposed in [1004]. It is meant for time-series forecasting and was used in a LSTM NN in [1004]. It is defined as

$$f(z_1, \ldots, z_n) = \sqrt{\frac{\sum_{j=1}^{n} \left(\bar{0}z - z_j\right)}{n}}, \tag{4.248}$$

where

$$\bar{0}z = \frac{\sum_{j=1}^{n} z_j}{n}, \tag{4.249}$$

n is the number of time-series samples in the given period [1004, 1005]. Unfortunately, no more details about the application of the VBAF were provided in [1004]; thus, it remains unclear whether the VBAF was applied only directly to the inputs, or it was used on intermediary representations of a NN.

4.2.48 *Chaotic activation functions*

The chaotic activation functions (CAFs) listed in this work are AFs that use a recursive definition to produce a chaotic behavior.

4.2.48.1 *Hybrid chaotic activation function*

The hybrid chaotic activation function (HCAF) is a multi-output type of AF proposed in [1006]. Neuron i in layer l takes an input z_i^l, applies the logistic sigmoid AF and then maps the outputs using logistic map function to individual outputs going to the neurons in layer $l+1$ [1006]. Therefore, a single neuron in layer l emits a different activation value to each neuron in the layer $l+1$ [1006].

For a neuron i, the HCAF first applies the logistic sigmoid function to produce activation a_i

$$a_i = f(z_i) = \sigma(z_i), \tag{4.250}$$

40 Kayim and Yilmaz named the function originally only *volatility activation function*.

then the first value going to the neuron 1 in the following layer is calculated as

$$c_{i,1} = ra_i \left(1 - a_i\right), \tag{4.251}$$

and the output values going to the other neurons in the following layer are calculated recursively as

$$c_{i,j} = rc_{i,j-1} \left(1 - c_{i,j-1}\right), \tag{4.252}$$

where j is the number of a neuron in a following layer and r represents an excitatory rate in a neuron [1006]. Reid and Ferens used $r = 4$ as this value produces a chaotic behavior of the logistic map [1006]; generally, values below 0 or above 4 lead to the output to become unbounded, values between 0 and 1 lead to convergence toward the zero, values between 1 and 3 lead to convergence to a fixed number, values between 3 and 3.5 lead to a periodic solution and only values between 3.5 and 4 produce chaotic behavior [1006].

4.2.48.2 *Fusion of chaotic activation function (FCAF)*

Similarly as HCAF, also the fusion of chaotic activation function (FCAF) [1007] uses a recursive definition for computing the output of a neuron. The FCAF is defined[41] for hidden units as[42]

$$f(z_{i+1}) = rz_i \left(1 - z_i\right) + z_i + a - \frac{b}{2\pi} \sin\left(2\pi z_i\right) \tag{4.253}$$

and for the output units as

$$f(z_{i+1}) = rz_i \left(1 - z_i\right) + z_i + a - \frac{b}{2\pi} \sin\left(2\pi z_i\right) + \exp\left(-cz_i^2\right) + d, \tag{4.254}$$

where r, a, b, c, and d are fixed parameters [1007]; the suitable values for the parameter r are discussed in Section 4.2.48.1 where an equivalent parameter is used.

4.2.48.3 *Cascade chaotic activation function (CCAF)*

The cascade chaotic activation function (CCAF) was introduced in [1008] and is recursively defined for the neuron $i+1$ in a given layer using the preceding neuron i from the same layer as

$$f(z_{i+1}) = a \cdot \sin\left(\pi \cdot b \cdot \sin\left(\pi z_i\right)\right), \tag{4.255}$$

where a and b are two fixed parameter from the interval $[0, 1]$ [1008].

41 Kabir et al. did not explicitly defined what the index i denotes but most likely it denotes the i-th neuron in a given layer.

42 The formula given in [1007] probably missed a minus sign after the parameter a.

4.3 ADAPTIVE ACTIVATION FUNCTIONS

The activation function introduces non-linearities to neural networks and is crucial for network's performance [46]. Even though it might be suboptimal, the same activation function is usually used for the whole network or at least for all neurons in a single layer. Over the last few decades, there have been several attempts to use activation functions that might differ across neurons (e.g., [871, 1009–1012]). The adaptive activation functions — i.e., functions that have a trainable parameter that changes their shape — have been receiving more attention recently (e.g., [871, 1013–1015]) and might become a new standard in the field. One of the first descriptions of the general adaptive activation function (AAF) approach is available in [1009] where Wu, Zhao, and Ding described an AF[43] that has one or more trainable parameters that are trained together with the rest of the network's weights [1009]. The simplest forms just add a parameter to a particular neural network that controls one of its properties (e.g., slope), while the more complex ones allow for the learning of a large number of activation functions (e.g., adaptive spline activation functions in [1012]).

4.3.1 *The ReLU-based family of adaptive functions*

The are numerous ReLU extensions that are adaptive [11]. Some of the adaptive activations have a non-adaptive counterpart — e.g., PReLU (see Section 4.3.1.1), which is basically a LReLU with an adaptive parameter of leakiness.

4.3.1.1 *Parametric rectified linear unit (PReLU)*

However, AAFs might be very useful even in the simplest form with a single added parameter — an AAF called PReLU was used to obtain a state-of-the-art result on the ImageNet Classification in 2015, the first surpassing human-level performance [871]. The PReLU generalize the ReLU by adding a parameter that controls the slope of the activation function for negative inputs (the ReLU is constant at zero for negative inputs) that is learned with other weights [1016]:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \frac{z_i}{a_i}, & z_i < 0, \end{cases} \tag{4.256}$$

where a_i is an optimized parameter for each neuron/filter i. The LReLU [869] is essentially a PReLU but with the parameter a_i fixed and not trainable (see Section 4.2.6.2 for LReLU details). PReLUs are better than ReLUs for verification-friendly NNs [1017].

43 The authors used the name trainable activation function (TAF) rather than the adaptive activation function (AAF) that is used throughout this work.

4.3.1.2 *Positive parametric rectified linear unit (PReLU⁺)*

The positive PReLU is an adaptive variant of the SlReLU (see Section 4.2.6.5) proposed in [1018]; it is also a special case of, for example, DPReLU, Dual Line, and piecewise linear unit (PiLU). It is defined as

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ 0, & z_i < 0, \end{cases} \tag{4.257}$$

where a_i is a trainable parameter [1018].

4.3.1.3 *Margin Relu*

The margin (MarReLU)[44] is an adaptive variant of the Shifted ReLU where the shift a_i is determined as the channel-by-channel expectation value of the negative response [1019]. It is defined as

$$f(z_i) = \max(z, a_i) = \begin{cases} z, & z_i - a_i \geq 0, \\ a_i, & z_i - a_i < 0. \end{cases} \tag{4.258}$$

4.3.1.4 *Funnel parametric rectified linear unit (FunPReLU)*

The funnel rectified linear unit (FunReLU)[45] and funnel (FunPReLU) are 2D AFs proposed in [1020]. The FunReLU and FunPReLU introduce a spatial context into the AF by comparing the input to a funnel condition instead of the zero that is used as the threshold in ReLU and PReLU [1020]. The FunReLU is defined as

$$f(z_{c,m,n}) = \max(z_{c,m,n}, \mathrm{t}(z_{c,m,n})), \tag{4.259}$$

where $z_{c,m,n}$ is the input on the c-th channel at the 2D spatial position m, n and $\mathrm{t}(z_{c,m,n})$ is the spatial context from a 3×3 window[46]

$$\mathrm{t}(z_{c,m,n}) = \sum_{m-1 \leq h \leq m+1, n-1 \leq w \leq n+1} z_{c,h,w} \cdot p_{c,h,w} \tag{4.260}$$

and $p_{c,h,w}$ denotes the coefficients on this window [1020]. The FunPReLU is defined similarly. The FunReLU was, for example, used in [1021, 1022].

4.3.1.5 *React-PReLU (RPReLU)*

The react- (RPReLU) is an adaptive variant of the PReLU with vertical and horizontal shifts; it is defined as

$$f(z_i) = \begin{cases} z_i - a_c + b_c, & z_i \geq a_c, \\ c_c(z_i - a_c) + b_c, & z_i < a_c, \end{cases} \tag{4.261}$$

44 Heo et al. abbreviated it as MReLU, but this abbreviation is used for the mirrored rectified linear unit (see Section 4.2.6.28) in this work.
45 Ma, Zhang, and Sun originally named the unit FReLU but its abbreviation would collide with the flexible ReLU.
46 Other sizes were also tested in [1020] but Ma, Zhang, and Sun found 3×3 to work the best.

where a_c, b_c, and c_c are trainable parameters for each channel c and z_i denotes the input to the neuron i in the channel c [1023]; a_c controls the horizontal shift, b_c controls the vertical shift, and c_c is the slope parameter for negative inputs as in the original PReLU.

4.3.1.6 *Smooth activation unit (SAU)*

The smooth activation unit (SAU) is a smoothed variant of the PReLU[47] using the convolution operation with the Gaussian function [1024]. It is defined as

$$f(z_i) = \left(\mathrm{PReLU}_{a_i} * \phi_{b_i}\right)(z_i), \tag{4.262}$$

where $*$ is the convolution operation, PReLU_{a_i} is the PReLU[48] parametrized by a_i[49] and $\phi_{b_i}(x)$ is the Gaussian function parameterized by b_i inversely controlling the deviation of the function [1024]. The resulting AF is then

$$f(z_i) = \frac{1}{2b_i}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{b_i^2 z_i^2}{2}\right) + \frac{1+\frac{1}{a_i}}{2}z_i + \frac{1-\frac{1}{a_i}}{2}z_i \cdot \mathrm{erf}\left(\frac{b_i z_i}{\sqrt{2}}\right), \tag{4.263}$$

where a_i and b_i are either fixed or trainable parameters [1024].

4.3.1.7 *Smooth maximum unit (SMU)*

The smooth maximum unit (SMU) [1025] is an AAF that uses a smooth approximation of the absolute value function. The SMU is defined as

$$f(z_i) = \frac{(1+a_i)\,z_i + (1-a_i)\,z_i\,\mathrm{erf}\left(b_i\,(1-a_i)\,z_i\right)}{2} \tag{4.264}$$

where a_i and b_i are learnable parameters [1025]. This smooth approximation of the absolute value function using the Gaussian error function could be used to create a whole class of AFs similarly as in Section 4.3.54.

4.3.1.8 *Leaky Learnable ReLU (LeLeLU)*

An adaptive LReLU variant named leaky learnable ReLU (LeLeLU) was proposed in [1026]. It is a LReLU with learnable scaling parameter:

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ 0.01 a_i z_i, & z_i < 0, \end{cases} \tag{4.265}$$

where a_i is a trainable parameter for each neuron i [1026].

47 The principle could be, however, applied to other AFs.
48 Biswas et al. used the LReLU in the definition of the SAU but since they consider the parameter a_i trainable, we stick to the usage of PReLU in the defintion.
49 To conform to the used definition of the PReLU unit, we will use the slope scaling by $\frac{1}{a_i}$ even though authors originally used the a_i for slope scaling of negative inputs.

4.3.1.9 *Parametric rectified exponential unit (PREU)*

Similarly as PReLU extends the ReLU (see Section 4.3.1.1), the PREU extends the swish and ELU inspired REU [932]. It is defined as

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ a_i z_i \cdot \exp(b_i z_i), & z_i < 0, \end{cases} \tag{4.266}$$

where a_i and b_i are trainable parameters for each neuron/filter i [932]. The advantage of PREU is that it uses the negative information near zero [11] — unlike the ReLU.

4.3.1.10 *Randomly translational PReLU (RT-PReLU)*

A randomly translational PReLU (RT–PReLU) an equivalent extension to PReLU as is RT–ReLU to ReLU (see Section 4.2.6.11) [883]. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i + b_i \geq 0, \\ \frac{z_i}{a_i}, & z_i + b_i < 0, \end{cases} \tag{4.267}$$

where a_i is a trainable parameter and b_i is a stochastic parameter for each neuron i randomly sampled from the Gaussian distribution at each iteration, $b_i \sim \mathrm{N}\left(0, \sigma^2\right)$, where σ^2 is the variance of the Gaussian distribution. The offset b_i is set to zero during the test phase [11]. The authors Cao et al. set the $\sigma^2 = 0.75^2$ for their experiments [883]. It is also possible to have the parameter b_i sampled for each neuron i, but the a_i is shared by all neurons in a channel c [1027].

4.3.1.11 *Probabilistic activation (ProbAct)*

A probabilistic class of activation functions ProbAct that adds a random noise to any activation function [1028]. It is defined as

$$f(z) = g(z) + \sigma e, \tag{4.268}$$

where $g(z)$ is any function (either fixed or trainable) defining the mean of the probabilistic activation, $e \sim \mathrm{N}(0, 1)$ is a random value sampled from a standard normal distribution, and σ is either fixed or learnable parameter controlling the range of the perturbation [1028]. σ can be either a global learnable parameter or different for each neuron i [1028]. The ProbAct used in [1028] is a ReLU based ProbAct defined as

$$f(z) = \max(0, z) + \sigma e, \tag{4.269}$$

which resembles NReLU (see Section 4.2.6.6) that adds random noise for output values for the positive inputs z [861]. A similar concept for sigmoid and tanh activation was used in [1029] (see Section 4.3.17).

The chaotic injections presented in [1030] represent a similar approach; however, the injections are not stochastic but rather defined using the chaos theory. Furthermore, Reid, Ferens, and Kinsner discuss several approaches for injections of a chaotic value s_n into a ReLU: $\mathrm{ReLU}\left(z + s_n\right)$, $\mathrm{ReLU}\left(z \cdot s_n\right)$, $\mathrm{ReLU}\left(z + z \cdot s_n\right)$, $\mathrm{ReLU}\left(z\right) + s_n$, $\mathrm{ReLU}\left(z\right) \cdot s_n$, $\mathrm{ReLU}\left(z\right) + zs_n$, and $\mathrm{ReLU}\left(z\right) + \mathrm{ReLU}\left(z\right) \cdot s_n$ [1030].

4.3.1.12 *Adaptive offset activation function (AOAF)*

Another ReLU variant with adaptive shift termed adaptive offset activation function (AOAF) was defined in [1031]. The AOAF introduces two hyperparameters and one data-dependent adaptive parameter; it is defined as

$$f(z_i) = \max\left(0, z_i - ba_i\right) + ca_i, \tag{4.270}$$

where b and c are predefined, fixed parameters and a_i is the mean value of the inputs of neuron i [1031]. The recommended values for the parameters b and c are $b = c = 0.17$ as it yielded the best image classification accuracy in the experiments [1031].

4.3.1.13 *Dynamic leaky ReLU (DLReLU)*

An error based dynamic leaky ReLU (DLReLU) was proposed in [1032] (under the name of Dynamic ReLU — DReLU — but this naming collides with DReLU established in [1033, 1034]; see Section 4.3.1.14 and Section 4.3.55.3). The DLReLU is a LReLU where the leakiness depends on the test error from the previous epoch [11]

$$f(z) = \begin{cases} z, & z \geq 0, \\ ab_t z, & z < 0, \end{cases} \tag{4.271}$$

where $a \in (0, 1)$ is a predefined parameter controlling the leakiness similarly as in LReLU (see Section 4.2.6.2) and b_t is a dynamic parameter computed for current training epoch t as the test erroch from the previous epoch $t-1$, i.e., $b_t = \mathrm{MSE}_{t-1}$ [1032].

A version exp–DLReLU was proposed to deal with deeper networks with more than seven hidden layers in order to avoid too large error values causing the training to fail [1032]:

$$f(z) = \begin{cases} z, & z \geq 0, \\ ac_t z, & z < 0, \end{cases} \tag{4.272}$$

where $c_t = \exp(-b_t) = \exp\left(\mathrm{MSE}_{t-1}\right)$ [1032].

The advantage of DLReLU and exp–DLReLU is that the changes in leakiness are big at the beginning of the training due to higher test error and diminish towards the end [1032]. A similar effect could be obtained by a schedule of the leakiness parameter in the LReLU.

4.3.1.14 *Dynamic ReLU (DReLU)*

Similar approach to the ABReLU is presented by the DReLU [1033] (not to be confused with identically named activations in [1032, 1034]),

$$f(z_i) = \begin{cases} z_i, & z_i - a_i \geq 0, \\ a_i, & z_i - a_i < 0, \end{cases} \tag{4.273}$$

where a_i is a threshold value that is computed as the midpoint of the range of input values for each batch; e.g., if the values range from -4 to 8, then $a_i = \frac{-4+8}{2} = 2$ [1033]. The DReLU can be considered to be a variant of DisReLU (see Section 4.2.6.44) with data-dependent determination of the shifting point.

4.3.1.15 *Flexible ReLU (FReLU)*

The FReLU is a ReLU extension with zero-like property and the ability to capture negative information [1035]. The zero-like property is the ability to push activation means closer to zero [1035] as this might speed up learning [874]. The FReLU builds on the ability to shift the AF

$$f(z_i) = \mathrm{ReLU}\left(z_i + a_i\right) + b_i, \tag{4.274}$$

where a_i and b_i would be optimized parameters [1035]. However, since the parameter a_i can be learned by the bias term of the neuron to whose output is the activation function applied, the authors Qiu, Xu, and Cai formulate the FReLU as

$$f(z_i) = \mathrm{ReLU}\left(z_i\right) + b_i = \begin{cases} z_i + b_i, & z_i \geq 0, \\ b_i, & z_i < 0, \end{cases}, \tag{4.275}$$

where b_i is a trainable parameter [1035].

4.3.1.16 *Adaptive shifted ReLU (ShiLU)*

An adaptive shifted ReLU (ShiLU) [889] is another adaptive variant of the ReLU activation; it is a variant that adds a trainable slope and a vertical shift:

$$f(z_i) = a_i \mathrm{ReLU}\left(z_i\right) + b_i = a_i \cdot \max\left(0, z\right) + b_i, \tag{4.276}$$

where a_i and b_i are trainable parameters for each neuron i [889]. They used the name *shifted ReLU*, but that name is already taken by the non-adaptive Shifted ReLU; hence, the full name is adaptive shifted ReLU throughout this work to avoid confusion.

4.3.1.17 *StarReLU*

The StarReLU [1036] is an adaptive version of the RePU of power 2 using a similar approach as the ShiLU; it is defined as

$$f(z_i) = a_i\left(\text{ReLU}\left(z\right)\right)^2 + b_i, \tag{4.277}$$

where a_i and b_i are trainable parameters for each neuron i [1036]. If the parameters are not used in an adaptive manner, Yu et al. recommend setting $a_i = 0.8944$ and $b_i = -0.4472$ [1036].

4.3.1.18 *Adaptive HardTanh*

An adaptive variant of HardTanh was used in [894]; it is defined as

$$f(z) = \text{HardTanh}\left(a_t\left(z - b\right)\right), \tag{4.278}$$

where a_t is a scale factor for each epoch t such that $1 \leq a_1 \leq a_2 \leq \ldots \leq a_t \leq \ldots \leq a_T$, T is the total number of training epochs and b is an adaptive parameter trained using BP with other parameters of the NN [894]. The parameters a_t are set such that the function starts in a similar shape as a regular HardTanh (see Section 4.2.6.18) and gradually approaches the sign function [894]. This allows for training a network that will gradually become a binary NN where each activation is the sign function which can be used for speeding the inference [894].

4.3.1.19 *Attention-based ReLU (AReLU)*

Attention-based ReLU (AReLU) [1037] is a adaptive ReLU variant that uses ELSA — element-wise attention mechanism proposed in [1037]. It is defined as

$$f(z_l) = \begin{cases} \left(1 + \sigma(b_l)\right) z_l, & z_l \geq 0, \\ C(a_l) z_l, & z_l < 0, \end{cases}, \tag{4.279}$$

where a_l and b_l are learnable parameters for each layer l, $\sigma(x)$ is the logistic sigmoid function, $C(x)$ is a function that clips the input into $[0.01, 0.99]$ [1037]. The derivative of $C(a_l)$ is handled by just not using the BP when $a_l < 0.01$ or $a_l > 0.99$ [1037]. While Chen, Li, and Xu observe that the parameters a_l and b_l are insensitive to the initialization, they recommend initializing $a_l = 0.9$ and $b_l = 2.0$ as a larger initial value of b_l can speed up the convergence [1037]. The AReLU was found to outperform CELU, ELU, GELU, LReLU, Maxout, Relu, RReLU, SELU, sigmoid, softplus, swish, tanh, adaptive piece-wise linear unit (APLU), Padé activation unit (PAU), PReLU, and self-learnable activation function (SLAF) in experiments with various learning rates in [1037]. The performance of AReLU was validated under different settings in [698, 821, 1038].

4.3.1.20 *Dual parametric ReLU (DPReLU) and Dual Line activation function*

A DPReLU [1039] extends the concept of PReLU even further:

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ b_i z_i, & z_i < 0, \end{cases} \tag{4.280}$$

where a_i and b_i are trainable parameters for each neuron i; these are initialized the same as PReLU — $a_i = 1$, $b_i = 0.01$ [1039]. The DPReLU was also later proposed independently in [1040] under the name *fully parametric ReLU* and the abbreviation FReLU (which is already used in the literature for the flexible ReLU; see Section 4.3.1.15).

4.3.1.21 *Dual Line*

The DPReLU was further extended into a Dual Line activation function that adds a shift parameter

$$f(z_i) = \begin{cases} a_i z_i + m_i, & z_i \geq 0, \\ b_i z_i + m_i, & z_i < 0, \end{cases} \tag{4.281}$$

where a_i and b_i are trainable parameters for each neuron i the same as in DPReLU and m_i is an additional trainable shift parameter for each neuron or filter i; m_i was initialized to $m_i = -0.22$ [1039].

4.3.1.22 *Piecewise linear unit (PiLU)*

An AF very similar to the Dual Line is the piecewise linear unit (PiLU) proposed in [1041]; it just extends the Dual Line concept by adding horizontal shifts. It is defined as

$$f(z_i) = \begin{cases} a_i z_i + c_i(1 - a_i), & z_i \geq c_i, \\ b_i z_i + c_i(1 - b_i), & z_i < c_i, \end{cases} \tag{4.282}$$

where a_i, b_i, and c_i are adaptive parameters for each neuron i [1041]. The PiLU geneneralizes, for example, ReLU, LReLU, PReLU, SlReLU, DPReLU, and Dual Line.

4.3.1.23 *Dual parametric family of activation functions*

The DPReLU approach (see Section 4.3.1.20) can be extended to a general concept transforming any activation function $g(z)$:

$$f(z_i) = \begin{cases} a_i g(z_i) + m_i, & z_i \geq 0, \\ g(z_i) + m_i, & z_i < 0, \end{cases} \tag{4.283}$$

where $g(z_i)$ is any activation function and a_i and m_i are trainable parameters for each neuron i [1039]. The functions of this family are called dual parametric activation functions (DPAFs) throughout this text.

4.3.1.24 *Fully parameterized activation function (FPAF)*

Similar approach to DPAF (see Section 4.3.1.23) was proposed under the name of fully parameterized activation function (FPAF) in [1040]; the FPAF is defined as

$$f(z_i) = \begin{cases} a_i g_1(z_i), & z_i \geq 0, \\ b_i g_2(z_i), & z_i < 0, \end{cases} \tag{4.284}$$

where a_i and b_i are trainable parameters for each neuron i and $g_1(z_i)$ and $g_2(z_i)$ can be any function [1040]. The FPAF, in contrast to the family of DPAFs, has no trainable shift but allows for learnable slopes for both parts of the piecewise function.

4.3.1.25 *Elastic PReLU (EPReLU)*

The same as EReLU extends the concept of ReLU (see Section 4.2.6) [918], the Elastic (EPReLU) extends the PReLU — it adds a varying coefficient to the positive part of the PReLU [918]:

$$f(z_i) = \begin{cases} k_i z_i, & z_i \geq 0, \\ \frac{z_i}{a_i}, & z_i < 0, \end{cases} \tag{4.285}$$

where a_i is the optimized parameter, k_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(1-\alpha, 1+\alpha)$ where $\alpha \in (0,1)$ [918]. A modified training procedure for EPReLU is also proposed — the neuron weights and the trainable parameter a_i are updated with $k_i = 1$ in odd epochs, while in even epochs, the a_i is kept fixed, the parameter k_i is sampled from the uniform distribution, and only the neuron weights are updated [918]. It was shown that the EPReLU leads to improved performance over the ReLU, PReLU, EReLU, APLU, network in network (NIN), and maxout unit networks on several datasets [918].

4.3.1.26 *Paired ReLU*

A paired ReLU [1042] is a concept similar to CReLU (see Section 4.2.6.34), but it introduces four trainable parameters. It is defined as

$$\boldsymbol{f}(z) = \begin{bmatrix} \max\left(a_i z_i - b_i, 0\right) \\ \max\left(c_i z_i - d_i, 0\right) \end{bmatrix}, \tag{4.286}$$

where a_i, b_i, c_i, and d_i are trainable parameters for each neuron i [11, 1042]. The parameters a_i and c_i are scale parameters and b_i and d_i are trainable thresholds; the inital values of scale parameters are $a_i = 0.5$ and $c_i = -0.5$ [1042].

4.3.1.27 *Tent*

The tent is a ReLU-based AF proposed in [1043]; it is defined as

$$f(z_i) = \max\left(0, a_i - |z_i|\right), \tag{4.287}$$

where a_i is a trainable parameter [1043]. Rozsa and Boult recommend using batch normalization and initializing $a_i = 1$ [1043]. Also, having a weight decay on the parameter a_i during training proved beneficial for certain tasks [1043].

4.3.1.28 *Hat*

The hat [1044] is an AAF very similar to the tent AF — the only difference is that the tent AF is centered around zero while the hat is positive only for positive inputs. The hat AF is defined as

$$f(z_i) = \begin{cases} 0, & z_i < 0, \\ z_i, & 0 \leq z_i \leq \frac{a_i}{2}, \\ z_i, & \frac{a_i}{2} \leq z_i \leq a_i, \\ 0, & z_i > a_i, \end{cases} \tag{4.288}$$

where a_i is can be either fixed or trainable parameter [1044]. Wang, Xu, and Zhu used $a_i = 2$ for the fixed variant in [1044]; this value was also used in [1045].

4.3.1.29 *ReLU memristor-like activation function (RMAF)*

The ReLU memristor-like activation function (RMAF) is an activation function similar to the swish AF (see Section 4.3.3.1) and it also attempts to leverage the negative values [1046]. It is defined as

$$f(z_i) = \left[b \frac{1}{\left(0.25\left(1+\exp(-z_i)\right)+0.75\right)^c} \right] a_i z_i, \tag{4.289}$$

where a_i is a trainable parameter initialized $a_i = 1$ for each neuron i or it is a fixed hyperparameter and b and c are fixed hyperparameters [1046].

4.3.1.30 *Parametric tanh linear unit (PTELU)*

A parametric tanh linear unit (PTELU) [1047] is an adaptive function that, for positive inputs, behaves just as a ReLU; however, the negative part is parameterized tanh function [11]. It can also be seen as an extension of the PReLU (see Section 4.3.1.1). It is an adaptive variant of ThLU (see Section 4.2.7.1). It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \tanh\left(b_i z_i\right), & z_i < 0, \end{cases} \tag{4.290}$$

where a_i and b_i are trainable parameters for each neuron i; $a_i \geq 0$ and $b_i \geq 0$ [1047]. It has output range of $[-a_i, \infty)$ [11]. The parameter a_i controls the saturation value, and the parameter b_i controls the convergence rate [1047]. While the AF resembles an adaptive extension of an ELU activation functions, the author Gupta and Duggal decided to use tanh function for the negative inputs because it gives a higher gradient for small negative inputs and saturates earlier than $\exp(z) - 1$ and thus the noise-robust deactivation state earlier and faster [1047]. The nonadaptive variant of PTELU with $a_i = 1$ and $b_i = 1$ was proposed in [815].

4.3.1.31 *Tangent linear unit (TaLU)*

The tanh linear unit (TaLU) [1048] is an AF similar to the PTELU. The TaLU is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \tanh(z_i) & a_i < z_i < 0, \\ \tanh(a_i) & z_i \leq a_i, \end{cases} \tag{4.291}$$

where $a_i < 0$ is either a learnable[50] or fixed parameter [1048].

4.3.1.32 *PTaLU*

The PTaLU[51] is a variant of TaLU with another learnable parameter [1048]. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq b_i, \\ \tanh(z_i) & a_i < z_i < b_i, \\ \tanh(a_i) & z_i \leq a_i, \end{cases} \tag{4.292}$$

where a_i and b_i are trainable parameters [1048]. Mercioni and Holban used initial values $a_i = -0.75$ and $b_i = 1$ in [1048].

4.3.1.33 *TanhLU*

The tanhLU[52] is a parametric combination of the tanh and a linear function proposed in [1049]. It is defined as

$$f(z_i) = a_i \cdot \tanh(b_i z_i) + c_i z_i, \tag{4.293}$$

where a_i, b_i, and c_i are trainable parameters for each neuron i [1049].

4.3.1.34 *TeLU*

Despite the similar name, the tanh exponential linear unit (TeLU)[53] [844] is quite different from the PTELU. The TeLU is closely related to the mish and TanhExp activations, but, unlike these two AFs, it also has an additional adaptive parameter. It is defined as

$$f(z_i) = z_i \cdot \tanh(\mathrm{ELU}(a_i z_i)), \tag{4.294}$$

where a_i is either learnable or fixed scaling parameter [844].

50 The variant with the adaptive parameter was named *TaLU learnable* by the authors.
51 Not an abbreviation but a name given by Mercioni and Holban in [1048].
52 Not an abbreviation but a name given by Shen et al. in [1049].
53 [844] used the TeLU as the name and not as an abbreviation; nevertheless, the long name tanh exponential linear unit fits the usual naming convention and, therefore, it is used in this work.

4.3.1.35 *Tanh based ReLU (TReLU)*

A TReLU was proposed in [1050]; however, it is is only a special case of previously proposed PTELU (see Section 4.3.1.30). It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \tanh\left(b_i z_i\right), & z_i < 0, \end{cases} \tag{4.295}$$

where b_i is a trainable parameter for each neuron i[1050]. This function is identical to the PTELU with its parameter a_i fixed to $a_i = 1$. Another special case of PTELU was proposed in [1051] also under the name of TReLU — this time, the parameter a_i becomes a predefined fixed parameter, and b_i becomes fixed to $b_i = 1$. This function is denoted TReLU variant 2 (TReLU2) in this work and is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a \tanh\left(z\right), & z < 0, \end{cases} \tag{4.296}$$

where a is fixed[54] parameter [1051].

4.3.1.36 *Rectified linear tanh (ReLTanh)*

A ReLTanh is a piecewise adaptive activation function that improves traditional tanh activation function [1052] — it replaces the positive and negative saturated regions of the tanh activation functions with straight lines whose slopes are identical to the slope of the tanh at the thresholds [1052]. It is defined as

$$f(z_i) = \begin{cases} \tanh'(a_i)(z_i - a_i) + \tanh(a_i), & z_i \leq a_i, \\ \tanh(z_i), & a_i < z_i < b_i, \\ \tanh'(b_i)(z_i - b_i) + \tanh(b_i), & z_i \geq b_i, \end{cases} \tag{4.297}$$

where $\tanh'(x)$ is the derivative of the tanh function

$$\tanh'(x) = \frac{4}{\left(\exp(x) + \exp(-x)\right)^2}, \tag{4.298}$$

and $a_i \in [a_{\text{low}}, a_{\text{high}}]$ and $b_i \in [b_{\text{low}}, b_{\text{high}}]$ are two trainable parameters that may be defined for each neuron i but are rather recommended to be shared by a whole layer l [1052] in order to decrease computational burden. The limits a_{low}, a_{high}, b_{low}, and b_{high} for the parameters are to constraint the learnable range and are predefined hyperparameters. Wang et al. used $a_{\text{low}} = -\infty$, $a_{\text{high}} = -1.5$, $b_{\text{low}} = 0$, and $b_{\text{high}} = 0.5$ in their work [1052]. The initial values were set to $a_i = -1.5$ and $b_i = 0$ for all layers (the parameters were shared layer-wise) in order to speed up the training process in early stages by the larger gradients [1052].

54 While Nakhua et al. used the parameter fixed during their experiments, they also speculated that making it learnable might improve the performance.

4.3.1.37 *Bendable linear unit (BLU)*

A BLU [1053] is an adaptive function that allows for any interpolation between the identity function and a rectifier [11, 1053]. It is defined as

$$f(z_i) = a_i \left(\sqrt{z_i^2 + 1} - 1 \right) + z_i, \tag{4.299}$$

where $a_i \in [-1, 1]$ is a trainable parameter for each neuron or filter [1053]. One of the main advantages of the BLU is that it can model an identity function; the identity function is useful because its gradient cannot vanish or explode, and it also allows for a layer to be "skipped" [1053] — it is one of the reasons of why ResNets [13] became so popular [1053] as it is rather hard to learn an identity transformation using conventional neural network and the architecture with skip connections allows for easy learning of the identity mapping [13]. Unless the magnitude $|a_i|$ is exactly 1, the derivative of BLU is non-zero for both positive and negative inputs (similarly to LReLU and in contrast to vanilla ReLU and ELU) [1053]. BLU has a slope higher than 1 for positive inputs for a_i approaching 1 (or for negative inputs for a_i approaching -1) [1053]; this property helps to avoid vanishing gradient problems [945, 1053]. Another useful benefit is that BLU are C^∞ continuous [1053], which can be theoretically exploited for speeding up the optimization [1053], e.g., [1054–1057]. It was also shown that smooth activation functions provide better signal propagation [648].

4.3.1.38 *Rectified BLU (ReBLU)*

A variant of the BLU (see Section 4.3.1.37) was proposed in [1018] under the name rectified BLU (ReBLU). It is defined as

$$f(z_i) = \begin{cases} \mathrm{BLU}(z_i), & z_i > 0, \\ 0, & z_i \leq 0, \end{cases} = \begin{cases} a_i \left(\sqrt{z_i^2 + 1} - 1 \right) + z_i, & z_i > 0, \\ 0, & z_i \leq 0, \end{cases} \tag{4.300}$$

, where a_i is a trainable parameter [1018].

4.3.1.39 *DELU*

The DELU[55] activation function [889] is a ReLU variation that utilizes the SiLU function (see Section 4.2.3). It is defined as

$$f(z_i) = \begin{cases} (a_i + 0.5)z_i + |\exp(-z_i) - 1|, & z_i \geq 0, \\ z_i \sigma(z_i), & z_i < 0, \end{cases} \tag{4.301}$$

where a_i is a trainable parameter for each neuron i and $\sigma(z_i)$ is the logistic sigmoid function [889].

55 DELU is not an abbreviation but rather a name given by Pishchik.

4.3.1.40 *Soft clipping mish*

A ReLU variant called soft clipping mish (SC-mish) was proposed in [1058]. It adds soft clipping to the positive inputs using the mish AF; it is defined as

$$f(z_i) = \max\left(0, z_i \cdot \tanh\left(\text{softplus}(a_i z_i)\right)\right), \tag{4.302}$$

where a_i is a fixed parameter [1058]; Mercioni and Holban used $a_i = 1$. It also has a variant where the parameter a_i is trainable. Such a variant is called soft clipping learnable mish (SCL-mish). When using the SCL-mish, Mercioni and Holban initalized the parameter $a_i = 0.25$ [1058].

4.3.1.41 *Soft clipping swish*

Yet another AF proposed by Mercioni and Holban is the soft clipping swish (SC-swish) [1059–1061]. This function is very similar to SC-mish and is defined as

$$f(z) = \max\left(0, z \cdot \sigma\left(z\right)\right), \tag{4.303}$$

where $\sigma(z)$ is the logistic sigmoid [1059].

4.3.1.42 *Parametric swish (p-swish)*

The parametric swish (p-swish) [1062] is another AF proposed by Mercioni and Holban. It is defined as

$$f(z_i) = \begin{cases} a_i z_i \sigma\left(b_i z_i\right), & z_i \leq c_i, \\ z_i, & z_i > c_i, \end{cases} \tag{4.304}$$

where a_i, b_i, and c_i are either trainable or fixed parameters (or combination thereof) [1062]. The parameters were initialized to $a_i = 1$, $b_i = 1$ and $c_i = 0$ in experiments in [1062]. An AF named R_S similar to the p-swish was independently proposed in [1063]; it is equivalent to a p-swish with fixed $a_i = 1$.

4.3.1.43 *Parametric exponential linear unit (PELU)*

Similarly as PReLU extends the concept of ReLU, the parametric exponential linear unit (PELU) [1064] extends the concept of ELU (see Section 4.2.6.48). The PELU builds on a parameterization that separately controls the saturation point, the decay, and the slope:

$$f(z_i) = \begin{cases} c_i z_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{b_i}\right) - 1\right), & z_i < 0, \end{cases} \tag{4.305}$$

where a_i, b_i, and c_i are trainable parameter for each neuron i [1064]. However, the PELU introduces only two new parameters — a_i controlling the saturation point, and b_i controlling the decay — to control the shape of the activation function; the slope is not controlled separately through another parameter as it could lead to non-differentiability at $z_i = 0$; therefore the slope is set such that the derivatives on both sides of zero are equal which leads to $c_i = \frac{a_i}{b_i}$ [1064] and therefore the PELU is defined as

$$f(z_i) = \begin{cases} \frac{a_i}{b_i} z_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{b_i}\right) - 1 \right), & z_i < 0. \end{cases} \tag{4.306}$$

The PELU combined with mixing different activation functions which use an adaptive linear combination or hierarchical gated combination of activation function was shown to perform well [1065] — see Section 4.3.1.45.

4.3.1.44 *Extended exponential linear unit (EDELU)*

An adaptive function called extended exponential linear unit (EDELU)[56] was proposed in [1066]. This function is the same as the PELU, but it omits the vertical scaling for positive inputs, adds a parameter controlling the threshold, and uses inverse definitions of the parameters present in the PELU. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq c_i, \\ \frac{(\exp(a_i z_i) - 1)}{b_i}, & z_i < c_i, \end{cases} \tag{4.307}$$

where $a_i \geq 0$[57] and $b_i \geq 0$[58] are trainable PELU parameters and $c_i \geq 0$ is the novel parameter for controlling the threshold that has to satisfy the relationship

$$b_i c_i = \exp(a_i c_i) - 1; \tag{4.308}$$

while the $c_i = 0$ is always a solution of the equation, there are other solutions for $b_i > a_i > 0$ [1066].

4.3.1.45 *Adaptive combination of PELU and PReLU*

Two different activation functions can be mixed together, as shown in [1065]. One such example of mixed activation function is

$$f(z_i) = a_i \cdot \text{LReLU}(z_i) + (1 - a_i)\, \text{ELU}(z_i), \tag{4.309}$$

where a_i is a combination coefficient that might be learned from the data [1065]. Another mixing approach was shown for combining PReLU and PELU:

$$f(z_i) = \sigma(a_i z_i)\text{PReLU}(z_i) + (1 - \sigma(a_i z_i))\, \text{PELU}(z_i), \tag{4.310}$$

where $\sigma(x)$ is the logistic sigmoid [1065]. Qian et al. also proposed other mixing schemes such as hierarchical activation, winner-take-all selection whose performance were shown on the MNIST [45], CIFAR-10 and CIFAR-100 [243] datasets; see [1065] for details.

56 Authors called the function *extendeD ELU* resulting in an abbreviation DELU but that name is already taken by an AF proposed a few months earlier in [889].

57 The PELU equivalent would be $\frac{1}{b_i}$.

58 The PELU equivalent would be $\frac{1}{a_i}$.

4.3.1.46 *Fast exponential linear unit (FELU)*

An ELU variant called fast exponential linear unit (FELU) aiming at efficient training and network inference was proposed in [1067] — it is inspired by fast approximation of the exponential function proposed in [1068] to replace the exponential in the ELU:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \left(2^{\frac{z_i}{\ln(2)}} - 1\right), & z_i < 0, \end{cases} \tag{4.311}$$

where a_i is a trainable parameter controlling the soft saturation region [1067].

4.3.1.47 *P+FELU*

Adem proposed variant of the FELU function named P+FELU; this variant has an added parameter and is defined as

$$f(z_i) = \begin{cases} z_i + b, & z_i \geq 0, \\ a_i \left(2^{\frac{z_i}{\ln(2)}} - 1\right) + b, & z_i < 0, \end{cases} \tag{4.312}$$

where a_i is a trainable parameter same as the original FELU and b is the added trainable parameter [1069].

4.3.1.48 *Multiple parametric exponential linear unit (MPELU)*

A PELU extension, multiple parametric exponential linear unit (MPELU) [1070], uses two trainable parameters to allow for a combination of a ReLU and ELU [11]. The multiple parametric exponential linear unit (MPELU) is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \left(\exp\left(b_i z_i\right) - 1\right), & z_i < 0, \end{cases} \tag{4.313}$$

where a_i and b_i are trainable parameters for each neuron i [11, 1070]. The ReLU, certain parameterizations of PELU, and ELU are special cases of the MPELU [1070]. A special method for weight initialization of neurons with MPELU units was also proposed; depending on a particular initialization, the method can become the initialization for ELU networks or for ReLU networks [1070]. The MSRA[59] filler approach [871] can be considered as a special case of the MPELU initialization [1070]. The MPLU initialization is a similar approach to LSUV initialization [868], but unlike the LSUV initialization, it provides an analytic solution for ELU and MPELU and therefore it has lower computational costs [1070]. It was also shown that the MPELU works better with batch normalization compared to the vanilla ELU [1070]. The performance of the MPELU was empirically shown on the CIFAR-10 and CIFAR-100 datasets [243] using multiple neural network architectures [1070], e.g. nine-layer deep NIN [1072] or even a ResNet with 1001 layers [13].

59 The initialization method was unnamed in the original paper [871] but was later named *Microsoft Research Asia* (MSRA) filler [1071].

4.3.1.49 *P-E2-ReLU*

The AAF named P-E2-ReLU is combining two ELUs and a ReLU using two adaptive parameters [1073]. It is defined as

$$f(z_i) = a_i \cdot \mathrm{ReLU}(z_i) + b_i \cdot \mathrm{ELU}(z_i) + (1 - a_i - b_i) \cdot (-\mathrm{ELU}(-z_i)), \quad (4.314)$$

where a_i and b_i are trainable parameters for each neuron i [1073]. The parameters were initialized to $a_i = 0.4$ and $b_i = 0.3$ in experiments in [1073]. Jie et al. mentioned that other combinations could be considered and called this family P-E2-XU. One such combination is denoted P-E2-Id and is defined as

$$f(z_i) = a_i z_i + (1 - a_i) \cdot (mathrmELU(z_i) - \mathrm{ELU}(-z_i)), \quad (4.315)$$

and another is named P-E2-ReLU-1

$$f(z_i) = a_i \cdot \mathrm{ReLU}(z_i) + (1 - a_i) \cdot (mathrmELU(z_i) - \mathrm{ELU}(-z_i)), \quad (4.316)$$

whera a_i is a trainable parameter in both AAFs [1073]. The parameter was initialized to $a_i = 0.5$ in experiments in [1073].

4.3.1.50 *Soft exponential*

The soft exponential activation function is an adaptive activation function that is able to interpolate between logarithmic, linear, and exponential functions [1074]. It is defined as

$$f(z_i) = \begin{cases} \frac{\exp(z_i)-1}{a_i} + a_i, & a_i > 0, \\ z_i, & a_i = 0, \\ -\frac{\ln(1-a_i(z_i+a_i))}{a_i}, & a_i < 0, \end{cases} \quad (4.317)$$

where a_i is a trainable parameter [1074]. The soft exponential activation functions is continuously differentiable with respect to z_i and also with respect to a_i [1074]; furthermore, for any constant a_i, the function is monotonic [1074]. When $a_i = -1$, the function becomes $f(z_i) = \ln(z_i)$, while for $a_i = 0$ it becomes linear function $f(z_i) = z_i$ and for $a_i = 1$, it is the exponential function $f(z_i) = \exp(z_i)$ [1074].

4.3.1.51 *Continuously differentiable ELU (CELU)*

A CELU was proposed in [1075]; CELU is very similar to the original parameterization of ELU [11] but reformulated such that the derivative at $z_i = 0$ is 1 for all values of a_i [1075]. CELU is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{a_i}\right) - 1\right), & z_i < 0, \end{cases} \quad (4.318)$$

where a_i is a learnable parameter for each neuron i. Its main advantages are that its derivative with respect to z_i is bounded and that it contains both the linear transfer function and ReLU [1075].

4.3.1.52 *Erf-based ReLU (ErfReLU)*

The Erf-based ReLU (ErfReLU) [1076] is an AAF similar to the ELU. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \text{erf}\left(z_i\right), & z_i < 0, \end{cases} \tag{4.319}$$

where a_i is a learnable parameter for each neuron i and $\text{erf}\left(z_i\right)$ is the Gauss error function [1076].

4.3.1.53 *Parametric scaled exponential linear unit (PSELU)*

A parametric scaled exponential linear unit (PSELU) [1077] is basically a SELU (see Section 4.2.7.11) where the parameters a and b controlling the behavior are trainable. It is defined as

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ a_i b_i \left(\exp\left(z_i\right) - 1\right), & z_i < 0, \end{cases} \tag{4.320}$$

where a_i and b_i are trainable parameters for each neuron i.

4.3.1.54 *Leaky parametric scaled exponential linear unit (LPSELU)*

A leaky parametric scaled exponential linear unit (LPSELU) [1077] is a leaky extension of the PSELU (see Section 4.3.1.53) to avoid small gradients hindering the learning process [1077]:

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ a_i b_i \left(\exp\left(z_i\right) - 1\right) + c_i z_i, & z_i < 0, \end{cases} \tag{4.321}$$

where a_i and b_i are trainable parameters for each neuron i and c_i is either a predefined constant or a trainable parameter [1077].

4.3.1.55 *Leaky parametric scaled exponential linear unit with reposition parameter (LPSELU_RP)*

The LPSELU can be extended by a reposition parameter similarly as FReLU extends ReLU (see Section 4.3.1.15) [1077]; such function is called LPSELU_RP and is defined as

$$f(z_i) = \begin{cases} a_i z_i + m_i, & z_i \geq 0, \\ a_i b_i \left(\exp\left(z_i\right) - 1\right) + c_i z_i + m_i, & z_i < 0, \end{cases} \tag{4.322}$$

where a_i and b_i are trainable parameters for each neuron i, and c_i is either a predefined constant or a trainable parameter the same as for LPSELU (see Section 4.3.1.54) and m_i is a trainable reposition parameter [1077]. It was empirically observed that the shift parameter m_i converges to a small negative value, which supports the hypothesis that the negative output of activation functions is important [1077].

4.3.1.56 *Shifted ELU family*

A family of several activation functions, shifted exponential linear unit, was proposed in [1078]; functions in this family have either vertical or horizontal shift of an ELU activation function that can be either constant or trainable. An ELU with fixed horizontal shift is ShELU, with fixed vertical SvELU and PELU (see Section 4.3.1.43) with trainable horizontal shift is PShELU [1078]. The ShELU is defined as

$$f(z) = \begin{cases} z+b, & z+b \geq 0, \\ a\left(\exp\left(z+b\right)-1\right), & z+b < 0, \end{cases} \tag{4.323}$$

where a is a fixed parameter similarly as in the vanilla ELU and b is novel, preset parameter controlling the horizontal shift [1078]. The SvELU is defined similarly:

$$f(z) = \begin{cases} z+b, & z \geq 0, \\ a\left(\exp\left(z\right)-1\right)+b, & z < 0, \end{cases} \tag{4.324}$$

where a is a fixed parameter similarly as in the vanilla ELU and b is novel, preset parameter controlling the vertical shift [1078]. Grelsson and Felsberg define also a variant of PELU with horizontal shift called PSheLU:

$$f(z_i) = \begin{cases} \frac{a_i}{b_i}\left(z_i+c_i\right), & z_i+c_i \geq 0, \\ a_i\left(\exp\left(\frac{z_i+c_i}{b_i}\right)-1\right), & z_i+c_i < 0, \end{cases} \tag{4.325}$$

where a_i and b_i are trainable parameters of the original PELU, and c_i is a novel trainable parameter controlling the horizontal shift for each neuron i [1078]. For some reason, Grelsson and Felsberg did not propose a PELU with vertical shift (PSvELU), but it could be defined in a similar manner

$$f(z_i) = \begin{cases} \frac{a_i}{b_i}z_i+c_i, & z_i \geq 0, \\ a_i\left(\exp\left(\frac{z_i}{b_i}\right)-1\right)+c_i, & z_i < 0, \end{cases} \tag{4.326}$$

where a_i, b_i are trainable parameters of the original PELU and c_i is a novel trainable parameter controlling the vertical shift. Note that the shifted activation functions with horizontal shifts are equivalent to non-shifted variants with biases that are individual for each neuron and not shared in the same tiling pattern as the convolutional kernel [1078].

4.3.1.57 *Tunable swish (T-swish)*

The tunable swish (T-swish) proposed in [1079] is an AAF combining the ELU, E-swish (see Section 4.3.3.4) and swish (see Section 4.3.3.1) as it has trainable parameters for both horizontal and vertical scaling for negative inputs. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq c_i, \\ a_i z_i \cdot \sigma(b_i z_i), & z_i < c_i, \end{cases} \tag{4.327}$$

where a_i, b_i, and c_i are either fixed or trainable parameters for each neuron i [1079].

4.3.1.58 *Rectified parametric sigmoid unit (RePSU)*

The rectified parametric sigmoid unit (RePSU) is an AAF proposed in [1080]; it consists of a linear combination of two components — rectified parametric sigmoid shrinkage unit (RePSKU) and rectified parametric sigmoid stretchage unit (RePSHU). It is defined as

$$f(z_i) = a_i \mathrm{RePSKU}_{b_i,c_i,d_i,e_i}(z_i) + (1 - a_i)\,\mathrm{RePSHU} b_i, c_i, d_i, e_i\,(z_i)\,, \quad (4.328)$$

where

$$\mathrm{RePSKU}_{b_i,c_i,d_i,e_i}(z_i) = \begin{cases} \frac{z_i - b_i}{1+\exp\left(-\mathrm{sgn}(z_i-c_i)\left(\frac{|z_i-c_i|}{d_i}\right)^{e_i}\right)}, & z_i \geq b_i, \\ 0, & z_i < b_i, \end{cases} \quad (4.329)$$

$$\mathrm{RePSHU}_{b_i,c_i,d_i,e_i}(z_i) = \begin{cases} 2z_i - \mathrm{RePSKU}_{b_i,c_i,d_i,e_i}(z_i) & z_i \geq b_i, \\ 0, & z_i < b_i, \end{cases} \quad (4.330)$$

and a_i, b_i, c_i, d_i, and e_i are parameters (common for both $\mathrm{RePSKU}_{b_i,c_i,d_i,e_i}(z_i)$ and $\mathrm{RePSHU} b_i, c_i, d_i, e_i\,(z_i)$) [1080]. The RePSU is a generalization of the smooth sigmoid-based shrinkage (SSBS) function [1081] used for image denoising [1080].

4.3.1.59 *Parametric deformable exponential linear unit (PDELU)*

An adaptive activation function parametric deformable exponential linear unit (PDELU) [1082] is based on the premise that shifting the mean value of the output closer to zero speeds up the learning [1082]. The PDELU is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i\left(\left[1 + (1-b)\,z_i\right]^{\frac{1}{1-b}} - 1\right), & z_i < 0, \end{cases} \quad (4.331)$$

where a_i is a trainable parameter for each neuron i and b is a fixed hyperparameter controlling the degree of deformation [1082]. The Cheng et al. recommend setting $b = 0.9$ [1082]. The authors found that the MSRA initialization method [871] is consistent with PDELU [1082]. The performance of PDELU was empirically shown on the CIFAR-10 and CIFAR-100 datasets [243] and on the ImageNet dataset [48] where it outperformed ReLU, APLU, LReLU, PReLU, SReLU, ELU, MPELU (and other) activation functions [1082].

4.3.1.60 *Elastic exponential linear unit (EELU)*

An adaptive variant of the ELU function that has a stochastic component was proposed in [1027] — the elastic exponential linear unit (EELU). The EELU combines EReLU (see Section 4.2.6.38) and MPELU (see Section 4.3.1.48) and is defined as

$$f(z_i^c) = \begin{cases} k_i^c z_i^c, & z_i^c \geq 0, \\ a^c\left(\exp\left(b^c z_i^c\right) - 1\right), & z_i^c < 0, \end{cases} \tag{4.332}$$

where a^c and b^c are trainable parameters shared among all neurons of a channel c and k_i^c is a randomly sampled noise parameter for each neuron i in channel c during the training stage [1027] and set to 1 during the testing stage. The k_i^c is sampled coefficient from Gaussian distribution with a random standard deviation that is truncated from 0 to 2; k_i^c is therefore sampled as

$$k_i^c = \max\left(0, \min\left(s_i^c, 2\right)\right) \tag{4.333}$$

where

$$s_i^c \sim \mathrm{N}\left(1, \sigma^2\right), \tag{4.334}$$

$$\sigma \sim \mathrm{U}(0, \epsilon), \epsilon \in (0, 1], \tag{4.335}$$

where $\mathrm{N}\left(1, \sigma^2\right)$ is Gaussian distribution with mean 1 and variance σ^2, U denotes the uniform distribution [1027]. The ϵ is a hyperparameter; the authors recommend smaller values, e.g., 0.1 or 0.2 [1027].

The training algorithm is also modified and works in two steps — first, the EELU parameter and the weights are updated with fixed $k_i^c = 1$, and then weights are updated with random k_i^c and fixed EELU parameters [1027]. The authors also recommend using the MPELU initialization [1070] method [1027].

4.3.1.61 *Parametric first power linear unit with sign (PFPLUS)*

The parametric first power linear unit with sign (PFPLUS) is an AAF proposed in [940, 1083]. It is defined as

$$f(z_i) = a_i z_i \cdot (1 - b_i z_i)^{\mathrm{H}(z_i) - 1}, \tag{4.336}$$

where $\mathrm{H}(z_i)$ is Heaviside step function (see Section 4.2.1)

$$\mathrm{H}(z_i) = \begin{cases} 1, & z_i \geq 0, \\ 0, & z_i < 0. \end{cases} \tag{4.337}$$

and $a_i > 0$ and $b_i > 0$ are trainable parameters for each neuron i [940]. For example, the PFPLUS is similar to the ReLU when $a_i = 0.2$ and $b_i = 10$ and similar to a linear mapping when $a_i = 5$ and $b_i = 0.1$ [940].

4.3.1.62 *Parametric variational linear unit (PVLU)*

The parametric variational linear unit (PVLU) is an adaptive variant of the VLU proposed [880]. It is defined as

$$f(z_i) = \mathrm{ReLU}(z_i) + a_i \sin(b_i z_i) = \max(0, z_i) + a_i \sin(b_i z_i), \tag{4.338}$$

where a_i and b_i are trainable parameters [880].

4.3.2 *Sigmoid-based adaptive functions*

Many different adaptive activation functions based on the sigmoid family were proposed in the literature [11], one of the earliest examples is a logistic sigmoid activation function with shape autotuning [1084]. The function proposed by Yamada and Yabuta uses a single parameter controlling both the amplitude and the slope of the activation function [691, 1084]. The proposed adaptive function is defined as

$$f(z) = 2\frac{1-\exp(-az)}{a(1+\exp(-az))}, \tag{4.339}$$

where $a \in (0, \infty)$ is a learnable parameter [691].

4.3.2.1 *Generalized hyperbolic tangent*

The generalized hyperbolic tangent [1085] introduces two trainable parameters that control the scale of the activation function:

$$f(z_i) = \frac{a_i(1-\exp(-b_i z_i))}{1+\exp(-b_i z_i)}, \tag{4.340}$$

where a_i and b_i are trainable parameters for each neuron i [1084]. A non-adaptive version with fixed parameters was used for document recognition in [795] in order to improve convergence toward the end of the learning session [795] (see Section 4.2.2.3).

4.3.2.2 *Trainable amplitude*

A more general approach was introduced in [1086], which used networks with a trainable amplitude of activation functions; the same approach was later used for recurrent neural networks [1087]. The class of adaptive functions with a trainable amplitude is defined as

$$f(z_i) = a_i g(z_i) + b_i, \tag{4.341}$$

where a_i and b_i are trainable parameters for each neuron i. The a_i determines the trainable amplitude and the b_i trainable offset. These parameters can be either different for each neuron or may be shared by a whole layer or even a whole network [1086].

4.3.2.3 *Adaptive slope sigmoidal function (ASSF)*

A adaptive slope sigmoidal function (ASSF) based on the work of Yamada and Yabuta, Yamada and Yabuta was used in [1089, 1090]. It is defined as

$$f(z) = \sigma(a \cdot z), \tag{4.342}$$

where σ is the logistic sigmoid and a is a global trainable parameter [1090]. The ASSF was also rediscovered by Mercioni, Tiron, and Holban in [1091].

4.3.2.4 *Slope varying activation function (SVAF)*

A slope varying activation function (SVAF) was proposed in [1092]

$$f(z) = \tanh(a \cdot z), \tag{4.343}$$

where a is a global trainable parameter. The slope varying activation function was proposed together with a BP modification that has two different learning rates [1092]. The slope varying activation function was implemented as a modification of the BP algorithm rather; a different example of modification of the BP algorithm resulting in an adaptive activation function is presented in [1093].

4.3.2.5 *TanhSoft*

The TanhSoft is a family of AAFs proposed in [1094] that combine the softplus and tanh that contains three notable cases — TanhSoft-1, TanhSoft-2, and TanhSoft-3 [1094, 1095].

The general TanhSoft is defined as

$$f(z_i) = \tanh(a_i z_i + b_i \exp(c_i z_i)) \ln(d_i + \exp(z_i)), \tag{4.344}$$

where a_i, b_i, c_i, and d_i are either trainable or fixed parameters [1094]; $a_i \in (-\infty, 1]$, $b_i \in [0, \infty)$, $c_i \in (0, \infty)$, and $d_i \in [0, 1]$ [1094].

The first AF, named TanhSoft-1, is defined as

$$f(z_i) = \tanh(a_i z_i) \ln(1 + \exp(z_i)), \tag{4.345}$$

where a_i is a trainable parameter [1094, 1095]; it can be obtained from the general TanhSoft by setting $b_i = 0$ and $d_i = 1$ [1094]. The second AF from [1095], TanhSoft-2, is defined as

$$f(z_i) = z_i \tanh(b_i \exp(c_i z_i)), \tag{4.346}$$

where b_i and c_i are trainable parameters [1094, 1095]. The TanhSoft-2 can be obtained from the general TanhSoft by setting $a_i = 0$ and $d_i = 0$ [1094]. The last AF from [1095], TanhSoft-3, is defined as

$$f(z_i) = \ln(1 + \exp(z_i) \tanh(a_i z_i)), \tag{4.347}$$

where a_i is a trainable parameter [1095]. It can be obtained from the general TanhSoft by setting $b_i = 0$ and $d_i = 1$.

4.3.2.6 *Parametric sigmoid (psigmoid)*

An adaptive variant of logistic sigmoid named parametric sigmoid (psigmoid)[60] was proposed in [1096, 1097].[61] Similarly as in generalized hyperbolic tangent, it introduces two scaling parameters to a logistic sigmoid:

$$f(z_i) = a_i \sigma (b \cdot z_i), \tag{4.348}$$

where a_i is a trainable parameter for each neuron or channel i and b is a global trainable parameter [1096].

4.3.2.7 *Parametric sigmoid function (PSF)*

A parametric sigmoid function (PSF) is a continuous, differentiable, and bounded function proposed in [1098, 1099][62] and is defined as

$$\mathrm{PSF}(z) = \frac{1}{(1+\exp(-z))^m}, \tag{4.349}$$

where m is a global trainable parameter [11, 1100]. The parameter m controls the slope of the sigmoid and the position of the maximum derivative; the envelope of the relevant derivatives for different values of m is also a sigmoid function [1098]. The larger values of m improve the gradient flow [11]. The PSF is only one instance of a larger class of activation functions proposed in [1101].

4.3.2.8 *Slope and threshold adaptive activation function with tanh function (STAC-tanh)*

The slope and threshold adaptive activation function function with tanh function (STAC-tanh) was proposed in [1102]. It is basically a tanh based equivalent of the improved logistic sigmoid with adaptive parameters. It is defined as

$$f(z_i) = \begin{cases} \tanh -a_i + b_i (z_i + a_i), & z_i < -a_i, \\ \tanh z_i, & -a_i \leq z_i \leq a_i, \\ \tanh a_i + b_i (z_i - a_i), & z_i > a_i, \end{cases} \tag{4.350}$$

where a_i and b_i are trainable parameters [1102].

4.3.2.9 *Generalized Riccati activation (GRA)*

The generalized Riccati activation (GRA) is an adaptive variant of a sigmoid AF proposed in [1103]. It is defined as

$$f(z_i) = 1 - \frac{a_i}{a_i + (1+b_i z_i)^{c_i}}, \tag{4.351}$$

where a_i, b_i, and c_i are adaptive parameters — $b_i > 0$ and $c_i > 0$ [1103].

60 Not to be confused with parametric sigmoid function (PSF) from Section 4.3.2.7.
61 It seems that this AAF was first proposed in 2010 in [1097] and then independently in 2021 in [1096].
62 [1099] contains the definition equivalent to $f(z) = \mathrm{PSF}\left(\frac{z}{2}\right)$.

4.3.3 *Adaptive sigmoid-weighted linear units*

There are several AFs that are based on the SiLU but have an adaptive parameter; the most common example is the swish AF, but there are also other popular functions based on the same principle.

4.3.3.1 *Swish*

A swish activation function [668] is an adaptive variant of the SiLU [816] (see Section 4.2.3); it is also the member of the LAAF class (see Section 4.3.15):

$$f(z_i) = z_i \cdot \sigma(a_i z_i), \tag{4.352}$$

where $\sigma(z)$ is the logistic sigmoid, a_i is either a fixed hyperparameter or a trainable parameter [668]. The swish has an output range of $(-\infty, \infty)$ [11]. The parameter a_i controls the amount of non-linearity the swish activation has [11]. The swish might also be considered a member of the family of activate or not activation functions (ACONs) [1104]; it is then named ACON-A. The parametric SiLU (PSiLU) is another name for the swish activation used in [1018].

4.3.3.2 *Adaptive hybrid activation function (AHAF)*

A swish variant with vertical scaling was proposed in [1105] under the name adaptive hybrid activation function (AHAF). It is defined as

$$f(z_i) = a_i z_i \cdot \sigma(b_i z_i), \tag{4.353}$$

where a_i and b_i are trainable parameters [1105].

4.3.3.3 *Parametric shifted SiLU (PSSiLU)*

The parametric shifted SiLU (PSSiLU) is a swish based AAF proposed in [1018]. It is defined as

$$f(z_i) = \frac{z_i \cdot (\sigma(a_i z_i) - b_i)}{1 - b_i}, \tag{4.354}$$

where a_i and b_i are trainable parameters [1018].

4.3.3.4 *E-swish*

E-swish [1106] is an AAF inspired by the swish [668] activation function (see Section 4.3.3.1); the E-swish has a scaling parameter that allows for vertical scaling of the activation function [1106]. The name of the activation function is not chosen well as the E-swish is rather extending the SiLU (see Section 4.2.3) and not swish which is its adaptive variant.[63] The function is defined as

$$f(z) = az \cdot \sigma(z), \tag{4.355}$$

63 Calling the SiLU as swish is quite common in the literature, e.g., exponential swish, generalized swish, and TS-swish.

where $\sigma(z)$ is the logistic sigmoid and a is a preset parameter [1106] — however, the parameter a is considered to be trainable in review [11]. Alcaide recommends setting $a \in [1, 2]$ to avoid exploding gradients that are hypothesized to more likely occur for higher values of a [1106]. The E-swish was found to outperform the SiLU (called swish in the paper) on the the MNIST [45], CIFAR-10 and CIFAR-100 [243] datasets using the Wide ResNet (WRN) [55] architecture [1106].

4.3.3.5 *ACON-B*

The ACON family conists of swish AF and several extensions; one is named ACON-B and is defined as

$$f(z_i) = (1 - b_i)\, z_i \cdot \sigma\left(a_i\,(1 - b_i)\, z_i\right) + b_i z_i, \tag{4.356}$$

where a_i and b_i are trainable parameters [1104]. The b_i is initalized to 0.25 and a_i to 1.[64]

4.3.3.6 *ACON-C*

The ACON-C is another member of the ACON family from [1104]. It is defined as

$$f(z_i) = (c_i - b_i)\, z_i \cdot \sigma\left(a_i\,(c_i - b_i)\, z_i\right) + b_i z_i, \tag{4.357}$$

where a_i, b_i, and c_i are trainable parameters [1104, 1107]. Ma et al. used initial values $a_i = 1$, $b_i = 0$, and $c_i = 1$ in [1104].

Ma et al. also proposed a general extension to the ACON family named MetaACON which uses a small NN to determine the value of the parameter a_i; they used the variant ACON-C for the experiments with MetaACON resulting in MetaACON-C[65] [1104]. The MetaACON was used to improve YOLOv7 [1108] in [1109]. Kan et al. extented the ACON AFs into an AF they named CBAC[66] [1110]. The ACONs were used, for example, in [1104, 1107, 1109, 1111–1123]. The 1Dmeta-ACON is a MetaACON extension proposed in [1124].

4.3.3.7 *Parameterized self-circulating gating unit (PSGU)*

The Parameterized self-circulating gating unit (PSGU) [1125] is related to the LiSHT and GTU activation functions as it is basically a LiSHT with gated input with learnable scaling parameter. It is defined as

$$f(z_i) = z_i \cdot \tanh\left(a_i \sigma\left(z_i\right)\right), \tag{4.358}$$

where a_i is a learnable parameter and $\sigma(z)$ is the logistic sigmoid function [1125]. Li et al. also propose a novel initialization method for NNs with the

64 There is no initial value for a_i in ACON-B mentioned explicitly in [1104]; however, there is one for its extension ACON-C.

65 The implementation of MetaACON-C and other AFs from the ACON family is available at `https://github.com/nmaac/acon`.

66 No further description is provided in [1110].

PSGU AF and show that it is more suitable for the use with PSGU than other common methods [1125]. The PSGU is shown to outperform ReLU, mish, swish, PATS and GELU using various NIN and ResNet architectures [1125]. The PSGU was also proposed in [829] under the name TSReLU learnable (TSReLUl) as the adaptive variant of TSReLU. Mercioni, Tat, and Holban used $a_i = 0.5$ as the initial value [829].

4.3.3.8 *Tangent-bipolar-sigmoid ReLU learnable (TBSReLUl)*

Similarly as TSReLUl is an adaptive variant of TSReLU, the TBSReLU learnable (TBSReLUl) [829] is an adaptive variant of TBSReLU. This variant is defined as

$$f(z) = z_i \cdot \tanh\left(a_i \frac{1 - \exp(-z_i)}{1 + \exp(-z_i)}\right). \tag{4.359}$$

where a_i is a trainable parameter [829]. Mercioni, Tat, and Holban used $a_i = 0.5$ as the initial value [829].

4.3.3.9 *PATS*

The AF named PATS[67] [1126] is very similar to PSGU, but it uses arctan and a random scaling parameter instead of the tanh and the adaptive parameter in PSGU. It is defined as

$$f(z_i) = z_i \tan^{-1}(a_i \pi \sigma(z_i)), \tag{4.360}$$

where $\sigma(z)$ is the logistic sigmoid function and

$$a_i \sim \mathrm{U}(l, u), \tag{4.361}$$

is sampled during training[68] from the uniform distribution with bounds l and u such that $0 < l < u < 1$ [1126]. The authors experimented with fixed, deterministic values of $a_i \in \{\frac{1}{4}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}\}$ — the value $\frac{5}{8}$ led to lowest test error on the CIFAR-10 [243]; they also deemed that suitable values for l and u are $\frac{1}{2}$ and 34 respectively [1126]. However, only fixed variant with $a_i = \frac{5}{8}$ was used in the follow-up works such as [679].

4.3.3.10 *Adaptive quadratic linear unit (AQuLU)*

The adaptive quadratic linear unit (AQuLU) is an adaptive SiLU variant proposed in [823]; it is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq \frac{1-b_i}{a_i}, \\ a_i z_i^2 + b_i z_i, & -\frac{b_i}{a_i} \geq z_i < \frac{1-b_i}{a_i}, \\ 0, & z_i < -\frac{b_i}{a_i}, \end{cases} \tag{4.362}$$

where a_i and b_i are trainable parameters for each neuron i [823].

67 Not an abbreviation.
68 Unfortunately, the author did not specify what happens during the test phase in [1126], one can only assume that the expected value is used.

4.3.3.11 *Sinu-sigmoidal linear unit (SinLU)*

Another adaptive SiLU variant is the sinu-sigmoidal linear unit (SinLU), which adds an adaptive term using the sine function to the linear part of the SiLU [1127]. The SinLU is defined as

$$f(z_i) = (z_i + a_i \sin(b_i z_i)) \cdot \sigma(z_i), \tag{4.363}$$

where $\sigma(z_i)$ is the logistic sigmoid function and a_i and b_i are trainable parameters for each neuron i [1127].

4.3.3.12 *ErfAct*

An AAF based on the Gauss error function was proposed in [1128]. The AAF is named ErfAct and is defined as

$$f(z_i) = z_i \cdot \operatorname{erf}(a_i \exp(b_i z_i)), \tag{4.364}$$

where a_i and b_i are trainable parameters for each neuron i and $\operatorname{erf}(x)$ is the Gauss error function [1128].

4.3.3.13 *Parametric serf (pserf)*

An adaptive version of the serf AF named parametric serf (pserf) was proposed in [1128]. It is defined as

$$f(z_i) = z_i \cdot \operatorname{erf}(a_i \ln(1 + \exp(b_i z_i))), \tag{4.365}$$

where a_i and b_i are trainable parameters for each neuron i and $\operatorname{erf}(x)$ is the Gauss error function [1128].

4.3.3.14 *Swim*

The swim is an adaptive variant of the PFLU (see Section 4.2.7.7) independently proposed in [1129]. It is defined as

$$f(z_i) = z_i \cdot \frac{1}{2}\left(1 + \frac{a_i z_i}{\sqrt{1 + {z_i}^2}}\right), \tag{4.366}$$

where a_i is either fixed or trainable parameter for each neuron i [1129]. Abdool and Dear used fixed $a_i = 0.5$ in their experiments in [1129].

4.3.4 *Tuned softmax (tsoftmax)*

A softmax (see Section 4.2.5) variant named tuned softmax (tsoftmax) was proposed in [860]; it is defined as

$$f(z_j) = \frac{\int \exp(z_j)}{\sum_{k=1}^{N} \int \exp(z_k)} + c, \tag{4.367}$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons and c is an adaptive parameter [860].

4.3.5 *Generalized Lehmer softmax (glsoftmax)*

The generalized Lehmer softmax (glsoftmax) is a softmax variant proposed in [1130]. It is defined as

$$f(z_j) = \frac{\exp\left(\text{LNORM}\left(z_j\right)\right)}{\sum_{k=1}^{N} \exp\left(\text{LNORM}\left(z_k\right)\right)}, \tag{4.368}$$

where $\text{LNORM}\left(z_j\right)$ is a generalized Lehmer-based Z-score-like normalization with four trainable parameters a_i, b_i, c_i, and d_i defined in [1130]:

$$\text{LNORM}\left(z_i\right) = \frac{z_i - M_{a_i,b_i}}{\text{GLM}_{c_i,d_i}\left(\boldsymbol{z} - M_{a_i,b_i}\right)}, \tag{4.369}$$

$$M_{a_i,b_i} = \text{GLM}_{a_i,b_i}\left(\boldsymbol{z}\right), \tag{4.370}$$

$$\text{GLM}_{\alpha,\beta}\left(\boldsymbol{x}\right) = \frac{\ln\left(\frac{\sum_{k=1}^{N} \alpha^{(\beta+1)x_k}}{\sum_{k=1}^{N} \alpha^{\beta x_k}}\right)}{\ln\left(\alpha\right)}, \tag{4.371}$$

$\boldsymbol{x}$ is a vector of elements x_k, $k = 1, \ldots, N$ and $\boldsymbol{z} - M_{a_i,b_i}$ represents a vector with elements $z_k - M_{a_i,b_i}$, $k = 1, \ldots, N$ [1130].

4.3.6 *Generalized power softmax (gpsoftmax)*

The generalized power softmax (gpsoftmax) is another softmax variant proposed in [1130]. It is defined as

$$f(z_j) = \frac{\exp\left(\text{PNORM}\left(z_j\right)\right)}{\sum_{k=1}^{N} \exp\left(\text{PNORM}\left(z_k\right)\right)}, \tag{4.372}$$

where $\text{PNORM}\left(z_j\right)$ is a generalized power-based Z-score-like normalization with four trainable parameters a_i, b_i, c_i, and d_i defined in [1130]:

$$\text{PNORM}\left(z_i\right) = \frac{z_i - M_{a_i,b_i}}{\text{GPM}_{c_i,d_i}\left(\boldsymbol{z} - M_{a_i,b_i}\right)}, \tag{4.373}$$

$$M_{a_i,b_i} = \text{GPM}_{a_i,b_i}\left(\boldsymbol{z}\right), \tag{4.374}$$

$$\text{GPM}_{\alpha,\beta}\left(\boldsymbol{x}\right) = \frac{\ln\left(\sum_{k=1}^{N} \alpha^{\beta x_k}\right) - \ln\left(N\right)}{\beta \ln\left(\alpha\right)}, \tag{4.375}$$

$\boldsymbol{x}$ is a vector of elements x_k, $k = 1, \ldots, N$ and $\boldsymbol{z} - M_{a_i,b_i}$ represents a vector with elements $z_k - M_{a_i,b_i}$, $k = 1, \ldots, N$ [1130].

4.3.7 *Adaptive radial basis function (ARBF)*

The adaptive (ARBF) was used in [1131]. It is defined as

$$f(z_i) \exp\left(-\frac{(z_i - a_i)^2}{2b_i^2}\right), \tag{4.376}$$

where a_i and b_i are adaptive parameters for each neuron i [1131]. The parameter a_i controls the center while the parameter b_i controls the width [1131].

4.3.8 *Parametric Gaussian error linear unit (PGELU)*

The AAF named parametric Gaussian error linear unit (PGELU) was proposed in [1132] as the result of noise injection. It is an GELU (see Section 4.2.3.1) adaptive variant defined as

$$f(z_i) = z \cdot \Phi\left(\frac{z}{a}\right), \tag{4.377}$$

where $\Phi(z)$ is the standard Gaussian CDF and a is a global learnable parameter representing the root mean square (RMS) noise [1132].

4.3.9 *Parametric flatted-T swish (PFTS)*

A parametric flatted-T swish (PFTS) [1133] is an adaptive extension of the FTS (see Section 4.2.6.46); PFTS is identical to FTS except for that the parameter T is adaptive — i.e.:

$$f(z_i) = \mathrm{ReLU}(z_i) \cdot \sigma(z_i) + T_i = \begin{cases} \frac{z_i}{1+\exp(-z_i)} + T_i, & z_i \geq 0, \\ T_i, & z_i < 0, \end{cases} \tag{4.378}$$

where T_i is a trainable parameter for each neuron i [1133]; the parameter T_i is initialized to the value -0.20 [1133].

4.3.10 *Parametric flatten-p mish (PFPM)*

The parametric flatten-p mish (PFPM) is an AAF proposed in [1134]; it is defined as

$$f(z_i) = \begin{cases} z_i \tanh\left(\ln\left(1 + \exp\left(z_i\right)\right)\right) + p_i, & z_i \geq 0, \\ p_i, & z_i < 0, \end{cases} \tag{4.379}$$

where p_i is a trainable parameter [1134].

4.3.11 *Gaussian error unit (GEU)*

The AAF named Gaussian error unit (GEU) was proposed in [1132] as the result of noise injection. It is defined as

$$f(z_i) = \Phi\left(\frac{z}{a}\right), \tag{4.380}$$

where $\Phi(z)$ is the standard Gaussian CDF and a is a global learnable parameter representing the RMS noise [1132]. The GEU multiplied by z becomes the PGELU (see Section 4.3.8).

4.3.12 *Scaled-gamma-tanh activation function (SGT)*

The scaled-gamma-tanh (SGT) AF is a piecewise polynomial function proposed in [1135]. It is defined as

$$f(z_i) = \begin{cases} az_i^{b_i}, & z_i \geq 0, \\ cz_i^{d_i}, & z_i < 0, \end{cases} \tag{4.381}$$

where a and c are fixed, predefined parameters and b_i and c_i are trainable parameters for each neuron or filter i [1135].

4.3.13 *RSign*

An adaptive variant of the sign function was used in [1023]. It is called react-sign (RSign) and is defined as

$$f(z_i) = \begin{cases} 1, & z_i \geq a_c, \\ -1, & z_i < a_c, \end{cases} \tag{4.382}$$

where a_c is an adaptive threshold for each channel [1023]. An extension was used in [1136], where Ding, Liu, and Zhou used multiple RSign functions for each channel.

4.3.14 *P-SIG-RAMP*

An AAF combining the logistic sigmoid and ReLU was proposed in [1073] under the name P-SIG-RAMP. The P-SIG-RAMP is defined as

$$f(z_i) = a_i\sigma(z_i) + (1-a_i) \cdot \begin{cases} 1, & z_i \geq \frac{1}{2b_i}, \\ b_iz_i + \frac{1}{2}, & -\frac{1}{2b_i} < z_i < \frac{1}{2b_i}, \\ 0, & z_i \leq -\frac{1}{2b_i}, \end{cases} \tag{4.383}$$

where $a_i \in [0,1]$ and b_i are trainable parameters [1073].

4.3.15 *Locally adaptive activation function (LAAF)*

A general class of slope varying functions called locally adaptive activation function (LAAF) was proposed in [1137, 1138]:

$$f(z_i) = g(a_i \cdot z_i), \tag{4.384}$$

where a_i is a trainable parameter for each neuron i and g is any activation function; Jagtap, Kawaguchi, and Karniadakis used logistic sigmoid, tanh, ReLU, and LReLU as g in their LAAFs in [1137]. The corresponding activations are thus given by

$$f(z_i) = \sigma(a_i z_i) = \frac{1}{1 + \exp(-a_i z_i)}, \tag{4.385}$$

$$f(z_i) = \tanh(a_i z_i) = \frac{\exp(a_i z_i) - \exp(-a_i z_i)}{\exp(a_i z_i) + \exp(-a_i z_i)}, \tag{4.386}$$

$$f(z_i) = \mathrm{ReLU}(a_i z_i) = \max(0, a_i z_i), \tag{4.387}$$

and

$$f(z_i) = \mathrm{LReLU}(a_i z_i) = \max(0, a_i z_i) - b \max(0, -a_i z_i), \tag{4.388}$$

where b is the LReLU leakiness parameter [1137]. To accelerate the convergence, Jagtap, Kawaguchi, and Karniadakis add additional fixed parameter to the expression:

$$f(z_i) = g(n a_i z_i), \tag{4.389}$$

where $n > 1$ is a fixed parameter [1137]. It was found that this additional parameter improves both the convergence rate and the solution accuracy [1137].

4.3.15.1 *Adaptive slope hyperbolic tangent*

A tanh activation function with adaptive slope was used in an MLP architecture in [1139]. The used activation function is defined as

$$f(z_i) = \tanh(a_i z_i), \tag{4.390}$$

where a_i is a trainable parameter for each neuron i.

4.3.15.2 *Parametric scaled hyperbolic tangent (PSTanh)*

A parametric activation function similar to the swish but based on the tanh function instead of the logistic sigmoid called parametric scaled hyperbolic tangent (PSTanh) was proposed in [688]. It is defined as

$$f(z_i) = z_i \cdot a_i (1 + \tanh(b_i z_i)), \tag{4.391}$$

where a_i and b_i are trainable parameters for each neuron i [688]. The function is also very similar to the PTELU (see Section 4.3.1.30) as for $z_i > 0$ and $a_i \approx 1$, the output is close to z_i [688] (the exact distance depends on the parameters a_i and b_i).

4.3.15.3 *Scaled sine-hyperbolic function (SSinH)*

An AF similar to PSTanh is the scaled sine-hyperbolic function (SSinH) [1140]; it is defined as

$$f(z_i) = a_i \sinh(b_i z_i), \tag{4.392}$$

where a_i and b_i are trainable scaling parameters and sinh is the hyperbolic sine [1140].

4.3.15.4 *Scaled exponential function (SExp)*

Husain, Ong, and Bober also proposed scaled exponential function (SExp) along with the SSinH in [1140]. It is defined as

$$f(z_i) = a_i \left(\exp\left(b_i z_i\right) - 1\right), \tag{4.393}$$

where a_i and b_i are trainable scaling parameters and sinh is the hyperbolic sine [1140].

4.3.15.5 *Logmoid activation unit (LAU)*

A learnable LAU was proposed in [1141, 1142]; which utilise two learnable parameters a_l and b_l for each network layer l

$$f(z_{i,l}) = z \ln\left(1 + a_l \sigma\left(b_l \cdot z_{i,l}\right)\right), \tag{4.394}$$

where $z_{i,l}$ is the output of the neuron i in layer l without the activation function and σ is the logistic sigmoid [1142]. The author used initial values of the parameters $a_l = b_l = 1$ for each network's layer l and trained these parameters together with the rest of the network's weights [1142].

4.3.15.6 *Cosinu-sigmoidal linear unit (CosLU)*

The cosinu-sigmoidal linear unit (CosLU) is an adaptive activation function proposed in [889] that is based on the logistic sigmoid. It is defined as

$$f(z_i) = \left(z + a_i \cos\left(b_i z_i\right)\right) \sigma\left(z_i\right), \tag{4.395}$$

where a_i and b_i are trainable parameters for neuron i and $\sigma(z_i)$ is the logistic sigmoid function [889]. The cosine amplitude is controlled by the parameter a_i, whereas its frequency is controlled by the parameter b_i.

4.3.15.7 *Adaptive Gumbel (AGumb)*

An activation function adaptive Gumbel (AGumb) is based approach of viewing activation functions as a combination of unbounded and bounded components where the bounded component is based upon a cumulative distribution function of a continuous distribution [1143]. While the logistic sigmoid activation is a CDF of the symmetric logistic distribution, the AGumb is based on the Gumbel distribution [1143]. It is defined as

$$f(z_i) = 1 - \left(1 + a_i \cdot \exp\left(z_i\right)\right)^{-\frac{1}{a_i}}, \tag{4.396}$$

where $a_i \in \mathbb{R}^+$ is trainable parameter for each neuron i [1143].

4.3.16 *Shape autotuning adaptive activation function (SAAAF)*

The shape autotuning adaptive activation function (SAAAF)[69] is an AAF proposed in [1144]. It is defined as

$$f(z_i) = \frac{z_i}{\frac{z_i}{a_i} + \exp\left(-fracz_i b_i\right)}, \tag{4.397}$$

where $a_i \geq 0$ and $b_i \geq 0$ are trainable parameters for neuron i and $0 < \frac{b_i}{a_i} < \mathrm{e}$ [1144].

4.3.17 *Noisy activation functions*

Stochastic variants of saturing activation functions such as the logistic sigmoid or hyperbolic tangent were proposed in [1029] where an additional noise is injected to the activation function when it operates in the saturation regimes [1029]. The noisy activation function is defined as

$$f(z_i) = a\mathrm{h}(z_i) + (1-a)\,\mathrm{u}(z_i) - \mathrm{sgn}(z_i)\mathrm{sgn}(1-a)c\Big(\sigma\Big(p_i\left(\mathrm{h}(z_i) - \mathrm{u}(z_i)\right)\Big)\Big)^2 \epsilon, \tag{4.398}$$

where $\mathrm{h}(z_i)$ is any saturating activation function such as hard-tanh or hard-sigmoid, $\mathrm{u}(z_i)$ is its linearization using first-order Taylor expansion around zero, c is a hyperparameter changing the scale of the standard deviation of the noise, p_i is a trainable parameter adjusting the magnitude of the noise for each neuron i, a is a hyperparameter influencing the mean of the added term, and $\sigma(x)$ is the logistic sigmoid function [1029]. ϵ is the added noise; it is defined as $\epsilon = |\xi|$ if the noise term ξ is sampled from half-normal distribution and as $\epsilon = \xi$ if the noise term ξ is sampled from normal distribution with mean 0 and variance 1 [1029].

Gulcehre et al. also experimented with adding noise to the input of the activation function, resulting in an activation function defined as

$$f(z_i) = \mathrm{h}\left(z_i + s(z_i)\epsilon\right), \tag{4.399}$$

where $s(z_i)$ is either fixed parameter $s(z_i) = b$ or it is a trainable term

$$s(z_i) = c\left(\sigma\left(p_i\left(\mathrm{h}(z_i) - \mathrm{u}(z_i)\right)\right)\right)^2, \tag{4.400}$$

where the meaning of c, σ, p_i, $\mathrm{h}(z_i)$, and $\mathrm{u}(z_i)$ is same as in Eq. (4.398) [1029].

A similar concept in ReLU settings is the ProbAct activation function (see Section 4.3.1.11).

69 Zhou et al. named the function as *shape autotuning activation function* but the resulting abbreviation SAAF is already taken by smooth adaptive activation function (see Section 4.3.28). Since the proposed function is an AAF, we term it as such to avoid the abbreviation collision.

4.3.18 *Fractional adaptive activation functions*

Fractional adaptive activation functions (FAAFs) were proposed in [1145–1149] as a generalization of several activation functions using the fractional calculus (see [1150] for a general introduction to the fractional calculus). Generally, for any activation function $f(z)$, its generalization $g(z)$ using fractional derivatives is defined as the $a - th$ fractional derivative of f:

$$g(z) = D^a f(z), \tag{4.401}$$

where a can be a learnable[70] parameter [1145]. The FAAFs proposed in [1147] were further evaluated in [1148].

4.3.18.1 *Fractional ReLU*

The fractional ReLU (FracReLU) is defined as

$$f(z_i) = \frac{z_i^{1-a_i}}{\Gamma(2-a_i)}, \tag{4.402}$$

where $\Gamma(x)$ is the Gamma function and a_i is a trainable parameter [1145]. The FracReLU was later independently proposed in [1147] under the name FReLU (but this abbreviation is already taken by flexible ReLU).

4.3.18.2 *Fractional softplus*

The fractional softplus (FracSoftplus) is using the softplus function to generalize sigmoid-like functions through fractional derivatives [1145]. It is defined as

$$f(z_i) = D^{a_i} \ln(1 + \exp(z_i)), \tag{4.403}$$

which is then computed as

$$f(z_i) = \lim_{h \to 0} \frac{1}{h^{a_i}} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(a_i + 1) \ln(1 + \exp(z_i - nh))}{\Gamma(n+1)\Gamma(1 - n + a_i)}, \tag{4.404}$$

where a_i is a trainable parameter [1145]. Particularly interesting cases are when $a_i = 0$ as it is the softplus function, $a_i = 1$ logistic sigmoid, and $a_i = 2$ which leads to a bell-like shape [1145].

4.3.18.3 *Fractional hyperbolic tangent*

The fractional tanh (FracTanh) is another fractional generalization proposed in [1145]; it is defined as

$$f(z_i) = D^{a_i} \tanh(z_i), \tag{4.405}$$

which is then computed as

$$f(z_i) = \lim_{h \to 0} \frac{1}{h^{a_i}} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(a_i + 1) \tanh(z_i - n \cdot h)}{\Gamma(n+1)\Gamma(1 - n + a_i)}, \tag{4.406}$$

where a_i is a trainable parameter [1145]. The function becomes the tanh for $a_i = 0$ and the quadratic hyperbolic secant function for $a_i = 1$.

70 [1147] did not specified whether the parameter is trainable but [1145] explicitly uses a trainable a.

4.3.18.4 *Fractional adaptive linear unit*

The fractional adaptive linear unit (FALU) [1146] is yet another AAF based on fractional calculus[71] It can be seen as the fractional generalization using the $a_i - th$ fractional derivative of the swish function:

$$f(z_i) = D^{a_i} z_i \sigma(b_i z_i), \tag{4.407}$$

where a_i and b_i are trainable parameters and σ is the logistic sigmoid function [1146]. The fractional derivative is then calculated as

$$f(z_i) = \lim_{h \to 0} \frac{1}{h^{a_i}} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(a_i + 1) z_i \sigma(b_i z_i)}{\Gamma(n+1)\Gamma(1 - n + a_i)}. \tag{4.408}$$

However, as this calculation is not practical, Zamora-Esquivel, Rhodes, and Nachman use following approximation for $a_i \in [0, 2]$ and $b_i \in [1, 10]$:

$$f(z_i) \approx \begin{cases} g(z_i, b_i) + a_i \sigma(b_i z_i)\left(1 - g(z_i, b_i)\right), & a_i \in [0, 1], \\ g(z_i, b_i) + a_i \sigma(b_i z_i)\left(1 - 2h(z_i, b_i)\right), & a_i \in (1, 2], \end{cases} \tag{4.409}$$

where

$$g(z_i, b_i) = z_i \sigma(b_i z_i), \tag{4.410}$$

$$h(z_i, b_i) = g(z_i, b_i) + \sigma(z_i)\left(1 - g(z_i, b_i)\right), \tag{4.411}$$

and a_i and b_i are the two previously mentioned trainable parameters [1146]. The FALU was shown to outperform ReLU, GELU, ELU, SELU, and kernel activation function (KAF) on the MNIST [45], CIFAR-10 [243], ImageNet [48, 817], and Fashion MNIST [950] datasets for several tested architectures [1146].

4.3.18.5 *Fractional leaky ReLU (FracLReLU)*

The fractional LReLU (FracLReLU) is the fractional variant of the LReLU (see Section 4.2.6.2) proposed in [1147]. It is defined using fractional calculus as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} 0.1 z_i, & z_i < 0, \end{cases} \tag{4.412}$$

where $a_i \in (0, 1)$ is a fixed parameter [1147]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2 - a_i)} z_i^{1 - a_i}, & z_i \geq 0, \\ \frac{b}{\Gamma(2 - a_i)} z_i^{1 - a_i}, & z_i < 0. \end{cases} \tag{4.413}$$

71 The FALU was published in [1146] without any links to [1145] even though it was proposed by the same first author and it uses the same principles.

4.3.18.6 *Fractional parametric ReLU (FracPReLU)*

The fractional PReLU (FracPReLU) is the fractional variant of the PReLU (see Section 4.3.1.1) proposed in [1147]. It is defined using fractional calculus as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} b_i z_i, & z_i < 0, \end{cases} \tag{4.414}$$

where $a_i \in (0,1)$ is a fixed parameter and b_i is a trainable parameter [1147]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ \frac{b_i}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i < 0. \end{cases} \tag{4.415}$$

4.3.18.7 *Fractional ELU (FracELU)*

The fractional ELU (FracELU) is the fractional variant of the ELU (see Section 4.2.6.48) proposed in [1147]. It is defined using fractional calculus as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} b \left(\exp(z_i - 1)\right), & z_i < 0, \end{cases} \tag{4.416}$$

where $a_i \in (0,1)$ and b are fixed parameters [1147]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ b \sum_{k=0}^{\infty} \left(\frac{1}{k!} \cdot \frac{\Gamma(k+1)}{\Gamma(k+1-a_i)} z_i^{k-a_i} \right) - b \frac{1}{\Gamma(1-a_i)} z_i^{-a_i}, & z_i < 0. \end{cases} \tag{4.417}$$

4.3.18.8 *Fractional SiLU (FracSiLU)*

The fractional SiLU (FracSiLU) is the fractional variant of the SiLU (see Section 4.2.3) proposed in [1147]. It is defined using fractional calculus as

While Job et al. intended the fractional SiLU (FracSiLU) to be the fractional variant of the SiLU (see Section 4.2.3), they used a wrong definition of the SiLU. Here we present both the FracSiLU from the [1147] and the FracSiLU that fit the definition of SiLU — the definition from [1147] will be denoted as FracSiLU variant 1 (FracSiLU1) whereas the variant we derived as FracSiLU variant 1 (FracSiLU2). The Job et al. used this definition[72] of SiLU:

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} z_i \sigma(z_i), & z_i < 0. \end{cases} \tag{4.418}$$

72 Job et al. referenced [742, 1065] for their definition of SiLU; however, the [742] contains the SiLU definition from Section 4.2.3 and [1065] does not mention SiLU at all.

Then the FracSiLU1 is defined as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} z_i \sigma(z_i), & z_i < 0, \end{cases} \tag{4.419}$$

where $\sigma(z_i)$ is the logistic sigmoid [1147]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ \sum_{k=0}^{\infty} \left((-1)^k + \frac{\left(2^{k+1}-1\right) B_{k+1} \Gamma(k+2)}{\Gamma(k+2-a_i)(k+1)!} \right) z_i^{k+1-a_i}, & z_i < 0, \end{cases} \tag{4.420}$$

where B_n is n-th Bernoulli's number [1147].

When using the SiLU definition from Section 4.2.3, the FracSiLU2 is then defined as

$$f(z_i) = D^{a_i} z_i \sigma(z_i). \tag{4.421}$$

Since Job et al. made no assumption about the sign of z_i, the fractional derivative of FracSiLU2 is computed as

$$f(z_i) = sum_{k=0}^{\infty} \left((-1)^k + \frac{\left(2^{k+1}-1\right) B_{k+1} \Gamma(k+2)}{\Gamma(k+2-a_i)(k+1)!} \right) z_i^{k+1-a_i}, \tag{4.422}$$

where B_n is n-th Bernoulli's number.

4.3.18.9 *Fractional GELU (FracGELU)*

Similarly as for FracSiLU, Job et al. intended the fractional GELU (FracGELU) to be the fractional variant of the GELU (see Section 4.2.3.1), but they used a wrong definition of the GELU. Here we present both the FracGELU from the [1147] and the FracGELU that fit the definition of GELU — the definition from [1147] will be denoted as FracGELU variant 1 (FracGELU1) whereas the variant we derived as FracGELU variant 1 (FracGELU2). The Job et al. used this definition[73] of GELU:

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \Phi(z), & z < 0, \end{cases} \tag{4.423}$$

where $\Phi(z)$ is the standard Gaussian CDF [1147]. Then the FracGELU1 is defined as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} z_i \cdot \Phi(z_i), & z_i < 0. \end{cases} \tag{4.424}$$

The fractional derivative of FracGELU1 is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ 0.5 \frac{z_i^{1-a_i}}{\Gamma(2-a_i)} - \frac{1}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \frac{1}{k!} \left(-\frac{1}{2}\right)^k \frac{z_i^{2(k+1)-a_i}}{2k+1} \frac{\Gamma(2k+3)}{\Gamma(2k+3-a_i)}, & z_i < 0. \end{cases} \tag{4.425}$$

73 Job et al. referenced [742, 1065] for their definition of SiLU; however, neither [742] nor [1065] contains a definition of GELU.

When using the GELU definition from Section 4.2.3.1, the FracGELU2 is then defined as

$$f(z_i) = D^{a_i} z_i \cdot \Phi(z_i). \tag{4.426}$$

Since Job et al. made no assumption about the sign of z_i, the fractional derivative of FracGELU2 is computed as

$$f(z_i) = 0.5 \frac{z_i^{1-a_i}}{\Gamma(2-a_i)} - \frac{1}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \frac{1}{k!} \left(-\frac{1}{2}\right)^k \frac{z_i^{2(k+1)-a_i}}{2k+1} \frac{\Gamma(2k+3)}{\Gamma(2k+3-a_i)}. \tag{4.427}$$

4.3.19 *Scaled softsign*

An activation function called scaled softsign [889] is an adaptive variant of the softsign activation (see Section 4.2.2.13) with variable amplitude. It is defined as

$$f(z_i) = \frac{a_i z_i}{b_i + |z_i|}, \tag{4.428}$$

where a_i and b_i are trainable parameters for each neuron i [889]. The parameter a_i controls the range of the output while the parameter b_i controls the rate of transition between signs [889].

4.3.20 *Parameterized softplus ($s_{+}2L$)*

Parameterized softplus is an adaptive variant of a softplus activation function that allows for vertical shifts [699]. It is defined as

$$f(z_i) = \ln(1 + \exp(z_i)) - a_i, \tag{4.429}$$

where $a_i \in [0, 1]$ is a trainable parameter for each neuron i [699]. Vargas et al. also proposed a non-adaptive variant with fixed a_i that is denoted as $s_{+}2$ [699].

4.3.21 *Universal activation function (UAF)*

The so-called universal activation function (UAF) is a softplus based AAF proposed in [968]. It is defined as

$$f(z_i) = \ln\left(1 + \exp\left(a_i(z_i + b_i) + c_i z_i^2\right)\right) - \ln\left(1 + \exp\left(d_i(z_i - b_i)\right)\right) + e_i, \tag{4.430}$$

where a_i, b_i, c_i, d_i, and e_i are trainable parameter for each neuron i [968]. For example, the UAF is able to well approximate the step function, logistic sigmoid, tanh, ReLU, LReLU, and Gaussian function [968].

4.3.22 *Learnable extended activation function (LEAF)*

The learnable extended activation function (LEAF) is an AAF proposed in [1151] that is able to replace several existing AFs. It is defined as

$$f(z_i) = (a_i z_i + b_i)\,\sigma(c_i z_i) + d_i, \tag{4.431}$$

where $\sigma(x)$ is the logistic sigmoid and a_i, b_i, c_i, and d_i are trainable parameters for each neuron i [1151]. The Table 4.1 contains a list of AFs that are equivalent to a particular LEAF parameterization.

equiv. AF	a_i	b_i	c_i	d_i
ReLU	1	0	$+\infty$	0
SiLU	1	0	1	0
tanh	0	2	2	-1
logistic sigmoid	0	0	1	0
swish	1	0	a_i	0
AHAF	a_i	0	b_i	0

Table 4.1: **AF equivalent to LEAF parameterizations**
The list of AFs that have an equivalent LEAF parameterization.

4.3.23 *Generalized ReLU (GReLU)*

Theb generalized ReLU (GReLU) is an AF based on the UAF (see Section 4.3.21) [1130]. It is defined as

$$f(z_i) = \frac{1}{b_i}\log_{a_i}\left(1 + a_i^{b_i z_i}\right) = \frac{\ln\left(1 + a_i^{b_i z_i}\right)}{b_i \ln(a_i)}, \tag{4.432}$$

where a_i and b_i are trainable parameters [1130].

4.3.24 *Multiquadratic activation function (MAF)*

The multiquadratic activation function (MAF) was used in [1152, 1153]. It is defined as

$$f(z_i) = \sqrt{||z_i - a_i||^2 + b_i^2}, \tag{4.433}$$

where a_i and b_i are trainable parameters [1153] a_i is the slope coefficient and b_i is the bias coefficient [1153].

4.3.25 *EIS activation functions*

The EIS[74] is a family of AAFs proposed in [1155] with three notable examples EIS-1, EIS-2, and EIS-3 [1154, 1155].

74 The EIS is a name given by Biswas et al.; it is not an abbreviation.

The general EIS is defined as

$$f(z_i) = \frac{z_i \left(\ln\left(1+\exp\left(z_i\right)\right)\right)^{a_i}}{\sqrt{b_i + c_i z_i^2} + d_i \exp\left(-e_i z_i\right)}, \tag{4.434}$$

where a_i, b_i, c_i, d_i, and e_i are either trainable parameters or fixed hyperparameters; $a_i \in [0,1]$, $b_i \in [0,\infty)$, $c_i \in [0,\infty)$, $d_i \in [0,\infty)$, $e_i \in [0,\infty)$ and b_i, c_i, and d_i cannot be equal to zero at the same time [1155].

The EIS-1 is defined as

$$f(z_i) = \frac{z_i \ln\left(1+\exp\left(z_i\right)\right)}{z_i + d_i \exp\left(-e_i z_i\right)}, \tag{4.435}$$

where d_i and e_i are trainable parameters [1154, 1155]. It can be obtained from the general EIS by setting $a_i = 1$, $b_i = 0$, $c_i = 1$ [1155].

The EIS-2 is defined as

$$f(z_i) = \frac{z_i \ln\left(1+\exp\left(z_i\right)\right)}{\sqrt{b_i + c_i z_i^2}}, \tag{4.436}$$

where b_i is a trainable parameter [1155]. It can be obtained from the general EIS by setting $a_i = 1$, $d_i = 0$ [1155]; however, the EIS-2 from [1154] also fixes $c_i = 1$.

And finally, the EIS-3 is defined as

$$f(z_i) = \frac{z_i}{1 + d_i \exp\left(-e_i z_i\right)}, \tag{4.437}$$

where d_i and e_i are trainable parameters [1154, 1155]; it can be obtained from the general EIS by setting $a_i = 0$, $b_i = 1$, $c_i = 0$ [1155].

The EIS family contains the softplus, swish, and ISRU as special cases [1155].

4.3.25.1 *Linear combination of parameterized softplus and ELU (ELUs$_+$2L)*

A linear combination of parameterized softplus and ELU (ELUs$_+$2L) [699] is an adaptive activation function combining ELUs and parameterized softplus activation functions. It is defined as

$$f(z_i) = b_i \mathrm{ELU}(z_i) + (1 - b_i)\mathrm{s}_+2\mathrm{L}(z_i), \tag{4.438}$$

where b_i is a trainable parameter for each neuron i, $\mathrm{ELU}(z_i)$ is the ELU activation function and $\mathrm{s}_+2\mathrm{L}(z_i)$ is the parameterized softplus activation function [699]. The variant with non-adaptive parameterized softplus is denoted as ELUs$_+$2 [699].

4.3.26 *Global-local neuron (GLN)*

The global-local neuron (GLN) is an AAF that is a convex combination of two AFs proposed in [1156]. It is defined as

$$f(z_l) = \sigma\left(a\right) \cdot \mathrm{global}(z_l) + \left(1 - \sigma\left(a\right)\right) \cdot \mathrm{local}(z_l) - b, \tag{4.439}$$

where a_l and b_l are trainable weights for each layer l and $\text{global}(z_l)$ and $\text{local}(z_l)$ are AFs capable of identifying the global and local characteristics respectively [1156, 1157]; the authors used $\text{global}(z_l) \sin(z_l)$ and $\text{local}(z_l) = \tanh(z_l)$ in [1156, 1157].

4.3.27 *Neuron-adaptive activation function*

A similar approach to trainable amplitude and generalized hyperbolic tangent is the so-called neuron-adaptive activation function (NAF) [1158–1160], which comprises of a linear combination of two activation functions with scalable amplitude:

$$f(z) = a \exp\left(-b \cdot (z)^2\right) + \frac{c}{1 + \exp(-d \cdot z)}, \tag{4.440}$$

where a, b, c, and d are trainable parameters that are shared by the whole network [1158]. The NAF was shown to perform superiorly on a few regression tasks [1158].

4.3.27.1 *Scaled logistic sigmoid*

A scaling variant of logistic sigmoid called scaled logistic sigmoid was proposed in [1161]. The function is defined as

$$f(z_i) = \frac{a_i}{1 + \exp(-b_i z_i)}, \tag{4.441}$$

where a_i and b_i are trainable parameters for each neuron i [1161]. Note that this activation is identical to the second part of the previously proposed NAF (see Section 4.3.27).

A variant combining scaled logistic sigmoid with scaled sine (SLS-SS) was also used in [1161]; it has four trainable parameters and is defined as

$$f(z_i) = a_i \cdot \sin(b_i z_i) + \frac{c_i}{1 + \exp(-d_i z_i)}, \tag{4.442}$$

where a_i, b_i, c_i, and d_i are trainable parameters [1161]. This activation function is a special case of another variant of NAF [1162]:

$$f(z_i) = a_i \cdot \sin(b_i z_i) + c_i \exp\left(-d_i \cdot (z)^2\right) + \frac{e_i}{1 + \exp(-f_i z_i)}, \tag{4.443}$$

where a_i, b_i, c_i, d_i, e_i and f_i are trainable parameters [1162].

4.3.28 *Adaptive piece-wise linear unit (APLU)*

Another generalization of ReLU is the adaptive piece-wise linear unit (APLU), which uses the sum of hinge-shaped functions as the activation function [1163]. An approach extending APLU is smooth adaptive activation function (SAAF) with piece-wise polynomial form and was specifically designed for regression and allows for bias–variance trade-off using a regularization term [1014].

APLU is defined as

$$f(z_i) = \max(0, z_i) + \sum_{s=1}^{S} a_i^s \max(0, -z_i + b_i^s), \tag{4.444}$$

where S is the number of hinges, i is the number of neurons, and a_i^s, b_i^s, $s \in 1, \ldots, S$ are trainable parameters per unit [873]. However, the optimizer might choose very large values of a_i^s and balance them by very small weights, which could lead to numerical instabilities; therefore, an L^2 penalty is added to the parameters a_i^s, b_i^s scaled by 0.001 [1163]. Another adaptive piecewise linear function was proposed in [1164], where a weighted combination of ReLUs with additional parameters was used.

4.3.29 *Simple piecewise linear and adaptive function with symmetric hinges (SPLASH)*

The simple piecewise linear and adaptive function with symmetric hinges (SPLASH) [1165] is an approach similar to the APLU. It is defined as

$$f(z_l) = \sum_{s=1}^{\frac{S+1}{2}} a_{l,s}^+ \max(0, z - b_{l,s}) + \sum_{s=1}^{\frac{S+1}{2}} a_{l,s}^- \max(0, -z - b_{l,s}), \tag{4.445}$$

where S is an odd number, $b_{l,s}$ and $-b_{l,s}$ are hinge parameters and $a_{l,s}^+$ and $a_{l,s}^-$ are scaling parameters for each layer l [1165] ; these max functions form $S+1$ continuous line segments with hinges at $b_{l,s}$ and $-b_{l,s}$ [1165]. While Tavakoli, Agostinelli, and Baldi tried different values for S, they found that using $S = 7$ usually works well [1165].

4.3.30 *Multi-bias activation (MBA)*

An approach similar to APLU and paired ReLU (see Section 4.3.1.26) termed multi-bias activation (MBA) [1166] uses the same activation but with multiple biases, which allows to learn more complex activations; in this it resembles paired ReLU as one input map leads to several output maps with activation with different biases [1166]. The weights that will be given to the output maps in the next layer are similar to the weights in the APLU; however, the MBA is able to provide cross-channel information due to multiple outputs for each activation [1166]. The MBA is defined as

$$f(z_i) = \begin{bmatrix} g(zi + b_{i,1}) \\ g(zi + b_{i,2}) \\ \cdots \\ g(zi + b_{i,k}) \\ \cdots \\ g(zi + b_{i,K}) \end{bmatrix}, \tag{4.446}$$

where $b_{i,k}$, $k = 1, 2, \ldots, K$ are trainable biases and $g(x)$ is any non-linear activation function [1166]; Li, Ouyang, and Wang used ReLU as the activation function $g(x)$ [1166].

4.3.31 *Mexican ReLU (MeLU)*

A Mexican ReLU (MeLU) is an activation function with a similar approach as the APLU, but it does not need any L^2 penalty [1167]. The MeLU is defined as

$$f(z_i) = \text{PReLU}(z_i) + \sum_{j}^{k-1} a_{i,j}\phi_{b_j c_j}(z_i), \tag{4.447}$$

where $a_{i,j}$ are trainable parameters for each neuron/filter i, and k is the total number of trainable parameters ($k-1$ for the sum and one for the PReLU), b_j and c_j are fixed constants that are chosen recursively (more details in [1167]); $\phi_{b_j c_j}(z_i)$ is defined as

$$\phi_{b_j c_j}(z_i) = \max\left(c_j - \left|z_i - b_j\right|, 0\right). \tag{4.448}$$

Maguolo, Nanni, and Ghidoni used $k = 4$ and $k = 8$ for their experiments; the trainable parameters $a_{i,j}$ were all initialized to zero which helps the training at the early stages by exploiting the properties of the ReLU (e.g., the MeLU is convex for many iterations at the beginning)[1167]. The advantage of the MeLU over the APLU is that it needs only half of the parameters while retaining the same representation power when the parameters are jointly optimized with the network's weights and biases [1167].

4.3.31.1 *Modified Mexican ReLU (MMeLU)*

The modified Mexican ReLU (MMeLU) is an MeLU inspired AF proposed in [1168]. It is defined as

$$f(z_i) = a_i \cdot \max\left(b_i - |z_i - c_i|, 0\right) + (1 - a_i)\,\text{ReLU}(z_i), \tag{4.449}$$

where a_i, b_i, and c_i are adaptive parameters estimated using Bayesian procedure outlined in [1168]; $a_i \in [0, 1]$, $b_i \in \mathbb{R}^+$ in , and $c_i \in \mathbb{R}$ [1168].

4.3.31.2 *Gaussian ReLU (GaLU)*

The Gaussian ReLU (GaLU) is a MeLU-inspired AAF proposed in [1169]. It uses the same basic form as MeLU has in Eq. (4.447) but it uses following $\phi_{b_j c_j}(z_i)$:

$$\phi_{b_j c_j}(z_i) = \max\left(c_j - \left|z_i - b_j\right|, 0\right) + \min\left(\left|z - b_j - 2c_j\right| - c_j, 0\right), \tag{4.450}$$

where b_j and c_j are similar parameters is in the original MeLU; more details about the parameters is available in [1169].

4.3.31.3 *Hard-Swish*

The Hard-Swish is an adaptive variant of a scaled Hard sigmoid activation [891]. It is defined as

$$f(z_i) = 2z_i \cdot \max\left(0, \min\left(0.2 b_i z_i + 0.5, 1\right)\right), \tag{4.451}$$

where b_i is either trainable or fixed parameter [891]. For $b_i \to \infty$, the Hard-Swish approaches the ReLU [891]. The Hard-Swish outperformed the logistic sigmoid, tanh, ReLU, LReLU, and swish on the MNIST dataset [45] in [891]. The ResNet [13], wide residual network (WRN) [55], and DenseNet [838] with glshardswish outperformed their variants with ReLU and swish on the CIFAR-10 [243] dataset [891].

4.3.32 *S-shaped rectified linear activation unit (SReLU)*

A S-shaped ReLU (SReLU) [873] consists of three piecewise linear functions that are controlled by four trainable parameters that are learned jointly with the whole network. The SReLU is able to learn both convex and non-convex functions; in particular, it is able to learn both ReLU and also sigmoidlike functions. It is similar to APLU (see Section 4.3.28), but APLU approximates non-convex functions, and it requires the rightmost linear function to have a unit slope and bias of zero [873]. SReLU is defined as

$$f(z_i) = \begin{cases} t_i^r + a_i^r(z_i - t_i^r), & z_i \geq t_i^r, \\ z_i, & t_i^r > z_i > t_i^l, \\ t_i^l + a_i^l(z_i - t_i^l), & z_i \leq t_i^l, \end{cases} \tag{4.452}$$

where t_i^r, t_i^l, a_i^r, and a_i^l are trainable parameters for each neuron i (or channel i in case of convolutional neural networks) [873]. The parameters t_i^r and t_i^l determine thresholds of an interval outside which the slope of the linear parts is controlled by parameters a_i^r and a_i^l, respectively. The authors Jin et al. show that the SReLU outperformed the ReLU, LReLU, PReLU, APLU, maxout unit and plain NIN on several visual tasks. The authors also recommend to initialize the parameters of SReLU to $t_i \in \mathbb{R}$, $a_i^r := 1$, $t_i^l := 0$, and $a_i^l \in (0, 1)$ which degenerates the SReLU into a LReLU and then keep these parameter fixed during several initial training epochs [873]. The SReLU can be seen as a more general concept to the later proposed piecewise linear unit (PLU) (see Section 4.3.34) and to the BLReLU [888] (see Section 4.2.6.24).

4.3.32.1 *N-activation*

The N-activation is activation very similar to a special case of SReLU[75] proposed in [1170]. The N-activation with trainable parameters a_i, and b_i is defined as

$$f(z_i) = \begin{cases} z_i - 2t_{i,\min}, & z_i < t_{i,\min}, \\ -z_i, & t_{i,\min} \leq z_i \leq t_{i,\max}, \\ z_i - 2t_{i,\max}, & z_i > t_{i,\max}, \end{cases} \tag{4.453}$$

where

$$t_{i,\min} = \min(a_i, b_i) \tag{4.454}$$

75 It would be a special case of SReLU if the the thresholds were directly trainable and not determined using the min and max functions.

and

$$t_{i,\max} = \max\left(a_i, b_i\right). \tag{4.455}$$

4.3.32.2 *ALiSA*

A special case of SReLU was later proposed under the name adaptive LiSA (ALiSA) in [1171]; it can be obtained by setting $t_i^r := 1$ and $t_i^l :== 0$:

$$f(z_i) = \begin{cases} a_i^r z_i - a_i^r + 1, & z_i \geq 1, \\ z_i, & t_i^r > z_i > t_i^l, \\ a_i^l z_i, & z_i \leq 0, \end{cases} \tag{4.456}$$

where a_i^r and a_i^l are adaptive parameters [1171]. Its nonadaptive variant is called simply linearized sigmoidal activation (LiSA) and has parameters a_i^r and a_i^l fixed [1171].

4.3.33 *Alternated left ReLU (All-ReLU)*

The alternated left ReLU (All-ReLU) was proposed in [1172] for usage in sparse neural networks. It is inspired by the SReLU [1172]. It is defined as

$$f(z_i) = \begin{cases} -az_i, & z_i \leq 0 \text{ and } l\%2 = 0, \\ az_i, & z_i \leq 0 \text{ and } l\%2 = 1, \\ z_i, & z_i > 0, \end{cases} \tag{4.457}$$

where a is a fixed parameter controlling the slope for negative inputs, l is the number of layers, and % is the modulo operation [1172].

4.3.34 *Piecewise linear unit (PLU)*

A PLU [1173] resembles two earlier proposed activation functions — the SReLU (see Section 4.3.32) and adaptive piece-wise linear unit (see Section 4.3.28); it can be even seen as a special case of the SReLU.

$$f(z_i) = \max\left(a_i\left(z_i + b\right) - b, \min\left(a_i\left(z_i - b\right) + b, z_i\right)\right), \tag{4.458}$$

where a_i is either a trainable parameter or a predefined constant [11] and b is a predefined constant [1173]; a variant with $a = 0.1$ and $b = 1$ was shown in [1173]. The advantage of the PLU compared to the SReLU is that it produces an invertible function (which is not always the case for the more general SReLU) [1173].

4.3.35 *Adaptive linear unit (AdaLU)*

The adaptive linear unit (AdaLU) [1174] is yet another piecewise linear AAF. It is defined as

$$f(z_i) = \begin{cases} c_i\left(z_i - a_i\right) + b_i, & \left(z_i - a_i\right) > 0 \text{ and } c_i\left(z_i - a_i\right) > e_i, \\ d_i\left(z_i - a_i\right) + b_i, & \left(z_i - a_i\right) \leq 0 \text{ and } d_i\left(z_i - a_i\right) > e_i, \\ e_i + b_i, & \text{otherwise,} \end{cases} \tag{4.459}$$

where a_i, b_i, c_i, d_i, and e_i are trainable parameters for each neuron i [1174]. The parameters a_i and b_i control the offsets; c_i and d_i control the slope of each linear part, and e_i is the saturation value [1174].

4.3.36 *Trapezoid-shaped activation function (TSAF)*

The trapezoid-shaped activation function (TSAF) [1175] (ref. from [904]) is an AF consisting of four ReLUs. It is defined as

$$f(z_i) = \frac{1}{c_i}\big(\mathrm{ReLU}\left(z_i - a_i + c_i\right) + \mathrm{ReLU}\left(z_i - a_i\right) + \mathrm{ReLU}\left(z_i + b_i - c_i\right) - \\ \mathrm{ReLU}\left(z_i - b_i\right)\big), \tag{4.460}$$

where a_i, b_i, and c_i are parameters[76] such that $a_i < b_i$ and $c_i \in (0, 1]$ [904].

4.3.37 *Adaptive Richard's curve weighted activation (ARiA)*

Another function motivated by the swish activation function is the Adaptive Richard's curve weighted activation (ARiA) [1176], which replaces the logistic sigmoid in the swish by Richard's curve [11]. Richard's curve [1177] is a generalization of the logistic sigmoid that is controlled by several hyperparameters. The Richard's curve is defined as [1176]:

$$\sigma_R(x) = A + \frac{K - A}{\left(C + Q \cdot \exp\left(-Bx\right)\right)^{\frac{1}{v}}}, \tag{4.461}$$

where A is the lower asymptote, K is the upper asymptote, C is a constant (typically equal to 1 [1176]), $v > 0$ controls the direction of growth and B is the exponential growth rate, Q controls the initial value of the function. The ARiA is defined as

$$f(z) = z \cdot \sigma_R(x), \tag{4.462}$$

where $\sigma_R(x)$ is the Richard's curve from Eq. (4.461) [1176]. As such, the ARiA has five hyperparameters controlling its behavior. To reduce the number of the hyperparameters, Adaptive Richard's curve weighted activation 2 (ARiA2) was also proposed [1176] that is defined by only two hyperparameters a and b

$$f(z) = z \cdot \left(1 + \exp\left(-bz\right)\right)^{-a}. \tag{4.463}$$

76 Pan et al. do not state whether they are used in trainable or fixed form.

The swish activation function is a special case of ARiA with $A = 1$, $K = 0$, $B = 1$, $v = 1$, $C = 1$, and $Q = a_i$, where a_i is the parameter of the swish activation function (see Section 4.3.3.1 for details) [1176]. The ARiA2 is a special case of ARiA with $K = 0$, $B = 1$, $v = 1$, $C = \frac{1}{a}$, and $Q = b$, where a and b are the ARiA2 hyperparameters [1176]. Patwardhan, Ingalhalikar, and Walambe reached best accuracy on the MNIST [45] dataset with a custom CNN using ARiA2 with $a = 1.5$ and $b = 2$; the best parameters for the DenseNet [838] were $a = 1.75$ and $b = 1$ [1176]. While the parameters were fixed in the experiments in [1176], they can also be trainable as is in the special case of the swish activation function.

4.3.38 *Modified Weibull function*

A modified Weibull function (MWF) is an Weibull-function-based AF proposed in [1140]. It is defined as

$$f(z_i) = \left(\frac{z_i}{a_i}\right)^{b_i - 1} \exp\left(-\left(\frac{z_i}{c_i}\right)^{d_i}\right), \tag{4.464}$$

where a_i, b_i, c_i, and d_i are trainable parameters [1140]. The parameter b_i determines the location of the peak of the AF [1140]. The polynomial term dominates for small input values while the exponential starts to dominate with larger values which reduces the output value as the input value further increases [1140].

4.3.39 *Sincos*

The sincos is another older AF proposed in [996]. It is defined as

$$f(z) = a \cdot \sin(bz) + c \cdot \cos(dz), \tag{4.465}$$

where a, b, c, and d are adaptive parameters [996].

4.3.40 *Combination of sine and logistic sigmoid (CSS)*

The combination of sine and logistic sigmoid (CSS)[77] is an AAF proposed in [1097]. It is defined as

$$f(z) = a \cdot \sin(bz) + c \cdot \sigma(dz), \tag{4.466}$$

where a, b, c, and d are adaptive parameters [1097].

4.3.41 *Catalytic activation function (CatAF)*

The catalytic activation function (CatAF) is an AAF that uses sinusoidal mixing of any AF and the identity to produce the final activation [1178]. It is defined as

$$f(z_i) = z_i \sin(a_i) + g(z_i) \cos a_i, \tag{4.467}$$

77 The function was unnamed in [1097]; we used this abbreviation to distinguish it from SinSig.

where a_i is a trainable parameter and $g(z_i)$ is any AF such as the ReLU [1178].

4.3.42 *Expcos*

An AAF combining an exponential function with the cosine was proposed in [1097]. It is called expcos in this work[78] and is defined as

$$f(z) = \exp\left(-az^2\right) \cdot \cos\left(bz\right), \tag{4.468}$$

where a and b are adaptive parameters [1097].

4.3.43 *Multi-bin trainable linear unit (MTLU)*

The multi-bin trainable linear unit (MTLU) can be seen as a conceptual extension of the SReLU (see Section 4.3.32) into more than three segments [11]:

$$f(z_i) = \begin{cases} a_{i,0}z + b_{i,0}, & z_i \geq c_{i,0}, \\ a_{i,1}z + b_{i,1}, & c_{i,0} < z_i \geq c_{i,1}, \\ \ldots \\ a_{i,k}z + b_{i,k}, & c_{i,k-1} < z_i \geq c_{i,k}, \\ \ldots \\ a_{i,K}z + b_{i,K}, & c_{i,K-1} < z_i, \end{cases} \tag{4.469}$$

where $a_{i,0}, \ldots, a_{i,K}$, and $b_{i,0}, \ldots, b_{i,K}$ are trainable parameters for each neuron/filter and K and $c_{i,0}, \ldots, c_{i,K-1}$ are predefined hyperparameters [1179]. The authors used unifromly distributed anchors $c_{i,0}, \ldots, c_{i,K-1}$ [1179]. The main disadvantage besides the higher number of additional parameters is the higher number of non-differentiable points [11]. The MTLU was also named continuous piecewise nonlinear activation function (CPN)$_{\mathrm{m}}$ in [1180]. The CPN$_{\mathrm{mc}}$ is a MTLU variant with continuity constraint proposed in [1180].

An AF with the same form as the MTLU with only minor differences was proposed in [1181, 1182] under the name piecewise linear unit (PWLU); Zhu et al. also proposed its 2D extension in [1182]. Unlike the MTLU, it uses a uniformly spaced demarcation points $c_{i,k}$ [1181]. Another PWLU variant named non-uniform piecewise linear unit (N-PWLU) allows for learnable intervals on which the function is piecewise linear, and also it leverages cumulative definition for efficient learning [1183]. Multistability analysis of such piecewise linear AFs is analyzed in [1184]. An analysis of a number of regions of piecewise linear NNs is available in [1185].

78 The function was originally unnamed in [1097].

4.3.44 *Continuous piecewise nonlinear activation function CPN*

A variant of the MTLU named CPN where the $c_{i,k}$ was used in [1180]. It is defined as

$$f(z_i) = \begin{cases} a_{i,0}z + b_{i,0} + c_{i,0}g(z_i), & z_i \geq d_{i,0}, \\ a_{i,1}z + b_{i,1} + c_{i,1}g(z_i), & d_{i,0} < z_i \geq d_{i,1}, \\ \dots \\ a_{i,k}z + b_{i,k} + c_{i,k}g(z_i), & d_{i,k-1} < z_i \geq d_{i,k}, \\ \dots \\ a_{i,K}z + b_{i,K} + c_{i,K}g(z_i), & d_{i,K-1} < z_i, \end{cases} \tag{4.470}$$

where $a_{i,0}, \dots, a_{i,K}$, $b_{i,0}, \dots, b_{i,K}$, $c_{i,0}, \dots, d_{i,K-1}$ are trainable parameters for each neuron/filter, $g(z_i)$ is a non-linear function such as the logistic sigmoid and K and $c_{i,0}, \dots, d_{i,K-1}$ are predefined hyperparameters [1180].

Gao et al. also proposed a variant named CPN_{nl}, which introduces a non-linear term for each small interval and does not enforce the uniform division of the activation space [1180]. It is defined as

$$f(z_i) = \max\left\{p_{i,0}(z), p_{1,0}(z), \dots, p_{i,k}(z), \dots, p_{i,K}(z)\right\}, \tag{4.471}$$

where

$$p_k(z) = a_{i,k}z_i + b_{i,k}\text{SiLU}(z_i) + c_i, \quad k = 0, 1, \dots, K, \tag{4.472}$$

where K is the number of functions and $a_{i,k}$, $b_{i,k}$, and $c_{i,k}$ are learnable coefficients for $k = 0, 1, \dots, K$ [1180].

4.3.45 *Look-up table unit (LuTU)*

A piecewise activation function look-up table unit (LuTU) [1186–1188] is a learnable activation function that consists of several points defining the function; the values between the points are obtained using either linear interpolation or smoothing with single period cosine mask function [1186]. Similar adaptive activation function using linear interpolation was used in [1189, 1190]. A look-up table of anchor points $\{a_{i,j}, b_{i,j}\}, j = 0, 1, \dots, n$ that are uniformly spaced with step s, $a_{i,j} = a_0 + s \cdot j$, controls the shape of the activation functions. The step s, anchor points a_0, and n are predetermined hyperparameters, and therefore $a_{i,j}$ are predetermined values for which the output values $b_{i,j}$ are learnable parameters. Linear-interpolation-based function is defined as

$$f(z_i) = \frac{1}{s}\left(b_{i,j}\left(a_{i,j+1} - z_i\right) + b_{i,j+1}\left(z_i - a_{i,j}\right)\right), \quad a_{i,j} \geq z \geq a_{i,j+1}, \tag{4.473}$$

where $a_{i,j}$ are hyperparameters defined by the step s and initial point a_0 shared for all points and $b_{i,j}$ are trainable parameters for each neuron i [1186].

Wang, Liu, and Foroosh used $a_0 = -12$, step $s = 0.1$ and $n = 240$ to cover the interval $[-12, 12]$ using 241 anchor poiints for each neuron. Therefore, for any input value between $a_{i,j}$ and $a_{i,j+1}$, the output is linearly interpolated from $b_{i,j}$ and $b_{i,j+1}$ [1186]. However, such a definition might lead to unstable gradients [1186]; therefore, a variant of LuTU with cosine smoothing was also proposed. The smoothing function is defined as

$$r(x,\tau) = \begin{cases} \frac{1}{2\tau}\left(1+\cos\left(\frac{\pi}{\tau}x\right)\right), & -\tau \geq x \geq \tau, \\ 0, & \text{otherwise}, \end{cases} \tag{4.474}$$

where τ is a hyperparameter controlling the period (2τ) of the cosine function [1186]. The smoothed variant of the LuTU is then defined as

$$f(z_i) = \sum_{j=0}^{n} y_j r\left(z_i - a_{i,j}, ts\right), \tag{4.475}$$

where t is an integer defining the ration between τ and s [1186]. The formula in Eq. (4.475) can be further simplified as it is not necessary to sum over all $j \in \{0, 1, \ldots, n\}$ as the smoothing function has a truncated input domain, more details in [1186].

4.3.46 *Maxout unit*

Maxout unit returns the maximum of multiple linear functions per each unit i [1191]:

$$f(z_i) = \max_{k\in\{1,\ldots,K\}} w_i^k z_i + b_i^k \tag{4.476}$$

where K is the number of linear functions. The maxout unit can also be used directly on inputs of the neuron as shown in [1191] (by replacing $w_i^k z_i$ with $\boldsymbol{x_i}^T \boldsymbol{w}_i^k$ where $\boldsymbol{x_i} \in \mathbb{R}^d$ is the vector of individual inputs to a neuron i and $w \in \mathbb{R}^d$ are trainable weights [1191]) but the equation presented here uses only the hidden state for simplicity. The advantage of maxout unit is that it is a universal approximator of a convex function [873, 1191]; however, it cannot learn non-convex functions [873] and introduces a high number of additional parameters per neuron [873, 1191]. While some works show that maxout unit perform superiorly [1191, 1192], other experiments show that ReLU, which is a special case of maxout, performs better [1193]. Furthermore, since the maxout unit is more complex than regular ReLU, the training is relatively slower [1193].

Empirical comparison of the maxout unit with ReLU, LReLU, SELU and tanh is available in [1193]; with ReLU, tanh, sigmoid and VLReLU in [868].

4.3.47 *Adaptive blending unit (ABU)*

An approach mixing several activation functions was described in [1194] where adaptive blending unit (ABU) was introduced. The ABU is a weighted sum of several predefined activations [1194]. It is defined as

$$f(z_l) = \sum_{j=0}^{n} a_{j,l} g_j(z_l), \tag{4.477}$$

where $g_j(z_l)$ is an activation function from a pool of n activation functions and $a_{j,l}$ is a weighting parameter that is trained for each layer l and activation function $g_j(z_l)$. The ABU was first proposed as a special case of a general framework called TAF already in 1997 [1009]. The blending weights $a_{j,l}$ are initialized to $\frac{1}{n}$ but are then trained alongside the weights of the NN [1194]. Sütfeld et al. used tanh, ELU, ReLU, swish, and the identity as the pool of activation functions g_j but they admit that no exhaustive search was performed to select this set and that there might be other pools that perform better [1194]. This approach was also used in [889] where ReLU, logistic sigmoid, tanh, and softsign activation functions were used.

However, similar approach was already proposed in [1186] where Wang, Liu, and Foroosh inspired by the mixture of Gaussian unit (MoGU) (see Section 4.3.47.10) generalized the concept to mixing several different activation functions

$$f(z_i) = \sum_{j=0}^{n} a_{i,j} g_j(z_i - b_{i,j}), \tag{4.478}$$

where $g_j(z_i)$ is an activation function from a pool of n activation functions, $a_{i,j}$ is a trainable weighting parameter of the function $g_j(z_i)$ and $b_{i,j}$ is a trainable parameter controlling the vertical shift of the function $g_j(z_i)$ for each neuron i [1186]. Furthermore, if $g_j(z_i)$ already contains a way for controlling its scale or shift, the parameters $a_{i,j}$ and $b_{i,j}$ can be discarded [1186]. This approach is identical to the ABU from [1194] if $b_{i,j} = 0$ and the parameters are shared by all neurons in the same layer and not learned for each neuron separately.

A very similar approach was proposed in [1195], where Manessi and Rozza use a linear combination of activation functions from a selected pool as the final activation function. The difference from the ABU is that the weights are constrained such that they sum up to 1 [1194, 1195]. Manessi and Rozza uses analyses the linear combination of identity, ReLU, and tanh activation functions [1195]. Sütfeld et al. analyzed the performance of unconstrained ABUs and ABUs with various constraints such as $\sum_{j=0}^{n} a_{j,l} = 1$, $\sum_{j=0}^{n} \left|a_{j,l}\right| = 1$, and two approaches enforcing $\sum_{j=0}^{n} a_{j,l} = 1$ and $a_{j,l} > 0$ — clipping of negative values $a_{j,l}$ before normalization and softmax normalization [1194]. It was found that the unconstrained ABU works the best on average on the selected tasks; however, some of the constrained variants performed better than the unconstrained ABU for particular tasks [1194].

Another variant of ABU (called by Klabjan and Harmon *activation ensemble*) was proposed in [1196] — the final activation is a weighted sum of activation functions; the weighting coefficients has to sum-up to 1 (similarly to [1195]).

However, unlike in the work [1195], the individual activation functions are scaled before the weighting to the interval $[0,1]$ using min–max scaling [1196]:

$$h_j(z) = \frac{g_j(z) - \min_k\left(g_j\left(z_k\right)\right)}{\max_k\left(g_j\left(z_k\right)\right) - \min_k\left(g_j\left(z_k\right)\right) + \epsilon}, \tag{4.479}$$

where g_j are individual activation functions, ϵ is a small number and k goes through all training samples in a minibatch.The final output is

$$f(z_i) = \sum_{j=0}^{n} a_{j,i} h_j(z_i), \tag{4.480}$$

where $a_{j,i}$ is a weight for each neuron i and activation function j, n is the total number of the individual activation functions in the ABU; the weights $a_{j,i} \in [0,1]$ are constrained such that

$$\sum_{j=0}^{n} a_{j,i} = 1. \tag{4.481}$$

4.3.47.1 *Trainable compound activation function (TCA)*

The trained compound activation function (TCA) [1197] is an AAF similar to the ABU [1194] and especially to its variant with the bias (see Eq. (4.478)) [1186]; however, unlike the form from Eq. (4.478) it uses horizontal scaling instead of the vertical. It was defined in [1197] as

$$f(z_i) = \frac{1}{k}\sum_{j=1}^{k} f_j\left(\exp\left(a_{i,j}\right) z_i + b_{i,j}\right), \tag{4.482}$$

where k is the number of mixed functions and $a_{i,j}$ and $b_{i,j}$, $j = 1, \ldots, k$, are scaling and translation trainable parameters for each neuron i and function j [1197]. The TCA was found to improve the performance of RBMs and DBNs [1197].

Later, Baggenstoss introduced a TCA also with vertical scaling parameters in [1198]. This slightly different variant is denoted as trained compound activation function variant 2 (TCAv2) throughout this work. TCAv2 is defined as

$$f(z_i) = \frac{\sum_{j=1}^{k} \exp\left(a_{i,j}\right) f_j\left(\exp\left(b_{i,j}\right) z_i + c_{i,j}\right)}{\sum_{j=1}^{k} \exp\left(a_{i,j}\right)}, \tag{4.483}$$

where k is the number of mixed functions and $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$, $j = 1, \ldots, k$, are scaling and translation trainable parameters for each neuron i and function j [1198].

4.3.47.2 *Average of a pool of activation functions (APAF)*

An average of a pool of activation functions (APAF) was used in [1199]; the output is defined as

$$f(z_i) = \frac{\sum_{j=0}^{n} a_{j,i} h_j(z_i)}{\sum_{j=0}^{n} a_{j,i}}. \tag{4.484}$$

Liao used the ReLU, logistic sigmoid, tanh, and the linear functions as the candidate functions in the pool [1199]. This approach was also used in [889].

4.3.47.3 *Gating adaptive blending unit (GABU)*

Yet another approach previously proposed employs a gated linear combination of activation functions for each neuron [1013] — the variant is called gating adaptive blending unit (GABU) throughout this work. This allows each neuron to choose which activation function (from an existing pool) it may use to minimize the error [1013]. A similar method uses just binary indicators instead of the gates [1200]. The gating variant of ABU from [1013] is defined as

$$f(z_i) = \sum_{j=0}^{n} \sigma\left(a_{j,i}\right) g_j(z_i), \tag{4.485}$$

where $\sigma\left(a_{j,i}\right)$ is the logistic sigmoid function acting as gating function and $a_{j,i}$ is a trainable parameter controlling the weight of the activation function g_j for each neuron i.

4.3.47.4 *Deep Kronecker neural networks*

The concept of ABUs was further generalized in the framework of Deep Kronecker neural networks (DKNNs) [1201], which provides an efficient way of constructing wide networks with adaptive activation functions while keeping the number of parameters low [1201]. DKNNs are equivalent to the feed-forward neural networks with an adaptive activation function f defined as

$$f(z_l) = \sum_{j=0}^{n} a_{l,j} g_j(b_{l,j} z_l), \tag{4.486}$$

where z_l is a preactivation of a neuron from a layer l, $a_{l,j}$ and $b_{l,j}$ are either trainable or fixed parameters and g_j, $j = 1, \ldots, n$ are fixed activation functions [1201].

4.3.47.5 *Rowdy activation functions*

Rowdy activation functions are a general class of activation functions that is a special case of DKNNs (see Section 4.3.47.4). A rowdy activation function is a DKNNs with any activation function (e.g., ReLU) that is the function g_0 from Eq. (4.486) and n other functions that are defined as

$$g_j(z_l) = c \cdot \sin\left(jcz_l\right), \tag{4.487}$$

or

$$g_j(z_l) = c \cdot \cos\left(jcz_l\right), \tag{4.488}$$

where $c \geq 1$ is a fixed scaling factor and $j = 1, \ldots, n$ [1201]. The rowdy activation functions introduce highly fluctuating, non-monotonic terms that remove saturation regions from the output of each layer in the network [1201] similarly as does the stochastic noise in [1029].

4.3.47.6 *Self-learnable activation function (SLAF)*

The SLAF [1202] can be considered to be a special case of the ABU where the function $g_j(z_i)$ are increasing powers of z_i:

$$f(z_i) = \sum_{j=0}^{k-1} a_{i,j} z_i^j, \tag{4.489}$$

where $a_{i,j}$ are learnable parameters for each neuron i and k is a hyperparameter defining the number of elements in the polynomial expression [11, 1202]. However, since the gradient is proportional to z_i and its powers, Goyal, Goyal, and Lall used mean-variance normalization over the training sample to avoid exploding or vanishing gradients [1202]. A similar concept was analyzed in [1203], where it was applied to the output neuron only. A similar approach was used independently in [1204], where authors used the equivalent of SLAF with $k = 6$. A quadratic variant (i.e., SLAF with $k = 2$) was used in [1205].

4.3.47.7 *Chebyshev polynomial-based activation function (ChPAF)*

A Chebyshev polynomial-based activation function (ChPAF) was proposed in [1206]. The function is defined as

$$f(z) = \sum_{j=0}^{k} a_j C_j(z), \tag{4.490}$$

where a_j, $j = 0, \ldots, k$ are learnable parameters shared by a whole network, k is a fixed hyperparameter denoting the maximum order of used Chebyshev polynomials, and $C_j(z)$ is a Chebyshev polynomial of order j defined as

$$C_{j+1}(z) = 2zC_j(z) - C_{j-1}(z) \tag{4.491}$$

with starting values $C_0(z) = 1$ and $C_1(z) = z$ [1206]. Deepthi, Vikram, and Venkatappareddy used polynomials of a maximum order of 3 in their experiments [1206]. The Chebyshev activation function was found to outperform several activation functions including ReLU, ELU, mish and swish while retaining fast convergence using the CIFAR-10 dataset [243] as shown in experiments [1206].

4.3.47.8 *Legendre polynomial-based activation function (LPAF)*

A Legendre polynomial-based activation function (LPAF) was used for the study of approximations of several non-linearities in [1207]. The activation is a linear combination of Legendre polynomials and is defined as

$$f(z) = \sum_{j=0}^{k} a_j G_j(z), \tag{4.492}$$

where a_j, $j = 0, \ldots, k$ are learnable parameters shared by a whole network, k is a fixed hyperparameter denoting the maximum order of used Legendre polynomials, and $G_j(z)$ is a Legendre polynomial of order j defined as

$$G_{j+1}(z) = \frac{2k+1}{k+1} z G_k(z) - \frac{k}{k+1} G_{k-1}(z_i), \tag{4.493}$$

with starting values $G_0(z) = 1$ and $G_1(z) = z$ [1207]. The LPAF was found to outperform ELU, ReLU, LReLU, and softplus on the MNIST [45] and Fashion MNIST [950] datasets [1207].

4.3.47.9 *Hermite polynomial-based activation function (HPAF)*

The Hermite polynomial-based activation function (HPAF) [1208] is an AAF similar to ChPAF and LPAF but it used the Hermite polynomials instead. It is defined as

$$f(z) = \sum_{j=0}^{k} a_j H_j(z), \tag{4.494}$$

where a_j is a trainable parameter and $H_j(z)$ is the Hermite polynomial

$$H_j(z) = (-1)^j \exp\left(z^2\right) \frac{\mathrm{d}^j}{\mathrm{d}z^j} \left(\exp\left(-z^2\right)\right), j > 0 \tag{4.495}$$

and

$$H_0(z) = 1. \tag{4.496}$$

4.3.47.10 *Mixture of Gaussian unit (MoGU)*

The mixture of Gaussian unit (MoGU) was proposed in [1186] as a byproduct of analysis of the behavior of the LuTU unit (see Section 4.3.45) as the shape of learned activation units with the cosine smoothing mostly composed of a few peaks and valleys [1186]. The MoGU is defined as

$$f(z_i) = \sum_{j=0}^{n} \frac{a_{i,j}}{\sqrt{2\pi\sigma_{i,j}^2}} \exp\left(-\frac{\left(z_i - \mu_{i,j}\right)^2}{2\sigma_{i,j}^2}\right), \tag{4.497}$$

where $a_{i,j}$, $\sigma_{i,j}$, and $\mu_{i,j}$ are trainable parameters for each neuron i and Gaussian j from the mixture [1186]. The parameter $a_{i,j}$ controls the scale, $\sigma_{i,j}$ controls the standard deviation, and $\mu_{i,j}$ controls the mean of the Gaussian j for neuron i [1186].

4.3.47.11 *Fourier series activation*

The Fourier series activation (FSA) was proposed in [1199]. It is defined as

$$f(z_i) = a_i + \sum_{j=1}^{r} \left(b_{i,j} \cos\left(j d_i z_i\right) + c_{i,j} \sin\left(j d_i z_i\right)\right), \tag{4.498}$$

where a_i, $b_{i,j}$, $c_{i,j}$, d_i are trainable parameters for each neuron i, and r is a fixed hyperparameter denoting the rank of the Fourier series [1199]; Liao used $r = 5$ throughout his experiments.

4.3.48 *Padé activation unit (PAU)*

Padé activation units (PAUs) [1209] are adaptive activations based on the Padé approximant [1210, 1211]. The PAU is defined as

$$f(z) = \frac{\sum_{j=0}^{m} a_j z^j}{1 + \sum_{k=1}^{n} b_k z^k}, \tag{4.499}$$

where m and n are hyperparameters denoting the order of the polynomials and a_j, $j = 0, \ldots, m$ and b_k, $k = 1, \ldots, n$ are trainable parameters that are globaly shared by all units [1209]. While the Padé approximation could be used to approximate particular activation function, the parameters a_j and b_k are optimized freely with other weights of the neural network [1209]. This PAU variant was for reinforcement learning in [1212] where Delfosse et al. observed that rational functions might replace some of the residual blocks in ResNets. To avoid numerical instabilities, a *safe PAU* ensures that the polynomial in the denominator cannot be zero [1209]; it is defined as

$$f(z) = \frac{\sum_{j=0}^{m} a_j z^j}{1 + \left|\sum_{k=1}^{n} b_k z^k\right|}. \tag{4.500}$$

The hyperparameters were set to $m = 5$ and $n = 4$ in experiments in [1209]. The notion of using rational functions in activations was further analyzed in [1213] where authors used activation function equivalent to Eq. (4.499) with distinct parameters for each layer to learn *rational neural networks*; the safe variant of PAU (Eq. (4.500)) was not used as it results in non-smooth activation function and expensive calculation of gradient during training [1213]. Boulle, Nakatsukasa, and Townsend used low degrees $m = 3$ and $n = 2$ in their work [1213]; this is in contrast to [1214] where rational functions of higher orders were used in a graph neural networkss.

4.3.49 *Randomized Padé activation unit (RPAU)*

The PAU can be extended similarly as RReLU extends ReLU, resulting in randomized Padé activation unit (RPAU) [1209]. Let $\boldsymbol{C} = \{a_0, \ldots, a_m, b_0, \ldots, b_n\}$ be coefficients of PAU activation (see Section 4.3.48). Then an additive noise is introduced into each coefficient $c_j \in \boldsymbol{C}$ during training for every input z_k such that $c_{j,k} = c_j + z_{j,k}$, where $z_{j,k} \sim \mathrm{U}(l_j, u_j)$, $l_j = (1-a)c_j$ and $u_j = (1+a)c_j$ [1209]. This results in RPAU:

$$f(z_k) = \frac{c_{0,k} + c_{1,k} z_k + c_{2,k} z_k^2 + \ldots + c_{m,k} z_k^m}{1 + \left|c_{m+1,k} z_k + c_{m+2,k} z_k^2 + \ldots + c_{m+n,k} z_k^n\right|}, \tag{4.501}$$

where z_k is output of a unit for training input k [1209].

4.3.50 *Enhanced rational activation (ERA)*

The enhanced rational activation (ERA) [1215] function is very similar to the original PAU (see Section 4.3.48); however, Trimmel et al. note similarly as

Boulle, Nakatsukasa, and Townsend that the safe version of PAU is costly to compute whereas the original PAU has undefined values on poles (values of z where the denominator in PAU is equal to zero). To avoid both the poles and the use of absolute value, a modified rational function without the poles is used [1215]. The ERA is defined as

$$f(z) = \frac{P(z)}{Q_C(z)} = \frac{\sum_{j=0}^{m} a_j z^j}{\epsilon + \Pi_{k=1}^{n}\left((z-c_k)^2 + d_k^2\right)}, \tag{4.502}$$

where a_j, $j = 0, \ldots, m$, c_k, and d_k, $k = 1, \ldots, n$ are trainable parameters for each layer and $\epsilon > 0$ is a small number helping to avoid numerical instabilities when d_k are small [1215]. In practice, Trimmel et al. used $\epsilon = 10^{-6}$ [1215]. The ERA in Eq. (4.502) can be rewritten using partial fractions, which reduces the number of operations and, therefore, leads to more efficient computation [1215]. Trimmel et al. used $m = 5$ and $n = 4$ for their experiments.

4.3.51 *Orthogonal Padé activation unit (OPAU)*

The orthogonal Padé activation unit (OPAU) is an extension of the PAU proposed in [1216]. It is defined as

$$f(z) = \frac{\sum_{j=0}^{m} a_j r_i(z)}{1 + \sum_{k=1}^{n} b_k r_k(x)}, \tag{4.503}$$

where a_j, $j = 0, \ldots, m$ and b_k, $k = 1, \ldots, n$ are trainable weights, m and n are fixed parameters, and $r_j(z)$ belongs to a set of orthogonal polynomials [1216]. The *sage OPAU* is defined[79] as

$$f(z) = \frac{\sum_{j=0}^{m} a_j r_i(z)}{1 + \sum_{k=1}^{n} |b_k|\,|r_k(x)|}, \tag{4.504}$$

with identical parameters as the OPAU from Eq. (4.503) [1216]. Biswas, Banerjee, and Pandey used six bases for orthogonal polynomials — Chebyshev polynomials (two variants), Hermite polynomials (also two variants), Laguerre, and Legendre polynomials — as shown in Table 4.2.

4.3.52 *Spline interpolating activation functions*

More complex approaches include spline interpolating activation functions (SAFs) [653, 1012, 1015, 1217–1228], which facilitate the training of a wide variety of activation functions using interpolation. One common example is the cubic spline interpolation that was used in [1015]. The SAFs are controlled by a vector $\boldsymbol{q} \in \mathbb{R}^k$ of internal parameters called *knots*, which are a sampling of the AF over k representative points [1015]. The output is computed using a spline interpolation using the closest knot and its p rightmost neighbors; $p = 3$ results in cubic interpolation [1015]. Spline-based activation functions were also used in the ExSpliNet — an interpretable approach combining

79 Using notation as described in the original article by Biswas, Banerjee, and Pandey [1216].

polynomial	definition
Chebyshev (first kind)	$r_0(z) = 1$, $r_1(z) = z$, $r_{n+1}(z) = 2zr_n(z) - r_{n-1}(z)$
Chebyshev (second kind)	$r_0(z) = 1$, $r_1(z) = 2z$, $r_{n+1}(z) = 2zr_n(z) - r_{n-1}(z)$
Laguerre	$r_0(z) = 1$, $r_1(z) = 1 - z$, $r_{n+1}(z) = \frac{(2n+1-z)r_n(z) - nr_{n-1}(z)}{n+1}$
Legendre	$r_n(z) = \sum_{k=0}^{\left[\frac{n}{2}\right]} (-1)^k \frac{(2n-2k)!}{2^n k!(n-2k)!(n-k)} z^{n-2k}$
Probabilist's Hermite	$r_n(z) = (-1)^n \exp\left(\frac{z^2}{2}\right) \frac{\mathrm{d}^n}{\mathrm{d}z^n}\left(\exp\left(-\frac{z^2}{2}\right)\right)$
Physicist's Hermite	$r_n(z) = (-1)^n \exp\left(z^2\right) \frac{\mathrm{d}^n}{\mathrm{d}z^n}\left(\exp\left(-z^2\right)\right)$

Table 4.2: **Polynomial bases used in OPAU**
List of polynomial bases used in the OPAU taken [1216]. The Chebyshev polynomials and the Laguerre polynomial have recurrent definitions; whereas the Legendre and the Hermite polynomials are defined by a single expression.

neural networks and ensembles of probabilistic trees [1229]. A set of fixed but highly redundant knots for spline interpolation was used in [1220], where the authors then relied on the sparsifying effect of L_1 regularization to nullify the coefficients that are not needed [1220]. Spline flexible activation functions were used for sound synthesis in [1230]. The usage of splines led to the creation of b-spline-based neural networks, e.g., [1231].

Similar to the SAF is the piecewise polynomial activation function (PPAF) [1232] that is also defined by a number of points where the function switches from one polynomial to another [1232]. López-Rubio et al. used zeroth-order, first-order, and third-order polynomials for the piecewise function [1232]; for example, zeroth-order PPAF uses step function and is defined as

$$f(z_i) = \begin{cases} 0, & z_i < q_{i,1}, \\ \frac{k}{m}, \quad q_{i,k} \leq & z_i < q_{i,k+1}, \\ 1, & z_i \geq q_{i,m}, \end{cases} \tag{4.505}$$

where $m - 1$ is the number of controlling points $q_{i,k}$ of a neuron i, $k \in \{1, 2, \dots, m-1\}$ [1232]. The position of control points is determined using the learning procedure outlined in [1232].

If there are no constrains and the AF is limited to linear splines, the AF can be also defined using one hidden layer with ReLUs [1226]:

$$f(z_i) = \sum_{k=1}^{K} a_{i,k} \mathrm{ReLU}\left(b_{i,k} z_i + c_{i,k}\right), \tag{4.506}$$

where $K \in \mathbb{N}$ and $a_{i,k}$, $b_{i,k}$, and $c_{i,k}$ are trainable parameters [1226].

4.3.53 *Truncated Gaussian unit (TruG)*

A truncated gaussian unit (TruG) [1233] is a unit in a probabilistic framework that is able to well approximate sigmoid, tanh, and ReLU. It is controlled by truncation points ξ_1 and ξ_2 and under the probabilistic framework described in [1233] is defined as

$$\mathrm{E}(h|z,\xi_1,\xi_2) = z + \sigma \frac{\phi\left(\frac{\xi_1 - z}{\sigma}\right) - \phi\left(\frac{\xi_2 - z}{\sigma}\right)}{\Phi\left(\frac{\xi_1 - z}{\sigma}\right) - \Phi\left(\frac{\xi_2 - z}{\sigma}\right)}, \tag{4.507}$$

where $\phi(x)$ is the probability density function (PDF) of a univariate Gaussian distribution with mean z and variance σ^2 and $\Phi(x)$ its CDF [1233]. The truncation points can be either selected manually or tuned with the rest of the weights [1233].

4.3.54 *Mollified square root function (MSRF) family*

Pan et al. used a smoothing approach on piecewise linear AFs to create a whole new family of AFs in [904]. The approach is based on the mollified square root function (MSRF) method. This smoothing approach was first used in [905] and then in the SquarePlus AF in [1234], which inspired Pan et al. in the creation of the MSRF family of AFs.

For example, the absolute value $|x|$ is not differentiable at $x = 0$, but it can be regularized by mollification as

$$|x|_\epsilon = \sqrt{x^2 + \epsilon}, \tag{4.508}$$

where ϵ is a small positive parameter and $\lim_{\epsilon \to 0^+} |x|_\epsilon = |x|$ [904].

4.3.54.1 *SquarePlus*

The SquarePlus [1234] is the first AF that used the mollification procedure described in Section 4.3.54 above. It is defined as

$$f(z) = \frac{1}{2}\left(z + |z|_\epsilon\right) = \frac{1}{2}\left(z + \sqrt{z^2 + \epsilon}\right). \tag{4.509}$$

The SquarePlus is very similar to the softplus (see Section 4.2.17) for $\epsilon = 4\left(\ln(2)\right)^2$ and they produce identical outputs at $z = 0$ [904].

4.3.54.2 *StepPlus*

As the SquarePlus approximates the ReLU, the StepPlus approximates the step function (see Section 4.2.1) similarly as the logistic sigmoid does [904]. It is defined as

$$f(z) = \frac{1}{2}\left(1 + \frac{z}{|z|_\epsilon}\right). \tag{4.510}$$

The sign function is smoothed into the BipolarPlus AF [904]

$$f(z) = \frac{z}{|z|_\epsilon}. \tag{4.511}$$

4.3.54.3 *LReLUPlus*

A smoothed variant of the LReLU called LReLUPlus is defined as

$$f(z_i) = \frac{1}{2}\left(z_i + a_i z_i + |(1 - a_i)\, z_i|_\epsilon\right), \tag{4.512}$$

where $|x|_\epsilon$ is the MSRF procedure [904] described in Section 4.3.54 and a_i is a fixed or trainable parameter.

A function equivalent to the LReLUPlus was independently proposed in [1025] under the name SMU-1. The only difference was that Biswas et al. used parameter μ that is the square root of ϵ from Section 4.3.54: $\epsilon = \mu^2$.

4.3.54.4 *vReLUPlus*

The vReLUPlus [904] is a MSRF smoothed variant of the vReLU (see Section 4.2.6.25); it is defined as

$$f(z) = |z|_\epsilon\,. \tag{4.513}$$

4.3.54.5 *SoftshrinkPlus*

The smoothed variant of the Softshrink (see Section 4.2.6.23) is named SoftshrinkPlus[80] [904] and is defined as

$$f(z) = z + \frac{1}{2}\left(\sqrt{(z-a)^2 + \epsilon} - \sqrt{(z+a)^2 + \epsilon}\right), \tag{4.514}$$

where a is a fixed parameter similar to the original Softshrink's thresholding parameter [904].

4.3.54.6 *PanPlus*

The MSRF procedure can be also used to smooth the pan AF (see Section 4.2.6.26) [904]; the resulting PanPlus [904] is defined as

$$f(z) = -a + \frac{1}{2}\left(\sqrt{(z-a)^2 + \epsilon} + \sqrt{(z+a)^2 + \epsilon}\right), \tag{4.515}$$

where a is a fixed thresholding parameter of the pan function [904].

4.3.54.7 *BReLUPlus*

The BReLUPlus [904] is a MSRF smoothed variant of the BReLU (see Section 4.2.6.16) defined as

$$f(z) = \frac{1}{2}\left(1 + |z|_\epsilon - |z-1|_\epsilon\right). \tag{4.516}$$

80 Pan et al. named the function STFPlus originally [904].

4.3.54.8 *SReLUPlus*

Another smoothed AF is the SReLUPlus which is the smoothed variant of the SReLU (see Section 4.3.32) [904]; it is defined as

$$f(z_i) = a_i z_i + \frac{1}{2}(a_i - 1)\left(|z_i - t_i|_\epsilon - |z_i + t_i|_\epsilon\right), \tag{4.517}$$

where a_i has similar role as in the original SReLU and t_i is a parameter for symmetric variant of SReLU with $t_i = t_i^r = t_i^l$ [904].

4.3.54.9 *HardTanhPlus*

Similarly, the smoothed variant of the HardTanh (see Section 4.2.6.18) named HardTanhPlus [904] is defined as

$$f(z) = \frac{1}{2}\left(|z+1|_\epsilon - |z-1|_\epsilon\right). \tag{4.518}$$

4.3.54.10 *HardshrinkPlus*

The smoothed variant of the Hardshrink (see Section 4.2.6.22) is named HardshrinkPlus[81] [904]; it is defined as

$$f(z) = z\left(1 + \frac{1}{2}\left(\frac{z-a}{\sqrt{(z-a)^2+\epsilon}} - \frac{z+a}{\sqrt{(z+a)^2+\epsilon}}\right)\right), \tag{4.519}$$

where a is a fixed parameter with a similar function as in the Hardshrink [904].

4.3.54.11 *MeLUPlus*

Pan et al. also provided a smoothed variant of the MeLU (see Section 4.3.31); however, the formula written in [904] is not the MeLU AF but rather its single component $\phi_{b_j c_j}(z_i)$. Nevertheless, the full smoothed MeLUPlus can be obtained easily as the combination of the LReLUPlus and the smoothed $\phi_{b_j c_j}^{\text{Plus}}(z_i)$ defined as

$$\phi_{b_j c_j}\text{Plus}(z_i) = \frac{1}{2}\left(c_j - |z_i - b_j|_\epsilon + \sqrt{\left(c_j - |z_i - b_j|_\epsilon\right)^2 + \epsilon}\right), \tag{4.520}$$

where b_j and c_j are the same parameters as in the MeLU.

4.3.54.12 *TSAFPlus*

The smoothed variant of the TSAF (see Section 4.3.36) named TSAFPlus [904] is defined as

$$f(z_i) = \frac{1}{c_i}\left(|z_i - a_i + c_i|_\epsilon + |z_i - a_i| + |z_i + b_i - c_i| - |z_i - b_i|\right), \tag{4.521}$$

where a_i, b_i, and c_i have a similar role as in the original TSAF [904].

81 Pan et al. named the function HTFPlus originally [904].

4.3.54.13 *ELUPlus*

Even the ELU (see Section 4.2.6.48) can be mollified into a "smoothed" variant named ELUPlus [904]. The smoothed variant is defined as

$$f(z) = \frac{1}{2}\left(z + |z|_\epsilon\right) + \frac{1}{2}\left(\frac{\exp(z) - 1}{a} + \left|\frac{\exp(z) - 1}{a}\right|_\epsilon\right), \tag{4.522}$$

where a is a fixed parameter[82] with a similar function as in the ELU [904].

4.3.54.14 *SwishPlus*

The mollified variant of the swish (see Section 4.3.3.1) named SwishPlus [904] is defined using the smoothed step function instead of the logistic sigmoid; it is, therefore, defined as

$$f(z) = z \cdot \mathrm{StepPlus}(z) = \frac{1}{2}\left(z + \frac{z^2}{|z|_\epsilon}\right). \tag{4.523}$$

4.3.54.15 *MishPlus*

The mollified variant of the swish (see Section 4.2.3.29) named MishPlus [904] is defined using the BipolarPlus and SquarePlus as

$$f(z) = z \cdot \mathrm{BipolarPlus}\left(\mathrm{BipolarPlus}(z)\right). \tag{4.524}$$

4.3.54.16 *LogishPlus*

The mollified variant of the logish (see Section 4.2.3.11) named LogishPlus [904] is defined as

$$f(z) = z \cdot \ln\left(1 + \mathrm{StepPlus}(z)\right). \tag{4.525}$$

4.3.54.17 *SoftsignPlus*

The mollified variant of the softsign (see Section 4.2.2.13) named SoftsignPlus [904] is defined as

$$f(z) = \frac{z}{1 + |z|_\epsilon}. \tag{4.526}$$

4.3.54.18 *SignReLUPlus*

Pan et al. provide a mollified version for an approximation of the SignReLU[83] (see Section 4.2.6.32) in [904]. They approximate the SignReLU as

$$\mathrm{SignReLU}(z) = \frac{1}{2}\left(z + |z|\right) + \frac{z - |z|}{2\,|1 - z|_\epsilon}. \tag{4.527}$$

82 Pan et al. used variant with inverse parameter $\frac{1}{a}$; we have used the same parameter variant as in the original ELU.

83 Pan et al. call it DLU throughout their work [904].

Using the approximation, they then define the SignReLUPlus as

$$\text{SignReLU}(z) = \frac{1}{2}\left(z + |z|_\epsilon\right) + \frac{z - |z|_\epsilon}{2\,|1 - z|_\epsilon}. \tag{4.528}$$

4.3.55 *Complex approaches*

The network in network (NIN) [1072], which uses a micro neural network as an adaptive activation function, represents a different approach. A combination of the NIN and maxout units called maxout-in-network (MIN) was shown to have good performance in [1235]. A similar approach is the wide hidden expansion (WHE) layer [1236], which is a sparselly connected layer with several activation functions that is used in place of a traditional activation function [1236].

Adaptive activation functions called NPF that are learned nonparametrically were proposed in [1237] where a Fourier series basis expansion is used for nonparametric estimation. Only one NPF is learned per filter in CNNs while different activation is learned in each neuron of a fully connected layer [1237]; the learning is in two stages where the network is first learned with ReLUs in the convolution layers and NPF in all others and only then the network is learned with all activation functions being the NPF [1237].

Yet another approach is learning activation functions using hypernetworks [1238] resting in *hyperactivations*. The hyperactivation consists of two parts — a shallow feed-forward neural network called activation network and a hypernetwork, which is a type of neural network that produces weights for another network [1238]. The hypernetwork is used for the normalization of the activation network. A single hyperactivation is learned for each layer in the neural network [1238]. A NN with a combination of more activation functions was used in [1239].

The adaptive activation function might also be trained in a semi–supervised manner [1240–1242].

4.3.55.1 *Variable activation function (VAF)*

Similarly to NIN, the variable activation function (VAF) subnetwork approach uses simple activation functions to produce more complex behavior [1243]; the activation is replaced by a small subnetwork with one hidden layer with k neurons and only one input and one output neuron [1243]. Specifically, VAF is defined as

$$f(z_l) = \sum_{j=1}^{k} a_{l,j}\, g\left(b_{l,j} z_l + c_{l,j}\right) + a_{l,0}, \tag{4.529}$$

where $a_{l,0}$, $a_{l,j}$, $b_{l,j}$, and $c_{l,j}$, $j = 1, \ldots, k$, are trainable parameters for each layer l and $g(x)$ is an activation function such as tanh or ReLU that were used in experiments with VAF in [1243]. The same concept of using subnetwork to learn the activation function was also proposed under the name of *activation function unit* (AFU) in [1244].

4.3.55.2 *Flexible activation bag (FAB)*

The flexible activation bag (FAB) [1245] is an approach similar to NIN and VAF as it uses a subnetwork to learn the AF for each layer l using a pool of K activations $f_k(z_l, \boldsymbol{a}_{l,k})$. It uses a shallow network with double head with ReLU activation in the first layer; then there are two separate heads[1245]. The first head predicts the parameters $\boldsymbol{a}_{l,k}$ of the individual AFs $f_k(z_l, \boldsymbol{a}_{l,k})$ in the bag squashed by a sigmoid AF, then the parameters are mapped into a valid range of each of the parameters [1245]. The second head is a selective head for selecting an appropriate AF by producing a score $s_{l,k}$ — it can be either discrete or continuous resulting in soft or hard selection [1245]. Klopries and Schwung used five selection methods — all of the functions are used ($s_{l,k} = 1$), hard selection, soft selection using logistic sigmoids, softmax selection, and Gumber-Softmax [1246] selection [1245]. The bag of activations used in the FAB consists of a constant function, linear function, exponential function, step function, ReLU, step function, sine function, tanh, logistic sigmoid, and Gaussian function (see [1245] for exact definitions with the adaptive parameters). Then the output of FAB is assembled as

$$f(z_l) = \sum_{k=1}^{K} s_{l,k} f_k(z_l, \boldsymbol{a}_{l,k}), \tag{4.530}$$

where $s_{l,k}$ are the selection scores of f_k; the $\boldsymbol{a}_{l,k}$ consists of parameters of the function f_k (the used AFs have from one to three parameters) and the f_k are the individual AFs from the bag of K functions [1245].

4.3.55.3 *Dynamic parameter ReLU (DY–ReLU)*

The dynamic parameter ReLU (DY–ReLU) (proposed under the name of dynamic ReLU in [1034] but that collides with previously proposed DReLUs in [1032, 1033]) is an activation function whose parameters are input dependent [1034]. The concept of DY–ReLU is similar to the WHE in [1236] and hyperactivations in [1238] as the DY–ReLU is an example of a hyperactivation. The dynamic activation function has two components — hyperfunction that computes parameters for the activation function and the activation function itself [1034]. The DY–ReLU piecewise linear function is computed as the maximum of multiple linear functions [11]. It is defined as

$$f(z_i) = \max_{1 \le k \le K} (a_{i,k} z_i + b_{i,k}), \tag{4.531}$$

where K is a hyperparameter and $a_{i,k}$ and $b_{i,k}$ are coefficients determined by the hyper function $\boldsymbol{\theta}(\boldsymbol{z})$ using all inputs z_i [1034]. The hyperfunction $\boldsymbol{\theta}(\boldsymbol{z})$ is a light-weight neural network [1034]. The parameters generated by the hyperfunction $\boldsymbol{\theta}(\boldsymbol{z})$ can be different for each filter i, or they can be shared in the whole layer [1034]. The DY–ReLU can be considered as a dynamic and efficient variant of Maxout (see Section 4.3.46) [1034].

4.3.55.4 *Random NNs with trainable activation functions*

A very different approach based on adaptive activation functions is presented in [1247] where a neural network with random weights is initialized, and the weights are not trained, but the activation functions are trained instead. The activation functions in [1247] are polynomial activation functions and are trained separately for each hidden neuron with random weights first; only then the weights of the output layer are estimated [1247]. Ertuğrul used five different adaptive variants of activation functions:

$$f(z_i) = \frac{1}{1 + \exp\left(-a_i z_i - b_i\right)}, \tag{4.532}$$

$$f(z_i) = \sin\left(a_i z_i + b_i\right), \tag{4.533}$$

$$f(z_i) = \exp\left(-a_i ||z_i - b_i||\right), \tag{4.534}$$

$$f(z_i) = \begin{cases} 1, & a_i z_i + b_i \leq 0, \\ 0, & \text{otherwise}, \end{cases} \tag{4.535}$$

and

$$f(z_i) = \sqrt{||z_i - a_i||^2 + b_i^2}, \tag{4.536}$$

where a_i and b_i are trainable parameters [1247].

4.3.55.5 *Kernel activation function (KAF)*

A kernel activation function (KAF) [1248] is a non-parametric function that uses kernel expansion together with a dictionary to make the activation flexible [11]. The KAF uses a weighted sum of kernel terms:

$$f(z_i) = \sum_{j=1}^{D} a_{i,j} \kappa\left(z_i, d_j\right), \tag{4.537}$$

where D is a fixed hyperparameter, $a_{i,j}$ are mixing coefficients and d_j, $j = 1, \ldots, D$ are called *dictionary elements* and $\kappa(z_i, d_j) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is 1D kernel function — Scardapane et al. consider only $a_{i,j}$ trainable and the dictionary elements d_j are uniformly spaced around zero [1248]. This has the advantage that the resulting model is linear in its parameters and, therefore, can efficiently optimized [1248].

The kernel function $\kappa(z, d_j)$ used in [1248] is the 1D Gaussian kernel defined as

$$\kappa(z, d_j) = \exp\left(-\gamma\left(z - d_j\right)^2\right), \tag{4.538}$$

where $\gamma \in \mathbb{R}$ is a fixed parameter called the *kernel bandwith* [1248]. Scardapane et al. recommend setting the kernel bandwidth to

$$\gamma = \frac{1}{6\Delta^2}, \tag{4.539}$$

where Δ is the distance betwen the grid points as adapting γ through back-propagation did not yield any gain in accuracy [1248]. The mixing coefficients $a_{i,j}$ can be initialized either randomly from a normal distribution — this provided good diversity for the optimization process [1248] — or using kernel ridge regression to approximate an activation function of choice [1248]. Scardapane et al. also proposed 2D-KAF that works over all possible pairs of incoming values and uses 2D Gaussian kernel [1248].

An extension of the KAF approach was presented in [1249] where activation function used was the sum of the KAF and RSigELU (see Section 4.2.7.15) or KAF and RSigELUD (see Section 4.2.7.19). Kernel methods are becoming more common in deep learning — e.g., fully kernected layers are replacing fully connected layers with a kernel-based approach in [1250].

4.3.56 *SAVE-inspired activation functions*

Brad produced several AF that are, supposedly, motivated by human behavior in [1251]. These AFs were created using the SAVE method [1252] and are mostly variations of the AFs listed above. For completeness' sake, a list of these AFs is included in our work in Table 4.3 also with the real-life motivations listed in [1251] — however, no deeper analysis or objective evaluation of these AFs was not provided in [1251].

4.4 NEURAL NETWORK ARCHITECTURES WITH PARALLEL CONNECTIONS

Neural networks are often not as homogeneous as the D–GEX networks and often include multiple parallel connections or units that are not directly connected [69]. One of the simplest parallel architectures is the so-called multi-column deep neural network (MCDNN) [1253, 1254], which is actually an ensemble of individual "columns" which are separately trained NNs. Other approaches include adding units that are composed of several parallel tracks, which might even differ in the number of layers — e.g., the Inception modules and their variants [52] for image classification. An important architecture with parallel connection is represented by the ResNet family of networks [13] with a residual skip connection, which adds the output of a layer to the output of the layer above. Both described approaches are still being researched and have resulted in networks such as *Inception-ResNet* [54] and *DenseNet* [838].

An approach similar to MCDNN is represented by Parallel Circuits (PCs) [1255, 1256], which partitions a network into several parallel columns of hidden layers that are not connected to each other. The PCs were developed mainly to reach weight reduction for speeding up the computations. Since PCs share an input and an output layer, the weight reduction occurs only when the network has two or more hidden layers [1256]. The networks with PCs were tested on five datasets from the UCI machine learning repository [956]; however, the experiments were done using only a CPU, and thus only small networks were tested — namely with 100 and 1,000 neurons. The sparsity introduced by the PCs acted as a regularizer and helped to reduce overfitting [1256].

formula	parameters	principle from [1251]
$a \sin(bz + c)$	a, b, c	activation of resonance
$a \tanh(bz + c)$	a, b, c	activation of resonance
$z + a$	a	introduction of neutral elements
$a\left(\text{ReLU}(z)\right)^b$	a, b	action against the wolf-pack spirit
$a \exp(bz)$	a, b	activation of centrifugal forces
$a_1\text{ReLU}(f_1(z)) + \ldots + a_n\text{ReLU}(f_n(z))$	$a_1, \ldots, a_n,$ $f_1(z), \ldots, f_n(z)$	application of multi-level connections
$\text{ReLU}(z - a)$	a	application of asymmetry
$a_1\text{ReLU}(z) + \ldots + a_n\text{ReLU}(z)$	$a_1, \ldots, a_n$	harmonization of individual goals with collective goal
az^{1-b}	$a\ , b$	transformation for value-added
$a(1 - \exp(-bz))$	$a\ , b$	transformation for value-added
$\text{ReLU}(f(z)g(z))$	$f(z), g(z)$	application of prisoner paradox
$\text{ReLU}\left(\frac{\exp(z)}{\frac{1}{1+\exp(-az)}}\right)$	a	application of prisoner paradox
$\text{ReLU}\left(\frac{az+b}{cz+d}\right)$	a, b, c, d	application of shipwrecked paradox

Table 4.3: **SAVE-inspired activations**
The list of SAVE-inspired AFs from [1251].

4.5 ARTIFICIAL DATA GENERATION

Part of the presented experiments uses artificial data as those allow for controllable scenarios and are therefore well suited for exploration of the presented transformative adaptive activation function (TAAF). There are many ways to generate artificial data — from the simple addition of random noise to existing samples and thus enlarging the datasets or definition of the sample distribution explicitly to the creation of highly nonlinear and complex data patterns using artificial neural networks. This work will focus only on the artificial data generated using neural networks since this is the method that was used throughout this work.

4.5.1 *Neural networks with random weights*

Most commonly, one encounters neural networks with random weights during initialization before training a neural network. The initial weights need to be set to some value before training, and random initialization is a common choice. This is because initializing all weights to the same value (e.g., zero) can cause problems such as weight symmetry that could never be broken during training, and therefore, it would result in getting stuck in local optima during training [1257]; it can also be shown that random weight initialization often gives a good starting point for optimization [1258]. More thorough overviews of initialization approaches are available in [1257, 1259, 1260].

However, usage of random weights is not limited to the initialization phase; neural networks with some random weights can often work as well (or even better) than the trained networks or be less costly to train [1261, 1262]. Deep learning models can take a long time to train because of their complex structure and large number of parameters. One solution is to use distributed training and powerful accelerators [1263–1265]. Another solution is to train only parts of the models and keep the rest fixed — either pretrained through transfer learning [1266–1268] or with random weights [1261, 1269, 1270]. Neural networks with random aspects are also used for energy-constrained devices, e.g., [1271].

In the cerebellum, which holds 80 % of all neurons in the human brain and plays a crucial role in the learning of precise movements, the granular layer receives and transforms various input signals to serve as the basis to generate responses helping to control the muscles [1272, 1273]. Researchers suggest that the mechanism of the transformation can be modeled using a random recurrent network that can generate necessary signal transformation as long as it operates in a state close to chaotic behavior [1272].

There are approaches where only a very small fraction of the weights is random, as in [1274], where the only randomness comes from a mixing weight a that is sampled from the interval $[0, 1]$ and is used for stochastically mixing two branches of a neural network which works as a regularization approach.

Neural networks with random weights are one possibility for nonlinear data generation as even random weights lead to meaningful patterns or extract good features [1275]. Neural networkss with random weights are sometimes used when training the whole network would be too computationally costly and it has been shown random weights extract good features as only retraining some of the top layers leads to comparative performances to fully trained neural networks [1276–1286]. The usage of random weights leads to a competitive performance and much faster training times compared to networks that are fully trained, and it is sometimes used for processing large-scale datasets [1281]. *Reservoir computing* is another name for similar approaches where a reservoir is generated randomly, and the so-called *readout* is then trained [1287]. There is also some evidence that a similar principle might exist in the brain [1288–1290].

The random vector functional link networks (RVFLNs) are one such case — they are a special class of a single hidden layer feed-forward neural networks that have the input layer connected both to the hidden layer and the output layer [1270, 1279, 1291–1294]. The RVFLNs are one of the first examples found in the literature that researched the usage of random weights in neural networks [1279, 1284]. The RVFLNs also represent one of the first examples of *skip-connections* since the input layer is connected both to the hidden layer and the output layer directly in a similar manner as ResNets [13, 54] — the impact of skip-connections in RVFLNs is researched in [1295] and was found to significant and positive. These networks can also be considered universal approximators under certain conditions [1293, 1294] and also leverage the random weights to avoid the somewhat resource-costly BP and gradient descents methods of feed-forward neural networks [1279, 1296]. While being less common than the feed-forward neural networks (FFNNs), these networks are still being utilized — for example, for thermal environmental prediction in [1296] — for their simplicity. They are being used for semi–supervised learning [1297–1299] or distributed learning [1300] and their applications include, for example, time-series classification [988], direction-of-arrival estimation [1299], conditional probability densities prediction [1301], biomedical classification [1302, 1303], effective solar power prediction [1304], short-term electric load forecasting [1305], COVID-19 spread forecasting [1306], face recognition [1303], handwritten digit recognition [1303], object recognition [1303], text classification [1303], visual tracking [1307], modelling thermal processes [1308], soft sensor modelling for sintering processes [1309], molten iron quality [1310] and word recognition [1311]; more application examples are available in [1280, 1295]. Few extensions of RVFLNs were proposed; for example, parsimonious random vector functional link network (pRVFLN) [1312], robust M-estimation-based RVFLN (M-RVFLN) [1310], RVFLN with ϵ-insensitive Huber loss function (ϵ-HRVFLN) [1302], multi-kernel RVFLN (MK-RVFLN) [1304], kernel RVFLN (K-RVFLN) [1308], MK-RVFLN with evaporation-based water cycle based parameter optimization (EVWCA-MKRVFLN) [1304], wavelet-coupled RVFLN (WCRVFLN) [1306], ensemble incremental learning [1305], sparse pre-trained RVFLN (SP-RVFLN) [1303], convolutional RVFLN (CRVFLN) [1307], and ensemble RVFLN based on negative correlation learning [1309, 1313].

A similar approach was used by Cao et al. in their variant of FFNNs with random weights [1315], which resemble RVFLNs but are missing the direct link from the input layer to the output layer [1314].

Stochastic configuration networks (SCNs) are an approach building up on the RVFLNs [1316] that builds up the network incrementally by adding hidden nodes and allows for fast convergence and good generalization performance. The first SCNs were based on least square optimization that suffered from scalability issues [1317], which resulted in so-called fast SCNs (FSCNs) [1317] that use incremental method for fitting the weights of the output nodes based on matrix decomposition and therefore have better performance when the hidden layer has a significant number of nodes [1317]. SCNs are often used in industrial applications — SCNs were, for example, used for prediction of component concentrations [1318], intrusion signal recognition [1319], soft sensor modeling [1320] of ammonia nitrogen concentration [1321], seawater ammonia nitrogen concentration [1322], sulfur dioxide and hydrogen sulfide concentration [1323], self-blast state detection [1324, 1325], industrial data classification [1326], modeling submergence depth of a pumping well [1327], fault diagnosis of power transformers [1328], and forecasting student learning performance [1329].

Extensions of SCNs include locality preserving SCN (LPSCN) [1320], SCN with rough set based attribute reduction (RS-SCN) [1322], SCN based on genetic algorithms (GA-SCN) [1321], broad SCNs [1330], ensemble SCN [1329], ensemble SCN based on negative correlation learning [1331] (similar to [1313] but with SCNs instead of RVFLNs), SCN with self-attention learning features [1326], SCN with hybrid bat-particle swarm optimization (G-BAPSO-SCN) [1332], Bayesian robust SCN based on a mixture of the Gaussian and Laplace distributions (MoGL-SCN) [1333], orthogonal SCN (OSCN) for filtering low-quality hidden nodes [1334], robust SCN for dealing with outliers or uncertainty [1335–1337], FSCN with an improved sparrow search algorithm (ISSA-FSCN) [1338], SCN with dropout [1319], bidirectional SCN (BSCN) [1339], chaotic sparrow search algorithm based SCN (CSSA-SCN) [1340], deep SCN (DSCN) [1341], AdaBoost based DSCN [1342], adaptive pruning regularization SCN (PRSCN) [1323], stochastic configured Bayesian neural network (SCBNN) [1343], and FPGA implementation of SCNs that was presented in [1344].

The ELMs are similar to RVFLNs; the hidden nodes are fixed and often randomly initialized and not trained [1152]; only the output layer is trained, and the ELM can be therefore thought of as linear models with nonlinear features [1278, 1345]. Even from a theoretical point of view, single-hidden-layer feedforward networks with random weights in the hidden layer are, under certain conditions, also universal approximators the same as networks with trainable weights as shown in [1346, 1347]. ELMs are also applicable in semi-supervised and unsupervised contexts [1348]. ELMs can be used, for example, image classification [1153, 1349, 1350], face recognition [1351–1355], 3D object recognition [1356], activity recognition [1357–1359], human gesture recognition [1360, 1361], stock market forecasting [1362], sales forecasting [1363], robot control [1364, 1365], location estimation [1366], wind speed forecasting [1367], question subjectivity identification [1368], gene expression based classification [628], river suspended sediment load prediction [1369, 1370], fault diagnosis [1371, 1372], and compound classification [692].

The performance of an ELM variant on various benchmark datasets from UCI repository [1373] is available in [1374]. ELM framework can be extended for cross-domain learing through domain adaptation ELMs (DAELMs) [1375] or to a multilayer neural network based one-class classification with ELM (ML-OCELM) [1376]; other extensions include daptive semi-supervised ELM (ASELM) [1368], kernel based ELM (K-ELM) [1377], online sequential ELM (OS-ELM) [1378–1380], online sequential fuzzy ELM (Fuzzy-ELM) [1378], structure-adjustable OS-ELM (SAO-ELM) [1379], dynamic forgetting factor based OS-ELM algorithm (DOS-ELM) [1381] and its multilayer variant (ML-DOS-ELM) [1381], fuzziness-based OS-ELM algorithm (FOS-ELM) [1382], adaptive deep hybrid kernel extreme learning machine (ADHKELM) [1372], hybrid radial basis function (RBF)-ELM NN [1153], incremental ELM (I-ELM) [1383] and convex incremental ELM (CI-ELM) [1383], coiflet wavelet-based optimization method-based ELM (cWOB-ELM) [1369, 1370], multi-layer extreme learning machine (ML-ELM), hierarchical extreme learning machine (H-ELM), densely connected (D-HELM) [1374], evolutionary optimized ELM (ES-ELM) [1384], error minimized extreme learning machine (EM-ELM) [1385], and bayesian extreme learning machine (BELM) [1386]. Wang et al. proposed random convolution nodes (RCNs) in [1387] and used NNs with RCNs for online sequential learning of respiratory motions; the output weights were computed analytically using the ELM approach [1387].

More details about ELMs, their extensions, and applications are available in reviews [1278, 1345, 1377].

The networks with random weights contain meaningful patterns, and it is possible to extract subnetworks that have comparative performance as networks of similar size but specifically trained for the task [1275, 1388]; therefore, the training might consist of selection of suitable subnetwork from a larger, randomly initialized network instead of weight training using, for example, some gradient descent method. In [1275], the authors show that they are able to find a subnetwork of Wide ResNet-50 [55] with randomly initialized weights that has a comparable performance to a smaller network, ResNet-34 [13] while also having slightly lower number of parameters — the work builds on top of [1389] where there are selected suitable subnetworks but those are still trained to reach sufficient accuracy.

The *No-Prop* algorithm for training neural networks is presented in [1283], and it is basically just using non-trainable, random weights of hidden neurons and training the top-layer using the steepest descent method. The authors show that such an approach is simpler and faster, and it often leads to performance identical to complete optimization using the back-propagation algorithm with gradient descent.

Not only does the last layer of a network or all of the layers have to be trainable, but it can be shown that many other variants work well — for example, also tunning a layer in the middle together with the last layer leads to better performance than just tunning the last layer [1277]. Furthermore, if one had selected only one block/layer for training, sometimes it is better to train other parts than the last layer as well as shown in [1277] where authors compared training reported a network's performance with respect to training only certain blocks (DenseNet [838] and WRNs [55] were used). Another possibility is to set the top layer to a fixed but not-random pattern as shown in [1390] where the top layer was kept fixed, but other layers were trained without significant impact on a network's performance.

Since recurrent neural networks (RNNs) are harder to train than classical FFNNs (e.g., due to the vanishing/exploding gradient [1391]), there are approaches that introduce some randomness also to this particular class of networks [1285, 1287]. The aforementioned reservoir computing (RC) [1290, 1392, 1393] is one such approach; a recurrent neural network called *reservoir* is randomly weighted and remains unchanged during training. Then the desired output signal is trained using the signals from the reservoir — the simplest approach, echo state networks (ESNs) [1394–1398], models the target signal as a linear combination of the signals from the reservoir using linear regression with least-square objective [1287] while more complex approaches might use quadratic programming to maximize margin in an SVM-like manner [1399]. Another example of RC is a *liquid state machine* [1400–1402], which is an independently discovered variant of ESNs first proposed in [1403]. RC is also used to model actual brain [1396, 1404], e.g. reservoir computing was used to model the granular layer in cerebellum in [1272, 1273, 1405] — the granular layer is the reservoir and long-term depression of the parallel fiber–Purkinje cell connections is the learning rule [1273]. The research of RC and ESNs themselves is often biologically motivated (e.g., [1406, 1407]) as it can offer insights into the brain [1396, 1404]. RC is also used to model short-term memory (STM) [1397, 1408–1414] that is hypothesised to be due to transient network activity [1408]. Nevertheless, it was found that NNs optimized for memory tasks might differ significantly from NNs optimized for prediction tasks [1415].

The effect of a deeply layered organization of RC models, an efficient RNN architecture, was investigated in terms of both occurrence of multiple time-scales and the increasing richness of the dynamics in [1416, 1417] and lead to onset of deep echo state network (DeepESN) [1417–1420]. The deep layering of recurrent models allows diversification of temporal representations in the layers of the hierarchy, leading to an increase in short-term memory capacity [1417] even though such layer stacking is just an architectural constraint — it is equivalent to fully connected architecture, where some connections were removed [1417]. The Deep ESNs have also less computational complexity and better predictive performance than shallow, single-layer ESNs [1421]. The inter-layer connectivity in DeepESNs plays a significant role in an ESN performance [1422]. The dynamical behavior of ESNs model is studied in [1423–1426] for shallow ESNs and in [1427] for DeepESNs; the hyperparameter optimization for ESNs models is studied in [1428]. A common usage of RC, ESNs and DeepESNs is regression, timeseries prediction and sequence processing [1429], e.g., [707, 1430–1452], but also classification [1453, 1454], e.g. emotion recognition [1455], word recognition [1402], room classification based on power consumption [1456], time series classification [1457–1461], graph classification [1462], and reconstruction of missing data [1463]. Another example is system modeling [1464] and identification [1465]. The more general RC models can be used, for example, for noisy image recognition [1466], reconstruction of unmeasured dynamical system variables [1467], emulation of chaotic systems with cryptography applications [1468], phoneme recognition [1469] and speech recognition [1470–1472], detection of epileptic seizures [1473, 1474], continuous digit recognition in audio [1475], robot control [1476], image classification [1471, 1477–1479], human action recognition in videos [1480], attack detection in smart grids [1481],

Few extensions of the ESN approach are the polynomial ESNs (PESNs) and their simplified variant (S-PESNs) [1431], variable memory length ESNs (VML-ESNs) [1482], double-reservoir ESNs (DRESNs) [1432], ESNs optimized using mutual information (MI-ESNs) [1483], deep belief echo state networks (DBENs) [1434], multiple reservoirs echo state networks (MR-ESNs) [1436], ESNs using differential evolution (ESN-DEs) [1437], particle swarm optimized ESNs (O-ESNs) [1484], probabilistic ESNs (π-ESNs) for density estimation [1455], leaky-integrator ESNs [1395, 1485], hybrid circle reservoir ESNs (HCR-ESNs) [1486], robust ESNs with correntropy induced loss function [1487], sinusoidal ESNs (SESNs) [1488] for periodic source signals, fast subspace decomposition echo state networks (FSDESNs) [1440], robust echo state networks (RESNs) based on Bayesian framework [1441], functional ESNs (FESNs) for time series classification [1457], time warp invariant echo state networks (TWIESNs) [1489], support vector echo-state vector machines (SVESMs) [1443], echo state graph neural networkss (ESGNNs) [1462, 1490]. There are also quantum reservoir computing extensions, e.g., [1491–1493].

Interestingly, RC can be efficiently implemented with a specialized hardware based on optical and optoelectronical systems, which can accomplish fast information processing with low energy consumption [1414], more examples and details are available in [1281, 1445, 1447, 1452, 1470, 1472, 1477, 1480, 1494–1520]. There are also integrated circuits and FPGA implementations (or simulations of such approaches) of RC models, e.g., [1407, 1446, 1478, 1518, 1521–1526]; and atomic switch network implementations [1527]; another approach to hardware implementation of RC is based on memristors [1528, 1529], which were shown to have good properties for a reservoir [1530], e.g., [1462, 1530–1537]. There are physical reservoirs [1538, 1539], for example, electrolytes and ion-based liquids [1479, 1540–1544], other physical reservoirs include nanomaterials [1281, 1545–1547], superconductors [1491, 1548–1550], semiconductor-based memristors [1535–1537, 1551, 1552] Brief review of hardware and physical implementations of RC is available in [1538] and in [1553] where an intelligent matter is discussed as one of the goals of neuromorphic computing.

The RC is well established from a theoretical point-of-view and works on two main principles — the reservoir does a conversion of spatiotemporal information into a spatial representation only, and it can be considered as a short-term memory as the inputs' influence will fade from the reservoir state over time [1290, 1554, 1555]. It was also shown that there are classes of RCs that are universal approximators [1556, 1557]. The RC approach can also be formulated as nonlinear vector autoregression [1558] without the need for an explicit reservoir with random weights resulting in next generation reservoir computing (NGRC) [1559]. More details about RC is available in reviews [1285, 1290, 1560–1562].

A similar approach is also used for graph neural networkss [1563], which utilizes random filters and adjusts the learning objective with regularized least square loss in order to speed up the training and facilitate the training even for very large graphs [1563]. Such an approach is called graph convolutional networks with random weights (GCN-RW) [1563] and was first proposed for node classification.

Another approach is the random weight NNs with trained activation function [1247], where the weights are kept random but the activation functions in the hidden layers are trained using linear regression from a selected pool of parameterized activation functions [1247].

Yet another application of random weights lies in ensemble learning. The Kowsari et al. introduces a new method, called random multimodel deep learning (RMDL), for selecting the best deep learning approach to solve classification problems [1269]. RMDL combines different deep learning techniques, such as deep neural networks (DNNs), RNNs, and CNNs, using parallel learning architecture to produce multiple random classification models. The method is evaluated on various datasets, including both text and image classification, and the results show that RMDL consistently outperforms conventional approaches like naïve Bayes, SVM, or a single deep learning model.

Neural networks with random weights are also helpful from a theoretical point of view as it is quite hard to create a theoretical framework explaining why neural networks work so well in practice [1262]. Neural networks with random weights are set to the random matrix framework in [1564] and also described in the context of other methods such as ridge regression.

4.5.2 *Synthetic data generation*

Neural networks can be used for generation of synthetic data as was already briefly discussed in Section 4.1.2.2 where a NNs were used for generation or augmentation of gene expression data. There are various reasons for generating data — from the need for data with known theoretical properties for demonstration learning algorithms (e.g., [1565]), enlarging smaller datasets through data augmentation [1566] to facilitate better learning and generalization to a generation of entirely fake data [1567, 1568] for aesthetic or other purposes [1569]. The line between data generation and other tasks such as translation between domains (e.g., image-to-image [1570–1572] or text-to-image [1573–1575] translation) is somewhat blurry since the translation task can be thought of as conditional data generation on some input. Therefore, there is no clear distinction between conditional data generation and transforming or mapping tasks, as the same tasks can be solved by models that are not primarily generative in the sense of allowing multiple different samples generated conditionally on a single input and by models that allow for unlimited sampling from the domain. A few examples of the former class include style transfer using feed-forward CNNs [1576], style transfer using perceptual loss [1577], style transfer using generative adversarial network (GAN) [1578, 1579],

The most important reasons for synthetic data generation/augmentation are:

controllable complexity and properties
Most common examples of synthetic data are usually textbook examples where the data are generated such that they exhibit simple, controllable properties that are required for a demonstration of a principle and usually nothing more, e.g., [1580]. Such data can be either simple points on a plane or quite complex datasets with texts and images.

scarcity
Synthetic data are often used because real data are scarce (e.g., due to costly manual annotation) and cannot be obtained in sufficient amount necessary for solving a problem in a sufficient quality [1581–1584]. The data can be either fully synthetic (e.g., [1583, 1585, 1586]) or augmented (e.g., [1584]) — the data can include real samples or be modifications of real samples [1584, 1587]. Data augmentation is often used to limit overfitting by producing more samples that are similar to the existing data [1587] or when the class distribution is imbalanced, and the samples of minority class synthesized instead of oversampled (e.g., [1588, 1589]).

privacy
The data are often hard to obtain due to privacy considerations [1581, 1590] — the data exist but cannot be released. This can be solved by fully synthetic datasets with similar properties or data anonymization techniques that often include synthetic data (the simplest example is the replacement of a person's name by a made-up name). Data privacy is important because even a learned model can leak private information even if the original dataset is very closely controlled. The leaks through the trained model might include *data extractions*, *model inversions* [1591, 1592] which falls under *attribute inference* [1593] (e.g., [1594–1597]), *membership inference attacks* [1593, 1594, 1596, 1598–1604] and are closely connected to concepts of *differential privacy* [1605–1609] and *differentially private learning* [1610–1619]. However, even synthetic data generation might lead to privacy leaks through an attack for disclosing whether the data from certain target individuals were used in the data generation [1620–1626]; therefore the protocol used for synthetic data generation has to be carefully designed if the process uses real data that are very confidential.

There are many domains that utilize synthetic or augmented data for machine learning purposes.

There are a lot of approaches that use neural network for synthetic data generation. Restricted Boltzmann machines (RBMs) [1627–1631] and deep belief networks (DBNs) [1632–1635] are one of the earliest methods [1636] and are followed by deep Boltzmann machines (DBMs) [1637] that can be seen as their extension. DBNs can also be used for extracting low-level features and dimensionality reduction [1638, 1639]. There are few extensions of DBNs — e.g., convolutional deep belief networks (CDBNs) [1635, 1640], mode isolation s (MI-CDBNs) [1641]. RBMs are a member of more general class [1642] — energy-based models (EBMs) (e.g., [1643, 1644]). An example of extension of RBMs are energy-based dropout [1645], stream-based RBMs [1646], FE-RBMs for classification [1647], Gaussian RBMs with binary auxiliary units (GARBMs) [1648], and parallel ensemble of RBMs [1649].

The autoregressive models [1650–1656] explicitly estimate distribution using modified autoencoder (AE) or recurrent architecture. An AE has two parts — an encoder and a decoder. The encoder maps the input to a latent variable z, and then the decoder maps the latent variable z back to the original space [1657, 1658].

Variational autoencoders (VAEs) [291, 1659, 1660] extend the idea of generative autoregressive or autoencoder models even further. While the AEs are learned to compress the input to a latent space, which then can be used for sampling new samples, the latent variable is not regularized, and therefore, it is hard to sample meaningful samples as there might not be parts of the latent space that do not correspond to any data point. It is possible to use the training data to estimate the distribution of samples in the latent space to facilitate sampling of meaningful points, but this can be elegantly solved by VAEs — the VAEs add constraint to the latent representation to regularize the latent space. The VAEs impose a constraint that the latent distribution of the inputs must follow a known, usually normal, distribution. Thanks to this constraint, the VAEs are able to learn smooth latent representations of the input data [1657, 1661]. Since the latent space follows such a distribution, the sampling of new data points is easy. However, even VAEs have a shortcoming — a sample generated by a VAE cannot often be consistently encoded [1662, 1663]. This issue was addressed by autoencoder VAEs (AVAEs) in which the encoder part of a VAE is trained using a notion of self-consistency leading to robust representations [1662]. There are also other VAE extensions, e.g. S3VAE [1664], C-DSVAE [1665], VQ-VAE [293, 1666, 1667], S-VAE [1565], VaDE [1668], NVISA [1669], InfoVAE [1670], β-VAE [1671], DRAW [1672], NVAE [1673], purifying VAE (PuVAE) [1674], VT-STOWER [1675], and CE-VAE [1580]. The VAEs can also be used to generate out-of-distribution data that are not present in the original training data using style transfer [491].

VAEs were used, for example, for image generation [1666, 1667, 1670–1673, 1676], video generation [1664, 1666], audio generation [1666], gesture generation from audio [1677], image clustering [1668, 1669], image feature extraction [1678], text clustering [1668], molecule clustering [1679], text style transfer [1675], analysis of biological data [1680], and generating gene expression samples [490, 491].

One of the well-known neural network techniques for synthetic data generation is a generative adversarial network (GAN) [1681]. GANs represent an approach where a generative model is trained together with the discriminative model using real data in a competitive manner — the generative model produces a synthetic sample, and the discriminative model then decides whether the sample is real or generated [1681]. More thorough reviews of GANs, their extensions and applications are available in [451, 1682–1686].

GANs extensions include VAEGAN (combining VAEs and GANs) [1676], Zero-VAE-GAN [1687], F-VAEGAN-D2 [1688], f-CLSWGAN [1689], CycleGAN [1578, 1690], bicycle GAN [1691], edge adversarial GAN (EGAN) [1692], StackGAN [1693] and its extension StackGAN++ [1694], GAWWN [1695], SinGAN [1696, 1697], S^2–GAN [1698], RDCGAN [1699], WGAN-GP [1589], auxiliary classifier GAN (AC-GAN) [1700], transformer based GAN (TransGAN) [1701], self-attention GAN (SAGAN) [1702], coupled GANs [1703], selective transfer GAN (STGAN) [1579], gradient-guided dual-branch GAN (GCD-GAN) [1704], GAN with residual inception modules (RI-GAN) [1705], LeicaGAN [1574] utilising prior knowledge, attentional GAN [1706], GAN with neural architecture search (AutoGAN)[1707], panoptic layout GAN (PL-GAN) [1708], deep fusion GAN (DF-GAN) [1709–1711], CRPGAN [1712], cross-modal attention gusion based GAN (CMAFGAN) [1713], graph GAN (GGAN) [1714], fused GAN with attention mechanism (AM-GAN) [1715], dual Generator attentional GAN (DGattGAN) [1716], ML-CGAN [1717], contrastive meta-learning GAN (CML-GAN) [1718], dual discriminator GAN (D2GAN) [1719], dual discriminator weighted mixture GAN (D2WMGAN) [1720], robust conditional GAN (RoCGAN) [1721], variance enforcing GAN (VARGAN) [1722], dual-stream GAN with phase awareness (DPGAN) [1723], geometry-aware GAN (GAGAN) [1724], GAN with residual partial modules (RePGAN) [1725], squeeze-excitation network-deep convolution GAN (SE-DCGAN) [1726], example attention GAN (EA-GAN) [1727], hyperbolic GAN (HGAN) [1728], partition-guided GAN [1729], and cascading residual–residual attention GAN (CRRAGAN) [1730].

The common tasks for GANs are image generation [1569, 1636, 1676, 1684, 1697, 1699–1701, 1717–1720, 1724, 1731–1748], image generation from text [1571, 1573, 1574, 1693–1695, 1706, 1709–1711, 1713, 1716, 1737, 1749–1765], semantic image synthesis [1766], text-guided image editing [1758, 1767–1769], image generation from embeddings [1688], image feature generation [1689], single image animation [1696], novel view synthesis [1770], audio enhancement [1723], audio-to-video [1771], image translation and editing [1578, 1691, 1712, 1726, 1768, 1772–1780] such as paint-to-image [1696], day-to-night [1571], sketch-to-image [1571, 1737, 1765, 1781], RGB to hyperspectral image [1782], image inpainting [1725, 1783–1789], restoration of ancient artworks, murals and texts [1727, 1790–1796], medical image translation and enhancement (fresh frozen samples to formalin-fixed, paraffin-embedding processed samples [1797], mapping one contrast to another in magnetic resonance imaging (MRI) [1798, 1799], contrast computed tomography (CT) images to non-contrast [1800], MRI-to-CT [1801], CT-to-MRI [1802], retinal images from vessel trees [1780, 1803], tumor segmentation [1804–1808]) [1809–1813], sketch extraction [1691, 1704], unsupervised image translation [1814] (more details in review [1815]), image editing [1579, 1696, 1816], super-resolution [1730, 1791, 1792, 1811, 1813, 1817–1835], generation high-quality images with high dynamic range [1836], image segmentation [1715], deblurring [1690, 1692, 1837–1842] and image haze removal [1705, 1782], and video editing [1778]; however, GANs can be also used password cracking [1843] (extended version

[1844]), gene expression inference [14], improving fault diagnosis [1845, 1846], defense against adversarial attacks [1847], learning data priors of 3D LiDAR data [1848], 3D model generation and manipulation [1849–1854], Alzheimer's disease staging using structural MRI (sMRI) [1855], generation of gene expression data [454], data imputation [1856, 1857], solving jigsaw puzzles [1858], texture generation [1714], image watermarking [1859], anomaly detection in medical images [1860–1863], medical image fusion [1864], anomaly detection in medical images [1865], object tracking [1866], and graph generation [1867]. It is also possible to translate networks learned for one task into networks solving other tasks — e.g., an approach that uses unconditional GANs or VAEs and uses them in conditional settings in [1868].

GANs are sometimes used for augmentation of the data and enlarging the training dataset for other methods [1588, 1589, 1846, 1869–1874]. For example, a GAN was used to augment simulated data that were used to supplement the real measured data in [1869], to enlarge small datasets in [1875], to generate synthetic samples for a minority class [1588, 1589, 1846, 1873, 1876–1878], mammographic images [1863, 1879], and brain tumor images [459, 1880, 1881]. The data augmentation and image generation are especially useful in medical fields [1874] where large datasets are often either costly to obtain or the observed characteristic is rare; furthermore, the generated data are often indistinguishable from data from real patients - e.g., generated SPECTs of cerebral ischemia were shown to be very faithful [1882]. Another example of dataset generation/augmentation in case there are only smaller datasets was presented in [1748], where authors used GANs to generate more samples to create an insulator image dataset.

There are other generative models, such as Plug and Play generative networkss (PPGNs) [1883] that describe a more general framework where there is a single generative network and a replaceable network that conditions the generative network what to draw [1883].

A slightly different generative approach is represented by diffusion models (DMs) [1884, 1885], also called denoising diffusion probabilistic models (DDPMs) [1575], that recently started to achieve state-of-the-art performance on several tasks [1886]. Diffusion models work by gradual denoising process [1886, 1887] — they first progressively destroy data by adding more and more noise and then learn to reverse this process [1887]. Sampling of a new sample is done by progressively denoising pure noise. Similarly to other generative models, the DMs can be used for image synthesis [1569, 1886, 1888–1909], video synthesis [1910–1923], text-to-image generation [1763, 1769, 1894, 1898, 1907, 1908, 1923–1953], text-to-video generation [1912, 1919, 1922, 1954–1956], motion synthesis [1957], text-to-motion [1918, 1958–1962], image-to-video [1963, 1964], image translation and editing [1892, 1894, 1901, 1908, 1911, 1926, 1928, 1943, 1944, 1949, 1951, 1965–1978], medical image translation and enhancement (MRI-to-CT [1979], accelerated MRI [1980, 1981], vessel segmentation [1982], brain tumor inpainting [1983]) [1874, 1965, 1984–1990], video translation and editing [1956, 1991], JPEG artifact correction [1992], semantic segmentation [1893, 1985, 1986, 1993–2000], text generation [1890, 2001], image-to-text [1905, 1907, 1908, 2002–2004], 3D image generation [1854, 1929, 2005–2013], text-to-3D [2007, 2008, 2014–2016], point cloud generation and reconstruction [1904, 2017–2019], novel-view synthesis [2011, 2020, 2021], 3D reconstruction [2011, 2021], scene synthesis [2022], vectorized sketch generation [2023], text and language generation [1907, 1942, 2024–2026], sentence completion [2024], super-resolution [1970, 1971, 1987, 1988, 2027–2029], inpainting [1967, 1970, 1971, 1988, 2030], sound-guided video editing [2031], sound-guided gesture synthesis [2032], audio synthesis [1910, 2033, 2034], text-to-speech [1942, 2035–2042] and text-to-audio [2043], audio enhancement [2035, 2044, 2045], time-series forecasting [2046–2049], time-series generation and imputation [2048, 2050, 2051], anomaly and out-of-distribution detection [2052, 2053], anomaly detection in medical images [2054, 2055], and change detection [2056] but also travelling salesman problem (TSP) [1893], combinatorial optimization [2057], height estimation [2058], Rickrolling [1930], image watermarking [2059], molecule and protein generation [2060–2075], material design [2076, 2077], and defense against adversarial attacks (adversarial purification) [2078–2087].

The DM principle can also be used together with other generative models [1887] such as combinations with autoregressive models [2049, 2088, 2089], VAEs [2090–2092], GANs [1965, 1982, 2000, 2093, 2094], NF models [2095–2099] and with energy-based models (EBMs) [2024, 2100]. There are also variants that produce images directly from the noise instead of gradual denoising, e.g., [1891]. More details about DMs and their usages are available in reviews [1769, 1887, 1923, 1984, 2025, 2035, 2046, 2060–2062, 2101–2103].

Generative models and generative artificial intelligence, in general, are a broad topic that was only superficially touched on in this work; for more details, see, for example, reviews [1682, 1760, 1874, 1887, 1942, 2104, 2105].

4.5.3 *Neural networks with random weights for data generation*

While the literature on neural networks with random weights and neural network for data synthesis is numerous (see Section 4.5.1 and Section 4.5.2 respectively), the usage of neural network with random weights for data synthesis is much rarer, this is partly due to the popularity of supervised data synthesis in recent years where the generators are trained to resemble particular domains and partly due to the lower need for data with a pattern generated at random. Some of the research focused on generative networks that have only partially random weights or are using other approaches that utilize some sort of randomness in the structure — as opposed to common generative networks for data generation where the only randomness comes from the random input to the network and from the random initialization of weights before the training process.

Two neural networks with random weights were used in [1143] to create simulated data for assessing the performance of proposed adaptive active functions. Farhadi, Nia, and Lodi used shallow networks with ten neurons in either one or eight fully connected hidden layers. The used activation functions in the data generation networks were ReLU and logistic sigmoid. Farhadi, Nia, and Lodi generated 10,000 samples with ten features and a binary target for each experiment with simulated data.

Another example of data generation networks with some random weights is represented by RCs (see Section 4.5.1 for more details) [2106, 2107]. It was first shown in a tutorial [2106] where an ESN was trained to be a sinewave generator; the network had a reservoir of 20 neurons whose internal weights were set to random values and were not changed during the training and a single readout unit that was connected with trainable weights to the reservoir. The trainable weights were fitted using linear regression instead of any gradient method as the readout is a linear combination of reservoir outputs and the sample size was small [2106]. There was also a variant with 100 neurons that allowed for tunable frequency of the sinewave generator.

A different example of data generation using RC is the limit cycle generation in [2107], where RC was used to simulate robot control using a recurrent network of spring and masses. First, the authors demonstrated the RC approach by generating three limit cycles — two defined by simple differential equations and the third being a Lissajous curve — and successfully generating them using a reservoir that simulated recurrent network of spring and masses with two linear readout units, one for each coordinate. Since the reservoir is not trained and fixed with random weights, the authors showed that a single reservoir could be used for all three tasks, which further supports their premise that even a robot's body could be a suitable physical reservoir for morphological computation [2107]. The generated limit cycles using RC were accurate and stable; to further show the stability of such systems, the author experimented with adding perturbations to the inputs. The first experiment replaced a single output with a constant that was fed to the system for 10s̃ instead of the actual value; it was shown that after the correct value was fed to the system once again, the system was able to return back to the desired trajectory after the disturbance disappeared [2107]. The second experiment introduced quite a strong constant horizontal force during the same time window as in the first experiment; this force was applied to all the nodes of the network and led the trajectories very far from the desired trajectory while being applied. Nevertheless, the system was also able to recover to the desired trajectory after the application of the force was stopped. The third experiment added perturbances in the form of white noise (more details in [2107]), and while the trajectory was quite off while the noise was applied, the trajectory fairly quickly returned back to the desired trajectory once the noise was removed. Therefore, it was shown that stable outputs can be produced even with randomly weighted reservoirs whose weights are not fitted during the training procedure [2107]. Similarly as in the first experiment with three limit cycles, the authors showed in another experiment that a single reservoir with fixed weights can be used for more tasks where different walking patterns (taken from [2108]) were generated only by refitting the readout unit [2107].

Another example of data generation with random weights is style transfer and natural texture synthesis in [1262], where authors present three popular inversion tasks for visualization. The inversions are applied on an untrained VGG [51] with random weights, and the authors show that they were able to reconstruct images with high perceptual quality and that the results were even better than using pre-trained VGG with the same architecture [1262]. The VGG with random weights was also used to synthesize natural textures.

While Gatys, Ecker, and Bethge failed at natural texture synthesis [2109], He, Wang, and Hopcroft hypothesized that it might have been due to their inappropriate scale of the weighting factors [1262]. He, Wang, and Hopcroft were able to synthesize natural textures of similar quality as had a trained VGG network in [2109] with VGG with random weights with automatic normalization to scale the squared correlation loss for different activation layers [1262].

The first experiment present in [1262] is an inversion of deep representation where He, Wang, and Hopcroft selected a few source images from the ILSVRC 2012 challenge [48] to be the examples for the inversion task; a monkey image was selected as the reference image. The VGG with stacked random weights (ranVGG) had 19 layers of random weights — 16 convolutional layers and three dense, fully connected layers — and five pooling layers with average pooling. He, Wang, and Hopcroft compares the performance of ranVGG, VGG with purely random weights, trained VGG, and the work [2110] of Mahendran and Vedaldi based on the AlexNet [49]. Both ranVGG and VGG with purely random weights showed lower reconstruction distances with lower variations than the trained VGG; furthermore, ranVGG had lower variations and lower reconstruction distances than the VGG with purely random weights and demonstrated a more stable and high performance [1262]. He, Wang, and Hopcroft also compared the perceptive quality of the reconstruction and noted that the ranVGG shows higher perceptive quality than the AlexNet from [2110].

The second experiment of He, Wang, and Hopcroft is texture synthesis where textures generated by the inversion task using an increasing number of convolutional layers are compared to the original image and the results obtained using pre-trained VGG from [2109]. It is shown that increasing the number of used convolutional layers improves the reconstruction up to the point where it is very similar to the pre-trained model; nevertheless, the pre-trained VGG outperforms the ranVGG when the original texture is neatly arranged [1262].

The last experiment present in [1262] is artistic style transfer; He, Wang, and Hopcroft selected one convolutional layer as the content layer and used the combination of a few other convolutional layers as the style. The results obtained using the ranVGG on several famous artworks were comparable to the work of Gatys, Ecker, and Bethge who used trained VGG [2111] — albeit the trained VGG resulted in slightly smoother lines and textures [1262]. The authors further complement the experiments by demonstrating an artistic style transfer from Chinese paintings to selected photographs.

While the work [1262] does not use the network with random weights in a feed-forward fashion for data generation but rather through inversion tasks, it still shows that random weights may be useful for several reasons. First, it is hard to develop theoretical foundations of deep learning with trained weights, but it might be easier with random weights (as is, for example, done in [1564]). Second, training deep neural networks is a very resource-intensive process; the ability to investigate network architectures without actually training them may speed up and smoothen the process of finding a suitable architecture [1262].

5 METHODS

The main improvements — transformative adaptive activation functions and checkerboard architectures — to the original D–GEX are described in this chapter. First, however, preliminaries such as the description of data (Section 5.1.1), data normalization (Section 5.1.1.2), and performance evaluation (Section 5.1.4) are described in Section 5.1. After such preliminaries, the main improvements are described. First, the transformative adaptive activation function is described in Section 5.2, and then the architectural improvements to the original D–GEX in the form of tower and checkerboard architectures are described in Section 5.3. Finally, technical and implementation details of the TAAFs are described in Section 5.4.

5.1 PRELIMINARIES

There are several common properties of most of the experiments that were run, and these are described in this section. In order to examine whether our novel transformative adaptive activation function in D–GEX model could lead to lower error, we have used the very same data as in [2]. Therefore, the data and their origin are briefly discussed in Section 5.1.1; the heterogeneity–aware dataset that is aiming at reducing bias due to the uninformed random data split present in the original paper is described in Section 5.1.1.1. While a similar experiment setup as in the original D–GEX paper was used, a different data normalization approach was used to reduce the influence of genes whose expression is near the noise levels; this normalization is described in more detail in Section 5.1.1.2.

The experiments on the same dataset as the original D–GEX are supplemented with experiments using artificially generated data that are similar to the microarray gene expression data that were used in the original D–GEX as the original data did not have perfectly independent samples and the sample independence is easy to ensure with artificially generated data. The process of the data generation is described more in-depth in Section 5.1.2. First, an overall methodology is described in Section 5.1.2.1

Furthermore, we have re-implemented the D–GEX and retrained it on the same data as the models with the novel TAAFs to ensure that the performance comparison (see Section 5.1.4) indeed reflects only the influence of the usage of the novel transformative adaptive activation functions and nothing else.

5.1.1 *Data*

We have used mainly gene expression data from the Affymetrix microarray platform curated by the Broad Institute. It was provided by the authors of the original D–GEX [2] and contains 129,158 profiles, each consisting of 22,268 probes. The data are also available at `https://cbcl.ics.uci.edu/public_data/D-GEX/`. We have replicated the data pre–processing process presented in [2] — we have removed the biological and technical replicates and have used the same set of target and landmark genes. We have used 942 landmark genes to reconstruct the expression of 9,518 genes. This data was split into two datasets: the first dataset called *full dataset* and the second *heterogeneity–aware dataset*. The full dataset contains all data after preprocessing (126,102 samples) and was split into a training, validation, and testing set (the training set has 88,256 samples, while the validation set has 18,895 samples, and the testing set has 18,951 samples). The validation dataset was used for model selection and parameter optimization, while the testing set was used for reporting the performance of selected models based on out-of-sample data. The Heterogeneity–aware dataset contains a subset of the full dataset and was used for testing to determine whether the performance of the models on the full dataset might have been due to possible information leakage between training and testing splits.

5.1.1.1 *Heterogeneity–aware dataset*

As in the original D–GEX paper [2], the data for most experiments were split into training and evaluation sets randomly; however, the data used contains different sets of samples that originated in the same experiment; thus such a split might have introduced bias to the reported results. To show that such bias, if present, is insignificant for our comparison, we have also run an experiment comparing our D–GEX reimplementation with D–GEX with TAAFs on a dataset, where the split was *GEO-* series aware (heterogeneity–aware dataset). We grouped the available samples from the full dataset by their *GEO-* series if such a grouping was obtainable from the sample ID. Then, we performed the split such that no group would have a sample in more than one split, which removed the potential information leakage between the splits. This resulted in a subset of the normalized data used consisting of 87,345 samples (the series information was not obtainable from the sample ID for some samples) split into training (52,407 samples), testing (17,469 samples), and validation (17,469 samples) sets with no series overlaps. Since the lower amount of samples available for training might negatively influence the training and the resulting model performance and since it resembles the approach of [2], most of the experiments were done using the full dataset and the heterogeneity–aware dataset was used only to verify that the model performance is not due to the bias caused by information leakage between the sets.

5.1.1.2 *Data normalization*

The data were preprocessed in the same manner as in [2] except for the last step — the scaling to a zero mean and unit standard deviation. Scaling each variable separately as in [2], however, removes the absolute differences in expression between individual genes. Moreover, it gives the same importance to all genes, including those whose expression is near noise levels, from the point of view of the error metrics. To keep the information about differences in expression levels, we scaled the data by transforming the whole data matrix to have zero mean and unit standard deviation without taking into account that there are different genes — thus, the more expressed genes will be proportionately more expressed even after the scaling. We believe that such scaling is more suitable in this case as the minimization of the error metrics during the fitting phase gives relatively higher importance to more expressed genes and less to the genes whose expression is near the noise level.

5.1.2 *Experiments with artificial data*

To further show the capabilities of TAAFs, we have run several experiments with artificial data as those are fully controllable, similarly as Farhadi, Nia, and Lodi generated artificial data for evaluation of their proposed activation function in [1143]. In general, a generative neural network was used to create the artificial dataset. These generative networks were randomly initialized, and their main purpose was to create a non-linear relationship between the input features in a fashion distantly similar to biological data.

5.1.2.1 *Methodology*

The experiments focus on a regression task from many features to many targets (e.g., 1,000 input features to predict 5,000 related targets for a single sample) — unlike the work of Farhadi, Nia, and Lodi where samples with ten features and a single binary target were generated [1143] as this would be insufficient in our case. The artificial data simulates data similar to gene expression data, which is what our work is mainly focused on. To achieve this, a densely connected neural network with L layers and N_l, $l = 1, \dots, L$ neurons was initialized with random weights. This network was then used to process randomly generated data to produce a dataset with non-linear relationships between the features and targets — samples are sampled from a given distribution with dimensionality equal to the number of inputs of the data generation network and then processed by the data generation network to get the samples with non-linear relationships between individual components similarly gene expression data have. We have also added some white noise to the targets, as data are rarely noiseless in practice. The exact layer configurations and parameters of the data generation process of the individual experiments are described in Section 6.3 — the common shared properties of the experiments are that the input dimension was 1,000 (similar dimension to the L1000 Luminex bead microarrays that are the targets of the D–GEX inference [2]), the data that were processed by the data generation networks were sampled from a normal distributions with zero mean (different standard deviations were used in different experiments), the white noise added to the samples after they were processed by the data generation network was also sampled from a normal distribution with zero mean, the output dimension was 5,000 (similar to the size of the D–GEX networks)

In most experiments, three data splits were used — *train* split for training the networks, *validation* for selection of a model checkpoint, and *test* for evaluation independent of the checkpoint selection process. The networks were usually trained for a fixed number of epochs, and the model was evaluated on the *validation* set after each epoch; the weights from the epoch with the lowest loss on the *validation* set and the last epoch were kept and are called as model checkpoints *loss* and *last* respectively.

5.1.3 *Baseline architectures and training procedure*

D–GEX, as proposed in [2], is a feedforward neural network consisting of one to three fully connected hidden layers, each having the same number of neurons. The output layer consists of one neuron per target with a linear activation function. Since we directly build upon the D–GEX architecture, we have used the same architecture as the baseline — only with varying number of layers and number of neurons in each layer. As in the original D–GEX, we have split the set $\mathcal{G}$ of 9,518 genes into two random subsets, each containing half of the genes to enable learning on GPUs with smaller memory. A separate network was then trained using each of the sets, and the final reconstruction consisted of outputs from both networks. The original D–GEX used *dropout* [143] as a regularization technique to improve the generalization of the network with three different dropout rates — 0%, 10%, and 25%. Since the D–GEX with the 25% dropout rate had the best generalization [2], we have used only this rate for our experiments. We have used the standard dropout and not one of its variants (see Section 2.2.3.2) as the standard dropout to keep the architecture similar to the original D–GEX and also because the standard dropout is very simple to interpret. All models were trained for 600 epochs (no improvement was observed near the end of the training). The performance of the model from each epoch was evaluated on the validation data, and only the best model from all epochs was used for further evaluation. The model optimization was done using the Nadam optimizer [292] (see Section 2.2.4.3 for a brief overview of the optimization process) with optimizer-specific parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and schedule decay $\eta = 0.004$; the batch size was set to 256 profiles. A fixed learning rate $\mu = 0.0005$ was used for experiments in Section 6.1 and the following schedule for experiments in Section 6.4 — the learning rate was set to 5×10^{-4} for epochs 1 – 400, 5×10^{-5} for epochs 401 – 475, 5×10^{-6} for epochs 476 –550, 5×10^{-7} for epochs 551 – 575, and 2.5×10^{-7} for epochs 576 – 600.

5.1.4 *Model evaluation*

To evaluate the model, we used the absolute error — first, we computed the mean absolute error (MAE) of prediction $\text{MAE}_m(s)$ of model m for sample $s \in \mathcal{S}$ over individual genes $g \in \mathcal{G}$ as in Eq 5.1 where $y(g, s)$ is the expression of gene g for sample s and $\widehat{y(g,s)}_m$ is the prediction of model m for the same target.

$$\text{MAE}_m(s) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \left| y(g,s) - \widehat{y(g,s)}_m \right| . \tag{5.1}$$

For further evaluation, we treat individual samples as independent (which is close enough to reality — our dataset probably contains small groups of samples that might be somewhat dependent, for example, having the same treatment, but it should be negligible for our size of dataset). Thus, for pairwise comparison, we compare error metrics over individual samples and not over individual genes that have ties to each other. The overall performance of model m is called mean mean absolute error (MMAE) and is defined as:

$$\text{MMAE}_m = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \text{MAE}_m(s). \tag{5.2}$$

To estimate the distribution of the MMAE, we employ bootstrap over $\text{MAE}_m(s)$, i.e., we resample the set of samples with a replacement to get a new set $\mathcal{S}'$ which is then used for MMAE calculation in each bootstrap iteration. Pairs of models are not compared only in terms of MMAEs but also using pairwise differences. The mean difference of absolute errors $\text{MDAE}_{m_1,m_2}(s)$ for models m_1 and m_2 and sample s is defined as:

$$\text{MDAE}_{m_1,m_2}(s) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \left(\left| y(g,s) - \widehat{y(g,s)}_{m_1} \right| - \left| y(g,s) - \widehat{y(g,s)}_{m_2} \right| \right) . \tag{5.3}$$

The MMDAE_{m_1,m_2} is defined as:

$$\text{MMDAE}_{m_1,m_2} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \text{MDAE}_{m_1,m_2}(s). \tag{5.4}$$

The pairwise nature of MMDAE_{m_1,m_2} and its distribution allow for an accurate comparison of two models even though their MMAEs are very close, and their confidence intervals (CIs) estimated using bootstrap on MAEs are overlapping. The distribution is estimated using bootstrap on $\text{MDAE}_{m_1,m_2}(s)$ in a similar manner as distribution of MMAE_m is estimated using $\text{MAE}_m(s)$.

To complement the model comparison based on mean MDAEs (MMDAEs), we have used the Student's paired t-test and the paired Wilcoxon rank test on MAEs of individual samples. These tests were used to test the hypothesis that the differences in MAEs for individual samples over all genes are significantly different.

5.1.4.1 *Pairwise evaluation of relative performance*

The evaluation focuses on determining whether the improved networks lead to better performance compared to the baseline networks. To answer this question, the prediction of improved and baseline with otherwise identical hyperparameters are compared on the level of MAEs of individual samples.

A MAE for each sample is calculated, and then using the Wilcoxon–signer rank test, it is determined whether the prediction of the network with TAAF has lower MAE in general compared to the prediction from the baseline network. It is considered a *win* if the Wilcoxon signed–rank test shows that the prediction of improved network has statistically significantly lower MAEs at the significance level $\alpha = 0.001$ and a *loss* if the baseline prediction has statistically significantly lower MAEs at the same significance level — we consider it a *tie* if it is neither.

5.1.5 *Evaluation of the practical impact*

Most of the evaluation of the models introduced focuses directly on the error of the gene expression inference; however, it is not entirely clear how lowering the error improves the accuracy of analyses applied to inferred data. To help clarify this, we show that the increased accuracy of the inference has both a statistically and practically significant impact on the accuracy of the differential gene expression (DGE) analysis.

5.1.5.1 *Differential gene expression analysis*

The differential gene expression (DGE) analysis is an analysis whose goal is to identify genes whose expressions are significantly different between two phenotypes [2112]. Usually, the DGE analysis is associated with statistical testing, and each gene has an associated p-value that is usually thresholded to identify the differentially expressed genes. The tests used for DGE analysis in this work are based on parametric empirical Bayes from the limma R package [2113, 2114], which borrows information between genes in a dynamic way [2113, 2115]. It uses a linear model that is fitted to each gene, which is used for moderating the residual variances [2113, 2116]. The procedures in the limma package allow for more reliable results for small data samples compared to other methods [2113]; more details are available in [2113].

5.1.5.2 *Used phenotypes*

There is no uniform phenotype annotation available for all of the samples. Therefore, we employed two distinct procedures to introduce a meaningful annotation for at least a limited sample subset. For the models trained on the full dataset, we ran hierarchical clustering on 2,000 samples randomly sampled from the test data of the full dataset (i.e., unseen during the training of the model), then we selected two large and relatively distinct clusters, each with more than 300 samples. In this way, we introduced two classes with naturally distinct expression profiles with a reasonable set of differentially expressed genes. Further in the text, we refer to these phenotypes as artificial. For the models trained on the heterogeneity–aware dataset, the phenotype information is available; therefore, we took the largest GEO- series (GSE2109[1]) and made sure it was in the test data of the heterogeneity-aware dataset. We used the original classes from this series as phenotype information for

1 Available at `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=gse2109`.

another set of DGE analyses. Further in the text, we refer to these phenotypes as real.

5.1.5.3 *Evaluation*

Since obtaining microarray data for DGE analysis is often very costly, we also show that the novel TAAFs improve performance even when using a low number of samples. We repeatedly sampled smaller datasets for different sample sizes where each half of the samples was from the same cluster and ran differential gene expression analysis using parametric empirical Bayes from the limma R package [2113, 2114] on the ground truth data (the actual gene expression) and the gene expressions inferred by the evaluated models. The threshold $\alpha = 0.01$ for the adjusted p–value was used to determine the differentially expressed (DE) genes. For each model and each sampled dataset of a given size, we calculated the F_1 score of the prediction of the DE genes compared to the DE genes from the ground truth data found for the same sample. Then we calculated the pairwise differences in the $F_{0.5}$, F_1, F_2 scores, accuracy, and Matthew's correlation coefficient (MCC) for compared models. The differences of all scores ($F_{0.5}$, F_1, F_2 scores, accuracy, and MCC) were tested using the Wilcoxon signed-rank test.

5.2 TRANSFORMATIVE ADAPTIVE ACTIVATION FUNCTION

We propose a novel adaptive activation function to further improve the original D–GEX that serves as our baseline. This proposal is based on an adaptive transformation of existing activation functions. The novel transformative adaptive activation function (TAAF) $g(f, z)$ [10] introduces four new parameters $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ per neuron, which transform the original activation function $f(z)$ (called *inner activation function* in the context of the TAAF):

$$g(f, z) = \alpha \cdot f(\beta \cdot z + \gamma) + \delta. \tag{5.5}$$

The output of a neuron with TAAF with inputs x_i is:

$$\alpha \cdot f\left(\beta \cdot \sum_{i=1}^{n} w_i x_i + \gamma\right) + \delta, \tag{5.6}$$

where x_i are individual inputs, w_i are its weights, and n is the number of incoming connections. If there is no unit x_i (i. e. unit constant), then the parameter γ is equivalent to the bias term of the neuron. The parameters are treated the same as other weights in the neural networks and are learned using back-propagation and gradient descent — the only difference is that parameters α and β are initialized to one and γ and δ are initialized to zero in every neuron.

The motivation for the added parameters is that they allow arbitrary translation and scaling of the original activation function, and this transformation may be different for each neuron (i. e., each neuron has four additional parameters that define the TAAF for that neuron). Furthermore, such an adaptive activation function removes the need to have a linear activation function in the last layer for regression tasks as is usually done. The usage of the linear function in the last layer requires a full set of weights for the incoming connections just for the ability to scale the output to an arbitrary range, while the proposed TAAF can do it with only four additional parameters.

The TAAF can also be viewed as a generalization of several existing adaptive activation functions — for example, the slope varying activation function [1092] is the TAAF with adaptive parameter β, and frozen $\alpha = 1$, and $\gamma, \delta = 0$, or the trainable amplitude [1086] is the TAAF with adaptive parameter α, and frozen $\beta = 1$, and $\gamma, \delta = 0$ (see Section 5.2.1.1 for full list). Other similar approaches also include parameters controlling slope but are focused only on a special, predefined function [1085, 1158] instead of allowing any activation function to be used as the inner function in the TAAF.

Loni et al. used an approach that is very similar to the TAAF in 2023; they tuned AFs by adding horizontal and vertical scaling parameters (equivalents to the TAAF's α and β) in [2117]. However, unlike the TAAF approach, they did not optimize the weights together with other weights in the network but rather used separate hyperparameter optimization regimes [2117]. More than a year after the publication of the preprint [444] proposing the TAAFs, a similar approach was proposed in [2118], where Liu et al. also used adaptive parameters for scaling and translating an AF.

5.2.1 *Motivation for individual parameters*

The TAAF parameters allow for arbitrary vertical and horizontal scaling and also arbitrary vertical and horizontal translations of the inner activation function $f(z)$.

Vertical and horizontal translation of the ELU activations were found to improve the learning in [1078], where Grelsson and Felsberg proposed ShELU activation with horizontal translation and SvELU activation with vertical translation (see Section 4.3.1.56); both activations have an additional fixed parameter controlling the translation that is not tuned. However, Grelsson and Felsberg also show that allowing the parameter to be adaptive further improves the learning performance together with an additional parameter for controlling the slope of the activation function.

The parameter α for vertical scaling of the inner activation function is quite often used in other activation functions (described in Section 5.2.1.1). Manual tuning of the parameter α can be used for controlling the gradient disappearance or overflow [972] if the inner activation function might be prone to it — e.g., Sun et al. used it with softplus as the inner activation function f. The parameter β can improve the convergence speed as shown in the concurrently published work [1138].

The parameter δ also allows for controlling the mean value of the activation as activations with mean outputs close to zero can improve the performance of a neural network [972] as they can speed up learning [1082].

5.2.1.1 *Activations as special cases of TAAFs*

As already mentioned above, the TAAF generalizes several other activation functions — while the individual parameters were often proposed individually in the literature, the TAAF provides a unique combination achieving better performance than if only some subset of parameters was used (see Section 6.1.4 for experimental results).

The scaled hyperbolic tangent [795] (see Section 4.2.2.3) can be considered as a special case of nonadaptive variant of TAAF if the TAAF is parametrized as $\alpha = a$, $\beta = b$, $\gamma = 0$, $\delta = 0$ and $f(z) = \tanh(z)$. Another case of nonadaptive TAAF is the E-Tanh (see Section 4.2.40) that uses a fixed parameter a for vertical scaling of the function; the TAAF equivalent is, therefore, $\alpha = a$, $\beta = 1$, $\gamma = 0$, $\delta = 0$ and $f(z) = \exp(z)\tanh(z)$.

The SSS (see Section 4.2.2.1) is also a special case of a nonadaptive TAAF as it is only a logistic sigmoid with horizontal scaling and translation; the TAAF equivalent is therefore $\alpha = 1$, $\beta = a$, $\gamma = -ab$, $\delta = 0$ and $f(z) = \sigma(z)$. Similarly, the VSF (see Section 4.2.2.2) is also a translated and scaled logistic sigmoid; its nonadaptive TAAF equivalent is $\alpha = a$, $\beta = b$, $\gamma = 0$, $\delta = -c$ and $f(z) = \sigma(z)$. Also, the Sloped ReLU (SlReLU; see Section 4.2.6.5) has a slope controlling parameter in a similar manner as the LReLU (and its variants) but for positive inputs. Its TAAF equivalent is $\alpha = a$, $\beta = 1$, $\gamma = 0$, $\delta = 0$ and $f(z) = \mathrm{ReLU}(z)$. There are several activation functions that use a parameter that modifies the range of the output of an activation function. One of them is the E-swish [1106] (see Section 4.3.3.4) which adds a parameter a that is the equivalent of the TAAF's parameter α — the E-swish is a special case of TAAF if $\alpha = a$, $\beta = 1$, $\gamma = 0$, $\delta = 0$ and $f(z) = z \cdot \sigma(z)$. The SGELU (see Section 4.2.3.2) also uses a parameter a for vertical scaling that controls the slope of the activation. While the parameter is fixed and nonadaptive, it can be tuned to reach better performance[822]. The SGELU can be considered as a special case of TAAF with fixed parameters: $\alpha = a$, $\beta = 1$, $\gamma = 0$, $\delta = 0$, and $f(z) = z \cdot \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right)$, where $\operatorname{erf}(x)$ is the Gauss error function.

The comb-H-sine activation (see Section 4.2.25) uses a fixed parameter a for input scaling; it can be considered as a special case of TAAFs with $\alpha = 1$, $\beta = a$, $\gamma = 0$, $\delta = 0$, and $f(z) = \sinh(z) + \sinh^{-1}(z)$.

The DRLU adds a parameter for horizontal shifting but this time it is a fixed predefined parameter a; therefore, it can be considered to be nontrainable equivalent of TAAF with $\alpha = 1$, $\beta = 1$, $\gamma = a$, $\delta = 0$, and $f(z) = \mathrm{ReLU}(z)$.

The DReLU (see Section 4.3.1.14) has a parameter a that shifts the basic ReLU both horizontally and vertically; it is TAAF equivalent for $\alpha = 1$, $\beta = 1$, $\gamma = -a$, $\delta = a$, and $f(z) = \mathrm{ReLU}(z)$. The only difference is the calculation of the value of a as the midpoint of the range of input values for each batch instead of optimizing it with the rest of a network's parameters. On the other hand, the DisReLU (see Section 4.2.6.44) employs the identical concept with a fixed, predefined parameter a instead of input dependent value — the other difference is that the parameter is defined with a negative sign. The DisReLU with parameter a is a special case of TAAF with $\alpha = 1$, $\beta = 1$, $\gamma = a$, $\delta = -a$, and $f(z) = \mathrm{ReLU}(z)$.

While the Flatted-T Swish (see Section 4.2.6.46) is a bit more complicated than a ReLU with additional parameters, it can also be considered as a special case of a TAAF but with more complicated function f — $\alpha = 1$, $\beta = 1$, $\gamma = 0$, $\delta = T$, and $f(z) = \text{ReLU}(z) \cdot \sigma(z)$, where T is the only parameter of the Flatted-T Swish.

The PSoftplus activation function (see Section 4.2.18) has two fixed parameters a and b for scaling and translation; it can be considered as a special case of the TAAF with $\alpha = a$, $\beta = 1$, $\gamma = 0$, $\delta = -ab$, and the function f is the softplus activation (see Section 4.2.17) — $f(z) = \ln(\exp(z) + 1)$.

The functions listed above are equivalent to TAAFs during the test phase or TAAFs with frozen, nonadaptive parameters. More Interestingly, many functions can be considered as a special case of TAAFs, including the property of adaptive parameters. One such function is the FReLU (see Section 4.3.1.15), which introduces parameters a_i and b_i for controlling the vertical and horizontal translation — the TAAF equivalent is with $\alpha = 1$, $\beta = 1$, $\gamma = a_i$, $\delta = b_i$, and $f(z) = \text{ReLU}(z)$. The ShiLU (see Section 4.3.1.16) is adaptive variant of ReLU that has adaptive vertical scaling using parameter a_i and vertical translation using parameter b_i; the TAAF equivalent is $\alpha = a_i$, $\beta = 1$, $\gamma = 0$, $\delta = b_i$, and $f(z) = \text{ReLU}(z)$.

The ABReLU [926] (see Section 4.2.6.42) has a parameter a_i for horizontal shifting of the function; it has the same function as the γ in TAAFs but its value is not optimized using gradient descent as in TAAFs but rather is calculated as the average of input activation map for each neuron i. The ABReLU is TAAF equivalent for $\alpha = 1$, $\beta = 1$, $\gamma = -a_i$, $\delta = 0$, and $f(z) = \text{ReLU}(z)$. The positive PReLU (see Section 4.3.1.2) is an adaptive variant of the SlReLU. Similarly, the pLogish [826] is a special case of nonadaptive TAAFs; the equivalent parameterization is $\alpha = \frac{a}{b}$, $\beta = b$, $\gamma = 0$, $\delta = 0$ and $f(z) = z \cdot \ln(1 + \sigma(z))$.

The AOAF (see Section 4.3.1.12) has three parameters, fixed b and c and adaptive parameter a_i that is calculated as the mean value of the inputs of neuron i; these parameters are used for translation of the activation function. The AOAF can be considered as a special case of TAAF but with a different scheme for updating the value of its parameters — $\alpha = 1$, $\beta = 1$, $\gamma = -ba_i$, $\delta = ca_i$, and $f(z) = \text{ReLU}(z)$.

The LeLeLU (see Section 4.3.1.8) is a LReLU with an added trainable parameter for scaling, thus its TAAF parameter is simply $\alpha = a_i$, $\beta = 1$, $\gamma = 0$, $\delta = 1$, and $f(z) = \text{LReLU}(z)$.

The RMAF (see Section 4.3.1.29) is a bit more complicated activation function that has one adaptive parameter a_i for vertical scaling and two fixed parameters b and c. Since the parameters b and c are fixed, the RMAF can be formulated using the TAAF framework — $\alpha = a_i$, $\beta = 1$, $\gamma = 0$, $\delta = 0$, and $f(z) = \left[b \frac{1}{(0.25(1+\exp(-z))+0.75)^c}\right] \cdot z$.

The RSign (see Section 4.3.13) is a sign function with horizontal shift; its TAAF formulation is therefore $\alpha = 1$, $\beta = 1$, $\gamma = -a_c$, $\delta = 0$, and $f(z) = \text{sgn}(z)$.

The paired ReLU (see Section 4.3.1.26) is a vector activation function that outputs two values instead of one; however, the same result can be obtained using two TAAFs that takes the same preactivation as the input and whose output values are then concatenated. The paired ReLU has four parameters a_i, b_i, c_i, and d_i, — one pair for each output value. In each pair, there is one parameter for horizontal scaling and one for horizontal translation. The TAAF based equivalent is $\alpha_1 = 1$, $\beta_1 = a_i$, $\gamma_1 = -b_i$, $\delta_1 = 0$, and $f_1(z) = \text{ReLU}(z)$ for the first TAAF and $\alpha_2 = 1$, $\beta_2 = c_i$, $\gamma_2 = -d_i$, $\delta_2 = 0$, and $f_2(z) = \text{ReLU}(z)$ for the second TAAF.

Similar approach to the paired ReLU is the MBA (see Section 4.3.30) which can be seen as multiple TAAFs applied to the same preactivation; in that case, each of K TAAFs would be defined as $\alpha = 1$, $\beta = 1$, $\gamma = b_{i,k}, k = 1, \ldots, K$, $\delta = 0$ and $f(z)$ can be any activation function — authors used the ReLU activation.

The SvELU and ShELU and its parametric variants (see Section 4.3.1.56), as briefly discussed in Section 5.2.1, introduce an additional parameter to the ELU activation controlling the translation. The ShELU introduces horizontal translation controlled by a fixed hyperparameter b; it is a TAAF equivalent with $\alpha = 1$ (the ELU, however, has its own parameter a for vertical scaling of the function for negative inputs), $\beta = 1$, $\gamma = b$, $\delta = 0$, and $f(z) = \text{ELU}(z)$. The SvELU introduces vertical translation instead of horizontal, it is a TAAF equivalent with $\alpha = 1$, $\beta = 1$, $\gamma = 0$, $\delta = b$, and $f(z) = \text{ELU}(z)$. The parametric variant PShELU combines the ShELU with the PELU and, as such, introduces two additional parameters controlling the slope a_i and b_i; these parameters, together with the ShELU's translation parameter c_i are adaptive. The exact TAAF equivalent is $\alpha = a_i$, $\beta = \frac{1}{b_i}$, $\gamma = \frac{c_i}{b_i}$, $\delta = 0$, and $f(z) = \text{ELU}(z)$. While Grelsson and Felsberg did not formulate the parameteric equivalent of the SvELU; it was formulated as the PSvELU in Section 4.3.1.56 in Eq. (4.326) — the TAAF equivalent parameterization is $\alpha = a_i$, $\beta = \frac{1}{b_i}$, $\gamma = 0$, $\delta = c_i$, and $f(z) = \text{ELU}(z)$. Similar AFs were proposed as variants of the HardTanh AF - the SvHardTanh introduces a fixed parameter for vertical shifts while the ShHardTanh introduces a fixed parameter for horizontal shifts. Their TAAF equivalents are $\alpha = 1$, $\beta = 1$, $\gamma = -a$, $\delta = 0$, and $f(z) = \text{HardTanh}(z)$ for ShHardTanh and $\alpha = 1$, $\beta = 1$, $\gamma = 0$, $\delta = a$, and $f(z) = \text{HardTanh}(z)$ for SvHardTanh.

Adem proposed a novel variant of the FELU by just adding a trainable parameter for vertical translation to the original AF; this is exactly what the TAAF does. Note that the original FELU is also adaptive and has its own scaling parameter a_i and its relation to the TAAFs is discussed in Section 5.2.1.2.

There is also an adaptive variant of HardTanh (see Section 4.3.1.18) that can be considered as a special case of TAAFs but with only parameter adaptive and the other is epoch dependent with a predefined schedule; the TAAF equivalent is $\alpha = 1$, $\beta = a_t$, $\gamma = -a_t b$, $\delta = 0$, and $f(z) = \text{HardTanh}(z)$ where a_t is the fixed parameter scheduled for each epoch t and b is optimized along with other parameters as is usual for TAAFs.

One of the adaptive activation function proposed earliest is the sigmoid function with shape autotuning (see Section 4.3.2, Eq. (4.339)). This function uses a single adaptive parameter $a \in (0, \infty)$ for controlling both the output range and the vertical scaling of the function; its equivalent within the TAAF framework is $\alpha = a$, $\beta = -a$, $\gamma = 0$, $\delta = 0$, and $f(z) = 2\frac{1-\exp(-z)}{(1+\exp(-z))}$. This approach was further extended into a generalized hyperbolic tangent (see Section 4.3.2.1), which separates the parameters for controlling the amplitude and the vertical scaling into a_i and b_i, which are adaptive parameters for each neuron i. The TAAF equivalent is $\alpha = a_i$, $\beta = -b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \frac{1-\exp(-z)}{(1+\exp(-z))}$.

A predecessor of TAAFs called trainable amplitude (see Section 4.3.2, Eq. (4.341)) introduces two additional adaptive parameters to any inner activation function $g(z)$; these two parameters a_i and b_i control vertical scaling and translation for each neuron i. Another general class of transformation of any activation functions was published in [1137] concurrently with our research [444] — the class of slope varying activation functions. This class adds a single adaptive parameter a to any activation function $g(z)$ allowing for horizontal scaling of the function; it is equivalent to a TAAF with $\alpha = 1$, $\beta = a$, $\gamma = 0$, $\delta = 0$, and $f(z) = g(z)$. This general approach was preceded by a special cases called SVAF (see Section 4.3.2.4) that uses hyperbolic tangent function as the inner activation $f(z)$ and ASSF (see Section 4.3.2.3) that uses logistic sigmoid as the inner activation. The psigmoid (see Section 4.3.2.6) is another AAF with scaling parameters. Unlike the SVAF, the psigmoid has both vertical and horizontal scaling parameters. Interestingly, only the vertical scaling parameter a_i is local for each neuron or channel — the horizontal scaling parameter b is global. It can be considered as a special case of TAAFs with some parameters shared and $\alpha = a_i$, $\beta = b$, $\gamma = 0$, $\delta = 0$, and $f(z_i) = \sigma(z_i)$. Another special case is the swish (see Section 4.3.3.1), an adaptive variant of the later proposed SiLU activation. The swish uses parameter a_i for horizontal scaling; its TAAF equivalent is $\alpha = 1$, $\beta = a_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = z \cdot \sigma(z)$. Another adaptive SiLU variant is the AHAF that employs both vertical and horizontal scaling; its TAAF equivalent is, therefore, $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = z \cdot \sigma(z)$. The adaptive slope hyperbolic tangent (see Section 4.3.15.1) is an adaptive function with horizontal scaling using parameter a_i with the TAAF parameterization $\alpha = 1$, $\beta =_a i$, $\gamma = 1$, $\delta = 1$, $f(z) = \tanh(z)$. The PSTanh (see Section 4.3.15.2) is an adaptive activation function that is a cross between the adaptive slope hyperbolic tangent and the slope hyperbolic tangent. The PSTanh has two scaling parameters — a_i for vertical scaling, b_i for horizontal scaling; the TAAF equivalent parameterization is $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = z \cdot (1 + \tanh(z))$. Similarly, the simpler SSinH (see Section 4.3.15.3) has also two scaling parameters a_i and b_i and its equivalent TAAF parameterization is $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \sinh(z)$. Another scaled AF is the SExp which uses exponential instead of the sinh function; its TAAF equivalent is $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \exp(z) - 1$.

Another sigmoid-based adaptive function that can be formulated within the TAAF framework is the PFTS (see Section 4.3.9) that is the combination of a ReLU and sigmoid activation with an adaptive parameter T_i for vertical translation — it is an adaptive variant of the FTS (see Section 4.2.6.46). The TAAF equivalent of PFTS is $\alpha = 1$, $\beta = 1$, $\gamma = 1$, $\delta = T_i$, and $f(z) = \mathrm{ReLU}(z) \cdot \sigma(z)$.

The parameterized softplus (see Section 4.3.20) has a parameter a_i for controlling vertical shift of the activation function; it is defined as $\alpha = 1, \beta = 1$, $\gamma = 0$, $\delta = -a_i$, and $f(z) = \ln(1 + \exp(z))$ within the TAAF framework albeit with the limiation of $\delta \in [-1, 0]$. The summary of activation functions found in the literature that can be formulated as special cases of TAAFs is in Table 5.1.

The scaled logistic sigmoid (see Section 4.3.27.1) is an adaptive function that is a special case of previously proposed NAF (see Section 4.3.27) that has parameters a_i and b_i for controlling the vertical and horizontal scale of the function; its TAAF equivalent is $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \frac{1}{1+\exp(-z)}$.

A different approach where only the activation functions are trained, and the networks are kept randomly initialized is presented in [1247] where five different adaptive activation functions with two parameters a_i and b_i each were used. Four of these activation can be formulated within the TAAF framework. The activation from Eq. (4.532) is equivalent to TAAF with $\alpha = 1$, $\beta = a_i$, $\gamma = b_i$, $\delta = 0$, and $f(z) = \frac{1}{1+\exp(-z)}$. The activation from Eq. (4.533) is equivalent to TAAF with $\alpha = 1$, $\beta = a_i$, $\gamma = b_i$, $\delta = 0$, and $f(z) = \sin(z)$. And finally, the activation from Eq. (4.534) is equivalent to TAAF with $\alpha = 1$, $\beta = a_i$, $\gamma = -a_i b_i$, $\delta = 0$, and $f(z) = \exp(-||z||)$. The activation from Eq. (4.535) could also be formulated within the TAAF framework even though it is only a step function with a variable threshold that is determined by two parameters — $\alpha = 1$, $\beta = a_i$, $\gamma = b_i$, $\delta = 0$, and

$$f(z) = \begin{cases} 1, & z \leq 0, \\ 0, & \text{otherwise}. \end{cases} \tag{5.7}$$

The final activation from [1247] shown in Eq. (4.536) cannot be formulated within the TAAF framework as only the parameter a_i has an equivalent parameter within the TAAF framework.

activation	year	section	source	adap.	param.	α	β	γ	δ	$f(z)$	note
scaled hyperbolic tangent	1998	4.2.2.3	[795]	✗	a, b	a	b	0	0	$\tanh(z)$	
E-Tanh	2022	4.2.40	[999]	✗	a	a	1	0	0	$\exp(z)\tanh(z)$	
SSS	2018	4.2.2.1	[793]	✗	a, b	1	a	$-ab$	0	$\sigma(z)$	
VSF	1995	4.2.2.2	[794]	✗	a, b, c	a	b	0	$-c$	$\sigma(z)$	
SlReLU	2017	4.2.6.5	[879]	✗	a	a	1	0	0	$\mathrm{ReLU}(z)$	
pLogish	2021	4.2.3.15	[826]	✗	a, b	$\frac{a}{b}$	b	0	0	$z \cdot \ln(1+\sigma(z))$	
E-Swish	2018	4.3.3.4	[1106]	✗	a	a	1	0	0	$z \cdot \sigma(z)$	
ABReLU	2021	4.2.6.42	[926]	✓	a_i	1	1	$-a_i$	0	$\mathrm{ReLU}(z)$	a_i calculated as the average of a neuron's input map
positive PReLU	2022	4.3.1.2	[1018]	✓	a	a	1	0	0	$\mathrm{ReLU}(z)$	
DRLU	2022	4.2.6.43	[927]	✗	a	1	1	a	0	$\mathrm{ReLU}(z)$	
AOAF	2022	4.3.1.12	[1031]	✓	a_i, b, c	1	1	$-ba_i$	ca_i	$\mathrm{ReLU}(z)$	a_i calculated as the average of a neuron's input map
DReLU	2018	4.3.1.14	[1033]	✓	a	1	1	$-a$	a	$\mathrm{ReLU}(z)$	a calculated as the midpoint of range of input values for each batch

DisReLU	2019	4.2.6.44	[928]	✗	a	1	1	a	$-a$	$\mathrm{ReLU}(z)$	
Flatted-T Swish	2018	4.2.6.46	[930]	✗	T	1	1	0	T	$\mathrm{ReLU}(z) \cdot \sigma(z)$	
PSoftplus	2019	4.2.18	[972]	✗	a, b	a	1	0	$-ab$	$\ln\left(\exp\left(z\right)+1\right)$	
SGELU	2019	4.2.3.2	[822]	✗	a	a	1	0	0	$z \cdot \mathrm{erf}\left(\frac{z}{\sqrt{2}}\right)$	
comb-H-sine	2022	4.2.25	[659]	✗	a	1	a	0	0	$\sinh\left(z\right)+\sinh^{-1}\left(z\right)$	
FReLU	2018	4.3.1.15	[1035]	✓	a_i, b_i	1	1	a_i	b_i	$\mathrm{ReLU}(z)$	
ShiLU	2023	4.3.1.16	[889]	✓	a_i, b_i	a_i	1	0	b_i	$\mathrm{ReLU}(z)$	
LeLeLU	2021	4.3.1.8	[1026]	✓	a_i	a_i	1	0	0	$\mathrm{LReLU}(z)$	
paired ReLU	2018	4.3.1.26	[1042]	✓	a_i, b_i, c_i, d_i	1	a_i, c_i	b_i, d_i	0	$\mathrm{ReLU}(z)$	concatenation of two TAAFs
RMAF	2020	4.3.1.29	[1046]	✓	a_i, b, c	a_i	1	0	0	$\left[\frac{b\cdot z}{(0.25(1+\exp(-z))+0.75)^c}\right]$	b and c are fixed
RSign	2020	4.3.13	[1023]	✓	a_c	1	1	$-a_c$	0	$\mathrm{sgn}(z)$	
RPReLU	2020	4.3.1.5	[1023]	✓	a_c, b_c, c_c	1	1	$-a_c$	b_c	$\mathrm{PReLU}(z)$	c_c is the parameter from PReLU
ShELU	2018	4.3.1.56	[1078]	✗	a, b	1	1	b	0	$\mathrm{ELU}(z)$	a is fixed parameter of the inner function f
SvELU	2018	4.3.1.56	[1078]	✗	a, b	1	1	0	b	$\mathrm{ELU}(z)$	a is fixed parameter of the inner function f
PShELU	2018	4.3.1.56	[1078]	✓	a_i, b_i, c_i	a_i	$\frac{1}{b_i}$	$\frac{c_i}{b_i}$	0	$\mathrm{ELU}(z)$	

PSvELU	—	4.3.1.56	—	✓	a_i, b_i, c_i	a_i	$\frac{1}{b_i}$	0	c_i	$\mathrm{ELU}(z)$	proposed in Section 4.3.1.56
ShHardTanh	2021	4.2.6.19	[895]	✗	a	1	1	$-a$	0	$\mathrm{HardTanh}(z)$	
SvHardTanh	2021	4.2.6.19	[895]	✗	a	1	1	0	a	$\mathrm{HardTanh}(z)$	
P+FELU	2022	4.3.1.47	[1069]	✓	b	1	1	0	b	$\mathrm{FELU}(z)$	The FELU is adaptive and has its own parameter a
Adaptive HardTanh	2021	4.3.1.18	[894]	✓	a_t, b	1	a_t	$-a_t b$	0	$\mathrm{HardTanh}(z)$	
sigmoid with shape autotuning	1992	4.3.2	[1084]	✓	a	a	$-a$	0	0	$2\frac{1-\exp(-z)}{(1+\exp(-z))}$	
generalized hyperbolic tangent	1996	4.3.2.1	[1085]	✓	a_i, b_i	a_i	$-b_i$	0	0	$\frac{1-\exp(-z)}{(1+\exp(-z))}$	
trainable amplitude	2001	4.3.2.2	[1086]	✓	a_i, b_i	a_i	1	0	b_i	$g(z)$	general approach allowing for any inner function
LAAF	2020	4.3.15	[1137]	✓	a	1	a	0	0	$g(z)$	general approach allowing for any inner function
SVAF	2009	4.3.2.4	[1092]	✓	a	1	a	0	0	$\tanh(z)$	
ASSF	2009	4.3.2.3	[1089]	✓	a	1	a	0	0	$\sigma(z)$	

psigmoid	2021	4.3.2.6	[1096]	✓	a_i, b	a_i	b	0	0	$\sigma(z)$	b is a global parameter
swish	2017	4.3.3.1	[668]	✓	a_i	1	a_i	0	0	$z \cdot \sigma(z)$	
AHAF	2022	4.3.3.2	[1105]	✓	a_i, b_i	a_i	b_i	0	0	$z \cdot \sigma(z)$	
PFTS	2020	4.3.9	[1133]	✓	T_i	1	1	0	T_i	$\mathrm{ReLU}(z) \cdot \sigma(z)$	
Adaptive slope hyperbolic tangent	2021	4.3.15.1	[1139]	✓	a_i	1	a_i	0	0	$\tanh(z)$	
PSTanh	2021	4.3.15.2	[688]	✓	a_i, b_i	a_i	b_i	0	0	$z \cdot (1 + \tanh(z))$	
SSinH	2021	4.3.15.3	[1140]	✓	a_i, b_i	a_i	b_i	0	0	$\sinh(z)$	
SExp	2021	4.3.15.4	[1140]	✓	a_i, b_i	a_i	b_i	0	0	$\exp(z) - 1$	
parameterized softplus	2023	4.3.20	[699]	✓	a_i	1	1	0	$-a_i$	$\ln(1 + \exp(z))$	$\delta \in [-1, 0]$
scaled logistic sigmoid	2007	4.3.27.1	[1161]	✓	a_i, b_i	a_i	b_i	0	0	$\frac{1}{1+\exp(-z)}$	special case of NAF
MBA	2016	4.3.30	[1166]	✓	$b_{i,k}, k = 1, \ldots, K$	1	1	$b_{i,k}$	0	$g(z)$	general approach allowing for any inner function; K TAAFs applied to same preactivation
Eq. (4.532)	2018	4.3.55.4	[1247]	✓	a_i, b_i	1	$-a_i$	b_i	0	$\frac{1}{1+\exp(-z)}$	unnamed AF
Eq. (4.533)	2018	4.3.55.4	[1247]	✓	a_i, b_i	1	a_i	b_i	0	$\sin(z)$	unnamed AF

Eq. (4.534)	2018	4.3.55.4	[1247]	✓	a_i, b_i	1	a_i	$-a_i b_i$	0	$\exp\left(-\|\|z\|\|\right)$	unnamed AF
Eq. (4.535)	2018	4.3.55.4	[1247]	✓	a_i, b_i	1	a_i	b_i	0	$\begin{cases} 1, & z \leq 0, \\ 0, & \text{otherwise}, \end{cases}$	unnamed AF

Table 5.1: **Activation functions as special cases of TAAFs**
Activation functionsthat can be formulated within the TAAF framework. The columns α, β, γ, δ and $f(z)$ show the equivalent formulation within the TAAF framework.

5.2.1.2 *Activations related to TAAFs*

Some of the activation functions proposed in literature employ similar concepts as the TAAFs but cannot be considered to be a special case of the TAAFs. Nevertheless, the motivation for the concepts remains similar as for TAAFs. One such example is the improved logistic sigmoid (see Section 4.2.2.7) that uses a fixed parameter a for controlling the slope of the outermost pieces of the piecewise function. However, since the central part of the piecewise function is not subjected to the controllable slope, it cannot be formulated as a special case of a TAAF. An AF very similar to the improved logistic sigmoid is the STAC-tanh (see Section 4.3.2.8) — the only difference is that it uses tanh instead of logistic sigmoid and its parameters a_i and b_i are adaptive. While it has a different shape, the RSigELU (see Section 4.2.7.15) also has parameter a for controlling the slope of the outermost components of the function. Similarly, the penalized hyperbolic tangent (see Section 4.2.2.9) has a fixed, slope-controlling parameter but only for negative inputs. The Hexpo (see Section 4.2.2.12) is an activation function with four fixed parameters that have similar functions as parameters α and β in TAAFs. The Hexpo is a piecewise function that is defined separately for positive and negative inputs — the parameter a is the equivalent of TAAFs α for positive inputs, and the parameter c is the equivalent for negative inputs; similarly, parameters b and b are equivalents of $\frac{1}{\beta}$.

Fixed slope controlling parameter in a piecewise function is also used in the LReLU, VLReLU, and OLReLU (see Section 4.2.6.2) where the parameter controls the slope of the "leaky" part of the activation function for negative inputs. Similarly, the SignReLU (see Section 4.2.6.32) uses a parameter a for controlling the slope for negative inputs; however, unlike the LReLU and its variant, the function is not linear for negative inputs. The DLReLU (see Section 4.3.1.13) also has a fixed parameter for controlling the slope for negative inputs as LReLU has, but it also has an additional parameter that scales the slope of the negative inputs further using the test error from the previous epoch. The RReLU (see Section 4.2.6.3) uses a stochastic slope controlling parameter during training and a fixed for inference when it becomes the LReLU. The EReLU (see Section 4.2.6.38) is similar to RReLU, but it uses stochastic parameters for controlling the slope for positive inputs instead of negative inputs.

The ELU (see Section 4.2.6.48) also has a parameter that linearly scales the function for negative inputs — albeit since the function is controlled by an exponential, the main reason for the parameter is to control to which value the ELU converges for inputs going to negative infinity. The SELU (see Section 4.2.7.11) has two parameters a and b controlling the slope — one (a) for the whole function and the other only for negative inputs (b). Since these parameters are fixed, it could be considered as a special case of TAAFs with the first parameter equivalent to TAAF's α that is fixed and with the TAAF's inner activation $f(z)$ being parameterized with another parameter b. Similarly, its extension LSELU (see Section 4.2.7.12) has one parameter for controlling the slope for all inputs; however, the LSELU is a sum of an ELU and linear function for negative inputs and slope of each component is controlled separately by parameters b and c. The sSELU (see Section 4.2.7.14) has two parameters a and b for vertically scaling the function separately for negative and positive inputs; it also has a parameter c for horizontally scaling the function for negative inputs similarly as does β in TAAFs. The RSigELUD (see Section 4.2.7.19) also has two parameters for controlling the slope of individual components of the function; it has parameter a for controlling the slope of the exponential component for inputs above one, and parameter b for controlling the slope for negative inputs. However, since it has no parameter for controlling the slope for inputs in the interval $[0, 1]$, where it is defined as a linear function, and since the parameter a does not control the slope of the whole function for inputs above one but rather only weights one component of the function, it cannot be considered as a special case of a TAAF but with different parameterization for positive and negative inputs as many other functions can.

The SoftModulusT (see Section 4.2.6.31) uses a fixed, predefined parameter a for scaling the input of the function similarly as the TAAF's parameter β albeit in an inverse form — $\beta \sim \frac{1}{a}$ — and only for the input going to the hyperbolic tangent function.

The NReLU (see Section 4.2.6.6) introduces a stochastic variant of the shift parameter γ — the mean value of the parameter is 0, and, therefore, it only introduces additive noise during training. The motivation behind NReLU is different from the motivation of the TAAFs, but nevertheless, the concept of the additive parameter to specific inputs resembles the TAAF's parameter γ. The RT–ReLU (see Section 4.2.6.11) also introduces stochastic translational parameter as the NReLU but samples the parameters from different distributions. The ReSP (see Section 4.2.6.14) also has a fixed parameter controlling the slope of the function for positive inputs only. On the other hand, the BLReLU (see Section 4.2.6.24) has a fixed parameter controlling the slope for the negative inputs and also for inputs above a threshold predefined by another parameter similar to the improved logistic sigmoid.

The Soft++ activation function (see Section 4.2.18.1) is a composition of a horizontally scaled softplus activation using parameter a and vertically scaled linear function using parameter b with an additional fixed offset. While it cannot be considered as a special case of the TAAF due to the composition of the two functions, the parameter a has an identical role as the parameter β in TAAFs, the parameter b scales the linear component similarly as parameter $\frac{1}{\alpha}$ and the linear offset can be defined as $\delta = -\ln(2)$.

While the activation function above do not have the parameters trainable — some of them use different adaptive schemes — there are also other activation functions that uses adaptive, trainable parameters similarly to TAAFs. One of them is the PReLU (see Section 4.3.1.1) that is basically a LReLU, but the parameter a is adaptive. The TAAF's parameter α is the equivalent of PReLU's parameter $\frac{1}{a}$ but only for negative inputs; there is no adaptive scaling for positive inputs. The RT–PReLU (see Section 4.3.1.10) is the PReLU but with additional stochastic parameter b that is randomly sampled and that controls the threshold of the piecewise function.

The PREU (see Section 4.3.1.9) has vertical scaling for the whole function, and, therefore, its parameter a is the direct equivalent of TAAF's parameter α.

However, it also introduces an equivalent for TAAF's parameter β but only for negative inputs; therefore, it cannot be considered a special case of the TAAF.

The AReLU (see Section 4.3.1.19) has two adaptive scaling parameters a_l and b_l as it has separate scaling of positive and negative inputs. However, its difference from the TAAF is much larger — the parameter b_l scaling positive inputs is transformed using the logistic sigmoid into interval $[1, 2]$ by $(1 + \sigma(b_l))$ and the parameter a_l for scaling negative inputs is clipped into interval $[0.01, 0.99]$.

The tanhLU (see Section 4.3.1.33) uses both vertical and horizontal scaling; however, since it has two components, it uses a separate parameterization for each of the components. The tanh component has a parameter a_i for vertical scaling and a parameter b_i for horizontal scaling, whereas the linear function has only a single parameter c_i for scaling as there is no difference between vertical and horizontal scaling of linear functions.

Separate adaptive parameters for controlling the slope are used in several adaptive activation functions. One of the simplest examples is the DPReLU (see Section 4.3.1.20), which is a piecewise linear function with one parameter controlling the slope for positive inputs and the other for negative inputs. The DPReLU is extended by an adaptive parameter m_i controlling vertical translation into the Dual Line activation function (see Section 4.3.1.21). Since the translation parameter m_i is shared by both piecewise components of the function, it is a direct equivalent of TAAF's parameter δ.

The PiLU (see Section 4.3.1.22) is another DPReLU extension; it generalizes the Dual Line by adding any horizontal shift — it has two parameters for vertical scaling (one for inputs below the threshold and one for inputs above the threshold) with similar function as the TAAF's α and one single parameter for the threshold which allows for the horizontal shift similarly as does the γ in TAAFs.

Similarly as TAAFs accept any inner activation function, the DPAFs (see Section 4.3.1.23) extends the Dual Line concept to use any suitable inner activation function; the DPAF uses an inner activation function $g(z_i)$ instead of the linear function from the Dual Line. It closely resembles the TAAF with α applied only for positive inputs, $\beta = 1$, $\gamma = 0$ and $\delta = m_i$. The FPAF (see Section 4.3.1.24) is very similar to DPAF, but it allows for two different inner functions, one for positive inputs and the other for negative inputs. Each of the inner functions has its own adaptive parameter for vertical scaling, but unlike DPAF, there is no adaptive translation parameter.

The EPReLU (see Section 4.3.1.25) also has two separate parameters for positive and negative inputs that control the vertical scaling; however, only the parameter a_i scaling the negative inputs is trainable; the parameter scaling the function for positive inputs is stochastic and sampled from a uniform distribution centered around 1 in each training epoch.

The PTELU (see Section 4.3.1.30) behaves as linear function for positive inputs and as a special case of TAAF for negative inputs with $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \tanh(z)$ where a_i and b_i are trainable parameters for each neuron i. The later proposed TReLU (see Section 4.3.1.35) is identical to PTELU but with fixed $\alpha = a_i = 1$ as it only has horizontal scaling for negative inputs.

The BLU (see Section 4.3.1.37) is an activation function that has two components — nonlinear function with adaptive scaling parameter a_i and a linear component. While the scaling parameter has a similar role as the TAAF's parameter α, its values are limited to the range $[-1, 1]$.

The PELU (see Section 4.3.1.43) extends the ELU by two parameters a_i and $\frac{a_i}{b_i}$ controlling the slope — separate parameters for positive and negative inputs — but it also has a horizontal scaling parameter $\frac{1}{b_i}$ for the exponential part of the PELU for negative inputs. The parameters a_i and b_i are formulated such that there is no non-differentiability at input $z = 0$. Another ELU extension FELU (see Section 4.3.1.46) uses adaptive scaling parameter a_i to control the soft saturation region for negative inputs. The MPELU (see Section 4.3.1.48) outputs identity for positive inputs, but it outputs nonlinearly transformed input for negative values that can be formulated within the TAAF framework — with $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \exp(z) - 1$ where a_i and b_i are MPELU's scaling parameters. The CELU (see Section 4.3.1.51) is similar to MPELU but it is reparameterized using a single parameter a_i such that its derivative at $z = 0$ is 1 — the only difference from the MPELU is that its TAAF reformulation for negative part is $\alpha = a_i$ and $\beta = \frac{1}{a_i}$.

The PSELU (see Section 4.3.1.53), which is the adaptive variant of SELU, has two trainable scaling parameters a_i and b_i that allow for vertical scaling of the function; the parameter a_i is scaling the whole function and as such is the exact equivalent of the parameter α while the parameter b_i scales the function only for negative inputs. The LPSELU (see Section 4.3.1.54) is identical to PSELU, but it adds a linear function for negative inputs to avoid small gradients — this linear function has slope controlled by another parameter c_i; i.e., the function has two parameters that control the slope only for negative inputs, b_i for the exponential part and c_i for the linear part, the parameter a_i controls the slope of the whole function. The LPSELU_RP (see Section 4.3.1.55) extends the LPSELU by the additional parameter m_i that controls the vertical translation of the whole function; this trainable parameter represents an exact equivalent of the TAAF's parameter δ.

The PDELU (see Section 4.3.1.59) introduces two parameters a_i and b; while a_i is an adaptive parameter that controls the scaling of the function for negative inputs, parameter b is a fixed hyperparameter controlling the shape of the nonlinear part of the activation. Similarly, the T-swish (see Section 4.3.1.57) has parameters a_i and b_i for vertical and horizontal scaling only for negative inputs.

The EELU (see Section 4.3.1.60) is an activation function with a stochastic component for positive inputs and function scaling for negative inputs. The function is scaled using parameter k_i for positive inputs, which is stochastic and is sampled from a Gaussian distribution with random variance (sampled from a uniform distribution) and is clipped into interval $[0, 2]$. Adaptive parameters a^c and b^c are used for vertical and horizontal scaling of the function for negative inputs and are shared by all neurons in channel c; the function for negative inputs can be formulated within TAAF framework using $\alpha = a^c$, $\beta = b^c$, $\gamma = 0$, $\delta = 0$, and $f(z) = \exp(z) - 1$.

The scaled softsign (see Section 4.3.19) is controlled by two adaptive parameters a_i and b_i; however, only the a_i has an equivalent within the TAAF framework — the parameter a_i controls the vertical scale of the function and thus it is the equivalent of the parameter α of the TAAF framework. The parameter b_i controls the rate of transition between signs, and as such, it does not have an equivalent within the TAAF framework.

The NAF (see Section 4.3.27) consists of parts; each one has one parameter for controlling its vertical scale. The parts also have two additional parameters for controlling the horizontal scale similarly as does β in the TAAFs — the first part has parameter b β^2 while the second uses parameter d that has the same function as β without any non-linear transformation. Function similar to NAF is the combination of scaled logistic sigmoid with scaled sine (SLS-SS; see Section 4.3.27.1) uses four parameters, one pair for controlling the horizontal and vertical scale of the logistic sigmoid and the other pair controlling the horizontal and vertical scale of the sine function; the function can be seen also as the combination of two TAAF based functions — the first with $\alpha = a_i$, $\beta = b_i$, $\gamma = 0$, $\delta = 0$, $f(z) = \sin(z)$ and the second with $\alpha = c_i$, $\beta = d_i$, $\gamma = 0$, $\delta = 0$, and $f(z) = \frac{1}{1+\exp(-z)}$.

The APLU (see Section 4.3.28) can be seen as sum of $S + 1$ TAAF based functions where the first function is just plain $\mathrm{ReLU}(z)$ while the others can be defined as TAAF equivalents with $\alpha = a_i^s$, $\beta = 1$, $\gamma = -b_i^s$, $\delta = 0$, and $f_s(z) = \mathrm{ReLU}(-z)$.

The MeLU (see Section 4.3.31) is an approach with the same representation power as the APLU but with a lower number of parameters; it consists of a sum of functions, each having its own trainable parameter for vertical scaling $a_{i,j}$.

Function combining sigmoid-like and ReLU functions is the SReLU (see Section 4.3.32); it is a piecewise function with linear function in the middle and with two trainable determining thresholds limiting the middle identity segment; it also has two trainable parameters controlling the slope of the outermost segments. Similarly, the LinQ (see Section 4.2.8.8) has one non-adaptive parameter for scaling the slope of the function but only for the parts that are outside the interval $[-2, 2]$. The PLU (see Section 4.3.34) can be considered as a special case of the SReLU enforcing invertibility of the function; it has only one trainable parameter a_i that determines the slope of two linear segments similarly as α does in TAAFs. The AdaLU (see Section 4.3.35) is a piecewise linear function with adaptive parameters for controlling the slope and shifts of individual components.

The MTLU (see Section 4.3.43) extends the SReLU approach into more than three segments; each of the K segments has a parameter $a_{i,k}, k = 0, \ldots, K$ that controls the slope of the respective segment (a local equivalent of α)and parameter $b_{i,k}, k = 0, \ldots, K$ that controls its translation (a local equivalent of δ); the segments are determined by parameters $c_{i,0}, \ldots, c_{i,K-1}$. The LuTU (see Section 4.3.45) is also a piecewise linear activation function where each segment has adaptive slope and bias — however, the function is defined by several anchor points instead of using direct equivalents of α and δ for each segment.

The maxout unit (see Section 4.3.46) returns a maximum of multiple linear functions; it can also be seen as returning maximum of K TAAFs; each with $\alpha = w_i^k$, $\beta = 1$, $\gamma = 0$, $\delta = b_i^k$, and $f(z) = z$, $k = 1, \ldots, K$.

The DY–ReLU (see Section 4.3.55.3) is a different approach compared to most of the adaptive functions in this list — it uses a hyperfunction for computing the parameters of the activation function. The activation function itself is a piecewise linear function that is defined as the maximum of multiple linear functions — it is a maximum of multiple independent TAAFs, each with two parameters that are equivalent to the TAAF parameters α and δ.

A similar approach to the maxout unit is the ABU and its variants (see Section 4.3.47) — using a weighted sum of activation functions instead of the maximum. This can be seen as a sum of TAAF based functions when the weight a_j, l is equivalent to the scaling parameter α for the relevant TAAF with any inner activation $g_j(z)$. The formulation of ABU as a sum of TAAF is beneficial for the extended variant with additional bias parameter (see Eq. (4.478)) — this ABU is a sum of n TAAF based functions with $\alpha_j = a_{i,j}$, $\beta_j = 1$, $\gamma_j = -b_{i,j}$, $delta_j = 0$ for any inner activation function. There are other ABU variants whose weights of individual inner activation functions have to sum up to 1 [1195, 1196] or that are employing min–max scaling [1196]. Another ABU variant called APAF divides the output by the sum of the weighting coefficients — the output is the weighted average of the inner activation functions. The GABU (see Section 4.3.47.3) is an ABU variant that uses gating functions for obtaining the scaling parameters of individual inner activation functions. The SLAF (see Section 4.3.47.6) is a special case of ABU that utilizes the increasing powers of the input as the individual inner activation functions. Similarly, the ChPAF (see Section 4.3.47.7) and LPAF (see Section 4.3.47.8) can be considered as ABU, but the inner functions are Chebyshev and Legendre polynomials instead.

The SinLU (see Section 4.3.3.11) uses vertical and horizontal scaling parameters a_i and b_i only for a single term in its definition that adds a sine function to the base linear function.

The KAF (see Section 4.3.55.5) is an activation function that uses kernel expansion with a dictionary; however, since Scardapane et al. used D fixed dictionary points, it can also be viewed as a sum of individually scaled functions with parameters $a_{i,j}$, $j = 1, \ldots, D$.

The PAU (see Section 4.3.48) extends the ABU concept even further; a PAU is basically a division of two SLAFs — i.e., the PAU is the division of two sums of individually transformed functions that are polynomials of increasing power. The ERA (see Section 4.3.50) is a function that is very similar to the PAU; however, the ERA is parameterized in such way that it can be rewritten using partial fractions reducing the number of operations — this formulation however holds even less similarities with the TAAF parameterization.

The MoGU (see Section 4.3.47.10) is, similarly to the ABU, also a sum of individually transformed functions. However, unlike ABU, the MoGU uses more TAAF parameters than just the α. It can be defined as a sum of n TAAF based functions with $\alpha_j = \frac{a_{i,j}}{\sigma_{i,j}}$, $\beta_j = \frac{1}{\sigma_{i,j}}$, $\gamma_j = \frac{-\mu_{i,j}}{\sigma_{i,j}}$, $\delta = 0$, and $f_j(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(z)^2\right)$, $j = 1, \dots, n$. Similarly, the TCA and TCAv2 (see Section 4.3.47.1) can be seen as a sum (TCA) or a weighted average (TCAv2) of k TAAF based functions. The TAAF based functions are using parameters $\beta i, j = \exp\left(a_{i,j}\right)$ and $\gamma i, j = \exp\left(b_{i,j}\right)$ in TCA and $\alpha i, j = \exp\left(a_{i,j}\right)$, $\beta i, j = \exp\left(b_{i,j}\right)$, and $\gamma i, j = \exp\left(c_{i,j}\right)$in TCAv2. Note that the sum of the functions in TCAv2 is divided by $\sum_{j=1}^{k} \exp\left(a_{i,j}\right)$ to obtain the weighted average of the functions.

The MSAF (see Section 4.2.2.24 is a sum of individually translated logistic sigmoids); it has a parameter for vertical translation a and each logistic sigmoid has another translation parameter b_k for horizontal translation. These translations, however, seem to be predefined and nonadaptive.

Similarly, the FSA (see Section 4.3.47.11) is also a sum of individually transformed functions; however, there are two different functions this time, and they are transformed using equivalents of both α and β — i.e., they have parameters for both horizontal and vertical scaling. Furthermore, there is also a single parameter a_i that controls the vertical translation similarly to the parameter δ in TAAFs.

The VAF (see Section 4.3.55.1) approach, published parallelly with the TAAFs, uses a specially defined subnetwork instead of a simple activation function; the resulting activation function from the subnetwork is equivalent to the sum of TAAFs in the most general sense — it has all four TAAF parameters in equivalent formulation and also allows for usage of any inner function. While the VAF is more general than TAAF, it also has significantly more parameters proportional to the size of the subnetwork.

The Table 5.2 summarizes the activation functions that uses concepts that are related to those used in TAAFs.

activation	year	details	source	adapt.	parameters	TAAF equiv.	note
improved logistic sigmoid	2019	Section 4.2.2.7	[801]	✗	a, b	α	controllable slope only for certain inputs
STAC-tanh	2021	Section 4.3.2.8	[1102]	✓	a_i, b_i	α	controllable slope only for certain inputs determined by adaptive thresholds
RSigELU	2021	Section 4.2.7.15	[949]	✗	a	α	controllable slope only for certain inputs
penalized hyperbolic tangent	2016	Section 4.2.2.9	[791]	✗	a	α	controllable slope only for certain inputs
Hexpo	2017	Section 4.2.2.12	[806]	✗	a, c; b, d	α; β	different parameters for negative and positive inputs
LReLU	2013	Section 4.2.6.2	[869]	✗	a	α	controllable slope only for certain inputs
VLReLU	2014	Section 4.2.6.2	[870]	✗	a	α	controllable slope only for certain inputs
OLReLU	2021	Section 4.2.6.2	[876]	✗	a	α	controllable slope only for certain inputs
DLReLU	2019	Section 4.3.1.13	[1032]	✗	ab_t	α	controllable slope only for certain inputs; slope controlled by a fixed parameter (a) but also using the test error from previous epoch (b_t)

SoftModulusT	2023	Section 4.2.6.31	[836]	✗	a	β	horizontal scaling only of the tanh component
SignReLU	2018	Section 4.2.6.32	[910]	✗	a	α	controllable slope only for certain inputs
RReLU	2015	Section 4.2.6.3	[872]	✗	a	α	controllable slope only for certain inputs, stochastic
EReLU	2018	Section 4.2.6.38	[918]	✗	a	α	controllable slope only for certain inputs
NReLU	2010	Section 4.2.6.6	[861]	✗	a	γ	stochastic parameter with zero mean
RT–ReLU	2018	Section 4.2.6.11	[883]	✗	a	γ	stochastic parameter with zero mean
ReSP	2018	Section 4.2.6.14	[886]	✗	a	α	controllable slope only for certain inputs
BLReLU	2016	Section 4.2.6.24	[888]	✗	a	α	controllable slope only for certain inputs
ELU	2016	Section 4.2.6.48	[874]	✗	a	α	controllable slope only for certain inputs
SELU	2017	Section 4.2.7.11	[945]	✗	a, b	α	separately controllable slope for positive and negative inputs

LSELU	2021	Section 4.2.7.12	[946]	✗	a, b, c	α	individual components has separate parameters for controlling the slope
sSELU	2021	Section 4.2.7.14	[946]	✗	a, b, c	α, β	individual components have separate parameters for controlling the slope
RSigELUD	2021	Section 4.2.7.19	[949]	✗	a, b	α	individual components have separate parameters for controlling the slope
Soft++	2020	Section 4.2.18.1	[973]	✗	a, b	α, β	one component is vertically scaled, the other horizontally scaled
PReLU	2015	Section 4.3.1.1	[871]	✓	a	α	controllable slope only for certain inputs
RT–PReLU	2018	Section 4.3.1.10	[883]	✓	a	α	controllable slope only for certain inputs; stochastic thresholding
PREU	2019	Section 4.3.1.9	[932]	✓	a, b	α, β	horizontal scaling only for negative inputs
AReLU	2020	Section 4.3.1.19	[1037]	✓	a_l, b_l	α	separate scaling for negative and positive inputs; parameter transformation
tanhLU	2022	Section 4.3.1.33	[1049]	✓	a_i, b_i, c_i	α, β	separate scaling for each component

DPReLU	2020, 2021	Section 4.3.1.20	[1039, 1040]	✓	a_i, b_i	α	proposed independently in [1039] and [1040]
Dual Line	2020	Section 4.3.1.21	[1039]	✓	a_i, b_i, m_i	α, δ	separate scaling for negative and positive inputs; common vertical translation
PiLU	2021	Section 4.3.1.22	[1041]	✓	a_i, b_i, c_i	α, γ	separate scaling for negative and positive inputs; common horizontal translation
DPAF	2020	Section 4.3.1.23	[1039]	✓	a_i, m_i	α, δ	slope scaling only for positive inputs; common vertical translation
FPAF	2021	Section 4.3.1.24	[1040]	✓	v	α	separate scaling for negative and positive inputs
EPReLU	2018	Section 4.3.1.25	[918]	✓	a_i, k_i	α	separate scaling for negative and positive inputs, stochastic scaling for positive inputs
PTELU	2017	Section 4.3.1.30	[1047]	✓	a_i, b_i	α, β	scaling for negative inputs only
TReLU	2019	Section 4.3.1.35	[1050]	✓	b_i	β	scaling for negative inputs only
BLU	2019	Section 4.3.1.37	[1053]	✓	a_i	α	scaling only the nonlinear component

PELU	2016	Section 4.3.1.43	[1064]	✓	a_i, b_i	α, β	separate vertical scaling for positive and negative inputs, horizontal scaling only for negative inputs
FELU	2019	Section 4.3.1.46	[1067]	✓	a_i	α	scaling only for negative inputs
MPELU	2018	Section 4.3.1.48	[1070]	✓	a_i, b_i	α, β	scaling only for negative inputs
CELU	2017	Section 4.3.1.51	[1075]	✓	a_i	α, β	scaling only for negative inputs, continuously diff.
PSELU	2020	Section 4.3.1.53	[1077]	✓	a_i, b_i	α	separate vertical scaling for positive and negative inputs
LPSELU	2020	Section 4.3.1.54	[1077]	✓	a_i, b_i, c_i	α	individual components have separate parameters for controlling the slope
LPSELU_RP	2020	Section 4.3.1.55	[1077]	✓	a_i, b_i, c_i, m_i	α, δ	individual components have separate parameters for controlling the slope
PDELU	2020	Section 4.3.1.59	[1082]	✓	a_i, b	α	scaling only for negative inputs, fixed b for shape control
T-swish	2022	Section 4.3.1.57	[1079]	✓	a_i, b_i, c_i	α, β	scaling only for negative inputs, c_i for threshold determination

EELU	2020	Section 4.3.1.60	[1027]	✓	a^c, b^c, k_i^c	α, β	vertical and horizontal scaling for negative inputs; vertical stochastic scaling for positive inputs
scaled softsign	2023	Section 4.3.19	[889]	✓	a_i, b_i	α	additional adaptive parameter
NAF	2000	Section 4.3.27	[1158]	✓	a, b, c, d	α, β	each component has its own scaling
SLS-SS	2007	Section 4.3.27.1	[1161]	✓	a_i, b_i, c_i, d_i	α, β	each component has its own scaling
APLU	2017	Section 4.3.28	[1014]	✓	a_i^s, b_i^s, $s = 1, \ldots, S$	α, γ	sum of S TAAFs
SReLU	2016	Section 4.3.32	[873]	✓	t_i^r, a_i^r, t_i^l, a_i^l	α	slope controllable only for the outermost segments
LinQ	2016	Section 4.2.8.8	[955]	✓	a	α	slope controllable only for the outermost segments outside the interval $[-2, 2]$
All-ReLU	2021	Section 4.3.33	[1172]	✗	a	α	slope controllable only for negative inputs, alternating between layers
PLU	2018	Section 4.3.34	[1173]	✓	a_i, b	α	b fixed; slope controllable only for the outermost segments

AdaLU	2022	Section 4.3.35	[1174]	✓	a_i, b_i, c_i, d_i, e_i	α, γ, δ	b controllable slopes and offsets for individual components
MTLU	2019	Section 4.3.43	[1179]	✓	$a_{i,0}, \ldots, a_{i,K}$, $b_{i,0}, \ldots, b_{i,K}$, $c_{i,0}, \ldots, c_{i,K-1}$	α, δ	separate parameters for indiv. segments
maxout unit	2013	Section 4.3.46	[1191]	✓	w_i^k, b_i^k, $k = 1, \ldots, K$	α, δ	maximum of individually transformed functions
DY–ReLU	2020	Section 4.3.55.3	[1034]	✓	$a_{i,k}$, $b_{i,k}$, $k = 1, \ldots, K$	α, δ	maximum of individually transformed functions; hyperfunction for parameter optimization
ABU	2020	Section 4.3.47	[1194]	✓	$a_{i,j}$	α	sum of individually transformed functions
ABU with bias	2018	Section 4.3.47	[1186]	✓	$a_{i,j}$, $b_{i,j}$	α, β	sum of individually transformed functions
ABU (constrained)	2018	Section 4.3.47	[1195]	✓	$a_{i,j}$	α	sum of individually transformed functions; their scaling parameter sum up to 1
TCA	2022	Section 4.3.47.1	[1197]	✓	$a_{i,j}$, $b_{i,j}$	β, γ	sum of individually transformed functions
TCAv2	2023	Section 4.3.47.1	[1198]	✓	$a_{i,j}$, $b_{i,j}$, $c_{i,j}$	α, β, γ	sum of individually transformed functions

activation ensemble	2019	Section 4.3.47	[1196]	✓	$a_{i,j}$	α	sum of individually transformed functions; their scaling parameter sum up to 1; min–max scaling
SinLU	2017	Section 4.3.3.11	[1127]	✓	a_i, b_i	α, β	scaling only of a single term
GABU	2016	Section 4.3.47.3	[1013]	✓	$a_{i,j}$	α	sum of individually transformed functions; gated
SLAF	2019	Section 4.3.47.6	[1202]	✓	$a_{i,j}$	α	sum of individually transformed functions
ChPAF	2023	Section 4.3.47.7	[1206]	✓	a_j, $j = 0, \dots, k$	α	sum of individually transformed functions
LPAF	2021	Section 4.3.47.8	[1207]	✓	a_j, $j = 0, \dots, k$	α	sum of individually transformed functions
KAF	2019	Section 4.3.55.5	[1248]	✓	$a_{i,j}$, d_j, $j = 1, \dots, D$	α	d_j fixed; sum of individually transformed functions
PAU	2020	Section 4.3.48	[1209]	✓	a_j, $j = 0, \dots, m$, b_k, $k = 1, \dots, n$	α	division of two sums of sum of individually transformed functions
MoGU	2018	Section 4.3.47.10	[1186]	✓	$a_{i,j}$, $\sigma_{i,j}$, $\mu_{i,j}$, $j = 1, \dots, n$	α, β, γ	sum of individually transformed functions
MSAF	2015	Section 4.2.2.24	[813]	✗	a, b_k, $k = 1, \dots, N$	γ, δ	sum of individually translated functions

FSA	2020	Section 4.3.47.11	[1199]	✓	a_i, $b_{i,j}$, $c_{i,j}$, d_i, $j = 0, \ldots, r$	α, β, δ	sum of individually transformed functions
VAF	2019	Section 4.3.55.1	[1243]	✓	$a_{l,0}$, $a_{l,j}$, $b_{l,j}$, $c_{l,j}$, $j = 1, \ldots, k$	α, β, γ, δ	sum of TAAFs

Table 5.2: **Activation functions related to TAAFs**
Activation functions that employs the same or similar concepts as TAAFs. The column *TAAF equiv.* lists the TAAF's parameters whose function the activation also employs in any manner.

5.2.1.3 *TAAF as output layer*

It is standard practice to use an output layer with a linear activation function as the sigmoidal activation functions such as hyperbolic tangent and logistic sigmoid have limited ranges. The original D–GEX architecture is no exception and uses a linear output layer. This, however, is no longer necessary with the use of TAAF as the scaling and translation allow for an arbitrary range. The modified network architectures with TAAFs in the output layer (denoted TAAFo) enable better performance than those with a linear activation in the output layer by increasing the network capacity. The need for a linear layer for some regression task could be, of course, solved by other approaches — e.g., another custom activation with only a single scaling parameter such as the activation function with trainable amplitude [1086] but the TAAF provides higher flexibility thanks to the added parameters. The usage of TAAFs in all layers is beneficial for two reasons:

ARCHITECTURE SIMPLIFICATION When TAAFs are used in all layers, the architecture is less complex and easier to understand and analyze than if different activation functions are used in different components of the neural network. Simpler and more coherent architectures are also simpler to debug.

EFFICIENT USAGE OF PARAMETERS Usage of a linear activation function in the output layer for purposes of output scaling introduces many unnecessary parameters — the linear layer needs $(N+1) \times M$ weights where N is the number of neurons in the previous layer, and M is the number of outputs of current layers. While a linear layer might be beneficial if the regression task naturally yields a solution as a linear combination of some nonlinear function, it introduces too many additional parameters if only output scaling is needed. Using TAAFs in the output layer instead of a linear activation functions often provides network capacity more similar to a network with $D+1$ layers with nonlinear activations in hidden layers and a linear activation in the output layer while keeping the number of parameters similar to the original network with D layers.

5.2.2 *Ensembles*

Integrating multiple neural networks (or other learners) into an ensemble very often leads to a better performance level than that of every single learner from the ensemble [2119–2122]. It is common practice to build ensembles of neural networks even for quite complex neural networks (e. g. [2123, 2124]). Ensemble usage is also a common practice when working with microarray data [2125] (e. g. an ensemble of *support vector machines* was used in [2126]). Further description of ensembles is out of the scope of this work; reviews are available in [2121, 2122, 2127, 2128].

We have evaluated ensembles consisting of different D–GEX architectures as the evaluation was without any significant computational overhead. Our ensemble selects a single D–GEX architecture as an expert for each gene based on one-half of the validation data; then only this expert is used to predict the expression of the given gene — this leads to better performance if some neural networks learned better prediction for some genes than the others. We have evaluated ensembles consisting of a maximum of four different architectures (a total of 984 ensembles for each activation function) based on models from Experiments 1 – 5 and selected those that performed best based on the second half of the validation data.

5.3 TOWER AND CHECKERBOARD ARCHITECTURES

One of the contributions of this work is the introduction of a novel architecture (published in [9]) for gene expression inference, which leads to significant improvements in the quality of the inference. The baseline model is a modification of D–GEX with TAAFs [10] which consists of three hidden, densely connected layers with 10,000 neurons in each layer — the largest D–GEX architecture consisted of only 9,000 neurons in each layer [2] but adding more neurons has proved beneficial — and an output layer. Each neuron contains the TAAF with a sigmoid as the inner activation function as in [10]; each hidden layer is with 25% dropout.

5.3.1 *Tower architecture (T–D–GEX)*

Since the baseline model consists of three densely connected layers, a further increase in the number of neurons in each layer is difficult as the number of connections (weights) increases quadratically, and even the baseline model was near the memory limitations of the used GPU. Thus, similarly to PCs, we introduce towers of dense layers that are not connected to each other, which allows for a significant increase in the number of neurons without the increase in the number of weights. Unlike the PCs [1255, 1256], the output layer is not densely connected to all the towers, but rather the outputs of individual towers are first averaged, and only then an output layer is added — otherwise, the gains from the tower architecture would be much smaller as the number of connections between last hidden layer and the output layer would not change. The D–GEX with the tower architecture is denoted T–D–GEX, the number of neurons in a single layer of a tower was determined such that networks with more columns have strictly fewer weights (yet more neurons) as shown in Table 5.3 and Figs. 5.2 and 5.3.

5.3.2 *Checkerboard architecture (C–D–GEX)*

The checkerboard architecture can be seen as an extension of the tower architecture. The tower architecture consists of towers of densely connected layers where each layer is connected to the layer that precedes it; there is no information flow between the towers — the towers share the input layer, and then their outputs are averaged before the output layer. The checkerboard architecture addresses this issue and divides each layer of a tower into halves — each half is connected to the same half of the same tower of the preceding

towers	neurons/tower	neurons	parameters
1	10,000	34,759	257,149,036
2	7,227	48,121	257,119,561
3	5,941	58,228	257,068,283
4	5,157	66,643	256,997,503
5	4,615	73,984	256,977,621
6	4,211	80,557	256,953,201
8	3,637	92,047	256,729,407
10	3,242	102,019	256,587,674
12	2,948	110,887	256,374,168

Table 5.3: **Number of parameters of used tower architectures**
The summary of the parameterization of used architectures — C–D–GEX, CR–D–GEX, T–D–GEX, and TR–D–GEX do not differ in number of parameters and neurons. Note that the total number of parameters remains approximately the same across architectures.

layer; however, it is connected to the other half of the same tower only every odd layer while every even layer it is connected to the other half of the neighboring tower resulting in a checkerboard-like pattern of densely connected blocks. Both checkerboard architectures used in this paper have the first hidden layer without a dropout. The D–GEX with the checkerboard architecture is denoted C–D–GEX.

5.3.3 *Skip connections*

Another improvement was the addition of a skip connection in a ResNet-like manner [13] — we have added a residual skip connection from first to second hidden layer to each tower; the output of the first hidden layer is added to the output of second hidden layer before proceeding to the third hidden layer. The whole architecture with skip connections compared to the original D–GEX is shown in Fig. 5.4 where a checkerboard variant (see Section 5.3.2) was used. The tower architecture (see Section 5.3.1) with skip connections is equivalent. Such networks are denoted TR–D–GEX and CR–D–GEX, respectively.

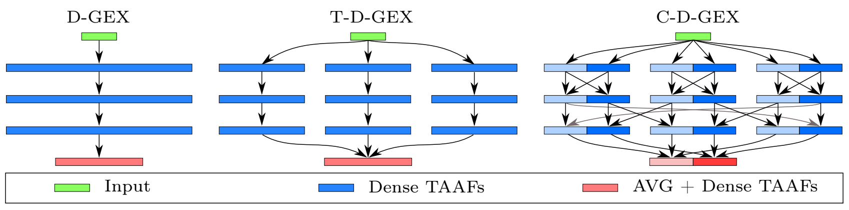


Figure 5.1: **Overview of used architectures**
The original D–GEX architectures and the novel architectures proposed in this paper. The outputs of the towers (or halves for C–D–GEX) are averaged before the output layer.

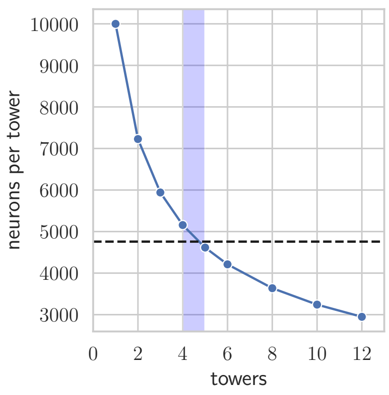


Figure 5.2: **Dependence of neurons per tower on number of towers**
The relationship between neurons per tower and number of towers when the number of weights is limited by the number of weights of a single tower with 10,000 neurons. The dashed line represents the number of neurons in the output layer, and the shaded region denotes the number of towers, for which the number of neurons in each layer of each tower is most similar to the number of output neurons.

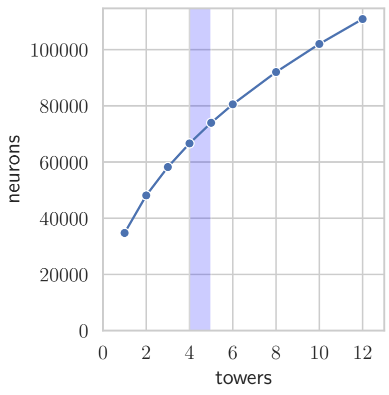


Figure 5.3: **Total number of neurons by the number of towers**
The relationship between the total number of neurons and the number of towers when the number of weights is limited by the number of weights of an architecture with a single tower with 10,000 neurons (i.e., the equivalent of the original D–GEX). The dashed line represents the number of neurons in the output layer, and the shaded region denotes the number of towers, for which the number of neurons in each layer of each tower is most similar to the number of output neurons.

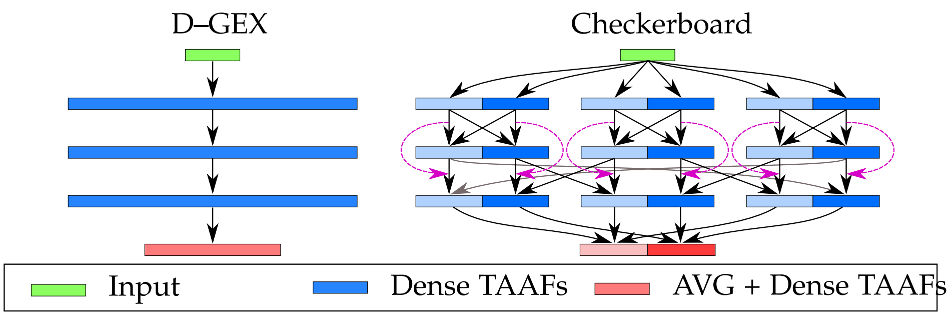


Figure 5.4: Diagram comparing the original D–GEX architectures and the checkerboard D–GEX architecture. The outputs of the towers are averaged before the output layer. Black arrows represent dense connections between blocks; purple dashed arrows represent skip connections.

5.4 IMPLEMENTATION

The work was implemented in python 3, the neural networks were implemented using the NN library *Keras* [209] and the computational framework *Tensorflow* [210]. Other packages used include *SciPy* [2129], *scikit-learn* [2130], *pandas* [2131], and *NumPy* [2132] for data manipulation and *Matplotlib* [2133] and *seaborn* [2134] for visualizations. The original implementation used in [10] is available at `https://github.com/kunc/TAAF-D-GEX`.

5.4.1 *Transformative adaptive activation function*

The TAAF was implemented by extending the Keras class *Layer*[2] as it allowed for seamless usage of the activation functions. To simplify the implementation, the TAAF split into several building blocks — two applications of a simpler linear transformation block called adaptive transformative unit (ATU) and a single application of the inner activation.

5.4.1.1 *Adaptive transformation unit*

The ATU is a basic building block of the TAAF. It has two adaptive parameters — a for linear scaling of the input and bias term b for translation:

$$\mathrm{ATU}(x) = a \cdot x + b. \tag{5.8}$$

The main part of the code of the ATU unit is then simple as:

```
class ATU(Layer):
    ...
    def call(self, inputs):
        outputs = self.alpha * inputs + self.beta
        return outputs
```

Listing 5.1: Core of implementation of the adaptive transformation unit

2 See the documentation at `https://keras.io/api/layers/base_layer/`.

5.4.1.2 *TAAF as the application of ATUs*

The TAAF itself can be formulated as one application of the ATU followed by an application of the inner activation function and finally followed by another ATU application. Using the functional API of Keras, it can be defined as

```python
def taaf(x, activation, name=""):
    """Transformative Adaptive Activation Function.
        It follows:
        ‘f(x) = alpha * f(beta*x + gamma) + delta‘,
        where f is a given activation function.
    """

    x = ATU(name=name + "TAAF_Bottom")(x)
    x = Activation(activation)(x)
    x = ATU(name=name + "TAAF_Top")(x)

    return x
```

Listing 5.2: TAAF implemented using ATUs

However, the implementation of TAAF using ATUs directly in a single function is a bit cumbersome to work with as it is not an extension of the Keras base Layer class. However, it is a similar concept but with more boilerplate code. The code below summarizes the implementation; some parts were omitted to highlight the core of the implementation; full code is available at `https://github.com/kunc/TAAF-keras`.

```python
class TAAF(Layer):
    def __init__(
        self,
        activation="tanh",
        alpha_initializer="ones",
        beta_initializer="ones",
        gamma_initializer="zeros",
        delta_initializer="zeros",
        ...
        **kwargs
    ):
        ...

        self.atu_bottom = ATU(
            alpha_initializer=beta_initializer,
            beta_initializer=gamma_initializer,
            ...
        )
        self.activation_layer = activations.get(activation)
        self.atu_top= ATU(
            alpha_initializer=alpha_initializer,
            beta_initializer=delta_initializer,
            ...
        )

        super(TAAF, self).__init__(**kwargs)
    ...
    def call(self, inputs):
        x = self.atu_bottom(x)
        x = self.activation_layer(x)
        x = self.atu_top(x)
        return x
```

Listing 5.3: TAAF implemented using ATUs

6 EXPERIMENTAL EVALUATION

After introducing TAAFs and tower and checkerboard architectures in previous Chapter 5, the goal of this chapter is to provide an empirical evaluation. First, we establish that TAAFs indeed improve the performance of the original D–GEX in Section 6.1, then we show that it has a practical measurable impact on subsequent analyses in Section 6.2, and then we show that TAAFs are also applicable outside the task of GE inference on several artificially generated multivariate regression datasets in Section 6.3. After establishing the performance of TAAFs, we show that the tower and checkerboard architectures further improve the performance of GE inference in Section 6.4 and that these improvements also have a practical impact on subsequent analyses in Section 6.5.

6.1 ESTABLISHING TAAF PERFORMANCE ON THE D-GEX MICROARRAY DATA

This set of experiments shows that the TAAFs improve the performance of the original NN model [2] for GE inference (see Section 5.1 for details about the task). We start by Section 6.1.1 Experiment 1: Usage of TAAFs where we show that just replacing the originally used tanh AFs by TAAFs also with tanh as the inner AF improves the performance of the NN without any further changes. Then, we further analyze the performance gains and show that additional modifications, such as replacing the activations in the last layer, improve the performance even further and that the same performance cannot be achieved by TAAFs without some of the parameters.

6.1.1 *Experiment 1: Usage of TAAFs*

The goal of this and the following experiments is to establish the improvement as a result of using the novel TAAFs in models trained on the full dataset. First, we compare the original D–GEX architectures equipped with the hyperbolic tangent (tanh) activation function to architectures equipped with the novel TAAF with hyperbolic tangent as the inner activation function. The results are shown in Tab 6.1, where the models are compared using the MMDAEs. The table shows the signed difference in absolute errors between the traditional hyperbolic tangent activation function and the adaptive activation function based on it — the novel transformative adaptive activation function was superior to the hyperbolic tangent activation function for all D–GEX architectures. Furthermore, the means (medians) of MMAEs for both models were significantly different using the paired Student's t-test (the Wilcoxon rank test) with $p < 0.0001$ for all D–GEX architectures tested.

TAAF tanh – tanh				
neurons	layers	MMDAE	95 % CI	
3,000	1	-0.016960	-0.017064	-0.016855
	2	-0.008421	-0.008472	-0.008370
	3	-0.015788	-0.015867	-0.015710
6,000	1	-0.018504	-0.018640	-0.018366
	2	-0.027463	-0.027548	-0.027376
	3	-0.041951	-0.042331	-0.041683
9,000	1	-0.020829	-0.021007	-0.020649
	2	-0.049515	-0.049631	-0.049394
	3	-0.063431	-0.063633	-0.063228

Table 6.1: **MMDAE summary TAAF tanh vs tanh**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the TAAF with tanh as inner activation function and classic tanh activation function for D–GEX with 25% dropout on the test data of the full dataset.

6.1.2 *Experiment 2: Replacing tanh with sigmoid activation function*

The performance of D–GEX with TAAF can be further improved by replacing the inner tanh activation function with a logistic sigmoid activation function. The comparison is shown in Tab 6.2. Since tanh is just a simple transformation of the logistic sigmoid, this can be thought of as a different initialization of the TAAF parameters, namely $\alpha := \frac{1}{2}$, $\beta := \frac{1}{2}$, and $\delta := \frac{1}{2}$. The original D–GEX benefited from the sigmoid activation more compared to the D–GEX with TAAFs (as shown in Tab 6.3), which shows that it is much more sensitive to the activation function used and that using the TAAFs adds some robustness to the model over different parameterizations. Furthermore, even the version of D–GEX with sigmoid activation function benefited significantly from the use of TAAFs, as presented in Tab 6.4.

6.1.3 *Experiment 3: TAAFs for capacity adjusted NNs*

The TAAFs introduce four additional parameters per neuron, which increase the capacity of the neural network, and the improvement might possibly be caused by the increase in the capacity. Indeed, it seems that increased capacity helps D–GEX as the architectures with more neurons have a lower prediction error for the same number of layers. We have reduced the number of neurons in each layer in the D–GEX with TAAFs such that the total number of parameters is the same as in the original D–GEX with the same architecture. The number of removed neurons was always lower than 30 as the number of added weights per neuron is insignificant compared to the number of weights of incoming connections. The improvement of the reduced D–GEX with TAAFs was from 0.0034 to 0.0068 across different D–

		TAAF sigmoid – TAAF tanh		
neurons	layers	MMDAE	95 % CI	
3,000	1	-0.005025	-0.005120	-0.004943
	2	-0.017574	-0.017725	-0.017420
	3	-0.010628	-0.010732	-0.010522
6,000	1	-0.004390	0-.004550	-0.004256
	2	-0.009468	-0.009560	-0.009376
	3	-0.002024	-0.002214	-0.001706
9,000	1	-0.004043	-0.004281	-0.003853
	2	-0.010392	-0.010511	-0.010271
	3	-0.001712	-0.001803	-0.001615

Table 6.2: **MMDAE summary TAAF tanh vs TAAF sigmoid**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the TAAF with tanh and sigmoid as inner activation functions for D–GEX with 25% dropout on the test data of the full dataset.

		sigmoid – tanh		
neurons	layers	MMDAE	95 % CI	
3,000	1	-0.018551	-0.018781	-0.018345
	2	-0.022399	-0.022543	-0.022253
	3	-0.021952	-0.022112	-0.021786
6,000	1	-0.018294	-0.018676	-0.017980
	2	-0.033569	-0.033709	-0.033429
	3	-0.038359	-0.038547	-0.038164
9,000	1	-0.019727	-0.020284	-0.019274
	2	-0.055361	-0.055534	-0.055192
	3	-0.058344	-0.058559	-0.058129

Table 6.3: **MMDAE summary sigmoid vs tanh**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the sigmoid and tanh activation functions for D–GEX with 25% dropout on the test data of the full dataset.

GEX architectures. The complete comparison of the reduced D–GEXs with the adaptive activation function based on sigmoid and the original D-GEXs is shown in Tab 6.5. We can observe that the reduction in the number of neurons had, as expected, only a small effect and that the network with TAAFs still significantly outperforms the original D–GEX. As this effect is negligible, most of the experiments throughout this work use identical architectures regarding the number of neurons, omitting this correction based on the number of trainable weights.

TAAF sigmoid – sigmoid				
neurons	layers	MMDAE	95 % CI	
	1	-0.003434	-0.003523	-0.003330
3,000	2	-0.003595	-0.003636	-0.003555
	3	-0.004464	-0.004508	-0.004419
	1	-0.004600	-0.004748	-0.004401
6,000	2	-0.003362	-0.003407	-0.003315
	3	-0.005617	-0.005674	-0.005563
	1	-0.005145	-0.005360	-0.004860
9,000	2	-0.004546	-0.004598	-0.004491
	3	-0.006799	-0.006876	-0.006724

Table 6.4: **MMDAE summary TAAF sigmoid vs sigmoid**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the TAAF with sigmoid as inner activation function and sigmoid activation function for D–GEX with 25% dropout on the test data.

TAAF sigmoid (reduced) – sigmoid					
neurons	layers	reduced	MMDAE	95 % CI	
	1	2990	-0.003384	-0.003476	-0.003279
3,000	2	2,997	-0.003503	-0.003543	-0.003464
	3	2,997	-0.004452	-0.004493	-0.004410
	1	5980	-0.004599	-0.004746	-0.004409
6,000	2	5,997	-0.003685	-0.003732	-0.003637
	3	5,997	-0.005680	-0.005734	-0.005627
	1	8971	-0.005130	-0.005346	-0.004849
9000	2	8,997	-0.004139	-0.004199	-0.004077
	3	8,997	-0.006811	-0.006882	-0.006740

Table 6.5: **MMDAE summary TAAF sigmoid (reduced) vs sigmoid**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the TAAF with sigmoid as inner activation function and sigmoid activation function for D–GEX with 25% dropout on the test data of the full dataset. The network with TAAF had a reduced number of neurons such that both networks had the same number of parameters — the final number of neurons in each layer is shown in the column *reduced*.

6.1.4 *Experiment 4: Importance of individual parameters*

To verify that all TAAF parameters improve performance, we trained neural networks with constrained TAAFs that had some of the parameters removed. We evaluated all 16 subsets of TAAF parameters (from the reduced TAAF equivalent to traditional sigmoid activation function to full TAAF with all four adaptive parameters) using three-layered D–GEX architecture with 6000 neurons in each layer. The networks with different subsets of TAAF parameters were pairwise evaluated based on MMDAE. Fig 6.1 shows the MMDAEs between all model pairs while Fig 6.2 shows whether model A (row) is significantly better than model B (column) based on the paired Wilcoxon rank test on samplewise MAEs at significance level $\alpha = 0.001$. The full TAAF is significantly better than all other combinations of parameters. This shows that the proposed TAAF with four parameters is the correct choice and that it outperforms the other adaptive activation functions it generalizes.

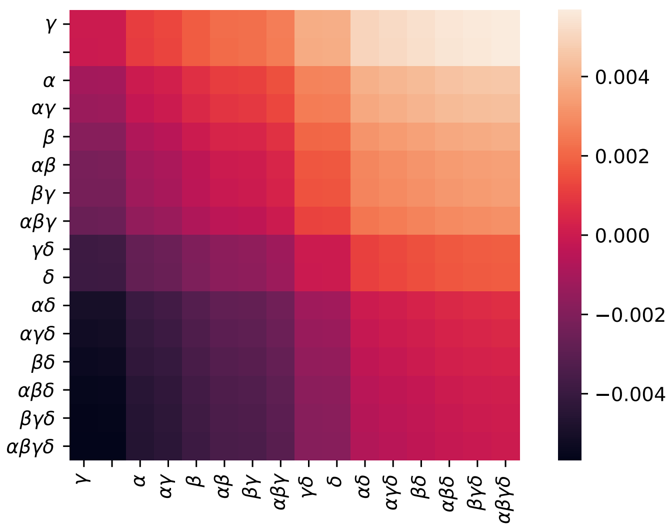


Figure 6.1: **MMDAE for different parameterizations**
The average mean difference in absolute errors for different model pairs. The model labels specify which adaptive parameters were used in the TAAF (e.g., $\alpha\beta$ means adaptive parameters α and δ were used).

6.1.5 *Experiment 5: TAAF in the output layer*

The networks with TAAF do not require the output layer to contain a linear activation function for regression tasks as the TAAF allows for scaling and translation. Using TAAFs in the output layer might lead to better performance, as shown in Tab 6.6, where networks with TAAFs in the output layer are compared with networks with a linear output layer. The usage of TAAFs in the output layer was beneficial for all architectures tested.

6.1.6 *Experiment 6: heterogeneity-aware data sampling*

We have also run an experiment comparing plain D–GEX and D–GEX with TAAFs (TAAFo) on the heterogeneity-aware data splits. The main focus of this experiment was to verify whether the possible bias due to information leakage between training and testing sets due to random splits in Experiments 1 – 5 is significant (if present at all).

Tab 6.7 shows the relative comparison of plain D–GEX and D–GEX with TAAFs (TAAFo) for networks with sigmoid and hyperbolic tangent inner

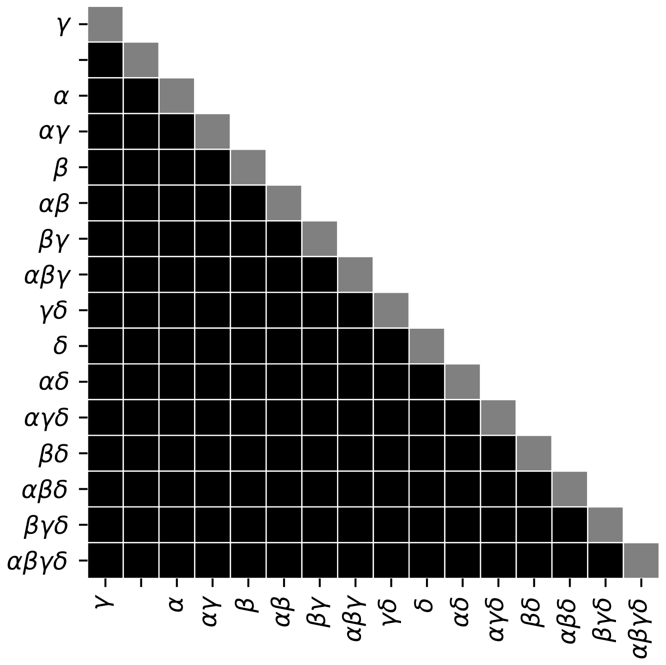


Figure 6.2: **Wilcoxon rank test for different parameterizations**
Testing for significant differences in samplewise errors — a cell is black if the model on the y-axis has a significantly lower MAE based on the paired Wilcoxon rank test at significance level 0.001 than a model on the x-axis. The model labels specify which adaptive parameters were used in the TAAF (e.g., $\alpha\beta$ means adaptive parameters α and δ were used).

TAAFo sigmoid – TAAF sigmoid

neurons	layers	MMDAE	95 % CI	
3,000	1	-0.000401	-0.000472	-0.000308
	2	-0.001015	-0.001091	-0.000945
	3	-0.001896	-0.001951	-0.001843
6,000	1	-0.000679	-0.000789	-0.000531
	2	-0.001654	-0.001718	-0.001591
	3	-0.002474	-0.002521	-0.002428
9,000	1	-0.000919	-0.001075	-0.000711
	2	-0.001864	-0.001935	-0.001796
	3	-0.001426	-0.001477	-0.001377

Table 6.6: **MMDAE summary TAAFo sigmoid vs TAAF sigmoid**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the TAAF with sigmoid as inner activation function and 25% dropout on the test data. `TAAFo sigmoid` denotes a network that contains TAAFs in the output layer while `TAAF sigmoid` uses a linear activation in the output layer.

activation functions. The networks with TAAFs performed statistically significantly better than the plain D–GEX even on this dataset, which had only

$\approx$ 60% of the training samples compared to the whole dataset (some samples from the originally provided data were missing the GEO- id and could not be used). This result demonstrates that the above-mentioned bias does not affect the comparative analyses in our manuscript.

		TAAFo sigmoid – sigmoid			TAAFo tanh – tanh		
neurons	layers	MMDAE	95 % CI		MMDAE	95 % CI	
3,000	1	-0.005339	-0.005449	-0.005221	-0.005927	-0.006007	-0.005847
	2	-0.006263	-0.006329	-0.006198	-0.021192	-0.021282	-0.021103
	3	-0.009114	-0.009185	-0.009042	-0.019252	-0.019341	-0.019160
6,000	1	-0.007214	-0.007338	-0.007082	-0.005080	-0.005174	-0.004990
	2	-0.005941	-0.006012	-0.005870	-0.005400	-0.005474	-0.005326
	3	-0.011624	-0.011709	-0.011539	-0.010166	-0.010256	-0.010071
9,000	1	-0.006664	-0.006785	-0.006539	-0.005402	-0.005515	-0.005293
	2	-0.006514	-0.006589	-0.006439	-0.007455	-0.007538	-0.007368
	3	-0.011349	-0.011446	-0.011250	-0.011921	-0.012038	-0.011806

Table 6.7: **Comparison of MMDAE of TAAFo sigmoid vs sigmoid and TAAFo tanh vs tanh**
The MMDAE and its 95 % CI estimated using bootstrap on samplewise MDAEs for the TAAF with sigmoid/tanh as inner activation function and sigmoid/tanh activation function for D–GEX with 25% dropout on the test set of the heterogeneity-aware sampled data.

6.1.7 *Overall comparison*

The best single network performs much better than our reimplementation of the original D–GEX — the MMAE of the best network ($3 \times 9,000$ TAAFo with sigmoid) is 0.1340 (the 95% CI estimated over samples is $[0.13316, 0.13486]$) compared to D–GEX with tanh activation function with an MMAE of 0.1637 (95% CI $[0.16279, 0.16458]$). Our proposed network performs better in 18,849 (99.75%) samples while worse in only 2 (0.001%) samples when the MAEs over genes for individual samples are compared using the paired Wilcoxon rank test at significance level $\alpha = 0.0001$.

All improvements to the original D–GEX are depicted in Fig 6.3, which shows the improvement of individual modifications. Fig 6.4 summarizes the individual improvements over the basic D–GEX with our proposed activation function. Tab 6.8 shows the absolute performance of the top ten D–GEXs.

6.2 PRACTICAL IMPACT OF TAAFS ON DIFFERENTIAL GENE EXPRESSION ANALYSIS ON THE D-GEX MICROARRAY DATA

To demonstrate that lowering the inference error established in Section 6.1 has a practical impact on applied tasks, we ran differential gene expression analyses as described in Section 5.1.5.1. We started with the artificial pheno-

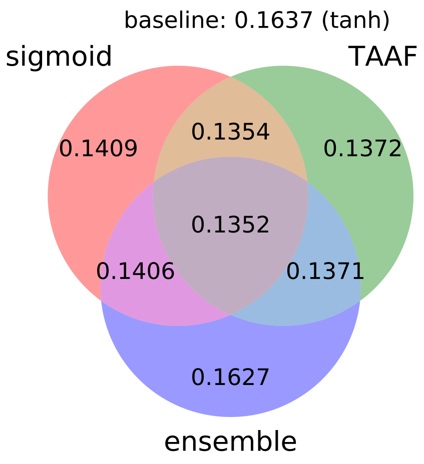


Figure 6.3: **Individual components of improvement**
A diagram depicting the performance for individual improvements over the standard D–GEX baseline with tanh activation function. The diagram shows the best MMAE over all D–GEX architectures for a given approach trained on the full dataset.

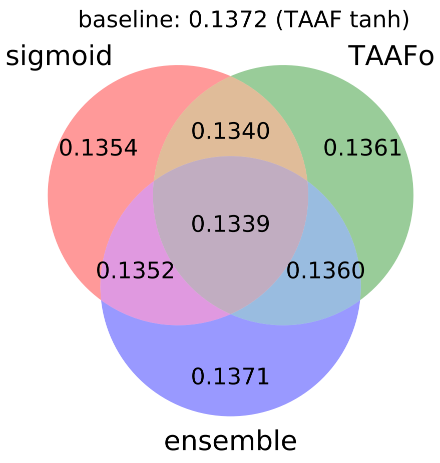


Figure 6.4: **Individual components of improvement including TAAFo**
A diagram depicting the performance for individual improvements over the D–GEX baseline already equipped with transformative adaptive activation functions. The diagram shows the best MMAE over all D–GEX architectures for a given approach trained on the full dataset.

types introduced by clustering; then, we continued with the real phenotypes taken from the original annotation available for a particular data series.

6.2.1 *Artificial phenotypes*

First, we ran the DGE analysis using the best model from Experiment 1, which contained models trained on the full dataset, using the artificial phenotypes. The randomly sampled balanced datasets had sizes ranging from 12 to 600, which is the usual sample size range for DGE analyses. The distribution

rank	MMAE	neurons	layers	type	activation
1	0.134015	9,000	3	TAAFo	sigmoid
2	0.134503	9,000	2	TAAFo	sigmoid
3	0.135430	8,997	3	TAAF (reduced)	sigmoid
4	0.135442	9,000	3	TAAF	sigmoid
5	0.136064	9,000	2	TAAFo	tanh
6	0.136367	9,000	2	TAAF	sigmoid
7	0.136774	8,997	2	TAAF (reduced)	sigmoid
8	0.136883	9,000	3	TAAFo	tanh
9	0.137154	9,000	3	TAAF	sigmoid
10	0.137189	6,000	3	TAAFo	sigmoid

Table 6.8: **10 best D–GEX architectures**
The 10 best D–GEX architectures in terms of MMAE on the test data of the full dataset. 25% dropout was used.

of values and the pairwise differences of F_1 score is shown in Fig 6.5 and Fig 6.6 for 5,000 repetitions for each sample size. The differences in all scores ($F_{0.5}$, $F_1 - F_{10}$ scores, accuracy, and MCC) were statistically significant for all sample sizes tested when using the Wilcoxon signed-rank test as all the p–values were $< 10^{-8}$.

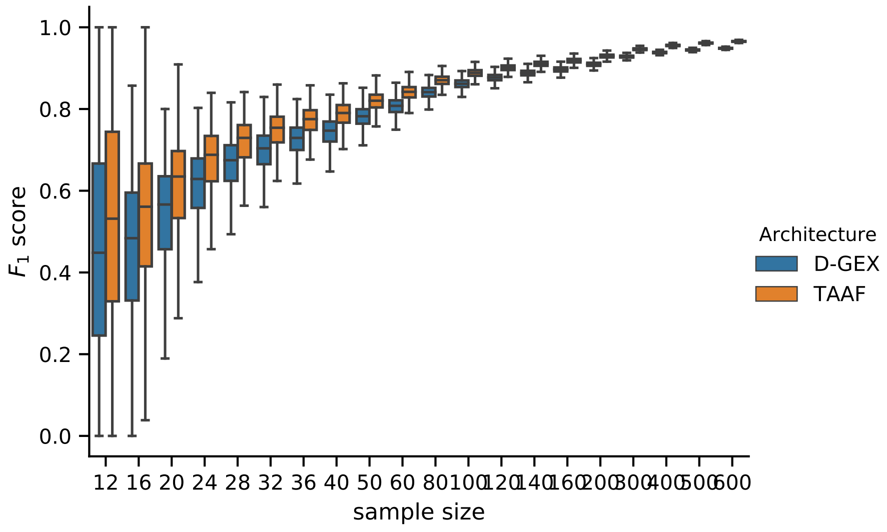


Figure 6.5: **Distribution of F_1 scores**
Distribution of the F_1 scores obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size. The whiskers show the 10th and 90th percentiles.

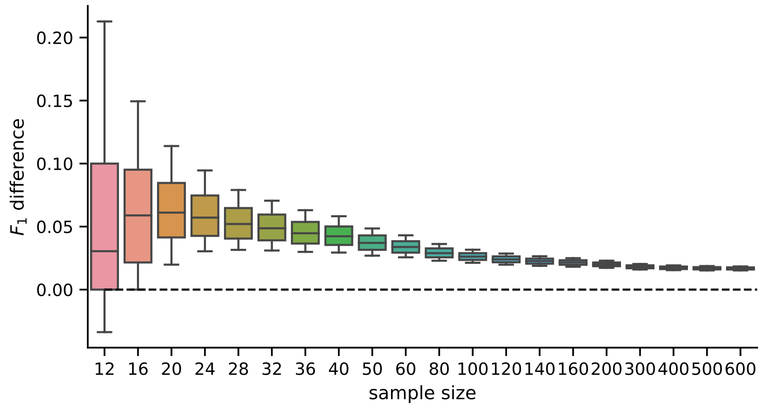


Figure 6.6: **Distribution of differences in F_1 scores**
Distribution of pairwise differences in the F_1 score obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size. The whiskers show the 10th and 90th percentiles.

6.2.1.1 *Impact on candidate rankings*

We also analyzed how the candidates' ranking for differentially expressed genes differs between the ground truth data and the inferred data using both methods. For the second part of the analysis, we selected 100 candidates for differentially expressed genes (i.e., genes ranked 1 – 100 by their statistical significance) and compared their ranks when using the inferred data by both methods. The MAE of the rankings for both methods is shown in Fig 6.7, and the pairwise difference in Fig 6.8. The difference was statistically significant for all sample sizes tested when using the Wilcoxon signed-rank test with significance level $\alpha = 10^{-8}$. Both rankings were becoming more similar to the ground truth candidate ranking with increasing sample sizes in general (the selections of the first 100 candidates are becoming more and more conservative with increasing sample size). However, the candidate rankings produced using data inferred by D–GEX with TAAFs were closer to the ground truth rankings.

For the third part of the analysis, we selected candidates for differentially expressed genes as those genes for whose the p–value was lower than the significance level $\alpha = 0.05$. The MAE of the rankings for both methods is shown in Fig 6.9 and the pairwise difference in Fig 6.10. The MAE here is generally higher than in the case of the selection of only the first 100 candidates, but that is because the number of candidates selected with the threshold $\alpha = 0.05$ is higher and increases with the sample size. The increased accuracy of the D–GEX with TAAFs impacts the ranking for larger sizes as it obviously represents the expression data more faithfully, and thus, the rankings are more similar to the ground truth data compared to the plain D–GEX.

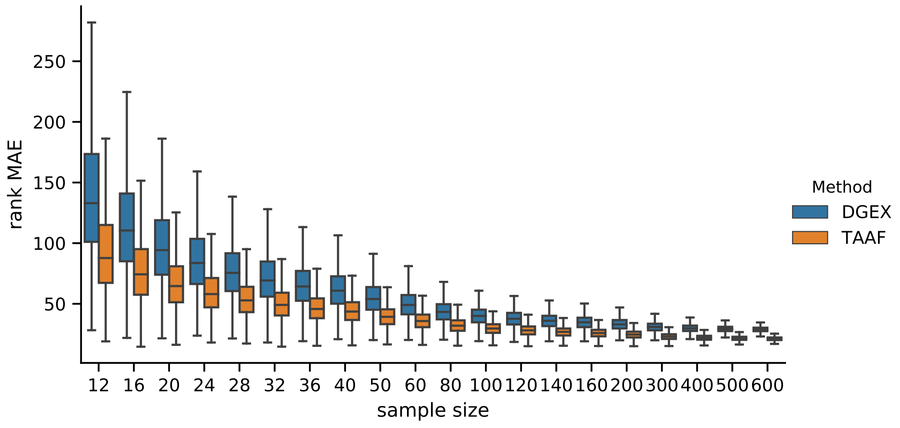


Figure 6.7: **Distribution of MAEs (first 100)**
Distribution of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the first 100 candidates for DE genes selected on the sampled ground truth data. The whiskers show the 10th and 90th percentiles.

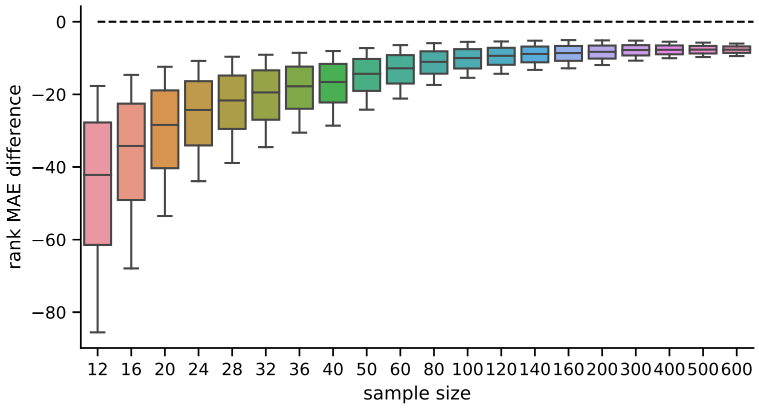


Figure 6.8: **Distribution of differences in MAEs (first 100)**
Distribution of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the first 100 candidates for DE genes selected on the sampled ground truth data. The whiskers show the 10th and 90th percentiles.

6.2.2 *Real phenotypes*

We ran the DGE analyses as in the previous experiment again; however, this time using real phenotypes and model trained on the heterogeneity-aware dataset (the same as in Experiment 6: heterogeneity-aware data sampling in Section 6.1.6) to show that the performance difference is not due to any potential information leakage to the test set and that the DGE analysis performance difference is present even for actual phenotypes. We used $3 \times 9,000$ architectures with hyperbolic tangent and sigmoid inner activation functions for both the plain D–GEX and D–GEX with TAAFs. During the sampling procedure, we ensured that the GSE2109 series was present in its

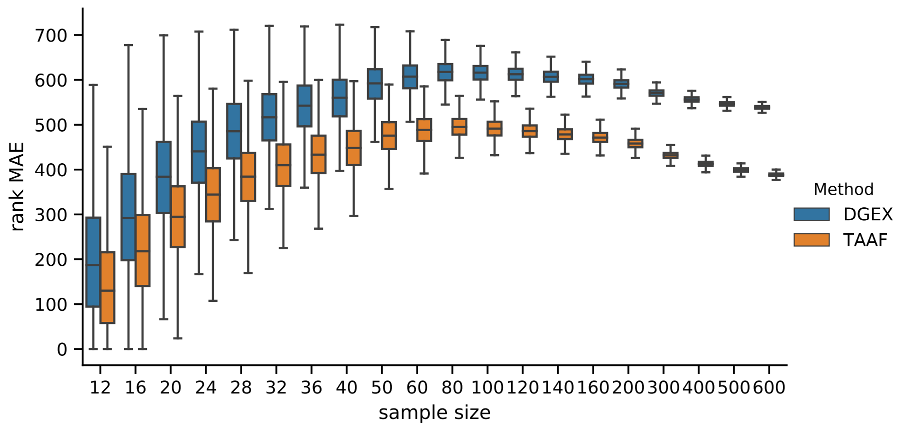


Figure 6.9: **Distribution of MAEs (p-value based)**
Distribution of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the DE candidates selected on the sampled ground truth data at significance level $\alpha = 0.05$. The whiskers show the 10th and 90th percentiles.

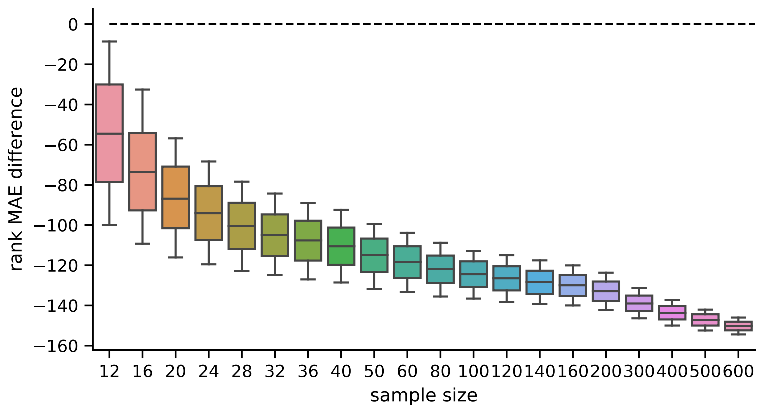


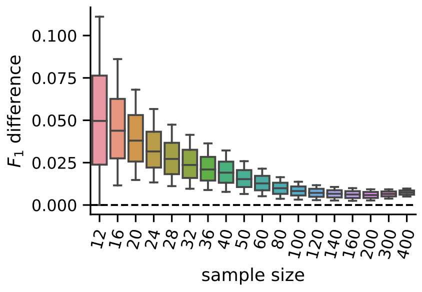
Figure 6.10: **Distribution of differences in MAEs (p-value based)**
Distribution of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the DE candidates selected on the sampled ground truth data at significance level $\alpha = 0.05$. The whiskers show the 10th and 90th percentiles.

entirety in the test data and used it for the DGE analysess. The GSE2109 series consists of samples from different tissues; these tissues were used for phenotype classes for the DGE analyses. We selected classes that had more than 100 samples and ended up with six classes, as shown in Tab. 6.9. We then ran a DGE analysis for every pair combination, resulting in 15 analyses. The sampled balanced dataset sizes ranged from 12 to 200 – 400 depending on the class size for the tissue; the actual maximum sample size for a particular class is shown in Tab. 6.9. The sampled datasets were balanced; thus, the smaller maximum sample size limit was used as the limit for the pair of classes.

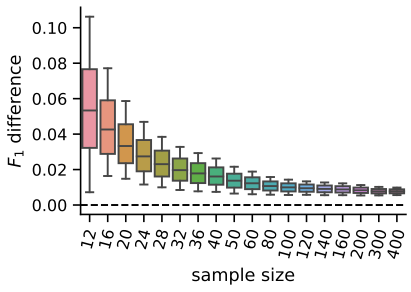
The distributions the pairwise differences of the F_1 score are shown in Fig 6.11, larger plots are available in the appendix in Fig. A.1 together with plots of distributions of the pairwise differences of other metrics — $F_{0.5}$ in Fig. A.2, F_2 in Fig. A.3, MCC in Fig. A.4, and accuracy in Fig. A.5. The differences in the accuracy, F_1, $F_{0.5}$, F_2, and MCC scores were also tested using the Wilcoxon signed-rank test with significance level $\alpha = 10^{-8}$. The D–GEX with TAAFs statistically significantly outperformed the plain D–GEX for most of the tasks and sample sizes, detailed results are shown in Fig 6.12 — the only exception is the *Ovary* $\times$ *Uterus* task, where both models performed very similarly for small sample sizes and no statistically significant performance difference was observed at the given significance level.

tissue	# samples	max sample size
Breast	351	200
Colon	292	200
Kidney	279	200
Ovary	198	150
Uterus	136	100
Lung	132	100

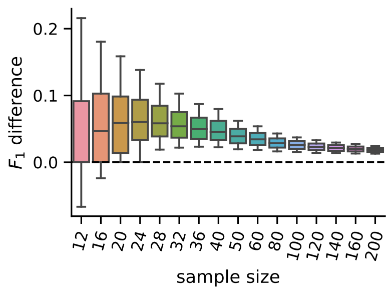
Table 6.9: **Overview of sample sizes of the GSE2109 series**
The number of samples for each tissue in the GSE2109 series.



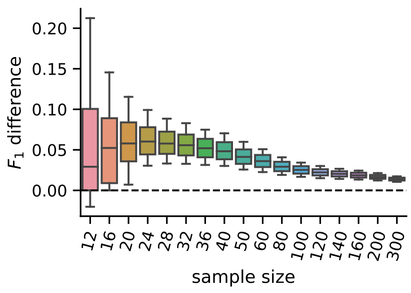
(a) Breast × Colon



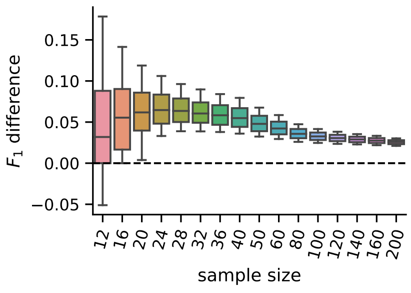
(b) Breast × Kidney



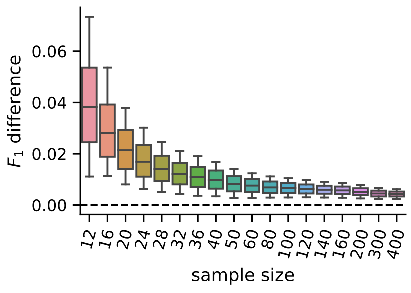
(c) Breast × Lung



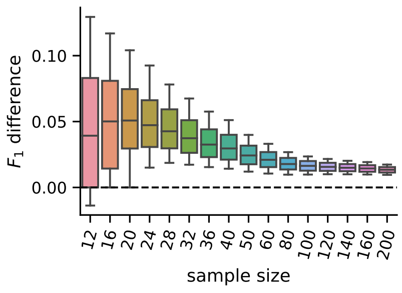
(d) Breast × Ovary



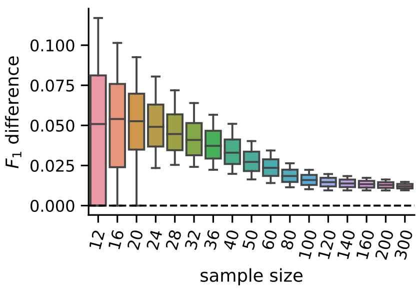
(e) Breast × Uterus



(f) Colon × Kidney

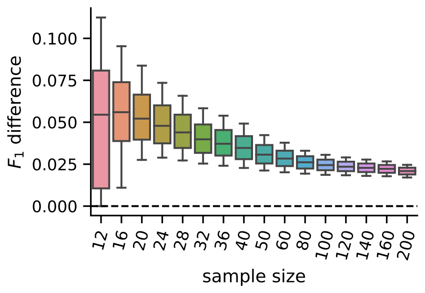


(g) Colon × Lung

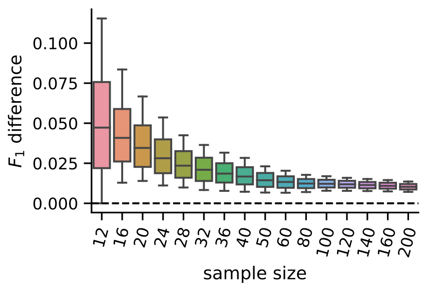


(h) Colon × Ovary

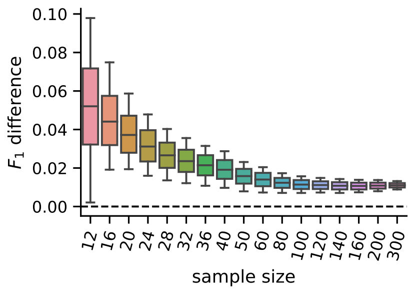
Figure 6.11: **Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles.



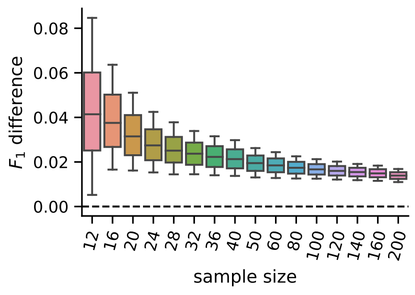
(i) Colon × Uterus



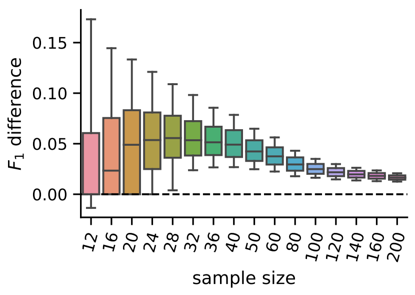
(j) Kidney × Lung



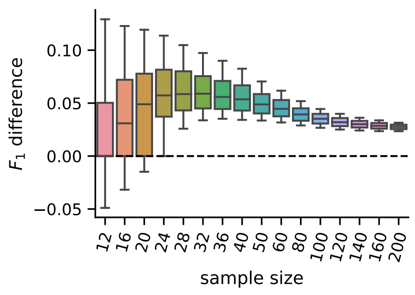
(k) Kidney × Ovary



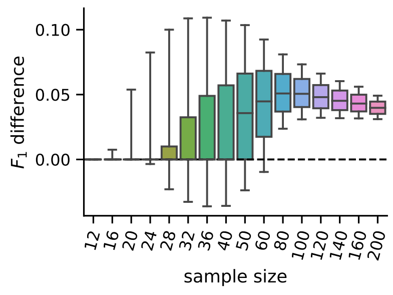
(l) Kidney × Uterus



(m) Lung × Ovary



(n) Lung × Uterus



(o) Ovary × Uterus

Figure 6.11: **(cont.) Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. 6.11.

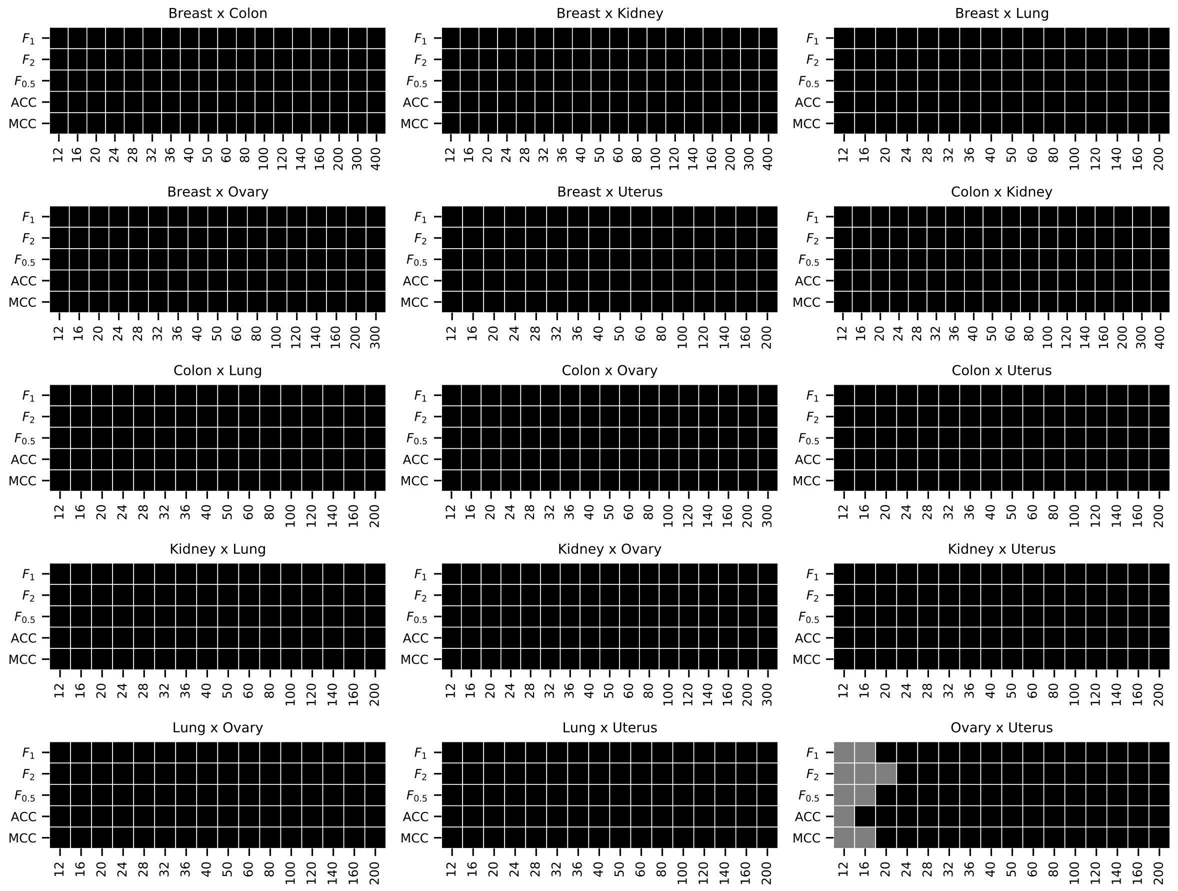


Figure 6.12: **Results of the Wilcoxon test for individual tasks pairs**
The results of the metrics pairwise comparison using the Wilcoxon signed-rank test with significance level $\alpha = 10^{-8}$. The cell is colored black if the D–GEX with TAAFs performance for the metric was statistically significantly better than the plain D–GEX, white if the plain D–GEX performed better, and grey if no statistically significant difference was found.

6.2.2.1 *Impact on candidate rankings*

The analysis of the impact on candidate rankings for the differentially expressed genes was also run for all 15 tissue pairs. First, we analyzed the difference in the rankings of the first 100 candidate genes selected by the DGE analysis using ground truth data. The results for individual tasks and sample sizes are shown in Fig. 6.13. Second, we analyzed the difference in the rankings of the candidate genes whose p-value from the DGE analysis on the ground truth data (for the particular sample) was above $\alpha = 0.05$ (leading to non-constant sizes of the candidate sets); the results are shown in Fig 6.14. The differences in MAE of the rank differences were tested using the Wilcoxon signed-rank test with significance level $\alpha = 10^{-8}$. The D–GEX with TAAFs was statistically significantly better for both candidate selection methods, all tasks, and all sample sizes with p-value $< 10^{-8}$. Therefore, it is safe to conclude that if there is some bias in the performance of the models on the full dataset, it is not significant for the model comparison, and the experiments and models trained on the full dataset are valid.

6.3 EXPLORING TAAF PERFORMANCE USING ARTIFICIAL DATA

This set of experiments shows that networks with TAAFs generally outperform the baseline for various architecture variants and parameterizations of D–GEX like networks. We have run four similar experiments using different parameterizations of the data generation networks as summarized in Table 6.10 (see Section 5.1.2 for a more general overview of data generation). The other parameters were the same for all of the data generation setups — the input dimension was 1,000, and the data were sampled from a normal distribution with zero mean and standard deviation as denoted in Table 6.10 (either 1 or 2). The output dimension was 5,000, and 50,000 samples were generated for each of the data sets (train, validation, and test). A noise was added to the resulting outputs from the network — depending on the variant, a normal noise with zero mean and standard deviation 0, 0.1, 0.5, 1.0, or 2.0 was used. Used generative networks were initialized using the Glorot initializer for weights and zeros for the bias term.

The experiments focused on the differences between networks with TAAFs and the baseline with respect to different activation functions of the inference network, different sizes, and also the sensitivity of learning due to the amount of noise applied to the dependent variable. Due to limited computational resources, not all of the subexperiments evaluated the same set of parameters but rather focused on slightly different ranges of parameters. Regarding the sizes of the inference networks, networks either with one or two hidden layers with 3,000 neurons or networks with one to three hidden layers with 1,000 neurons were used for subexperiments NN1 and NN2; the subexperiments NN3 and NN4 were also run with larger inference networks — one to three hidden layers either with 1,000, 3,000 or 6,000 neurons each. The inference networks in all four subexperiments used all either sigmoid, swish or hyperbolic tangent activation function and were without dropout or with

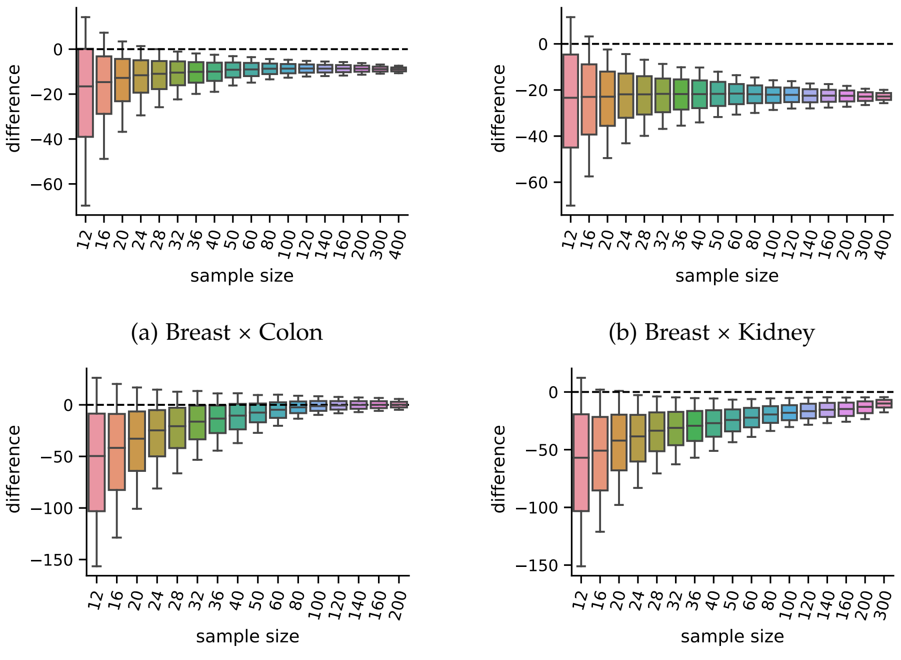
25 % dropout. The used values of individual parameterization of inference networks for each task are summarized in Table 6.11.

subexp. code	input std.	AF	hidden l. config.
NN1	1.0	sigmoid	1000-1000-1000
NN2	1.0	sigmoid	3000-5000-5000-5000
NN3	2.0	swish	1000-1000-1000
NN4	2.0	swish	3000-5000-5000-5000

Table 6.10: **Parameterization of data generation networks**
Used distinct parameterizations of data generation networks in experiments with artificial data.

6.3.1 *The general performance comparison*

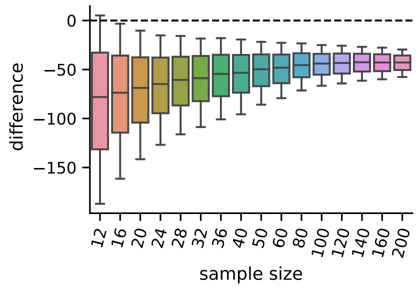
The networks with TAAFs generally performed better over the evaluated parameterizations. The OOS performance on the test set with the model checkpoint that has the lowest loss is shown in Table 6.12 while Table 6.13 shows the performance on the training data with the model from the last epoch.



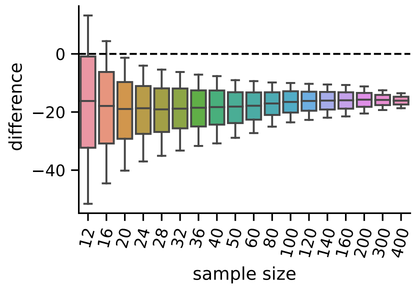
(a) Breast × Colon (b) Breast × Kidney

(c) Breast × Lung (d) Breast × Ovary

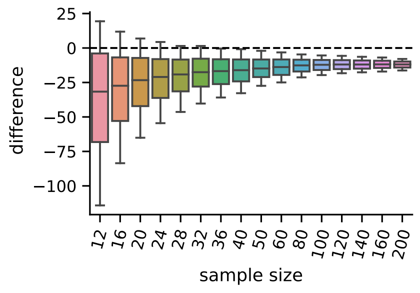
Figure 6.13: **Distributions of differences in MAEs (first 100) for the real phenotype**
Distributions of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the first 100 candidates for DE genes selected on the sampled ground truth data for different tissues. The whiskers show the 10th and 90th percentiles.



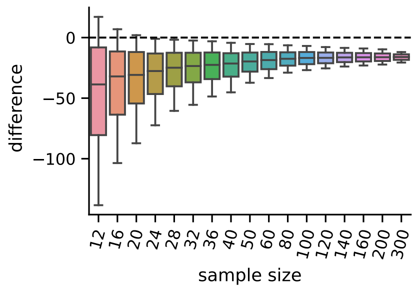
(e) Breast × Uterus



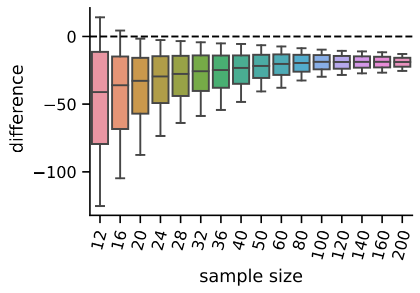
(f) Colon × Kidney



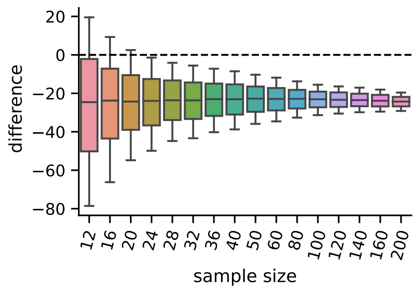
(g) Colon × Lung



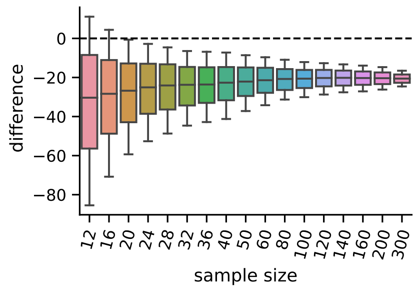
(h) Colon × Ovary



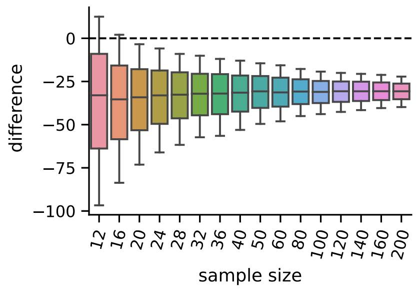
(i) Colon × Uterus



(j) Kidney × Lung

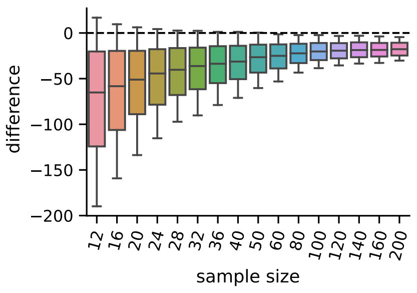


(k) Kidney × Ovary

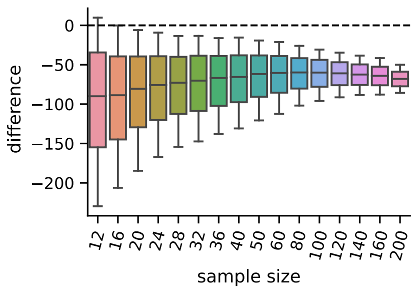


(l) Kidney × Uterus

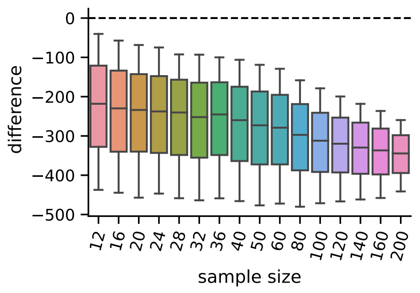
Figure 6.13: **(cont.) Distributions of differences in MAEs (first 100) for the real phenotype**
Distributions of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the first 100 candidates for DE genes selected on the sampled ground truth data for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. 6.13.



(m) Lung × Ovary



(n) Lung × Uterus



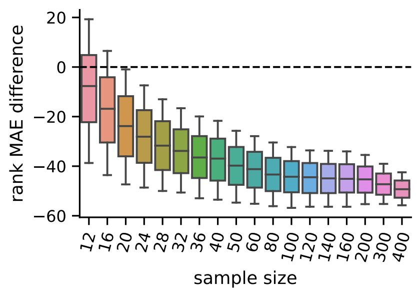
(o) Ovary × Uterus

Figure 6.13: **(cont.) Distributions of differences in MAEs (first 100) for the real phenotype**
Distributions of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the first 100 candidates for DE genes selected on the sampled ground truth data for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. 6.13.

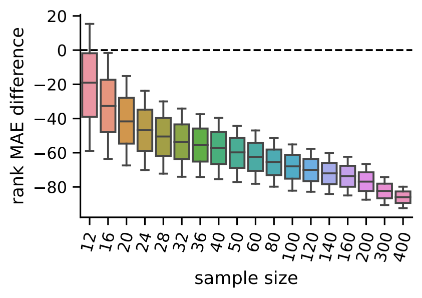
The networks with TAAFs are generally better in terms of statistically significantly lower MAEs of individual predictions when compared with the baseline. However, the usage of the TAAFs increases the learning capabilities of the network, and as such, these networks are more prone to overfitting as shown in Table 6.13 where the relative performance on the train set of the model checkpoint from the last epoch is shown — the networks with TAAFs perform similarly or slightly worse compared to the baseline on the noiseless targets as they were fitted to the noisy targets for which they show significantly better performance (due to overfitting). This phenomenon is also clear when the relative performance is broken by individual noise levels of the target as shown in Fig. 6.15 where the network with TAAFs performs consistently on the test set when the model checkpoint is selected on the validation set but the TAAFs dominance tend to decrease with increasing the noise levels on the training set with model from the last epoch when the performance is evaluated using the noiseless targets (i.e., the networks were trained using the noisy targets but evaluated using the noiseless targets) but there is no noteworthy degradation over the noisy targets as the networks with TAAFs overfitted more to the noisy targets compared to the baseline.

6.3.2 *Target noise variance's impact on performance*

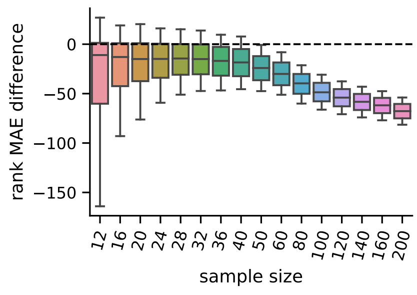
As shown in Fig. 6.15, the dominance of the networks with TAAFs over the baseline is not much influenced by the amount of the noise added to the target prior the training — with the exception the quality of prediction of the noiseless targets when learned on the noisy targets as TAAFs tend to overfit more than the baseline due to their higher learning capacity and as such increasing noise lead to significant drops in relative performance compared to the baseline. Nevertheless, the OOS performance on the *test* data of the networks with TAAFs is consistently better than the baseline, and therefore, the overfitting is not an issue when selecting the checkpoint performing the best on the *validation* set. The comparison of absolute performance of networks with TAAFs and the baseline is shown in Fig. 6.16, which shows the mean MMAE over relevant parameterization evaluated on the noiseless targets. The Fig. 6.17 shows the same information for the Noiseless target as Fig. 6.16 but since it shows the mean MMAE relative to the training error of the baseline, it shows the relationship not distorted by the MMAE component due to the noise which would make the plot unreadable. The error, in general, rises with higher amounts of noise of the targets as overfitting occurs (the in-sample error of the noiseless targets is higher than the unbiased oos error



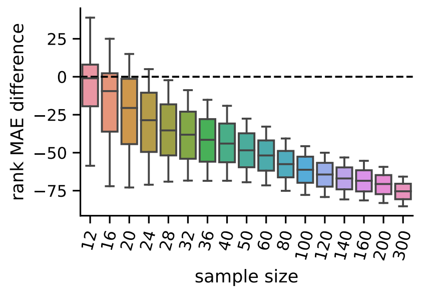
(a) Breast × Colon



(b) Breast × Kidney

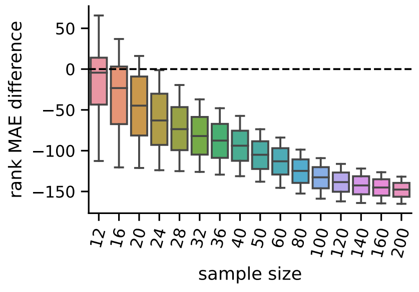


(c) Breast × Lung

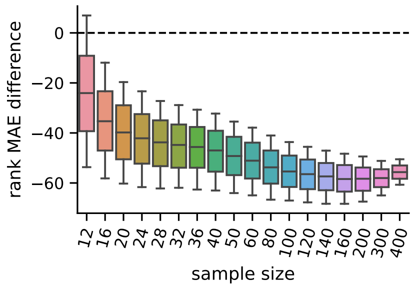


(d) Breast × Ovary

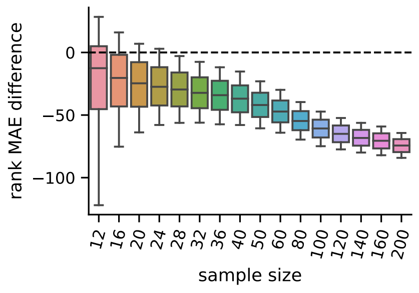
Figure 6.14: **Distributions of differences in MAEs (p-value based) for the real phenotype**
Distributions of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the DE candidates selected on the sampled ground truth data at significance level $\alpha = 0.05$ for different tissues. The whiskers show the 10th and 90th percentiles.



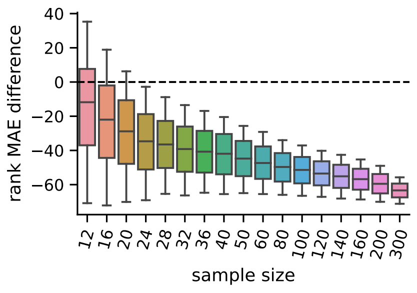
(e) Breast × Uterus



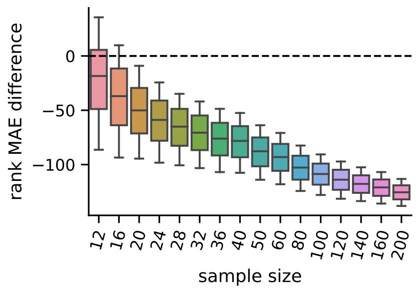
(f) Colon × Kidney



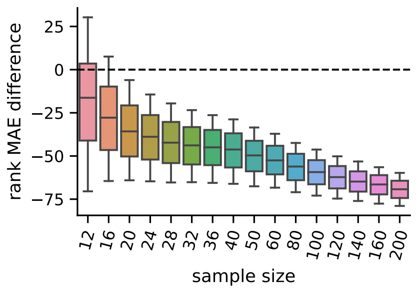
(g) Colon × Lung



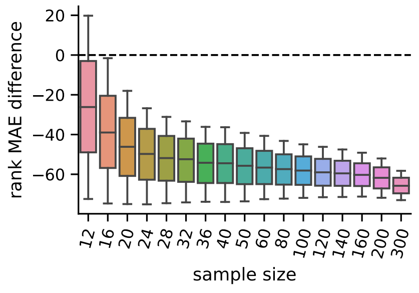
(h) Colon × Ovary



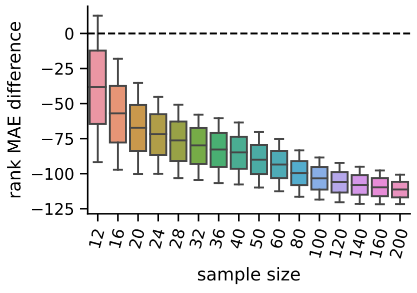
(i) Colon × Uterus



(j) Kidney × Lung

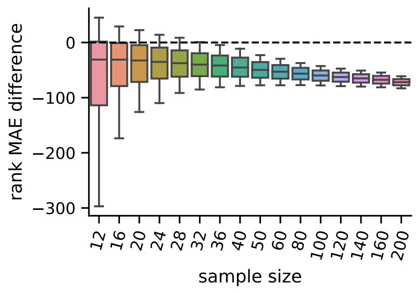


(k) Kidney × Ovary

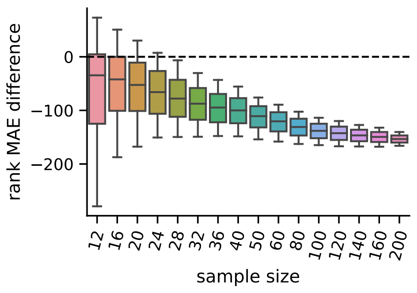


(l) Kidney × Uterus

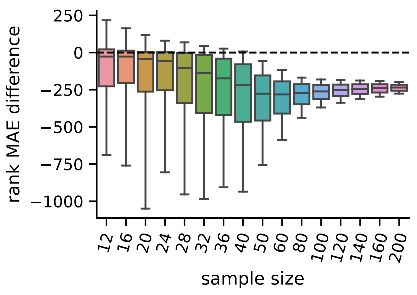
Figure 6.14: **(cont.) Distributions of differences in MAEs (p-value based) for the real phenotype**
Distributions of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the DE candidates selected on the sampled ground truth data at significance level $\alpha = 0.05$ for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. 6.14.



(m) Lung × Ovary



(n) Lung × Uterus



(o) Ovary × Uterus

Figure 6.14: **(cont.) Distributions of differences in MAEs (p-value based) for the real phenotype**
Distributions of pairwise differences of the MAE on rankings obtained by the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for the DE candidates selected on the sampled ground truth data at significance level $\alpha = 0.05$ for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. 6.14.

while being lower for the noisy targets that were used for training as shown in Fig. 6.17) for both baseline and TAAF-based networks and the task gets harder as more of the information required for training is overlaid by the white noise (the noiseless error is increasing with higher amounts of noise as shown in Fig. 6.16).

6.3.3 *Performance impact of layer configuration of the inference network*

The inference networks consist, similarly to the DGEX [2], of several densely connected hidden layers. Due to computational constraints, the experiments were limited to one to three hidden layers, all with 1,000, 3,000, or 6,000 neurons each. The network with TAAFs generally performs better than the baseline for most layer configurations when evaluated both on the noisy targets that were used for training and the real, noiseless targets as shown in Fig. 6.19. The networks with TAAfs performed similarly to the baseline only for the training data when using the model from the last epoch of training and evaluating using the noiseless data — even though the performance on the noisy targets that were used during training was significantly better than the baselines'. The dominance of the networks with TAAFs was slightly diminishing with larger networks for the experiment NN4, while experiments

parameter	task			
	NN1	**NN2**	**NN3**	**NN4**
hidden l. config.	1×1000	1×1000	1×1000	1×1000
	2×1000	2×1000	2×1000	2×1000
	3×1000	3×1000	3×1000	3×1000
	1×3000	1×3000	1×3000	1×3000
	2×3000	2×3000	2×3000	2×3000
			3×3000	3×3000
			1×6000	1×6000
			2×6000	2×6000
			3×6000	3×6000
dropout	0 %	0 %	0 %	0 %
	25 %	25 %	25 %	25 %
AF	sigmoid	sigmoid	sigmoid	sigmoid
	swish	swish	swish	swish
	tanh	tanh	tanh	tanh
target noise	0	0	0	0
				0.1
	0.25	0.25	0.25	0.25
	0.5	0.5	0.5	0.5
				1.0
	2.0	2.0	2.0	2.0
total number of variants	120	120	216	324

Table 6.11: **Configurations of inference networks**
Used distinct configurations of inference networks and noise variants for individual tasks. Note that the amount of target noise is not a property of the inference networks but rather a variant of the given task; however, since it influences the number of evaluated variants for each task, it is listed with the configurations of inference networks.

subexperiment	**noiseless target**				**noisy target**			
code	**win**	**loss**	**tie**	**win [%]**	**win**	**loss**	**tie**	**win [%]**
NN1	99	20	1	83.2	95	25	0	79.2
NN2	95	25	0	79.2	83	37	0	69.2
NN3	151	64	1	70.2	151	62	3	70.9
NN4	228	96	0	70.4	224	92	8	70.9

Table 6.12: **Summary of experiments with artificial data (test set, best validation error)**
The pairwise performance comparison of identical configurations using Wilcoxon signed–rank test on individual predictions evaluated on the test dataset with the network checkpoint with lowest error on the validation set. The comparison shows the performance evaluated both on the noiseless targets and the noisy targets.

subexperiment	**noiseless target**				**noisy target**			
code	**win**	**loss**	**tie**	**win [%]**	**win**	**loss**	**tie**	**win [%]**
NN1	41	79	0	34.2	111	9	0	92.5
NN2	41	79	0	34.2	110	9	1	92.4
NN3	111	105	0	51.4	194	22	0	89.8
NN4	147	177	0	45.4	244	80	0	75.3

Table 6.13: **Summary of experiments with artificial data (training set, last epoch)**The pairwise performance comparison of identical configurations using Wilcoxon signed–rank test on individual predictions evaluated on the train split with the network checkpoint from the last epoch. The comparison shows the performance evaluated both on the noiseless targets and the noisy targets used for training.

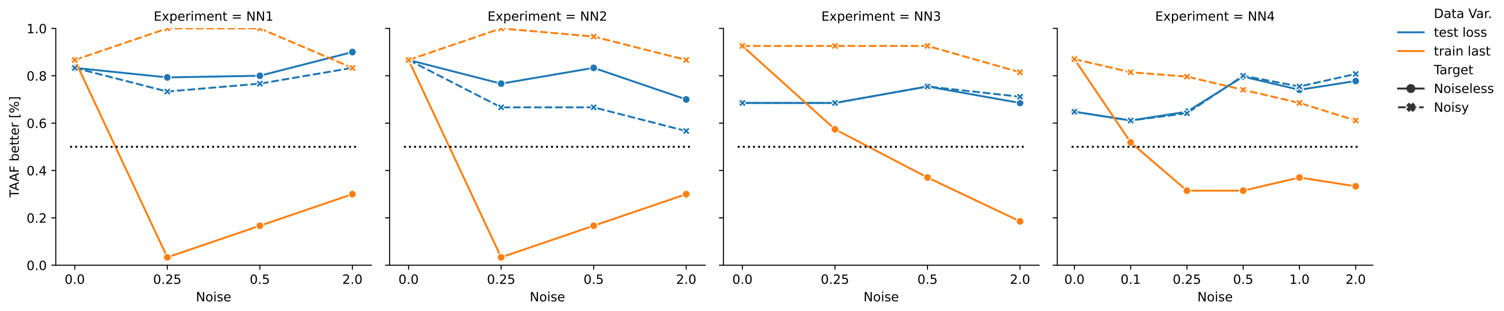


Figure 6.15: **TAAF dominance for individual noise levels**
The relative performance of TAAFs and baseline for different values of the noise added to the target prior the training. Only the OOS performance of the best model on the validation set and the in-sample performance of the model trained on the training set till the last epoch is shown. Shows the fraction of networks whose predictions were statistically significantly better than the baseline with identical parameterization.

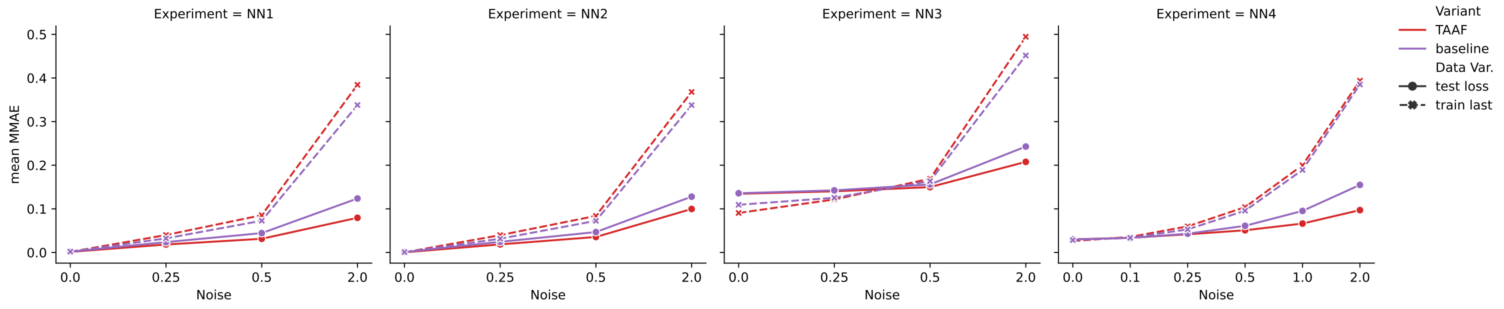


Figure 6.16: **Absolute performance for individual noise levels**
The absolute performance of TAAFs and baseline for different values of the noise added to the target prior to the training. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on noisy data but evaluated on noiseless targets.

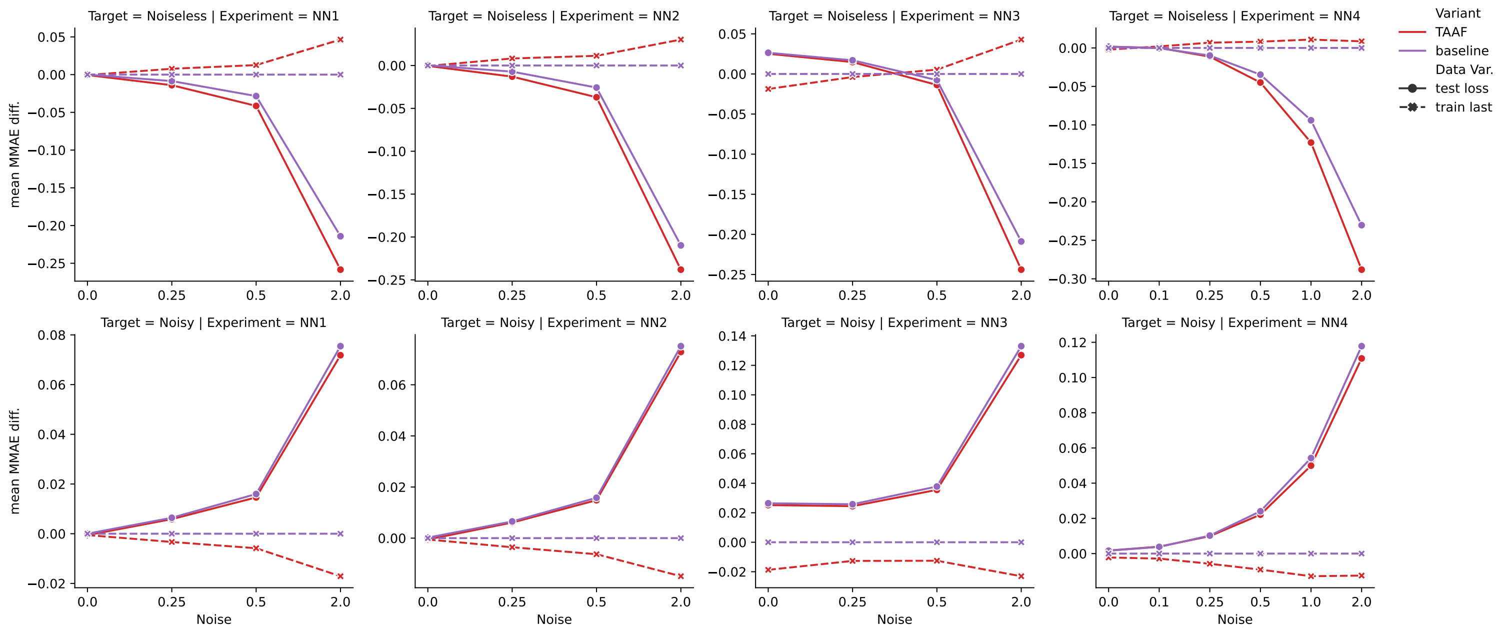


Figure 6.17: **Relative performance for individual noise levels**
The performance of TAAFs and baseline for different values of the noise added to the target prior to the training. It shows the difference of the mean MMAE for each data variant and network variant and the mean in-sample MMAE of the baselines with checkpoints from the last epoch computed for each noise level. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown.

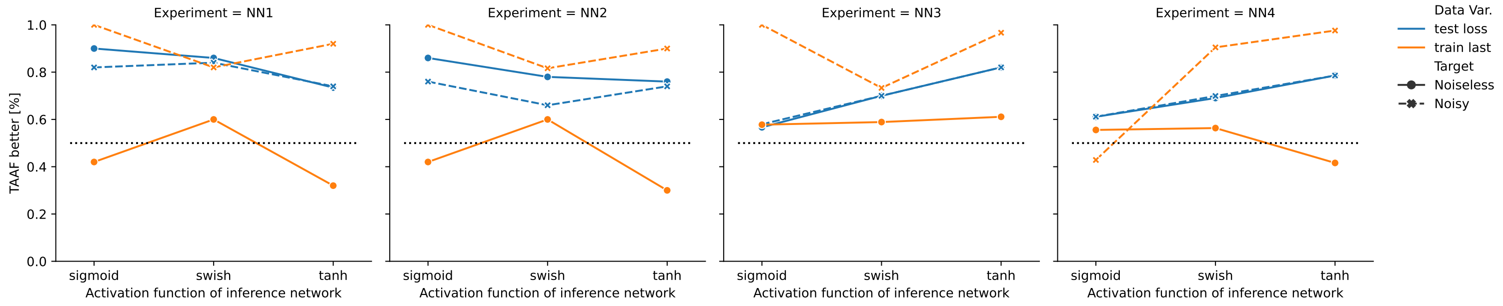


Figure 6.18: **TAAF dominance broken by inner activation network**
The relative performance of TAAFs and baseline for activations functions of the inference networks. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. Shows the fraction of networks whose predictions were statistically significantly better than the baseline with identical parameterization. The points are connected only for better readability of the plots; there is no inherent order of the activation functions.

NN1 – NN3 do not show significant differences in relative performance between the networks with TAAFs and the baseline with respect to the layer configuration. The absolute performance for targets without noise is shown in Fig. 6.20 and for targets with noise with standard deviation 2.0 in Fig. 6.21; complete set of variants with and without target noise for all tested noise levels is shown in Appendix A.2 in Figs. A.6 to A.10.

6.3.4 *Consistency of results over repetitions*

While the previous experiments compared the baseline and the networks with TAAFs over different parameterizations, only a single initialization for each generative network variant was used — resulting in four different generative networks. To show that the performance difference was not due to a particular initialization of the weights and therefore due to a chance, a single network variant was selected and run with 19 repetitions[1] with different initializations. The used data generation parameterization for the network experiment is the same as the NN3 network from the previous experiment — three hidden layers with 1,000 neurons each and the swish activation function, 1,000 input neurons and 5,000 output neurons. The results are shown in the Table 6.14, where the mean error of the samples for each relevant pair of baseline network with TAAFs were compared using Wilcoxon signed rank test at significance level $\alpha = 0.001$. The table also shows the median of differences of MMAEs between the baseline and the network with TAAFs (all the differences were also statistically significant using Wilcoxon signed rank test at significance level $\alpha = 0.001$). The actual MMAEs are shown in Fig. 6.22 for each network in the experiment (broken by the data split, model checkpoint, and usage of dropout). The networks with TAAFs outperformed the baseline in all cases for the test data set when the selected model was the one with the lowest loss over the test dataset and also in all cases on the training dataset if the model from the last epoch was used. The baseline tended to perform better on the test dataset when the last model from the epoch was used and also for the variant with 25% dropout on the training data set when the model with the lowest loss on the test data set was used.

However, the consistency of the results is more important as it shows that the performance differences are not just due to a realization of the random initialization of the weights of the data generation network. The initialization of the generative network influences the difficulty of the task and leads to the MMAE varying between the individual repetitions; however, the relative performance of the baseline and the network with TAAFs is consistent and without significant variance. This experiment shows that the results from the experiment from Section 6.3 are not weakened by the single initialization of the generative networks and that a single initialization is sufficient to draw results from — this is built upon also in the following experiments as doing multiple repetitions of the initialization in addition to an examined characteristic for each experiment would be very computationally costly.

1 Originally, 20 repetitions were planned, but results from one run got corrupted due to technical issues.

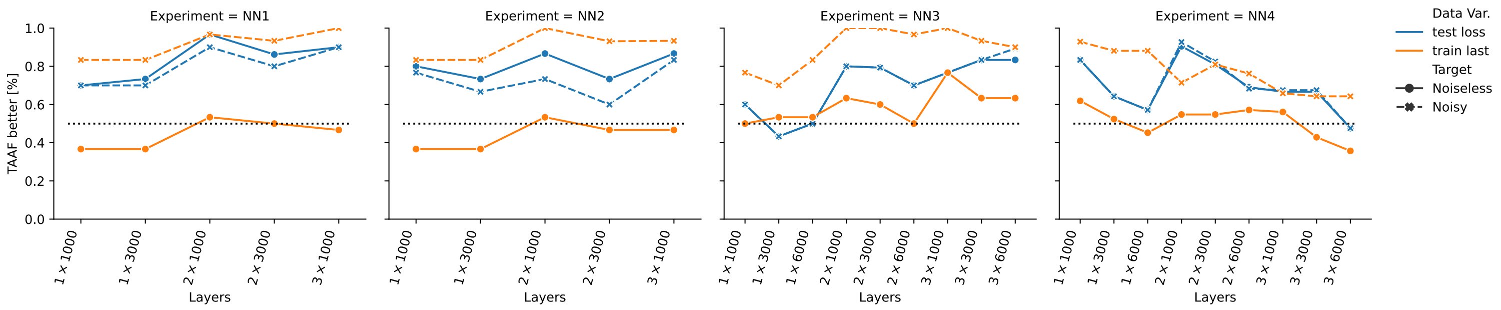


Figure 6.19: **TAAF dominance by layer configuration**
The relative performance of TAAFs and baseline for different configurations of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. Shows the fraction of networks whose predictions were statistically significantly better than the baseline with identical parameterization.

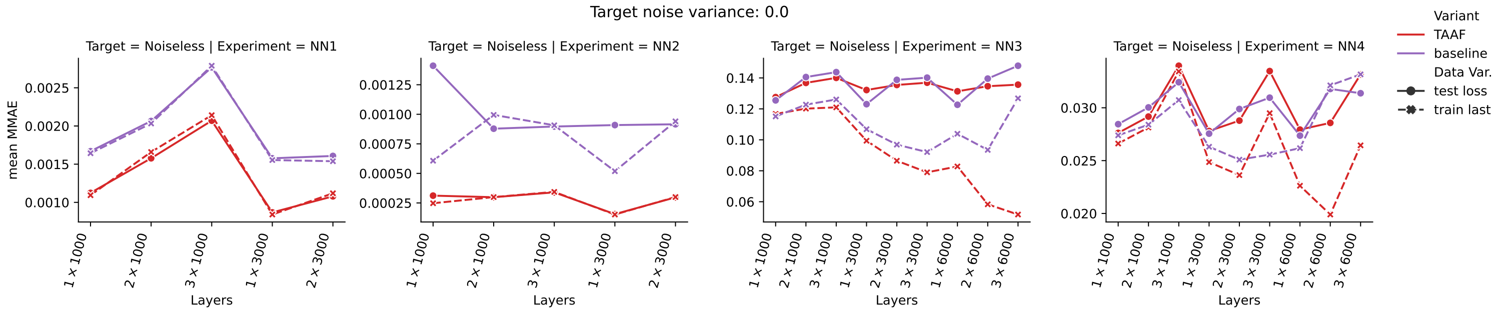


Figure 6.20: **Absolute performance by layer configuration without target noise**
The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on noiseless targets.

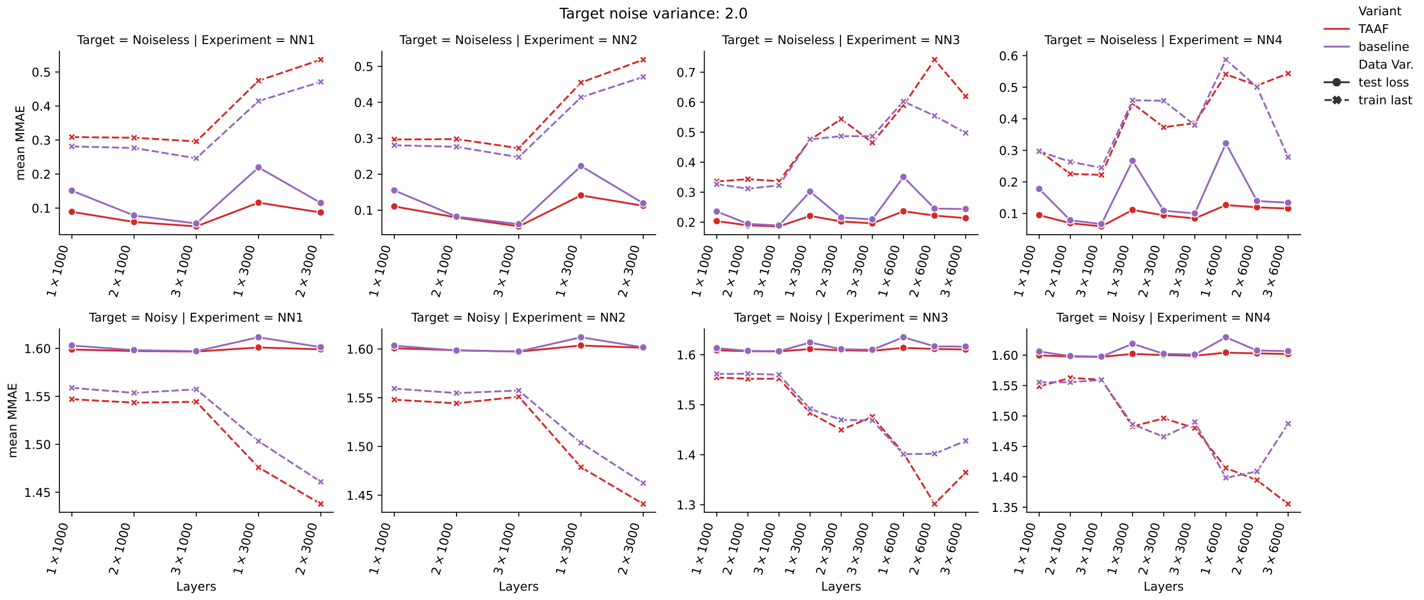


Figure 6.21: **Absolute performance by layer configuration with target noise**
The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on targets with noise with a standard deviation of 2.0.

Furthermore, the results from the Fig. 6.22 show that there are no significant differences between the results from the *validation* and *test* set as the process of selection of the best-performing network weight checkpoint introduces only a very negligible bias as the *validation* set is sufficiently large. Nevertheless, the *test* is used throughout most of the evaluation as the data are artificially generated and therefore not costly to obtain, and usage of independent *validation* and *test* set removes possible doubts about the results.

6.3.5 *Width of data generation networks*

This experiment focuses on the influence of the data generation setup on the relative performance of the baseline and TAAF networks, namely on the influence of the width of hidden layers in the data generation network. This setup is the same as in the previous experiment except for the size of the hidden layers. Each hidden layer had n neurons for each variant; $n \in \{250, 500, 750, 1000, 1500, 2000, 2500, \ldots, 9500\}$. Four networks were trained for each data variant — each network had either the *tanh* or *swish* activation function and was with 25% dropout or without any dropout.

6.3.5.1 *Results*

The overall error for each of the networks is shown in Fig. 6.24 — the difficulty of the task changes nonlinearly with the width of the generation network. The easiest task is when the generation network is very small (250 or 500 neurons in three hidden layers), but the difficulty peaks very quickly around a width of 1,000 to 3,000 neurons and slowly diminishes afterward as this pattern is present regardless of the inference network parameterization and on both *train* a *test* datasets. While the peak in difficulty coincides with the

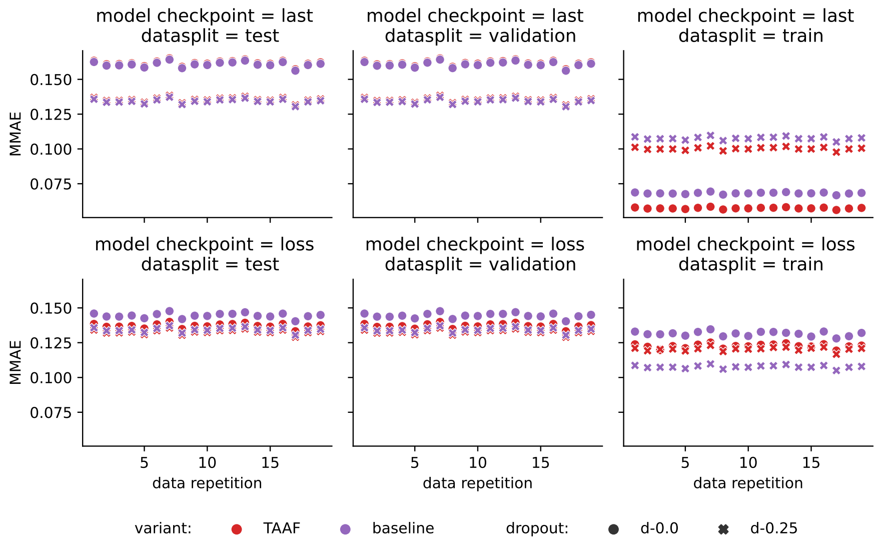


Figure 6.22: **Consistency of results over initializations**
The consistency of results over independent individual random initializations of the data generation network and input data samplings.

data set	**model checkpoint**	**TAAF better (no dropout)**	**TAAF better (25% dropout)**	**med. MMAE difference (no dropout)**	**med. MMAE difference (25% dropout)**
test	last	0/19	0/19	0.000929	0.001220
test	loss	19/19	19/19	-0.007395	-0.001559
train	last	19/19	19/19	-0.010781	-0.007428
train	loss	19/19	0/19	-0.009022	0.012887

Table 6.14: **Summary of relative performance on the artificial data**
Relative performance summary over independent individual random initializations of the data generation network and input data samplings. The relative performance is measured as the median of MMAE differences.

width of the inference networks used in the experiments, the behavior was not further analyzed.

The networks with the TAAFs dominated over the baseline in all cases on the test set when the network checkpoint was selected by the lowest loss over the test set and in all cases where the checkpoint from the last epoch was used on the train set. This is important as these are the two most interesting combinations; the rest of the combinations are less interesting. The *loss* checkpoint with the *test* data split shows the unbiased performance of a model selected under ideal conditions as the model checkpoint is selected independently from the training performance in order to reduce overfitting. On the other hand, the *last* checkpoint on the *training* data split shows the model's learning ability — how well it is able to fit the data. The comparison of the MMAEs of individual parameterizations of the baseline and the TAAF variant is shown in Fig. 6.25 for the *loss* checkpoint on the *test* set and in Fig. 6.26 for the *last* checkpoint for the *train* set; all shown pairwise differences in MAEs over samples were statistically significant when using Wilcoxon signed rank test.

The networks with TAAFs seem to be more prone to overfitting as when optimized till the last epoch, they reach better performance than the baseline on the training set used for the optimization, but they have worse performance on the test, and this effect is much more significant for variants without dropout as shown in Fig. 6.27. The networks with TAAFs with dropout do not show such behavior — the variant with swish network is always better. A detailed look at the relative performance of the tested variants on the test set with checkpoint from the last epoch is shown in Fig. 6.23. The network with swish activation function and the 25% dropout always performed better with TAAFs for all of the widths; however, it seems that the variant with tanh activation function and without dropout might have overfitted more than the baseline as it performed worse for nearly all of the widths — the 25% dropout helped and from a certain width, the TAAF based networks dominated (and thus performed worse only for widths 750, 1,000, 1,500, and 2,000 neurons).

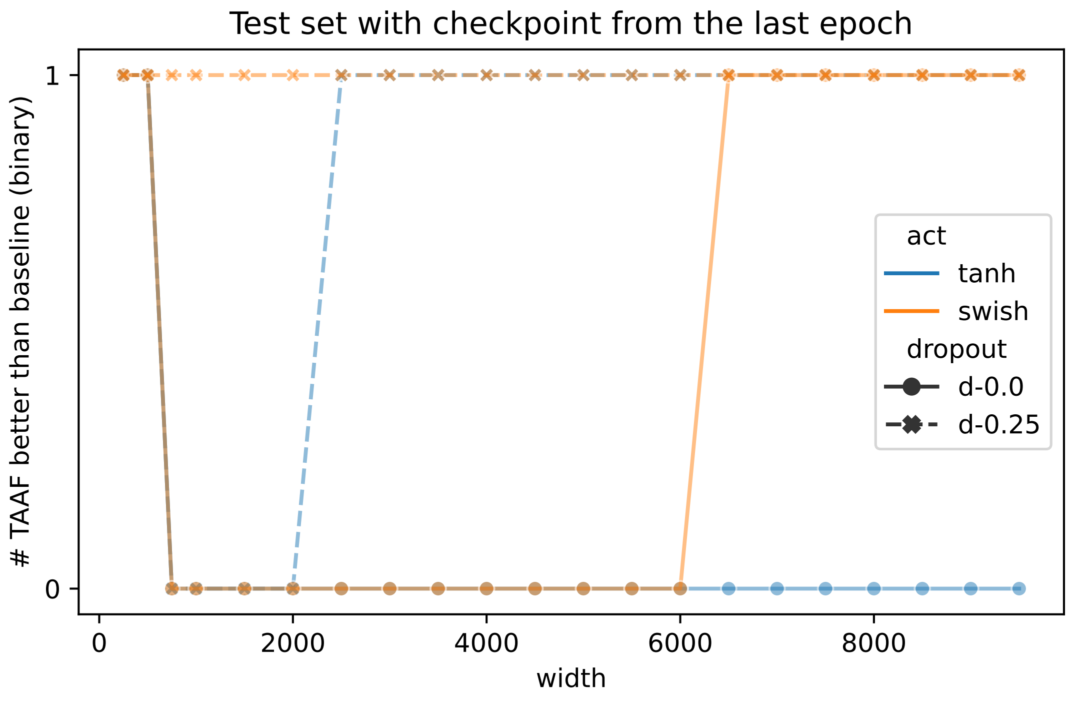


Figure 6.23: **TAAF dominance of last checkpoint on the test set**
The TAAF dominance on the test set for the model checkpoint from the last epoch based on the width of the generative network broken by different activation functions and dropouts of the inference neural network.

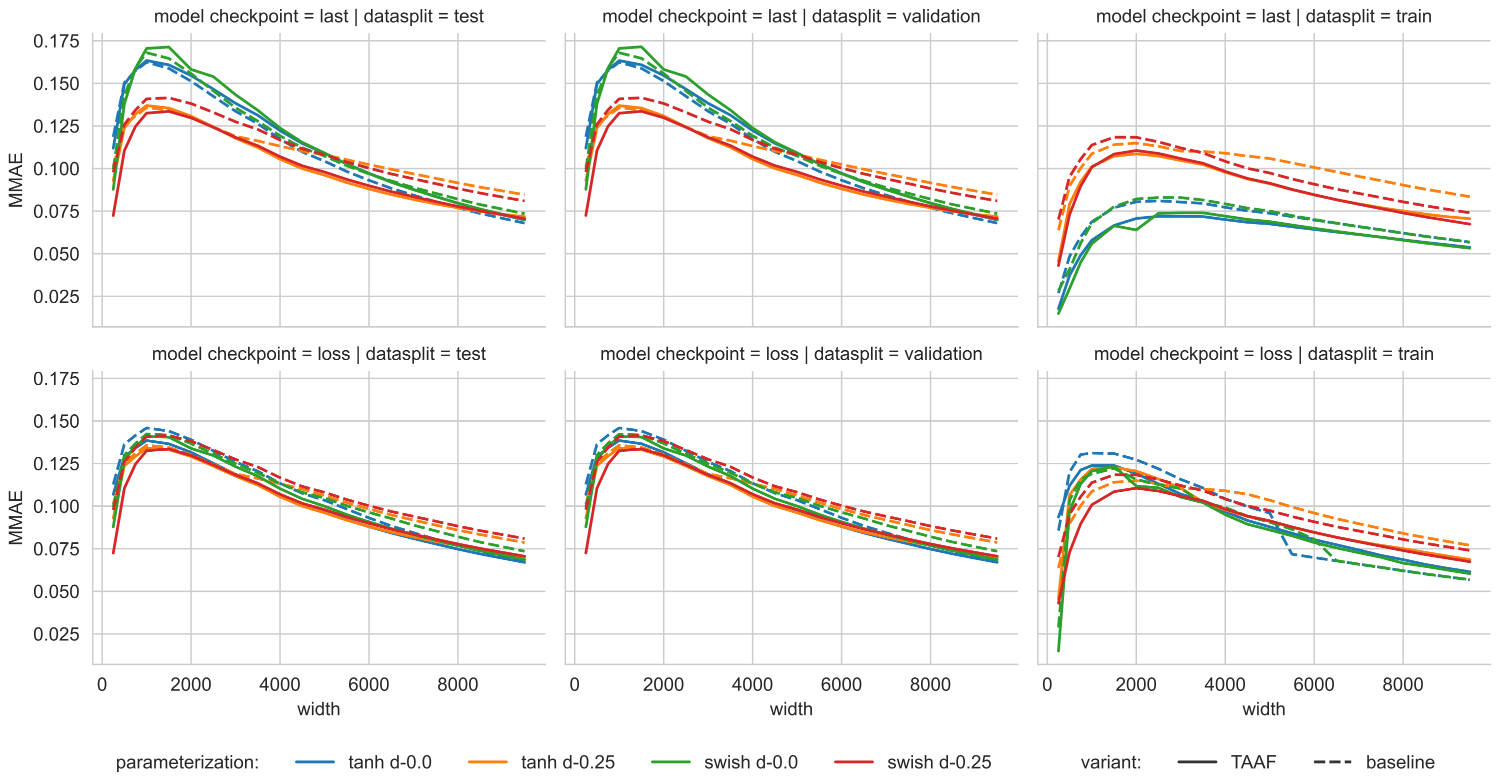


Figure 6.24: **Absolute performance by generative network width**
The absolute performance based on the width of the generative networks broken by data split and model checkpoint.

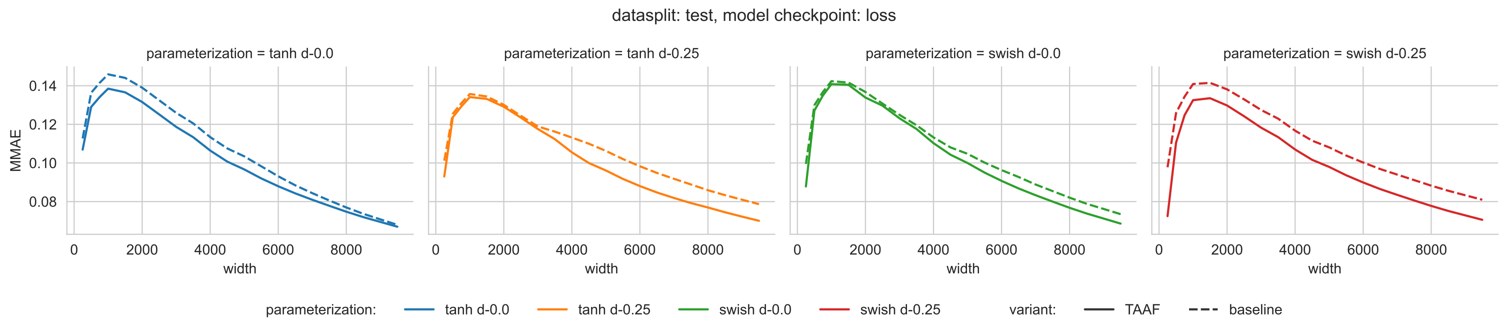


Figure 6.25: **Absolute test performance of best checkpoint by activations and dropouts**
The absolute performance on the test dataset for the model checkpoint with minimal loss on the validation dataset based on the width of the generative network broken by different activation functions and dropouts of the inference neural network.

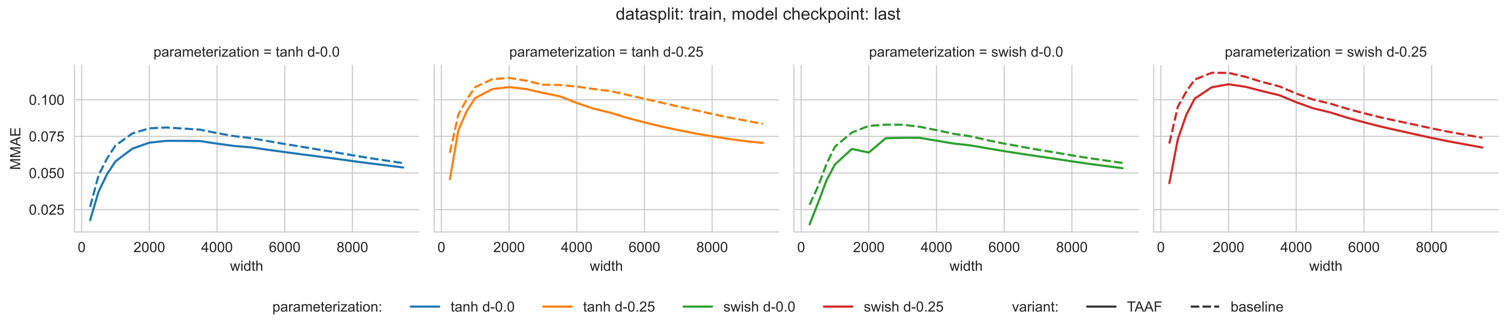


Figure 6.26: **Absolute training performance of last checkpoint by activations and dropouts**
The absolute performance on the training set for the model checkpoint from the last epoch based on the width of the generative network broken by different activation functions and dropouts of the inference neural network.

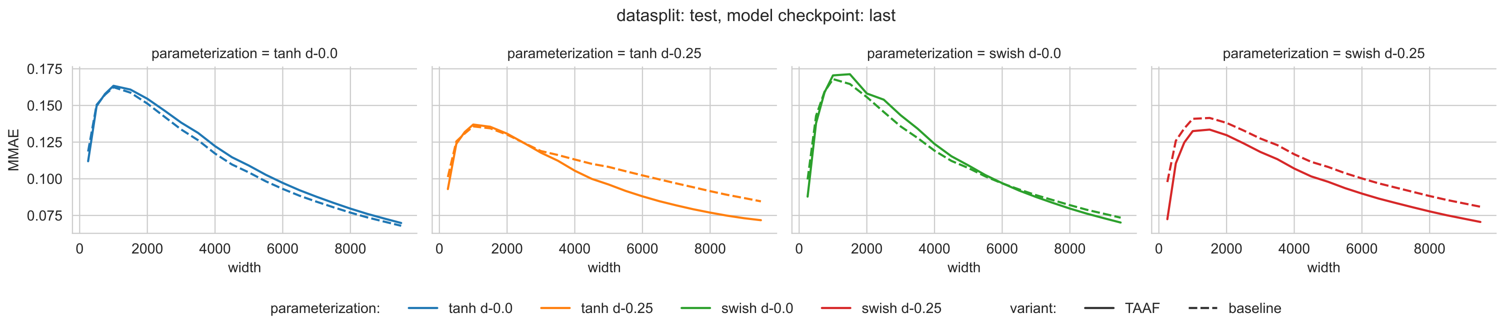


Figure 6.27: **Absolute test performance of last checkpoint by activations and dropouts**
The absolute performance on the test set for the model checkpoint from the last epoch based on the width of the generative network broken by different activation functions and dropouts of the inference neural network.

6.3.6 *Depth of data generation networks*

While the previous experiment focused on the width of the data generation network, this one focused on the depth of the data generation network. A similar setup was used with the exception that the width of hidden layers was fixed to 5,000 neurons, the depth was from the range $0, 1, 2, \ldots, 20$, and the output noise variance was 1.0.

6.3.6.1 *Results*

The networks with TAAFs performed better in at least half of the cases for each of the data set and model checkpoint combination; however, no dependence of the relative performance on the depth of the data generation network was observed as shown in Fig. 6.28.

6.4 ESTABLISHING THE ARCHITECTURAL IMPROVEMENTS USING D–GEX MICROARRAY DATA

We have evaluated both modifications of the baseline D–GEX architectures for nine different tower configurations. The configurations differ in the number of towers; their parameters are shown in Table 5.3, and the relationship between them is shown in Figs. 5.2 and 5.3. For each configuration, we have compared both possibilities for both configurations — tower or checkerboard architectures with or without the skip connection — resulting in a comparison of $2 \times 2 \times 8 = 32$ different pairs of networks. The detailed results for all four architectures and different numbers of towers are shown in Table 6.15. The relationship of MMAE and the number of towers for different architectures is shown in Fig. 6.29 — we can observe that the MMAE drops quickly and then starts to rise again slowly. The drop in MMAE at the beginning is due to an increase in the total number of neurons, which then increases the capacity of the network, making it able to better learn the relationships between the landmark and target genes. The relationship between MMAE and the

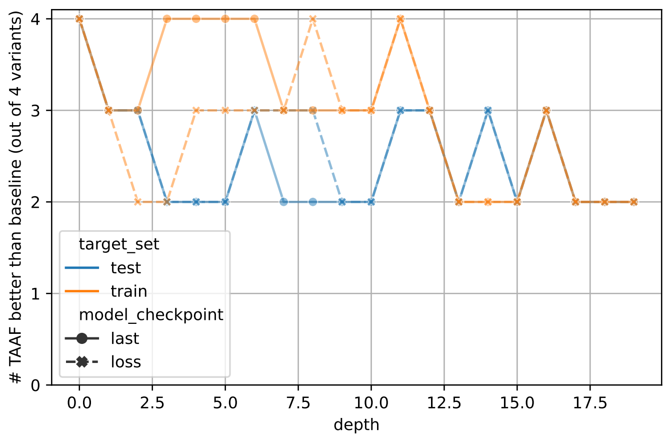


Figure 6.28: **TAAF dominance over generation network depths**
The TAAF dominance for various depths of the data generation network.

number of towers seems to slightly differ across the architectures — e.g., the TR–D–GEX reaches the minimum MMAE for the lowest number of towers compared to other architectures — and thus further research is needed.

The networks with five or more towers actually introduce a compressed representation as the layers in individual towers contain fewer neurons than the output layer — the last layer has to infer the gene expression from a lower number of inputs than the number of inferred genes. The number of towers for which the number of neurons is closest to the number of target genes is shown as a shaded region in Figs. 5.2, 5.3 and 6.29.

The baseline is the equivalent of the original D–GEX but with more neurons in each layer as the original D–GEX had at most 9,000 neurons in each layer, and the tested architectures are based on T–D–GEX with 10,000 neurons. The increase in the number of neurons, together with the learning rate schedule and increase in the number of training epochs, led to the improvement of the single tower D–GEX's MMAE from 0.134 to 0.131 even without the main architectural modifications. However, the proposed architectural changes, namely the CR–D–GEX (a checkerboard architecture with a skip connection from the first to the second layer), led to MMAE of 0.128 without any increase in the number of parameters of the network and only a slight increase in the running time which is due to more neurons (more operations to be performed).

6.4.1 *Statistical evaluation*

A Wilcoxon signed-rank test was used for pairwise comparison of individual models. The test was used to compare the means of MAEs of individual samples (which are assumed to be independent) at a significance level $\alpha = 10^{-4}$. The results for comparing different tower configurations are shown

# Towers	**T–D–GEX**	**C–D–GEX**	**TR–D–GEX**	**CR–D–GEX**
2	0.130187	0.130281	0.128623	0.128437
3	0.129839	0.129969	0.128548	0.128187
4	0.129735	0.129872	0.128568	0.128078
5	0.129729	0.129883	0.128617	**0.128053**
6	0.129707	0.129799	0.128677	0.128053
8	0.129760	0.129874	0.128864	0.128095
10	0.129804	0.129881	0.129078	0.128218
12	0.129889	0.129891	0.129291	0.128353
baseline ($3 \times 10,000$)				0.131301
D–GEX ($1 \times 9,000$, tanh) [2, 10]				0.163684
D–GEX with TAAFs ($3 \times 9,000$, TAAFo sigmoid) [10]				0.134015

Table 6.15: The MMAE of column based architectures on the test data. Architecture similar to the D–GEX with TAAFs [10] is the T–D–GEX with one tower, the overall best model is shown in **bold**.

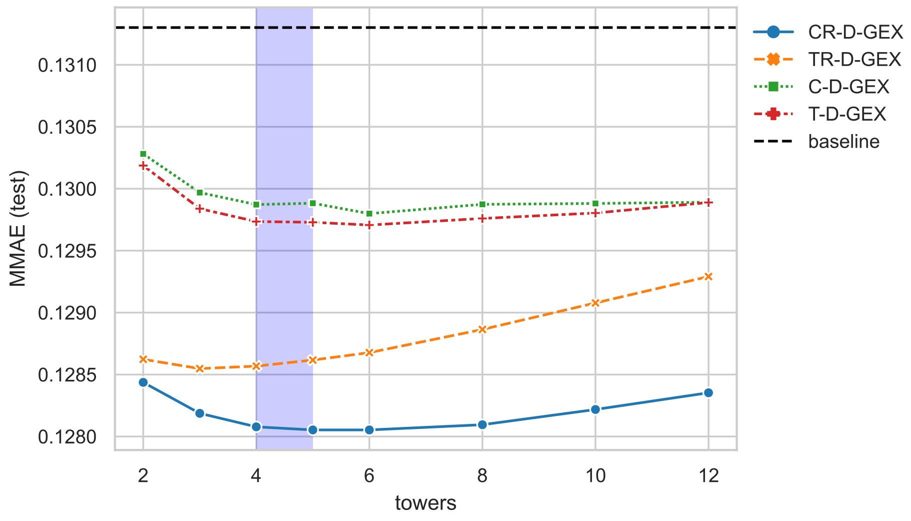


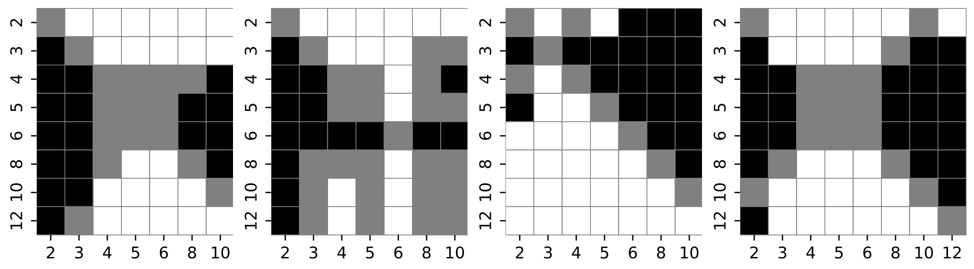
Figure 6.29: **MMAE progression based by number of towers**
The development of MMAE based on the number of towers for individual architectures. The shaded region denotes the number of towers for which the number of neurons in each tower is the most similar to the number of output neurons. The baseline is a single tower D–GEX with 10,000 neurons in each layer.

in Fig. 6.30 and generally confirm the U-shape of the model performance shown in Fig. 6.29. The comparison of different architectures for fixed tower configuration is shown in Fig. 6.31 — the checkerboard architecture CR–D–GEX is statistically significantly better for most configurations and not worse in all configurations. All of the tested architectures were also statistically significantly better than the baseline. We have also compared the best architecture (CR–D–GEX with five towers) with the best D–GEX with TAAFs from [10] ($3 \times 9,000$, TAAFo sigmoid) using the Wilcoxon signed-rank test and t-test on MAEs of individual samples and found that the CR–D–GEX has significantly lower MMAE with p–value $< 10^{-6}$. The 95% confidence interval for MMAE determined using bootstrap on samples' MAEs with 10^5 iterations was $[0.12717, 0.12891]$ for the CR–D–GEX with 5 towers and $[0.13317, 0.13487]$ for the D–GEX with TAAFs [10].

6.4.2 *Varying dropout rates in checkerboard architectures*

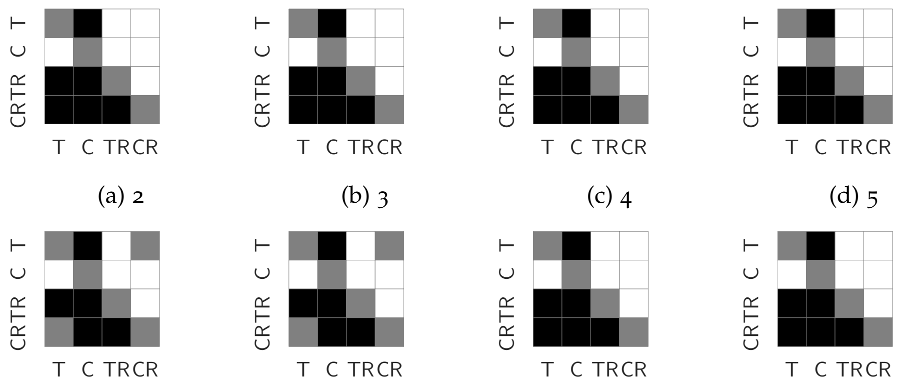
The previous experiments used a fixed value of dropout; however, this dropout value might not be optimal for the improved architecture even though it was performing well for the original D–GEX. These experiments show the performance evolution with respect to several chosen dropout values. The used architecture is the checkerboard architecture with five interconnected towers, each with 4,615 neurons in each layer.

We have evaluated nine different dropout rates from the interval $[0, 0.4]$ with a step of 0.05. The relationship between the MMAE on the test data and the dropout rate is shown in Fig. 6.32. We can observe a U-shaped curve; the networks were overfitting without a dropout; however, too high dropout rates



(a) T–D–GEX (b) C–D–GEX (c) TR–D–GEX (d) CR–D–GEX

Figure 6.30: **Pairwise statistical comparison by towers**
Results of pairwise Wilcoxon signed–rank test on the MAEs for individual samples for different number of towers. A cell in row *r* and column *c* is black if the model with *r* towers is statistically significantly better than the model with *c* towers, white if worse, and grey if no statistically significant difference was observed.



(a) 2 (b) 3 (c) 4 (d) 5

(e) 6 (f) 8 (g) 10 (h) 12

Figure 6.31: **Pairwise statistical comparison by architectures**
Results of pairwise Wilcoxon signed–rank test on the MAEs for individual samples for different architectures for fixed tower configuration. A cell in row *r* and column *c* is black if the model with architecture *r*-D–GEX is statistically significantly better than the model with architecture *c*-D–GEX, white if worse, and grey if no statistically significant difference was observed.

are also harmful as those increase redundancy. The pairwise comparison using Wilcoxon signed–rank test on the MAEs for individual samples for networks with different dropout rates with significance level $\alpha = 10^{-5}$ is shown in Fig. 6.33. The lowest MMAE was 0.1278 when the dropout rate was set to 0.15, and this improvement in MMAE is statistically significant at the significance level $\alpha = 10^{-5}$.

6.5 PRACTICAL IMPACT OF THE CHECKERBOARD ARCHITECTURE ON DIFFERENTIAL GENE EXPRESSION ANALYSIS

While the checkerboard architecture has statistically significantly lower prediction error than the D–GEX with TAAFs (see Section 6.1), the practical impact of this improvement remains unclear. We decided to demonstrate this impact on the frequent task of detection of differential gene expression similarly as in Section 6.2.1. The artificial phenotypes (see Section 5.1.5) were used to show the practical impact of the checkerboard architecture.

We have repeatedly sampled smaller datasets for different sample sizes (12–160) where each half of the samples was from the same cluster and have run differential gene expression analysis using parametric empirical Bayes from the limma R package [2113, 2114] on the ground truth data (the actual gene expression) and on the gene expressions inferred by the CR–D–GEX with five towers and the default D–GEX (TAAFo) [10] (see Section 6.4). We have used 10,000 repetitions for each sample size in this experiment.

The distribution of values and the pairwise differences of F_1 and MCC is shown in Figs. 6.34 to 6.37 for 10,000 repetitions for each sample size. The differences of all scores ($F_{0.5}$, F_1, F_2 scores, accuracy, and MCC) were statistically significant for all tested sample sizes when using the Wilcoxon signed-rank test as all the p–values were $< 10^{-8}$. Obviously, the advanced architectures can reasonably improve differential gene expression analysis and better approximate the gene sets reached with the original gene expression data. The improvement most strongly manifests for small sample sets, where

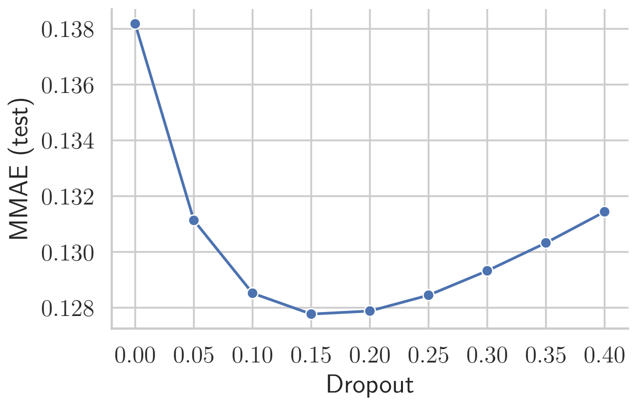


Figure 6.32: The development of MMAE for different dropout rates.

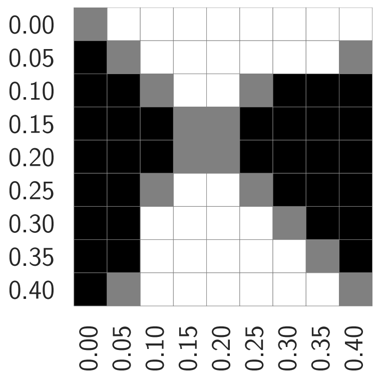


Figure 6.33: Results of pairwise Wilcoxon signed–rank test on the MAEs for individual samples for different dropout rates. A cell in row r and column c is black if the model with dropout r is statistically significantly better than the model with dropout c, white if worse, and grey if no statistically significant difference was observed.

even small changes in gene expression values may result in significant gene set changes.

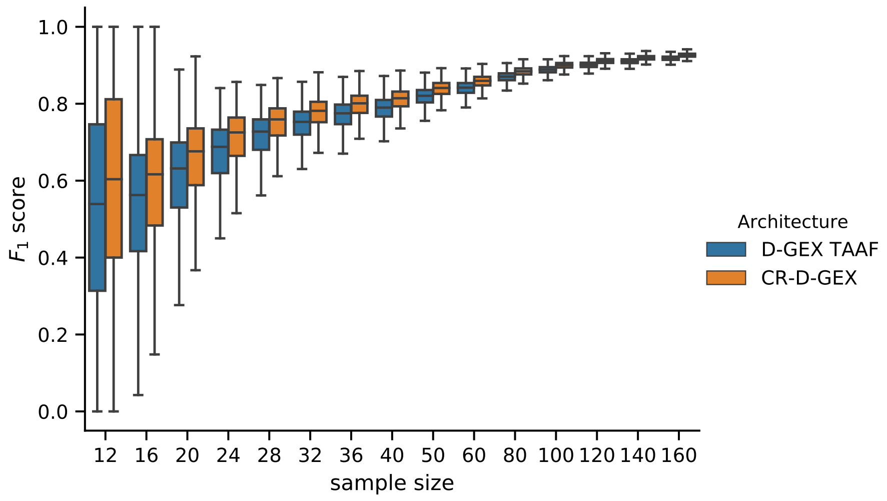


Figure 6.34: **F_1 scores by sample size**
Distribution of the F_1 scores obtained by the CR-D-GEX with 5 towers and the D–GEX with TAAF of 10,000 repetitions for each sample size. The whiskers show the 10th and 90th percentiles.

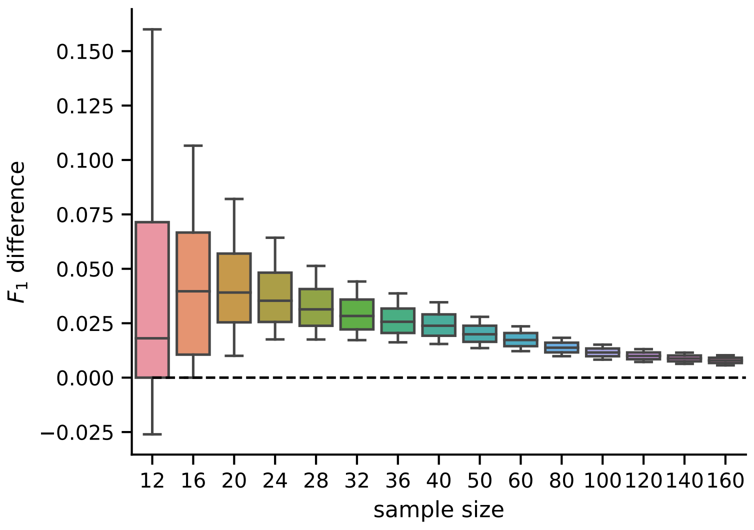


Figure 6.35: **Pairwise F_1 score differences by sample size**
Distribution of pairwise differences of the F_1 score obtained by the CR-D-GEX with 5 towers and the D–GEX with TAAF of 10,000 repetitions for each sample size. The whiskers show the 10th and 90th percentiles.

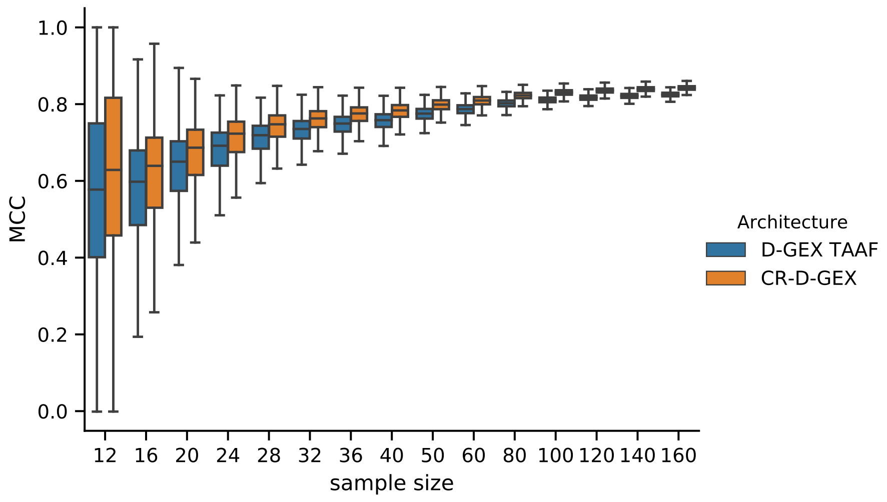


Figure 6.36: **MCCs scores by sample size**
Distribution of the MCCs obtained by the CR-D-GEX with 5 towers and the D–GEX with TAAF of 10,000 repetitions for each sample size. The whiskers show the 10th and 90th percentiles.

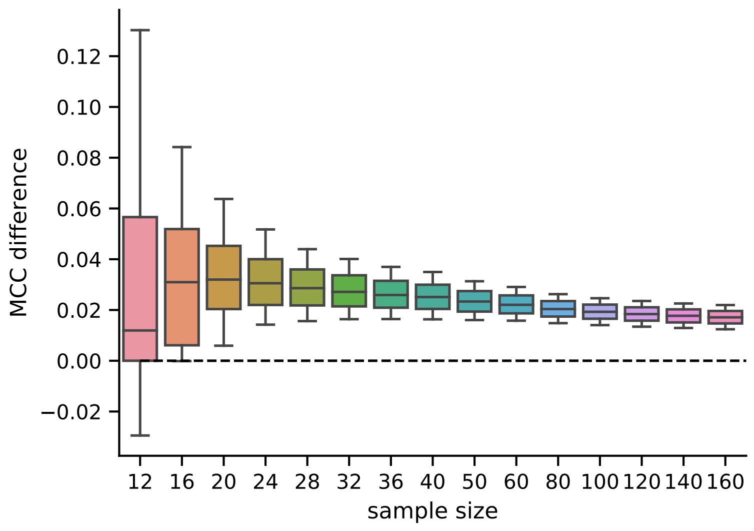


Figure 6.37: **Pairwise MCC differences by sample size**
Distribution of pairwise differences of the MCCs obtained by the CR-D-GEX with 5 towers and the D–GEX with TAAF of 10,000 repetitions for each sample size. The whiskers show the 10th and 90th percentiles
.

7 DISCUSSION

As shown in the previous chapter, the TAAFs and tower and checkerboard architectures improve the performance of GE inference from L1000 landmark genes. We first discuss the TAAFs themselves for the task of GE inference in Section 7.1 and for the artificial regression tasks in Section 7.2 and then we focus on the further improvements of the GE inference performance using the tower and checkerboard architectures in Section 7.3.

7.1 TRANSFORMATIVE ADAPTIVE ACTIVATION FUNCTIONS

The experimental results show that the proposed transformative adaptive activation functions lead to significant improvements in the gene expression inference for the L1000 platform in Section 6.1 and in the multivariate regression tasks using several artificially generated datasets in Section 6.3.

7.1.1 *TAAFs improve the performance*

The TAAFs can serve as a drop-in improvement for various architectures as shown by improving the D–GEX architecture for the GE inference task as was shown in Section 6.1.1 where we reimplemented the original D–GEX and used with the tanh AF that was originally used by Chen et al. in [2] and shown that TAAFs also with the tanh as the inner activation function f (see Section 5.2 for definition) lead to statistically significant improvement in the error metric MAE. We have re-implemented the D–GEX and did our own experiments instead of just comparing the error values from the original paper [2] for several reasons. One, the reimplementation allowed us total control over the experiments and limited any potential reporting bias by the authors of the original D–GEX. Two, the authors of the original D–GEX Chen et al. normalized all genes separately, which, we believe, is not suitable for the gene expression inference task from the L1000 profiles as it gives equal importance in the error metric to highly expressed genes and genes whose expressions are near noise levels which biases the inference model in unwanted direction (see Section 5.1.1.2 Data normalization). And three, the reimplementation allows us perfect control over technical details of the training procedure, and therefore, we can show that the performance gains are solely due to the usage of the TAAFs and not to any other changes in the training procedure (e.g., different schedule of the learning rates or batch size).

Our reimplementation of the original D–GEX has an MMAE 0.1637 while the best performing D–GEX with TAAFs as drop-in modification (i.e., only adding the TAAFs to the architecture and keeping the tanh as the inner AF of the TAAFs) has MMAE 0.1361 leading to $1 - \frac{0.1361}{0.1637} \approx 17\%$. As the authors of the concurrent works [14, 15] used identical normalization as Chen et al., which is not as suitable for the GE inference task as our normalization, it is not possible to directly compare our results with the much more complex approaches presented in [14, 15]. Nevertheless, we can provide a rough estimate of the D–GEX with TAAFs relative performance to the GAN based approaches used in [14, 15] by comparing the performance gains over the original vanilla D–GEX — the D–GEX with TAAFs provides $\approx 18\%$ improvement in the MMAE while the presented improvements using complex GAN approach have $1 - \frac{0.2997}{0.3204} \approx 6.5\%$ improvement [15] and $1 - \frac{0.2897}{0.3204} \approx 9.6\%$ in [14]. Even though this approximate comparison can be arbitrarily biased due to the normalization differences, it should be fairly unbiased unless the approaches presented in [14, 15] have performance significantly skewed toward particular genes based on their mean expression levels. On the other hand, both TAAFs and GANs from [14, 15] are not mutually exclusive and can potentially be used together to achieve even better performance. Furthermore, the TAAF approach is conceptually much more straightforward than the usage of GANs while reaching, at the very least, comparable performance.

After establishing the performance of TAAFs as drop-in modifications of the original D–GEX in Section 6.1.1, we further analyze the usage of TAAFs for the GE inference tasks. The usage of TAAFs in a NN leads to higher robustness in terms of sensitivity to the choice of the inner AF as shown in Section 6.1.2, where we have shown that the original D–GEX benefits highly from replacing the original tanh activation with the logistic sigmoid sigmoid activation. In contrast, the D–GEX with TAAFs is able to reach similar performance even with the original tanh as the inner activation — the choice of tanh and logistic sigmoid as the inner AFs can be seen as merely different initialization of the TAAFs. Initilization of the TAAFs with parameters $\alpha := \frac{1}{2}$, $\beta := \frac{1}{2}$, and $\delta := \frac{1}{2}$ and tanh as the inner AF f is equivalent to using the default initialization with the logistic sigmoid as the inner AF.

Nevertheless, the initialization of the TAAFs does play a role in the performance as the usage of the logistic sigmoid as the inner AF leads to improved performance over the tanh as the inner AF even for the architectures with TAAFs as shown in Table 6.2 even though, as discussed above, the relative performance gain is less noteworthy compared to AF replacement in the original D–GEX as can be seen from comparison of Table 6.2 and Table 6.3. Not surprisingly, the usage of TAAFs in the D–GEX variant with logistic sigmoids leads to performance gains similarly as in the previous experiment Section 6.1.1 Experiment 1: Usage of TAAFs with tanh AF as shown in Table 6.4.

7.1.2 *TAAF parameters*

While showing that TAAFs as drop-in modification can increase inference performance without any architecture modification was necessary to illustrate that TAAFs can be used easily in an existing setup, it also has some limitations. The most severe is that the TAAFs introduce four additional parameters, and the compared networks have a different number of parameters.

This is problematic as NNs with more parameters usually have higher capacity, and this could lead to higher performance in general. NNs with more parameters usually reach higher performance on the training data (under certain circumstances, the NNs can even memorize the dataset to have perfect performance but poor generalizability [2135–2138]); increased capacity can help the network if it is underfitting and thus increase its performance also on the test data [2136]. This also happens for the original D–GEX where the architectures with more neurons in each layer generally performed better than the NNs with fewer layers or neurons as shown in Fig. 1 in [2]. Therefore, the reported performance gains from experiments in Sections 6.1.1 and 6.1.2 could be, in theory, solely due to this increased capacity and not due to the better expressivity of the novel TAAFs; theoretically, it could be even worse and just using larger vanilla D–GEX with the same number of parameters as the D–GEX with TAAFs could have higher performance gain and the TAAFs would be actually harmful to the model.

To show that this is not the case and that TAAFs improve the performance of the GE inference due to more efficient parameter usage and not just because of the capacity increase that is solely due to the higher parameter count, the Experiment 3: TAAFs for capacity adjusted NNs was conducted. In this experiment, the TAAFs are not used as a drop-in modification of an existing architecture but are rather used in a slightly smaller neural network that has the same or lower number of parameters as the D–GEX variant it is compared to. To obtain the reduced architecture, neurons were uniformly removed from all hidden layers until the total parameter count was not higher than the parameter count of the respective D–GEX variant. The final size of the hidden layers is shown in Table 6.5 in the second column. Even the NNs with adjusted numbers of parameters by decreasing the width of hidden layers always outperformed the respective D–GEX variant that had the same or higher number of parameters (see Table 6.5); therefore the performance gains are due to the TAAFs themselves and not just due to the additional parameters introduced by the TAAFs.

As already briefly discussed in Section 6.1.3, the size reduction of the hidden layers was negligible as TAAFs introduce four parameters per neuron, but each additional neuron in a hidden layer introduces a weight for each incoming connection. Therefore, the performance drop of D–GEX variants with TAAFs due to the adjustment of a total number of parameters is negligible, and it is not surprising that the results from Sections 6.1.1 and 6.1.2 hold, and the variants with TAAFs still significantly outperform the original D–GEX architecture. Since the parameter reduction had only an insignificant effect, the rest of the comparisons of the two architectures are as drop-in modifications the same as in Sections 6.1.1 and 6.1.2 for clarity, easier analysis, and simpler running of experiments instead of comparing the original variant with the reduced TAAFs based architectures with adjusted number of parameters.

The proposed TAAF introduces four additional parameters, and so far, the importance of individual parameters has not been established. Since the TAAF can be viewed as a generalization of several previously established AAFs, the performance increase compared to the sigmoid activation function might be due only to those parameters that were already established as beneficial (e.g., trainable amplitude [1086]; see Section 5.2.1.1 and Table 5.1) and there might not be synergies from combining multiple already established parameters. Furthermore, since the proposed adaptive activation function is applied to the weighted sum of inputs in the neuron, parameter β might seem redundant:

$$g(f, \boldsymbol{x}) = \alpha \cdot f\left(\beta \cdot \sum_{i=1}^{n} w_i x_i + \gamma\right) + \delta, \tag{7.1}$$

where n is the number of inputs in the neuron, x_i are individual inputs and w_i are associated weights. This can be expressed without parameter β if we define $u_i = \beta w_i$:

$$g(f, \boldsymbol{x}) = \alpha \cdot f\left(\sum_{i=1}^{n} u_i x_i + \gamma\right) + \delta. \tag{7.2}$$

While parameter β seems to be redundant, redundancy by itself does not mean uselessness — in some cases, it can even improve the performance as shown in [2139–2141], where the authors introduced additional redundancy to neural networks to increase its performance, and in [2142] where the authors discuss redundancy in a biological context with a connection to the artificial neural network architecture ResNet [13]. Another example of apparent redundancy can be found in overspecified neural networks — it was shown that overspecified wide networks simplify the optimization surface for optimizers in the sense that it is easier to reach good optima [2143–2145]. However, redundancy does not always improve the performance; e.g., Lee et al. showed that the redundancy in the rank of NN parameters slows the training and that a regularization method reducing this redundancy improves both performance and training speed [2146].

Even though the redundancy represented by the β parameter is different from some of the referenced examples, we empirically show that it improves performance, and the improvement is statistically significant. The intuition behind redundancy in the form of additional parameters or overspecified networks is that "higher dimensions also mean more potential directions of descent, so perhaps the gradient descent procedures used in practice are more unlikely to get stuck in poor local minima and plateaus" [2143], which might be one of the reasons that the inclusion of the redundant parameter β was empirically shown to be beneficial in our work. Furthermore, Jagtap, Kawaguchi, and Karniadakis used the horizontal scaling parameter as the only adaptive parameter in their LAAFs that were concurrently[1] published to our work, and they show that it improves the speed of convergence in [1137]. Jie et al. consider AFs to be *expressively independent* if such form of redundancy is not present (see [1073] for definition) and assume that an AF that is not expressively independent is not a good choice [1073]; however, this is not the case as there are many AAFs that are not expressively independent and yet

1 The TAAF preprint [444] was publicly available in March 2019 while the LAAF preprint [2147] only in June; however the full work [1138] was published in a journal in March 2020 whereas TAAFs [10] only in December 2020.

improve the performance — e.g. SVAF, ASSF, swish, and LAAFs. Also, our findings show that the parameter β contributes to a better performance of TAAFs, and, therefore, the assumption of Jie et al. does not hold in practice.

We have empirically shown that all four TAAF parameters are beneficial in Section 6.1.4 Experiment 4: Importance of individual parameters, where all 16 subsets of trainable parameters were evaluated, and the full TAAF with all four parameters trainable statistically significantly outperformed other TAAF variants. This empirical evidence shows that the full TAAF is the correct choice, and the reported performance gain is not solely due to a component that was reported previously in the literature, such as the vertical scaling parameter α whose equivalents are present in many adaptive activation functions such as the trainable amplitude [1086] (see Section 4.3.2) and swish [668] (see Section 4.3.3.1) or the horizontal scaling parameter β present in, e.g., SVAF [1092] (see Section 4.3.2) or Adaptive slope hyperbolic tangent [1139] (see Section 4.3.15.1).

7.1.3 *Conceptual architectural simplification for regression tasks*

The first two experiments established the benefits of TAAFs and the second two experiments dealt with some of the limitations of the first two by showing that the increased performance of NNs with TAAFs is due to the AAFs themselves and not just due to the parameter increase, that the TAAFs provide an efficient way of increasing expressivity of NNs, and that all four introduced parameters are necessary and therefore the formulation of TAAFs is correct and cannot be reduced further without performance costs. The fifth experiment shows another benefit of the TAAFs in NNs for regression problems where the last layer usually has no activation to allow for any output range as most sigmoids, ReLUs and similar AFs have limited range. The TAAFs can be used in the last layer even for regression problems as their range is not limited thanks to the vertical scaling parameter α. This is shown in Section 6.1.5 Experiment 5: TAAF in the output layer, where a D–GEX model with TAAFs in hidden layers and no activation in last layer is compared to a D–GEX with TAAFs in all layers (denoted as TAAFo). The TAAFo variant leads to statistically significantly higher performance for all tested depths and widths of the D–GEX network. Another benefit introduced by TAAFs besides the improved performance is the conceptual simplification of the neural network as all layers have identical activation functions and there is no need for special treatment of the last layer even for regression problems.

Similarly as in the original D–GEX work [2], we used a random division of the data into the training, testing, and validation sets (see Section 5.1.1 for more details). While this approach has the benefit of simplicity, it has one potential issue — since the data consists of several different biological datasets, there might be introduced a bias into our results as samples from one biological dataset might be in in both the training and testing set and the reported test performance might not translate well to unseen data. Nevertheless, we show that it is not the case in Section 6.1.6 Experiment 6:

heterogeneity-aware data sampling, where we used the heterogeneity-aware dataset (see Section 5.1.1.1) where the data splits conform to the known *GEO-* series and all samples from a single *GEO-* series are in a single split (either training, testing, or validation). As shown in Table 6.7, the D–GEX with TAAFs outperformed statistically significantly the original D–GEX even on this heterogeneity-aware dataset; this shows that the potential bias mentioned above does not significantly affect (if present at all) the results of the presented comparative analyses. Since not all samples from the data provided originally were missing the *GEO-* id and therefore could not be matched to an existing *GEO-* series and thus were omitted, the heterogeneity-aware dataset had only $\approx 60\%$ of the training samples compared to the whole dataset, the full dataset containing the same samples as in [2] was used for most of the experiments rather than this reduced dataset since the potential bias does not significantly influences the results.

7.1.4 *Gene expression inference perspective*

So far, we have discussed the results regarding individual improvements to the NN architecture and established that the observed performance increases are valid. However, we have not looked at the results from the perspective of the gene expression task itself — which architecture has the best performance. This is done in Section 6.1.7 where all results from the previous experiments are compared, and 10 best-performing architectures are listed in Table 6.8. While the previously reported results focused on the pairwise differences to see that the modifications help all tested NN variants, Table 6.8 contains the absolute performance metric MMAE and lists the best models with the lowest test error irrespective of the underlying architecture. Generally, the best-performing models are the largest tested architectures with 9,000 neurons in either two or three hidden layers with the logistic sigmoid as the inner activation function of the TAAFs. The overall best-performing architecture with MMAE was the NN with three hidden layers, each with 9,000 neurons with TAAFs with the logistic sigmoid and the last layer contained the TAAFs instead of the linear activation present in the original D–GEX. This network was even better than the one from the Section 6.1.1 Experiment 1: Usage of TAAFs, where the TAAFs were used only as a drop-in modification of the original D–GEX — MMAE 0.1340 (CI $[0.13316, 0.13487]$) compared to 0.1637 (95% CI $[0.16279, 0.16458]$), or, relative to the D–GEX reimplementation, $1 - \frac{0.1340}{0.1637} \approx 18\%$ improvement compared to $1 - \frac{0.1361}{0.1637} \approx 17\%$ improvement over the D–GEX performance. It can be seen that the major improvement to the original D–GEX architecture is the usage of TAAFs; a more detailed breakout of the impact of individual improvements is shown in Figs. 6.3 and 6.4. Figure 6.3 shows the improvement to the regular D–GEX architecture (showed as a baseline) including switching the inner activation function from tanh to the logistic sigmoid, using TAAFs, and using an ensemble of several NNs (more about the ensembling in Section 5.2.2) while Fig. 6.4 builds on the D–GEX variant already equipped with TAAFs and its goal is to depict other improvements besides TAAFs. As already briefly discussed above, we can observe that the NNs TAAFs are much more robust with respect to their parameterization compared to the plain D–GEX as there is only a small difference between using TAAFs with the logistic sigmoid and the hyperbolic tangent functions. In contrast, this difference is very large for the plain D–GEX, as shown in Fig 6.3 — MMAE 0.1409 for D–GEX with logistic sigmoid activation and 0.1637 with tanh activation compared to the TAAF based networks where NN with logistic sigmoid activation has MMAE 0.1354 and 0.1372 when tanh activation is used instead. This robustness is important because the training of the networks is computationally costly, and thus, the parameter search possibilities are limited.

The impact of the different initialization of TAAFs is similar to the impact of the usage of TAAFs in the output layer — MMAE 0.1354 for using logistic sigmoid instead of tanh compared to 0.1361 for using TAAFs in the output layer; the baseline was 0.1372; furthermore both of these changes worked well together resulting in MMAE 0.1340. On the one hand, the impact of the used ensemble scheme resulted only in minor MMAE improvements; on the other hand, these improvements were present on the test data and were for free in the sense that the ensembles are from already trained models and no extra costly training process was necessary to obtain these ensembles. Nevertheless, the GE inference is more costly using this type of ensembling, and therefore, we believe that it is most suitable to use only a single model for most tasks instead of this kind of ensembling. These observations are limited to the used ensembling in this work; this work touched the ensembles only superficially, and only because they were available without any significant costs, there might be ensembling schemes that would lead to significantly better performances compared to a single model.

7.1.5 *Practical impact of TAAFs*

So far, we have focused on the error of the GE inference from the L1000 landmark genes. The limitation of the hitherto reported results was that the improvements in the inference errors are rather abstract, and it is unclear whether they have any practical significance. The practical significance of the reported improvements is shown in Section 6.2 where several differential gene expression analyses were run on the inferred data — specifically, we have run a DGE analysis on the data inferred by plain D–GEX and D–GEX with TAAFs for several sample sizes. We have run two sets of experiments, differential gene expression analyses with artificial phenotypes that arose from hierarchical clustering of the data and with real phenotypes belonging to the largest GEO- series present in the data (see Section 5.1.5 for more details). As most biological experiments suffer from rather small data sizes as obtaining data is usually costly, we have focused on sample sizes ranging from 12 to 600 samples for the artificial phenotypes and to 400 samples for the real phenotypes as those were limited by the available data of each class (see Table 6.9 for actual maximum sample sizes). To limit the influence of variance arising from the admittedly small sample sizes that are often present in biological data, each sample size was sampled with 5,000 repetitions. The results from the differential gene expression analyses can be considered as a classification task; therefore we have opted for accuracy, F_1, $F_{0.5}$, F_2, and MCC metrics to show the practical impact of the TAAFs on the GE inference task in Sections 6.2.1 and 6.2.2. This was complemented by analysis of candidate rankings in Sections 6.2.1.1 and 6.2.2.1 where the rankings of candidate genes from the DGE analysis on the ground truth data were compared to the rankings from differential gene expression analyses on the inferred data.

The results in Section 6.2 show that the NNs with TAAFs outperform the plain D–GEX in practical tasks and that the difference in the metrics on the practical tasks are statistically significant for all sample sizes for the artificial phenotype and for all sample sizes of all but one combination of classes where the D–GEX with TAAFs performed similarly as the plain D–GEX for smaller sizes (see Fig. 6.12). The empirical evaluation of the practical impact of the presented improvements is one of the important results of this work as no evaluation of practical impact was done in other works in the literature [2, 14, 15] where it was only assumed that the presented improvements would have any practical importance on the tasks faced by biologists and medical experts.

7.2 TAAFS FOR OTHER TASKS BESIDES GENE EXPRESSION INFERENCE

Up until now, the TAAFs were evaluated on the GE microarray data on the GE inference tasks from the L1000 landmark genes; however, the concept of TAAFs is general, and they can be used in many settings. In Section 6.3, the TAAFs are used for regression tasks with artificially generated data. We have chosen to show the performance of TAAFs using artificially generated data for several reasons — they are the usual reasons why researchers use artificial data that are discussed in Section 4.5.2 Synthetic data generation: control over the properties of the data and scarcity of data. By using the artificial data, the real, noiseless targets are available, and therefore, we know how much white noise is added to the data that is not predictable. For example, reaching a lower inference error than is due to the added white noise is a clear symptom of overfitting, as we can guarantee in artificially generated data that the present noise is indeed independent. Furthermore, we were looking for a sufficiently complex multivariate regression task outside the omics but there were datasets similarly large as the dataset used by the D–GEX that would have several hundreds of independent and dependent variables; there were smaller datasets such as, for example, *Wine Quality* [2148] or *Breast Cancer Wisconsin* [2149] from the UCI Machine Learning Repository [1373] but they were too small both in terms of number of features and number of samples. There are larger regression multivariate time-series datasets, but the time-series aspect would add too much complexity for the analyses, and we still would not have control over the dataset and would not know the properties of the noise and what would be the best achievable error; therefore, the usage of TAAFs for time-series prediction is left for future works. Four different regression datasets were generated; details about the generation process are available in Section 5.1.2.1. We have shown that the NNs with TAAFs, on average, outperform the vanilla variants in several experiments.

First, we have established that NNs with TAAFs generally improve the performance in Section 6.3.1. The results also show that the model from the training procedure that has the lowest error on the validation set should be generally used as otherwise NNs with TAAFs are more prone to overfitting, as can be seen from comparing the results on the test data with a model that has the lowest validation loss in Table 6.12 with the results on the training set with a model from the last training epoch in Table 6.13.

Second, the impact of the amount of noise in the target variables was analyzed in Section 6.3.2. The NNs with TAAFs are better in general for all amounts of noise when evaluated on the noisy targets used for training for both training (model from last epoch) and testing (model with best validation loss) data and also better when evaluated on the test data with model with best validation loss on the noiseless targets — they, however, perform generally worse on the training data with model from last epoch when evaluated on the noiseless targets while trained with the noisy targets; this suggests that the models are overfitting to the present noise. This in line with the observations from Fig. 6.22 in Section 6.3.4 where variants without dropout perform better on the training data when the model from last epoch is used and NNs with TAAFs even have much lower error than the variants without TAAFs while the models' error on the validation and test datasets have higher error without dropout regularization. We conclude from Fig. 6.22 that some form of regularization is recommended as the used 25% dropout improved the performance on the test data (this is not surprising as dropout improved performance even in the original D–GEX in [2]). Dropout is also discussed later in the context of TAAFs and the checkerboard architecture (see Sections 5.3.2 and 6.4.2).

We have also analyzed whether there is a noteworthy difference in the relative performance based on the layer configuration of the inference network in Section 6.3.3. No noteworthy changes in the pairwise comparison were observed for the tasks NN1 — NN3 while slightly diminishing tendency in the dominance of TAAFs with increasing number of weights or depth was observed for the task NN4; nevertheless, the NNs with TAAFs dominated the non-TAAF baseline variants in most of the cases even for the task NN4 as shown in Fig. 6.20.

7.2.1 *Impact of initialization of the data generation network*

Previously discussed results only focused on the tasks NN1 — NN4, where a single random initialization was used for each task; we show that the observed results are not due to a particular random initialization of the generative network but rather apply to the general class of tasks in Section 6.3.4 where multiple initializations of the generative network used in task NN3 were created and used for data generation. The results are consistent with only negligible variants across all the initialization, and the NNs with TAAFs always outperformed the non-TAAF variants in all repetitions when evaluated on the test data set when the selected model was the one with the lowest loss over the test dataset and when evaluated on the training dataset if the model from the last epoch was used.

7.2.2 *Depth and width of the data generation network*

We have also evaluated how changing the width and depth of the data generation network influences the relative performance of the NNs with TAAFs and the baselines in Section 6.3.5. Most importantly, the NNs with TAAFs dominated the baselines for all tested widths on the test set when the network checkpoint was selected by the lowest loss over the test set and also for all tested widths where the checkpoint from the last epoch was used on the train set as shown in Figs. 6.25 and 6.26. Other less important combinations of the model checkpoint and the data split are discussed in Section 6.3.5. The depth of the data generation network has almost no noteworthy impact on the relative performance of the TAAFs and baselines as shown in Section 6.3.6 with possibly small diminishing of TAAF dominance for tasks generated using deeper networks; nevertheless, the NNs with TAAFs in at least half of the tested variants for all evaluated depths of the data generation network.

7.3 TOWER AND CHECKERBOARD ARCHITECTURES

So far, we have discussed only the improvement in the activation functions to the original D–GEX, namely the transformative adaptive activation functions; however, we have also improved the architecture by using tower or checkerboard patterns for connecting blocks of neurons (see Section 5.3) instead of fully-connected layers as in the original D–GEX [2]. The tower and checkerboard architectures use interconnected blocks of fully connected layers to allow for a higher number of neurons while not increasing the overall number of trainable parameters as shown in Table 5.3 and Figs. 5.2 and 5.3. We have empirically shown that such patterns increase the performance of the network in Section 6.4. We have also introduced skip-connections in a ResNet manner (see Fig. 5.1 for a depiction of used architectures), which further improved the performance of the GE inference without introducing significant overhead.

The evaluated architectures had each 10,000 neurons with TAAFs in three hidden layers, which were split into 2 – 12 towers either with or without skip-connections. The skip-connections uniformly help as shown in Table 6.15 and Fig. 6.29 where all tested configurations with skip-connections outperformed all of the other configurations without skip-connections. The optimal number of towers differed slightly between the four tested architectures — the checkerboard and tower architectures without the skip connections plateaued after reaching four towers with only a slight decrease in performance with an increasing number of towers, whereas the variants with skip connections showed much more significant U-shaped performance curve. The tower architecture with skip-connections reached the lowest MMAE with three towers with statistically significantly better performance when compared to all other numbers of towers. The checkerboard with skip connections was the best-performing architecture and reached the lowest test error with five towers in a checkerboard pattern. This difference was also statistically significant for all tested numbers of towers, but this time with the exception of four and six towers where the dominance was not statistically significant as shown in Fig. 6.30. For the shown configurations with 2 – 12 towers, the checkerboard architecture was statistically significantly better for most of the configurations, while the difference was not statistically significant for two configurations as shown in Fig. 6.31. While direct comparison with GAN based models that were proposed concurrently with our work is not possible due to a different normalization (see Section 5.1.1.2 for details), the best overall model with TAAFs and checkerboard architecture reaches MMAE of 0.1278 which represents $1 - \frac{0.1278}{0.1637} \approx 21.9\%$ improvement over the reimplementation of the original D–GEX from [2], while the more complex GAN based approaches show $1 - \frac{0.2997}{0.3204} \approx 6.5\%$ improvement [15] and $1 - \frac{0.2897}{0.3204} \approx 9.6\%$ in [14] over the baseline D–GEX (note that these relative improvements are only rough estimates — see Section 7.1.1 for explanation). It was shown that the combination of skip-connection and the checkerboard architecture significantly improves the GE inference while being more conceptually simpler than its GAN based competitors [14, 15].

Additionally, we have run a small dropout analysis with the best-performing checkerboard model from Section 6.4 in Section 6.4.2 as the dropout rates were kept fixed throughout the previous experiments. The dropout of 25% that was used in most experiments was shown as sufficiently good, but the inference can be further improved by using a slightly lower value of 15% as shown in Fig. 6.32; the improvements are statistically significant as shown in Fig. 6.33.

7.3.1 *Practical impact of checkerboard architecture*

The same as for TAAFs, we have also shown that the improved inference performance has a practical impact on the DGE analysis in Section 6.5. The DGE analysis was run using the artificial phenotypes as described in Section 5.1.5 for sample sizes of 12 – 160 samples, which are realistic sample sizes for commonly run differential gene expression analyses. The checkerboard architecture with 5 towers and skip-connections significantly outperformed the D–GEX with TAAFo from Section 6.1, which was so far the best-performing model. The differences of all scores ($F_{0.5}$, F_1, F_2 scores, accuracy, and MCC) were statistically significant for all tested sample sizes when using the Wilcoxon signed-rank test as all the p–values were $< 10^{-8}$. To provide interpretation for the observed pairwise differences in F_1 score, the typical scenario was that CR–D–GEX with five towers reported much fewer false positive differentially expressed genes than TAAFo, sometimes at the cost of a very small increase in false negatives. The performance improvements were most noteworthy for smaller sample sizes, which is useful as these sample sizes are common in biological experiments and make running differential gene expression analyses and similar analyses difficult.

8 CONCLUSIONS

The analysis of gene expression is necessary for understanding cellular processes, disease mechanisms, and developmental pathways. The analysis was made significantly cheaper by the introduction of L1000 microarray platform that, instead of measuring the gene expression of all genes, measures the gene expression of only a handful of landmark genes and relies on computational models to infer the gene expression of the rest of the genes (see Section 3.3.3). In the beginnings, these computational models relied on linear regression [2] but were soon replaced by the neural network called D–GEX in [2].

This thesis presents several significant improvements to the original D–GEX that are conceptually simple and yet have a significant impact on the gene expression inference. The main innovation is the introduction of a novel class of adaptive activation functions called TAAF (see Section 5.2). The TAAFs introduce four adaptive parameters allowing for any horizontal and vertical scaling and translation of any inner activation function. The TAAFs improve the performance of the NNs for GE inference and also add some robustness to the choice of the activation function. Furthermore, The TAAFs generalize over 50 AFs proposed in the literature that can be considered special cases of the TAAFs (see Section 5.2.1.1). We have analyzed the TAAFs on both real microarray data of the GE inference task for the L1000 platform in Section 6.1 and several artificial regression datasets in Section 6.3 to show that TAAFs are applicable outside the omics field. Furthermore, it was shown in Section 6.2 that the improvement to the gene expression inference translates to the improvement of subsequent analyses and has, therefore, a statistically significant practical impact.

The second important improvement to the original neural network used for the GE inference was the introduction of tower and checkerboard architectures (Section 5.3) with skip-connections in a ResNet-like manner that further improve the NN with TAAFs and reach even better performance (Section 6.4); this improvement also translates to a statistical significant improvement in the subsequent analyses using the inferred data (Section 6.5).

While these improvements were shown on the gene expression inference task, they are not limited to the omics field and are applicable to more general classes of neural networks. The TAAFs generalize several activation functions that were proposed in the literature (see Section 5.2.1.1), and most of these special cases were shown in many different settings outside the gene expression inference and omics in general. Last but not least, we have presented a comprehensive list of 400 activation functions to simplify further research in the area and to help avoid repeated proposals of the activation functions already present in the literature.

8.1 FUTURE WORKS

Despite the scope of the presented work, there are many areas in which the work might be expanded in the future. While the overview of activation functions present in the literature (see Sections 4.2 and 4.3) can be updated indefinitely as new activation functions are proposed, there are many other directions in which the presented findings will be explored in the future.

8.1.1 *Gaining insights into TAAFs*

While this work demonstrated empirically that the TAAFs lead to improved performance for the GE inference and other methods, further theoretical insights explaining the performance gains are needed. A method to visually demonstrate the benefits of skip connections by visualizing low dimensional representation of the optimization landscape was presented in [2150]; attempts to use it for the evaluation of the NNs with and without TAAFs were presented in [2151]; nevertheless, the experimental evaluations of the hypothesis that TAAFs also simplify the optimization landscape remain inconclusive. Therefore, further research is needed into the effects the usage of TAAFs has on a neural network.

8.1.2 *Analysis of redundancy*

The TAAF parameter β is redundant in the sense that it does not add anything to the expressivity of the AF; nevertheless, it was shown that its inclusion statistically significantly improves the performance in Section 6.1.4 — some perceived redundancy might therefore be beneficial for the optimization. One future direction, therefore, includes the analysis of why the redundancy helps and whether there are other similar cases that could improve the optimization process.

8.1.3 *Simplification of usability*

While we have published the codes of the TAAFs in a public repository, usage of the presented models might still be difficult for researchers from other fields. Therefore, future works also include either a runnable application or an online inference tool where a researcher might just drop measured L1000 profiles, and the full inferred profiles will be returned.

8.1.4 *Extending applications*

The previous points focused on further improving the GE inference aspect of the presented work; however, there are many areas in which the presented TAAFs and checkerboard architectures might be applied inside and also outside the omics field. While results outside of the omics field were already presented in Section 6.3 Exploring TAAF performance using artificial data, there might be many other applications where the presented findings might improve the performance of currently used models. The TAAFs generalize many AFs present in the literature; therefore the simplest direction would be to evaluate the TAAFs on the same data as its special cases — e.g., the ABReLU (Section 4.2.6.42) was used to improve face retrieval using datasets PaSC [2152], LFW [2153], PubFig [2154], FERET [2155, 2156], AR [2157, 2158], ExYaleB [2159, 2160] and PolyU-NIRFD [2161], PShELU (Section 4.3.1.56) was used on the CIFAR-100 [243] dataset in [1078], and the PSTanh (Section 4.3.15.2) in *Capsule networks* [2162] for brain tumor classification dataset [2163] in [688] but also used MNIST dataset [45], Fashion MNIST [950] and the CIFAR-10 and CIFAR-100 datasets [243].

8.1.5 *Insights into tower and checkerboard architectures*

The number of towers in tower and checkerboard architectures influences the performance of the neural network as empirically demonstrated in Section 6.4. It is highly likely that the optimal number of towers is problem-dependent and also architecture-dependent. It is likely that the number of optimal towers would be different if a different number of neurons were used or if the number of targets were different. Moreover, the optimal number of towers also differs between the used architectures (see Fig. 6.29). Further analysis is needed to provide general recommendations on the number of towers to avoid costly grid searches.

8.1.6 *Generalization of the checkerboard architecture*

Other directions include a generalization of the checkerboard architecture — the checkerboard architecture divided each layer in each tower into halves and reconnected those in a certain pattern; however, dividing the layers into multiple folds and using more complex reconnection patterns might lead to networks with better performance and thus further improving the gene expression inference.

8.1.7 *Leaving the blocks behind*

The tower and checkerboard architectures use larger interconnected blocks of neurons to be able to use existing accelerators such as GPUs and libraries for efficient computation; however, there are also approaches that allow for efficient computation even with sparse matrices such as *Sparse Tensor Cores* by Nvidia [2164] or the WASAP-SGD optimization approach in [1172]. These could be used to extend the concept behind checkerboard architecture even further by no longer requiring that neurons be grouped in sufficiently large blocks for efficient computations.

8.1.8 *Dual transformative adaptive activation function*

While the TAAFs generalize many existing activation functions, there is a class of activation functions that cannot be generalized by the TAAFs and yet it is simple enough — these functions usually have different scaling for positive inputs (e.g., LReLU), see Section 5.2.1.2 for discussion of these functions. The TAAFs could be extended in dual transformative adaptive activation functions (DTAAFs) that would have the scaling parameters different for positive and negative inputs or, more generally, for inputs below and above a certain threshold. While the final formulation of DTAAFs would still be sufficiently simple, it might lead to even better performance as this concept works well for many existing activation functions.

8.1.9 *Generalized dual transformative adaptive activation function*

The DTAAF concept can be further generalized by allowing different inner functions for each piecewise definition, resulting in generalized dual transformative adaptive activation function (GDTAAF). While the resulting AAF would not be a simple encapsulation of the inner activation function, the resulting AAF would be able to generalize more AFs present in the literature and could have even higher performance. The GDTAAF would extend the approach presented in [815] where the authors use different AFs for positive and negative inputs.

8.1.10 *GAN-based approaches for GE inference*

Dizaji, Wang, and Huang used GAN-based approaches to improve the gene expression inference from the L1000 data in their works [14, 15]. As already discussed in Section 7.1, their results are not directly comparable with the results presented in this work; nevertheless, the rough estimate of performance shows inferior performance to TAAFs and checkerboard architectures. However, the presented approaches are not mutually exclusive with the methods used in [14, 15]. Therefore, one future direction is to use both TAAFs and maybe checkerboard architectures in the GAN based approaches from [14, 15] to see whether synergies are possible and such combination will improve the GE inference performance even further.

8.1.11 *TAAF initialization*

The initialization of the TAAF parameters has a significant influence on the overall performance, as shown in Sections 6.1.2 and 7.1.1. While the Section 6.1.2 has shown that even a different fixed initialization, where the same TAAF parameters are set to the same value for all neurons, can improve the performance, it is very likely that the performance can be improved even further by sampling the initial values of parameters from some distribution similarly as is done for the weights of the connections between neurons. Therefore, future work should evaluate various initialization of the TAAF parameters as it will most likely lead to an improved performance.

BIBLIOGRAPHY

[1] A. Subramanian et al. "A Next Generation Connectivity Map: L1000 Platform and the First 1, 000, 000 Profiles." In: *Cell* 171.6 (Nov. 2017), 1437–1452.e17. DOI: 10.1016/j.cell.2017.10.049. URL: https://doi.org/10.1016/j.cell.2017.10.049 (cit. on pp. xlviii, 1, 2, 38, 39).

[2] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie. "Gene expression inference with deep learning." In: *Bioinformatics* 32.12 (Feb. 2016), pp. 1832–1839. DOI: 10.1093/bioinformatics/btw074. URL: https://doi.org/10.1093/bioinformatics/btw074 (cit. on pp. xlvi, 2, 5, 6, 9, 17, 38, 41, 42, 187–190, 222, 229, 251, 267, 275–277, 279, 280, 282–285, 287).

[3] H.-C. Kuo, C.-T. Yao, B.-Y. Liao, M.-P. Weng, F. Dong, Y.-C. Hsu, and C.-M. Hung. "Weak gene–gene interaction facilitates the evolution of gene expression plasticity." In: *BMC Biology* 21.1 (Mar. 2023). ISSN: 1741-7007. DOI: 10.1186/s12915-023-01558-6. URL: http://dx.doi.org/10.1186/s12915-023-01558-6 (cit. on p. 2).

[4] E. B. Josephs, S. I. Wright, J. R. Stinchcombe, and D. J. Schoen. "The Relationship between Selection, Network Connectivity, and Regulatory Variation within a Population of Capsella grandiflora." In: *Genome Biology and Evolution* 9.4 (Apr. 2017), pp. 1099–1109. ISSN: 1759-6653. DOI: 10.1093/gbe/evx068. URL: http://dx.doi.org/10.1093/gbe/evx068 (cit. on p. 2).

[5] N. Mähler, J. Wang, B. K. Terebieniec, P. K. Ingvarsson, N. R. Street, and T. R. Hvidsten. "Gene co-expression network connectivity is an important determinant of selective constraint." In: *PLOS Genetics* 13.4 (Apr. 2017). Ed. by N. M. Springer, e1006402. ISSN: 1553-7404. DOI: 10.1371/journal.pgen.1006402. URL: http://dx.doi.org/10.1371/journal.pgen.1006402 (cit. on p. 2).

[6] A. Fakhry, R. Khafagy, and A.-A. Ludl. *GENER: A Parallel Layer Deep Learning Network To Detect Gene-Gene Interactions From Gene Expression Data*. 2023. DOI: 10.48550/ARXIV.2310.03611. URL: https://arxiv.org/abs/2310.03611 (cit. on pp. 2, 46).

[7] F. Emmert-Streib, M. Dehmer, and B. Haibe-Kains. "Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks." In: *Frontiers in Cell and Developmental Biology* 2 (Aug. 2014). ISSN: 2296-634X. DOI: 10.3389/fcell.2014.00038. URL: http://dx.doi.org/10.3389/fcell.2014.00038 (cit. on p. 2).

[8] K. Van Steen. "Travelling the world of gene-gene interactions." In: *Briefings in Bioinformatics* 13.1 (Mar. 2011), pp. 1–19. ISSN: 1477-4054. DOI: 10.1093/bib/bbr012. URL: http://dx.doi.org/10.1093/bib/bbr012 (cit. on p. 2).

[9] V. Kunc and J. Kléma. "On tower and checkerboard neural network architectures for gene expression inference." In: *BMC Genomics* 21.S5 (Dec. 2020). DOI: 10.1186/s12864-020-06821-6. URL: https://doi.org/10.1186/s12864-020-06821-6 (cit. on pp. 3, 222).

[10] V. Kunc and J. Kléma. "On transformative adaptive activation functions in neural networks for gene expression inference." In: *PLOS ONE* 16.1 (Jan. 2021). Ed. by H. Fröhlich, e0243915. DOI: 10.1371/journal.pone.0243915. URL: https://doi.org/10.1371/journal.pone.0243915 (cit. on pp. 3, 42, 193, 222, 225, 267, 268, 270, 278).

[11] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark." In: *Neurocomputing* 503 (Sept. 2022), pp. 92–108. DOI: 10.1016/j.neucom.2022.06.111. URL: https://doi.org/10.1016/j.neucom.2022.06.111 (cit. on pp. 3, 15, 48–51, 53–55, 58, 64, 68, 69, 71–73, 76, 78, 80–82, 87, 89, 105, 108, 109, 113, 114, 117, 120, 121, 126, 128–130, 150, 151, 153, 159, 169, 170).

[12] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete. "A survey on modern trainable activation functions." In: *Neural Networks* 138 (June 2021), pp. 14–32. DOI: 10.1016/j.neunet.2021.01.026. URL: https://doi.org/10.1016/j.neunet.2021.01.026 (cit. on pp. 3, 49, 50).

[13] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90. URL: https://doi.org/10.1109/cvpr.2016.90 (cit. on pp. 5, 17, 65, 117, 120, 149, 171, 174, 176, 223, 278).

[14] X. Wang, K. G. Dizaji, and H. Huang. "Conditional generative adversarial network for gene expression inference." In: *Bioinformatics* 34.17 (Sept. 2018), pp. i603–i611. DOI: 10.1093/bioinformatics/bty563. URL: https://doi.org/10.1093/bioinformatics/bty563 (cit. on pp. 6, 41, 42, 183, 276, 282, 285, 290).

[15] K. G. Dizaji, X. Wang, and H. Huang. "Semi-Supervised Generative Adversarial Network for Gene Expression Inference." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD'18*. New York: ACM Press, 2018. DOI: 10.1145/3219819.3220114. URL: https://doi.org/10.1145/3219819.3220114 (cit. on pp. 6, 41, 42, 276, 282, 285, 290).

[16] J. Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: https://doi.org/10.1016/j.neunet.2014.09.003 (cit. on pp. 9–13).

[17] K. J. Cios. "Deep Neural Networks—A Brief History." In: *Advances in Data Analysis with Computational Intelligence Methods*. Springer International Publishing, Sept. 2017, pp. 183–200. DOI: 10.1007/978-3-319-67946-4_7. URL: https://doi.org/10.1007/978-3-319-67946-4_7 (cit. on p. 9).

[18] J. Schmidhuber. "Deep Learning." In: *Encyclopedia of Machine Learning and Data Mining*. Springer US, 2016, pp. 1–11. DOI: 10.1007/978-1-4899-7502-7_909-1. URL: https://doi.org/10.1007/978-1-4899-7502-7_909-1 (cit. on p. 9).

[19] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/bf02478259. URL: https://doi.org/10.1007%2Fbf02478259 (cit. on p. 9).

[20] D. Hebb. *The Organization of Behavior*. New York: Wiley, 1949 (cit. on p. 9).

[21] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519. URL: https://doi.org/10.1037/h0042519 (cit. on pp. 9, 50).

[22] A. G. Ivakhnenko. "Polynomial Theory of Complex Systems." In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-1.4 (Oct. 1971), pp. 364–378. DOI: 10.1109/tsmc.1971.4308320. URL: https://doi.org/10.1109%2Ftsmc.1971.4308320 (cit. on pp. 9, 10).

[23] A. Ivakhnenko. "Heuristic self-organization in problems of engineering cybernetics." In: *Automatica* 6.2 (Mar. 1970), pp. 207–219. DOI: 10.1016/0005-1098(70)90092-0. URL: https://doi.org/10.1016/0005-1098(70)90092-0 (cit. on p. 9).

[24] R. Mehra. "Group method of data handling (GMDH): Review and experience." In: *1977 IEEE Conference on Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications*. IEEE, Dec. 1977. DOI: 10.1109/cdc.1977.271540. URL: https://doi.org/10.1109/cdc.1977.271540 (cit. on p. 9).

[25] R. D. Joseph. "Contribution to Perceptron Theory." PhD thesis. University of Cornell, 1961 (cit. on p. 10).

[26] S. Viglione. "4 Applications of Pattern Recognition Technology." In: *Mathematics in Science and Engineering*. Elsevier, 1970, pp. 115–162. DOI: 10.1016/s0076-5392(08)60492-0. URL: https://doi.org/10.1016/s0076-5392(08)60492-0 (cit. on p. 10).

[27] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. DOI: 10.1007/bf00344251. URL: https://doi.org/10.1007/bf00344251 (cit. on p. 10).

[28] A. E. Bryson. "A gradient method for optimizing multi-stage allocation processes." In: *Proc. Harvard Univ. Symposium on digital computers and their applications*. Vol. 72. 1961, p. 22 (cit. on p. 10).

[29] A. E. Bryson, Y.-C. Ho, and G. M. Siouris. "Applied Optimal Control: Optimization, Estimation, and Control." In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.6 (1979), pp. 366–367. DOI: 10.1109/tsmc.1979.4310229. URL: https://doi.org/10.1109/tsmc.1979.4310229 (cit. on p. 10).

[30] H. J. Kelley. "Gradient Theory of Optimal Flight Paths." In: *ARS Journal* 30.10 (Oct. 1960), pp. 947–954. DOI: 10.2514/8.5282. URL: https://doi.org/10.2514/8.5282 (cit. on p. 10).

[31] S. Dreyfus. "The numerical solution of variational problems." In: *Journal of Mathematical Analysis and Applications* 5.1 (Aug. 1962), pp. 30–45. DOI: 10.1016/0022-247x(62)90004-5. URL: https://doi.org/10.1016/0022-247x(62)90004-5 (cit. on p. 10).

[32] M. Minsky and S. A. Papert. *Perceptrons*. Cambridge: MIT press, 1969 (cit. on p. 10).

[33] S. Linnainmaa. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors." PhD thesis. Master's Thesis (in Finnish), Univ. Helsinki, 1970 (cit. on p. 11).

[34] S. Dreyfus. "The computational solution of optimal control problems with time lag." In: *IEEE Transactions on Automatic Control* 18.4 (Aug. 1973), pp. 383–385. DOI: 10.1109/tac.1973.1100330. URL: https://doi.org/10.1109/tac.1973.1100330 (cit. on p. 11).

[35] P. J. Werbos. “Applications of advances in nonlinear sensitivity analysis.” In: *System Modeling and Optimization*. Springer-Verlag, 1981, pp. 762–770. DOI: 10.1007/bfb0006203. URL: https://doi.org/10.1007/bfb0006203 (cit. on p. 11).

[36] Y. Lecun. “A theoretical framework for back-propagation.” In: *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*. Ed. by D. Touretzky, G. Hinton, and T. Sejnowski. Morgan Kaufmann, 1988, pp. 21–28. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf (cit. on p. 11).

[37] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition.” In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541. URL: https://doi.org/10.1162/neco.1989.1.4.541 (cit. on p. 11).

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1.” In: ed. by D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: http://www.cnbc.cmu.edu/~plaut/IntroPDP/papers/RumelhartETAL86.backprop.pdf (cit. on pp. 11, 13, 25).

[39] A. N. Kolmogorov. “On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition.” In: *Doklady Akademii Nauk*. Vol. 114. 5. Russian Academy of Sciences. 1957, pp. 953–956 (cit. on p. 11).

[40] K. Hornik, M. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators.” In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8. URL: https://doi.org/10.1016/0893-6080(89)90020-8 (cit. on p. 11).

[41] R. Hecht-Nielsen. “Theory of the backpropagation neural network.” In: *International Joint Conference on Neural Networks*. IEEE, 1989. DOI: 10.1109/ijcnn.1989.118638. URL: https://doi.org/10.1109/ijcnn.1989.118638 (cit. on p. 11).

[42] K. Levenberg. “A method for the solution of certain non-linear problems in least squares.” In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. DOI: 10.1090/qam/10666. URL: https://doi.org/10.1090/qam/10666 (cit. on p. 11).

[43] D. W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters.” In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (June 1963), pp. 431–441. DOI: 10.1137/0111030. URL: https://doi.org/10.1137/0111030 (cit. on p. 11).

[44] M. Riedmiller and H. Braun. “A direct adaptive method for faster backpropagation learning: the RPROP algorithm.” In: *IEEE International Conference on Neural Networks*. IEEE, 1993. DOI: 10.1109/icnn.1993.298623. URL: https://doi.org/10.1109/icnn.1993.298623 (cit. on p. 11).

[45] L. Deng. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web].” In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), pp. 141–142. DOI: 10.1109/msp.2012.2211477. URL: https://doi.org/10.1109/msp.2012.2211477 (cit. on pp. 11, 61, 64, 89, 91, 119, 130, 140, 149, 152, 160, 289).

[46] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 12, 13, 17, 18, 20–22, 24, 25, 48, 50, 51, 68, 105).

[47] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991 (cit. on p. 12).

[48] O. Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge.” In: *International Journal of Computer Vision* 115.3 (Apr. 2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y (cit. on pp. 12, 23, 58, 65, 124, 140, 186).

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on pp. 12, 68, 186).

[50] M. D. Zeiler and R. Fergus. “Visualizing and Understanding Convolutional Networks.” In: *CoRR* abs/1311.2901 (2013). URL: http://arxiv.org/abs/1311.2901 (cit. on p. 12).

[51] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: *CoRR* abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556 (cit. on pp. 12, 17, 82, 185).

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions.” In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. DOI: 10.1109/cvpr.2015.7298594. URL: https://doi.org/10.1109/cvpr.2015.7298594 (cit. on pp. 12, 13, 17, 20, 65, 171).

[53] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. "A survey of the recent architectures of deep convolutional neural networks." In: *Artificial Intelligence Review* 53.8 (Apr. 2020), pp. 5455–5516. DOI: 10.1007/s10462-020-09825-6. URL: https://doi.org/10.1007/s10462-020-09825-6 (cit. on p. 13).

[54] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (Feb. 2017). DOI: 10.1609/aaai.v31i1.11231. URL: https://doi.org/10.1609/aaai.v31i1.11231 (cit. on pp. 13, 171, 174).

[55] S. Zagoruyko and N. Komodakis. "Wide Residual Networks." In: *Procedings of the British Machine Vision Conference 2016*. British Machine Vision Association, 2016. DOI: 10.5244/c.30.87. URL: https://doi.org/10.5244/c.30.87 (cit. on pp. 13, 130, 149, 176, 177).

[56] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. "Aggregated Residual Transformations for Deep Neural Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.634. URL: https://doi.org/10.1109/cvpr.2017.634 (cit. on p. 13).

[57] D. Han, J. Kim, and J. Kim. "Deep Pyramidal Residual Networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017. URL: https://openaccess.thecvf.com/content_cvpr_2017/papers/Han_Deep_Pyramidal_Residual_CVPR_2017_paper.pdf (cit. on p. 13).

[58] F. Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.195. URL: https://doi.org/10.1109/cvpr.2017.195 (cit. on p. 13).

[59] X. Zhang, Z. Li, C. C. Loy, and D. Lin. "PolyNet: A Pursuit of Structural Diversity in Very Deep Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.415. URL: https://doi.org/10.1109/cvpr.2017.415 (cit. on p. 13).

[60] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer International Publishing, 2023. DOI: 10.1007/978-3-031-29642-0. URL: https://doi.org/10.1007/978-3-031-29642-0 (cit. on p. 13).

[61] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar. "A Survey on Deep Learning." In: *ACM Computing Surveys* 51.5 (Sept. 2018), pp. 1–36. DOI: 10.1145/3234150. URL: https://doi.org/10.1145/3234150 (cit. on p. 13).

[62] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on p. 13).

[63] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. "Deep learning for visual understanding: A review." In: *Neurocomputing* 187 (Apr. 2016), pp. 27–48. DOI: 10.1016/j.neucom.2015.09.116. URL: https://doi.org/10.1016/j.neucom.2015.09.116 (cit. on p. 13).

[64] A. Shrestha and A. Mahmood. "Review of Deep Learning Algorithms and Architectures." In: *IEEE Access* 7 (2019), pp. 53040–53065. DOI: 10.1109/access.2019.2912200. URL: https://doi.org/10.1109/access.2019.2912200 (cit. on pp. 13, 14).

[65] W. Rawat and Z. Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review." In: *Neural Computation* 29.9 (Sept. 2017), pp. 2352–2449. DOI: 10.1162/neco_a_00990. URL: https://doi.org/10.1162/neco_a_00990 (cit. on p. 13).

[66] S. Min et al. "Deep learning in bioinformatics." In: *Briefings in Bioinformatics* (July 2016), bbw068. DOI: 10.1093/bib/bbw068. URL: https://doi.org/10.1093/bib/bbw068 (cit. on pp. 13, 41).

[67] C. Angermueller et al. "Deep learning for computational biology." In: *Molecular Systems Biology* 12.7 (July 2016), p. 878. DOI: 10.15252/msb.20156651. URL: https://doi.org/10.15252/msb.20156651 (cit. on pp. 13, 41).

[68] M. Zamani and S. C. Kremer. "Neural Networks in Bioinformatics." In: *Intelligent Systems Reference Library*. Springer Berlin Heidelberg, 2013, pp. 505–525. DOI: 10.1007/978-3-642-36657-4_15. URL: https://doi.org/10.1007/978-3-642-36657-4_15 (cit. on p. 13).

[69] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. V. Essen, A. A. S. Awwal, and V. K. Asari. "A State-of-the-Art Survey on Deep Learning Theory and Architectures." In: *Electronics* 8.3 (Mar. 2019), p. 292. DOI: 10.3390/electronics8030292. URL: https://doi.org/10.3390/electronics8030292 (cit. on pp. 13, 48, 171).

[70] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications." In: *AI Open* 1 (2020), pp. 57–81. DOI: 10.1016/j.aiopen.2021.01.001. URL: https://doi.org/10.1016/j.aiopen.2021.01.001 (cit. on p. 13).

[71] D. Bacciu, F. Errica, A. Micheli, and M. Podda. "A gentle introduction to deep learning for graphs." In: *Neural Networks* 129 (Sept. 2020), pp. 203–221. DOI: 10.1016/j.neunet.2020.06.006. URL: https://doi.org/10.1016/j.neunet.2020.06.006 (cit. on p. 13).

[72] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. "Deep Learning for 3D Point Clouds: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (Dec. 2021), pp. 4338–4364. DOI: 10.1109/tpami.2020.3005434. URL: https://doi.org/10.1109/tpami.2020.3005434 (cit. on p. 13).

[73] Z. Wang, J. Chen, and S. C. H. Hoi. "Deep Learning for Image Super-Resolution: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.10 (Oct. 2021), pp. 3365–3387. DOI: 10.1109/tpami.2020.2982166. URL: https://doi.org/10.1109/tpami.2020.2982166 (cit. on p. 13).

[74] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. "Deep learning on image denoising: An overview." In: *Neural Networks* 131 (Nov. 2020), pp. 251–275. DOI: 10.1016/j.neunet.2020.07.025. URL: https://doi.org/10.1016/j.neunet.2020.07.025 (cit. on p. 13).

[75] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez. "A survey on deep learning in medical image analysis." In: *Medical Image Analysis* 42 (Dec. 2017), pp. 60–88. DOI: 10.1016/j.media.2017.07.005. URL: https://doi.org/10.1016/j.media.2017.07.005 (cit. on p. 13).

[76] A. Fourcade and R. Khonsari. "Deep learning in medical image analysis: A third eye for doctors." In: *Journal of Stomatology, Oral and Maxillofacial Surgery* 120.4 (Sept. 2019), pp. 279–288. DOI: 10.1016/j.jormas.2019.06.002. URL: https://doi.org/10.1016/j.jormas.2019.06.002 (cit. on p. 13).

[77] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu. "Object Detection With Deep Learning: A Review." In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (Nov. 2019), pp. 3212–3232. DOI: 10.1109/tnnls.2018.2876865. URL: https://doi.org/10.1109/tnnls.2018.2876865 (cit. on p. 13).

[78] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. "Efficient Processing of Deep Neural Networks: A Tutorial and Survey." In: *Proceedings of the IEEE* 105.12 (Dec. 2017), pp. 2295–2329. DOI: 10.1109/jproc.2017.2761740. URL: https://doi.org/10.1109/jproc.2017.2761740 (cit. on p. 13).

[79] A. Kamilaris and F. X. Prenafeta-Boldú. "Deep learning in agriculture: A survey." In: *Computers and Electronics in Agriculture* 147 (Apr. 2018), pp. 70–90. DOI: 10.1016/j.compag.2018.02.016. URL: https://doi.org/10.1016/j.compag.2018.02.016 (cit. on p. 13).

[80] Q. Zhang, L. T. Yang, Z. Chen, and P. Li. "A survey on deep learning for big data." In: *Information Fusion* 42 (July 2018), pp. 146–157. DOI: 10.1016/j.inffus.2017.10.006. URL: https://doi.org/10.1016/j.inffus.2017.10.006 (cit. on p. 13).

[81] P. Meyer, V. Noblet, C. Mazzara, and A. Lallement. "Survey on deep learning for radiotherapy." In: *Computers in Biology and Medicine* 98 (July 2018), pp. 126–146. DOI: 10.1016/j.compbiomed.2018.05.018. URL: https://doi.org/10.1016/j.compbiomed.2018.05.018 (cit. on p. 13).

[82] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. "Convolutional neural networks: an overview and application in radiology." In: *Insights into Imaging* 9.4 (June 2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9 (cit. on p. 13).

[83] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data." In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. DOI: 10.1109/msp.2017.2693418. URL: https://doi.org/10.1109/msp.2017.2693418 (cit. on p. 13).

[84] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. "Deep Learning for Computer Vision: A Brief Review." In: *Computational Intelligence and Neuroscience* 2018 (2018), pp. 1–13. DOI: 10.1155/2018/7068349. URL: https://doi.org/10.1155/2018/7068349 (cit. on p. 13).

[85] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. "Deep Reinforcement Learning: A Brief Survey." In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), pp. 26–38. DOI: 10.1109/msp.2017.2743240. URL: https://doi.org/10.1109/msp.2017.2743240 (cit. on p. 13).

[86] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. "State-of-the-art in artificial neural network applications: A survey." In: *Heliyon* 4.11 (Nov. 2018), e00938. DOI: 10.1016/j.heliyon.2018.e00938. URL: https://doi.org/10.1016/j.heliyon.2018.e00938 (cit. on p. 13).

[87] A. Tealab. "Time series forecasting using artificial neural networks methodologies: A systematic review." In: *Future Computing and Informatics Journal* 3.2 (Dec. 2018), pp. 334–340. DOI: 10.1016/j.fcij.2018.10.003. URL: https://doi.org/10.1016/j.fcij.2018.10.003 (cit. on p. 13).

[88] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. "Physics-informed machine learning." In: *Nature Reviews Physics* 3.6 (May 2021), pp. 422–440. DOI: 10.1038/s42254-021-00314-5. URL: https://doi.org/10.1038/s42254-021-00314-5 (cit. on p. 13).

[89] A. S. Lundervold and A. Lundervold. "An overview of deep learning in medical imaging focusing on MRI." In: *Zeitschrift für Medizinische Physik* 29.2 (May 2019), pp. 102–127. DOI: 10.1016/j.zemedi.2018.11.002. URL: https://doi.org/10.1016/j.zemedi.2018.11.002 (cit. on p. 13).

[90] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke. "The rise of deep learning in drug discovery." In: *Drug Discovery Today* 23.6 (June 2018), pp. 1241–1250. DOI: 10.1016/j.drudis.2018.01.039. URL: https://doi.org/10.1016/j.drudis.2018.01.039 (cit. on p. 13).

[91] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G. B. Kim, J. B. Seo, and N. Kim. "Deep Learning in Medical Imaging: General Overview." In: *Korean Journal of Radiology* 18.4 (2017), p. 570. DOI: 10.3348/kjr.2017.18.4.570. URL: https://doi.org/10.3348/kjr.2017.18.4.570 (cit. on p. 13).

[92] Z. Akkus, A. Galimzianova, A. Hoogi, D. L. Rubin, and B. J. Erickson. "Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions." In: *Journal of Digital Imaging* 30.4 (June 2017), pp. 449–459. DOI: 10.1007/s10278-017-9983-4. URL: https://doi.org/10.1007/s10278-017-9983-4 (cit. on p. 13).

[93] J. Chen and X. Ran. "Deep Learning With Edge Computing: A Review." In: *Proceedings of the IEEE* 107.8 (Aug. 2019), pp. 1655–1674. DOI: 10.1109/jproc.2019.2921977. URL: https://doi.org/10.1109/jproc.2019.2921977 (cit. on p. 13).

[94] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan. "Medical Image Analysis using Convolutional Neural Networks: A Review." In: *Journal of Medical Systems* 42.11 (Oct. 2018). DOI: 10.1007/s10916-018-1088-1. URL: https://doi.org/10.1007/s10916-018-1088-1 (cit. on p. 13).

[95] J. Amin, M. Sharif, A. Haldorai, M. Yasmin, and R. S. Nayak. "Brain tumor detection and classification using machine learning: a comprehensive survey." In: *Complex & Intelligent Systems* 8.4 (Nov. 2021), pp. 3161–3183. DOI: 10.1007/s40747-021-00563-y. URL: https://doi.org/10.1007/s40747-021-00563-y (cit. on p. 13).

[96] M. I. Razzak, S. Naz, and A. Zaib. "Deep Learning for Medical Image Processing: Overview, Challenges and the Future." In: *Lecture Notes in Computational Vision and Biomechanics*. Springer International Publishing, Nov. 2017, pp. 323–350. DOI: 10.1007/978-3-319-65981-7_12. URL: https://doi.org/10.1007/978-3-319-65981-7_12 (cit. on p. 13).

[97] S. Suganyadevi, V. Seethalakshmi, and K. Balasamy. "A review on deep learning in medical image analysis." In: *International Journal of Multimedia Information Retrieval* 11.1 (Sept. 2021), pp. 19–38. DOI: 10.1007/s13735-021-00218-1. URL: https://doi.org/10.1007/s13735-021-00218-1 (cit. on p. 13).

[98] H. A. Helaly, M. Badawy, and A. Y. Haikal. "A review of deep learning approaches in clinical and healthcare systems based on medical image analysis." In: *Multimedia Tools and Applications* (Sept. 2023). DOI: 10.1007/s11042-023-16605-1. URL: https://doi.org/10.1007/s11042-023-16605-1 (cit. on p. 13).

[99] E. S. Kumar and C. S. Bindu. "Medical Image Analysis Using Deep Learning: A Systematic Literature Review." In: *Emerging Technologies in Computer Engineering: Microservices in Big Data Analytics*. Springer Singapore, 2019, pp. 81–97. DOI: 10.1007/978-981-13-8300-7_8. URL: https://doi.org/10.1007/978-981-13-8300-7_8 (cit. on p. 13).

[100] A. M. Hafiz and G. M. Bhat. "A Survey of Deep Learning Techniques for Medical Diagnosis." In: *Information and Communication Technology for Sustainable Development*. Springer Singapore, June 2019, pp. 161–170. DOI: 10.1007/978-981-13-7166-0_16. URL: https://doi.org/10.1007/978-981-13-7166-0_16 (cit. on p. 13).

[101] Z. Liu, L. Tong, L. Chen, Z. Jiang, F. Zhou, Q. Zhang, X. Zhang, Y. Jin, and H. Zhou. "Deep learning based brain tumor segmentation: a survey." In: *Complex & Intelligent Systems* 9.1 (July 2022), pp. 1001–1026. DOI: 10.1007/s40747-022-00815-5. URL: https://doi.org/10.1007/s40747-022-00815-5 (cit. on p. 13).

[102] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. "Pre-trained models for natural language processing: A survey." In: *Science China Technological Sciences* 63.10 (Sept. 2020), pp. 1872–1897. DOI: 10.1007/s11431-020-1647-3. URL: https://doi.org/10.1007/s11431-020-1647-3 (cit. on p. 13).

[103] E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert, and D. V. Valen. "Deep learning for cellular image analysis." In: *Nature Methods* 16.12 (May 2019), pp. 1233–1246. DOI: 10.1038/s41592-019-0403-1. URL: https://doi.org/10.1038/s41592-019-0403-1 (cit. on p. 13).

[104] L. Jing and Y. Tian. “Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 4037–4058. DOI: 10.1109/tpami.2020.2992393. URL: https://doi.org/10.1109/tpami.2020.2992393 (cit. on p. 13).

[105] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao. “Deep Learning for Single Image Super-Resolution: A Brief Review.” In: *IEEE Transactions on Multimedia* 21.12 (Dec. 2019), pp. 3106–3121. DOI: 10.1109/tmm.2019.2919431. URL: https://doi.org/10.1109/tmm.2019.2919431 (cit. on p. 13).

[106] G. Eraslan, Ž. Avsec, J. Gagneur, and F. J. Theis. “Deep learning: new computational modelling techniques for genomics.” In: *Nature Reviews Genetics* 20.7 (Apr. 2019), pp. 389–403. DOI: 10.1038/s41576-019-0122-6. URL: https://doi.org/10.1038/s41576-019-0122-6 (cit. on p. 13).

[107] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida. “Deep learning in spiking neural networks.” In: *Neural Networks* 111 (Mar. 2019), pp. 47–63. DOI: 10.1016/j.neunet.2018.12.002. URL: https://doi.org/10.1016/j.neunet.2018.12.002 (cit. on p. 13).

[108] H. Wang, Z. Lei, X. Zhang, B. Zhou, and J. Peng. “A review of deep learning for renewable energy forecasting.” In: *Energy Conversion and Management* 198 (Oct. 2019), p. 111799. DOI: 10.1016/j.enconman.2019.111799. URL: https://doi.org/10.1016/j.enconman.2019.111799 (cit. on p. 13).

[109] G. B. Goh, N. O. Hodas, and A. Vishnu. “Deep learning for computational chemistry.” In: *Journal of Computational Chemistry* 38.16 (Mar. 2017), pp. 1291–1307. DOI: 10.1002/jcc.24764. URL: https://doi.org/10.1002/jcc.24764 (cit. on p. 13).

[110] J. E. Ball, D. T. Anderson, and C. S. Chan. “Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community.” In: *Journal of Applied Remote Sensing* 11.04 (Sept. 2017), p. 1. DOI: 10.1117/1.jrs.11.042609. URL: https://doi.org/10.1117/1.jrs.11.042609 (cit. on p. 13).

[111] T. van Klompenburg, A. Kassahun, and C. Catal. “Crop yield prediction using machine learning: A systematic literature review.” In: *Computers and Electronics in Agriculture* 177 (Oct. 2020), p. 105709. DOI: 10.1016/j.compag.2020.105709. URL: https://doi.org/10.1016/j.compag.2020.105709 (cit. on p. 13).

[112] B. Sahiner, A. Pezeshk, L. M. Hadjiiski, X. Wang, K. Drukker, K. H. Cha, R. M. Summers, and M. L. Giger. “Deep learning in medical imaging and radiation therapy.” In: *Medical Physics* 46.1 (Nov. 2018). DOI: 10.1002/mp.13264. URL: https://doi.org/10.1002/mp.13264 (cit. on p. 13).

[113] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso. “Deep Learning for Time Series Forecasting: A Survey.” In: *Big Data* 9.1 (Feb. 2021), pp. 3–21. ISSN: 2167-647X. DOI: 10.1089/big.2020.0159. URL: http://dx.doi.org/10.1089/big.2020.0159 (cit. on p. 13).

[114] M. Zhou, N. Duan, S. Liu, and H.-Y. Shum. “Progress in Neural NLP: Modeling, Learning, and Reasoning.” In: *Engineering* 6.3 (Mar. 2020), pp. 275–290. ISSN: 2095-8099. DOI: 10.1016/j.eng.2019.12.014. URL: http://dx.doi.org/10.1016/j.eng.2019.12.014 (cit. on p. 13).

[115] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao. “Review of Image Classification Algorithms Based on Convolutional Neural Networks.” In: *Remote Sensing* 13.22 (Nov. 2021), p. 4712. ISSN: 2072-4292. DOI: 10.3390/rs13224712. URL: http://dx.doi.org/10.3390/rs13224712 (cit. on p. 13).

[116] J. Maurício, I. Domingues, and J. Bernardino. “Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review.” In: *Applied Sciences* 13.9 (Apr. 2023), p. 5521. ISSN: 2076-3417. DOI: 10.3390/app13095521. URL: http://dx.doi.org/10.3390/app13095521 (cit. on p. 13).

[117] I. Santos, L. Castro, N. Rodriguez-Fernandez, Á. Torrente-Patiño, and A. Carballal. “Artificial Neural Networks and Deep Learning in the Visual Arts: a review.” In: *Neural Computing and Applications* 33.1 (Jan. 2021), pp. 121–157. ISSN: 1433-3058. DOI: 10.1007/s00521-020-05565-4. URL: http://dx.doi.org/10.1007/s00521-020-05565-4 (cit. on p. 13).

[118] A.-S. Maerten and D. Soydaner. *From paintbrush to pixel: A review of deep neural networks in AI-generated art*. 2023. DOI: 10.48550/ARXIV.2302.10913. URL: https://arxiv.org/abs/2302.10913 (cit. on p. 13).

[119] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is All you Need.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on p. 13).

[120] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (cit. on p. 13).

[121] H. Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. DOI: 10.48550/ARXIV.2302.13971. URL: https://arxiv.org/abs/2302.13971 (cit. on p. 13).

[122] A. Karpathy. *CS231n: Convolutional Neural Networks for Visual Recognition — Module 2: Convolutional Neural Networks*. 2023. URL: https://cs231n.github.io/ (visited on 10/11/2023) (cit. on pp. 13–17, 20–26, 68).

[123] J. Gu et al. "Recent advances in convolutional neural networks." In: *Pattern Recognition* 77 (May 2018), pp. 354–377. DOI: 10.1016/j.patcog.2017.10.013. URL: https://doi.org/10.1016/j.patcog.2017.10.013 (cit. on pp. 13, 20–22).

[124] I. H. Sarker. "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions." In: *SN Computer Science* 2.6 (Aug. 2021). ISSN: 2661-8907. DOI: 10.1007/s42979-021-00815-1. URL: http://dx.doi.org/10.1007/s42979-021-00815-1 (cit. on p. 13).

[125] M. Krichen. "Convolutional Neural Networks: A Survey." In: *Computers* 12.8 (July 2023), p. 151. ISSN: 2073-431X. DOI: 10.3390/computers12080151. URL: http://dx.doi.org/10.3390/computers12080151 (cit. on p. 13).

[126] A. Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network." In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. DOI: 10.1016/j.physd.2019.132306. URL: https://doi.org/10.1016/j.physd.2019.132306 (cit. on p. 14).

[127] M. Kaur and A. Mohta. "A Review of Deep Learning with Recurrent Neural Network." In: *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, Nov. 2019. DOI: 10.1109/icssit46314.2019.8987837. URL: https://doi.org/10.1109/icssit46314.2019.8987837 (cit. on p. 14).

[128] Y. Yu, X. Si, C. Hu, and J. Zhang. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures." In: *Neural Computation* 31.7 (July 2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199. URL: https://doi.org/10.1162/neco_a_01199 (cit. on p. 14).

[129] I. Kligvasser, T. R. Shaham, and T. Michaeli. "xUnit: Learning a Spatial Activation Function for Efficient Image Restoration." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00258. URL: http://dx.doi.org/10.1109/CVPR.2018.00258 (cit. on p. 14).

[130] A. Zemmari and J. Benois-Pineau. "Deep Neural Networks: Models and Methods." In: *Multi-faceted Deep Learning*. Springer International Publishing, Feb. 2012, pp. 5–38. DOI: 10.1007/978-3-030-74478-6_2. URL: https://doi.org/10.1007/978-3-030-74478-6_2 (cit. on p. 14).

[131] M. Aggarwal and M. N. Murty. "Deep Learning." In: *Machine Learning in Social Networks*. Springer Singapore, Nov. 2020, pp. 35–66. DOI: 10.1007/978-981-33-4022-0_3. URL: https://doi.org/10.1007/978-981-33-4022-0_3 (cit. on p. 14).

[132] V. Liermann, S. Li, and N. Schaudinnus. "Deep Learning: An Introduction." In: *The Impact of Digital Transformation and FinTech on the Finance Professional*. Springer International Publishing, 2019, pp. 305–340. DOI: 10.1007/978-3-030-23719-6_17. URL: https://doi.org/10.1007/978-3-030-23719-6_17 (cit. on pp. 14, 24, 25).

[133] M. L. Forcada. *Neural Networks: Automata and Formal Models of Computation*. 2000. URL: https://www.dlsi.ua.es/~mlf/nnafmc/pbook.pdf (visited on 10/11/2023) (cit. on p. 14).

[134] P. Mantini and S. K. Shah. "CQNN: Convolutional Quadratic Neural Networks." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. DOI: 10.1109/icpr48806.2021.9413207. URL: http://dx.doi.org/10.1109/ICPR48806.2021.9413207 (cit. on p. 14).

[135] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras. "Non-linear Convolution Filters for CNN-Based Learning." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.510. URL: http://dx.doi.org/10.1109/ICCV.2017.510 (cit. on p. 14).

[136] Z. Xu, F. Yu, J. Xiong, and X. Chen. *QuadraLib: A Performant Quadratic Neural Network Library for Architecture Optimization and Design Exploration*. 2022. DOI: 10.48550/ARXIV.2204.01701. URL: https://arxiv.org/abs/2204.01701 (cit. on p. 14).

[137] M. M. Noel and V. Muthiah-Nakarajan. *Computationally Efficient Quadratic Neural Networks*. 2023. DOI: 10.48550/ARXIV.2310.02901. URL: https://arxiv.org/abs/2310.02901 (cit. on p. 14).

[138] C. Chen, G. L. Zhang, X. Yin, C. Zhuo, U. Schlichtmann, and B. Li. *Computational and Storage Efficient Quadratic Neurons for Deep Neural Networks*. 2023. DOI: 10.48550/ARXIV.2306.07294. URL: https://arxiv.org/abs/2306.07294 (cit. on p. 14).

[139] F. Fan, J. Xiong, and G. Wang. "Universal approximation with quadratic deep networks." In: *Neural Networks* 124 (Apr. 2020), pp. 383–392. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.01.007. URL: http://dx.doi.org/10.1016/j.neunet.2020.01.007 (cit. on p. 14).

[140] M. U. Demirezen. "Quadratic Residual Multiplicative Filter Neural Networks for Efficient Approximation of Complex Sensor Signals." In: *IEEE Access* 11 (2023), pp. 75236–75268. ISSN: 2169-3536. DOI: 10.1109/access.2023.3297724. URL: http://dx.doi.org/10.1109/ACCESS.2023.3297724 (cit. on p. 14).

[141] T. Qi and G. Wang. "Superiority of quadratic over conventional neural networks for classification of gaussian mixture data." In: *Visual Computing for Industry, Biomedicine, and Art* 5.1 (Sept. 2022). ISSN: 2524-4442. DOI: 10.1186/s42492-022-00118-z. URL: http://dx.doi.org/10.1186/s42492-022-00118-z (cit. on p. 14).

[142] R. Rodriguez, O. O. Vergara Villegas, V. G. Cruz Sanchez, J. Bila, and A. Mexicano. "Arrhythmia disease classification using a higher-order neural unit." In: *2015 Fourth International Conference on Future Generation Communication Technology (FGCT)*. IEEE, July 2015. DOI: 10.1109/fgct.2015.7300253. URL: http://dx.doi.org/10.1109/FGCT.2015.7300253 (cit. on p. 14).

[143] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf (cit. on pp. 17, 69, 190).

[144] S. J. Nowlan and G. E. Hinton. "Simplifying Neural Networks by Soft Weight-Sharing." In: *Neural Computation* 4.4 (July 1992), pp. 473–493. DOI: 10.1162/neco.1992.4.4.473. URL: https://doi.org/10.1162/neco.1992.4.4.473 (cit. on p. 17).

[145] H.-i. Lim. "A Study on Dropout Techniques to Reduce Overfitting in Deep Neural Networks." In: *Lecture Notes in Electrical Engineering*. Springer Singapore, Dec. 2020, pp. 133–139. DOI: 10.1007/978-981-15-9309-3_20. URL: https://doi.org/10.1007/978-981-15-9309-3_20 (cit. on p. 17).

[146] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio. *An empirical analysis of dropout in piecewise linear networks*. 2013. DOI: 10.48550/ARXIV.1312.6197. URL: https://arxiv.org/abs/1312.6197 (cit. on p. 17).

[147] H. Zhang, S. Li, Y. Ma, M. Li, Y. Xie, and Q. Zhang. "Interpreting and Boosting Dropout from a Game-Theoretic View." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=Jacdvfjicf7 (cit. on p. 17).

[148] M. M. Bejani and M. Ghatee. "A systematic review on overfitting control in shallow and deep neural networks." In: *Artificial Intelligence Review* 54.8 (Mar. 2021), pp. 6391–6438. DOI: 10.1007/s10462-021-09975-1. URL: https://doi.org/10.1007/s10462-021-09975-1 (cit. on p. 17).

[149] Y. Gal and Z. Ghahramani. "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/076a0c97d09cf1a0ec3e19c7f2529f2b-Paper.pdf (cit. on p. 17).

[150] R. Moradi, R. Berangi, and B. Minaei. "A survey of regularization strategies for deep models." In: *Artificial Intelligence Review* 53.6 (Dec. 2019), pp. 3947–3986. DOI: 10.1007/s10462-019-09784-7. URL: https://doi.org/10.1007/s10462-019-09784-7 (cit. on pp. 17, 18).

[151] S. Wager, S. Wang, and P. S. Liang. "Dropout Training as Adaptive Regularization." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/38db3aed920cf82ab059bfccbd02be6a-Paper.pdf (cit. on p. 17).

[152] S. H. Khan, M. Hayat, and F. Porikli. "Regularization of deep neural networks with spectral dropout." In: *Neural Networks* 110 (Feb. 2019), pp. 82–90. DOI: 10.1016/j.neunet.2018.09.009. URL: https://doi.org/10.1016/j.neunet.2018.09.009 (cit. on pp. 17, 18).

[153] P. Baldi and P. J. Sadowski. "Understanding Dropout." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf (cit. on p. 17).

[154] Z. Liu, Z. Xu, J. Jin, Z. Shen, and T. Darrell. "Dropout Reduces Underfitting." In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 22233–22248. URL: https://proceedings.mlr.press/v202/liu23a.html (cit. on pp. 17, 18).

[155] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic. *Dropout as data augmentation*. 2015. DOI: 10.48550/ARXIV.1506.08700. URL: https://arxiv.org/abs/1506.08700 (cit. on p. 17).

[156] T. DeVries and G. W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017. DOI: 10.48550/ARXIV.1708.04552. URL: https://arxiv.org/abs/1708.04552 (cit. on p. 17).

[157] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. URL: https://proceedings.mlr.press/v48/gal16.html (cit. on p. 17).

[158] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. "Regularization of Neural Networks using DropConnect." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1058–1066. URL: https://proceedings.mlr.press/v28/wan13.html (cit. on p. 17).

[159] Z. Lian, X. Jing, X. Wang, H. Huang, Y. Tan, and Y. Cui. "DropConnect Regularization Method with Sparsity Constraint for Neural Networks." In: *Chinese Journal of Electronics* 25.1 (Jan. 2016), pp. 152–158. DOI: 10.1049/cje.2016.01.023. URL: https://doi.org/10.1049/cje.2016.01.023 (cit. on p. 17).

[160] S. Park and N. Kwak. "Analysis on the Dropout Effect in Convolutional Neural Networks." In: *Computer Vision – ACCV 2016*. Springer International Publishing, 2017, pp. 189–204. DOI: 10.1007/978-3-319-54184-6_12. URL: https://doi.org/10.1007/978-3-319-54184-6_12 (cit. on p. 17).

[161] H. Pan, X. Niu, R. Li, S. Shen, and Y. Dou. "DropFilterR: A Novel Regularization Method for Learning Convolutional Neural Networks." In: *Neural Processing Letters* 51.2 (Nov. 2019), pp. 1285–1298. DOI: 10.1007/s11063-019-10147-0. URL: https://doi.org/10.1007/s11063-019-10147-0 (cit. on p. 17).

[162] S. Liang, Y. Khoo, and H. Yang. "Drop-Activation: Implicit Parameter Reduction and Harmonious Regularization." In: *Communications on Applied Mathematics and Computation* 3.2 (Oct. 2020), pp. 293–311. DOI: 10.1007/s42967-020-00085-3. URL: https://doi.org/10.1007/s42967-020-00085-3 (cit. on p. 17).

[163] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. "Deep Networks with Stochastic Depth." In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 646–661. DOI: 10.1007/978-3-319-46493-0_39. URL: https://doi.org/10.1007/978-3-319-46493-0_39 (cit. on p. 17).

[164] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and B. Ottersten. "Training Very Deep Networks via Residual Learning with Stochastic Input Shortcut Connections." In: *Neural Information Processing*. Springer International Publishing, 2017, pp. 23–33. DOI: 10.1007/978-3-319-70096-0_3. URL: https://doi.org/10.1007/978-3-319-70096-0_3 (cit. on p. 18).

[165] J. Ba and B. Frey. "Adaptive dropout for training deep neural networks." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/7b5b23f4aadf9513306bcd59afb6e4c9-Paper.pdf (cit. on p. 18).

[166] S. Wang, T. Zhou, and J. Bilmes. "Jumpout: Improved Dropout for Deep Neural Networks with ReLUs." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6668–6676. URL: https://proceedings.mlr.press/v97/wang19q.html (cit. on p. 18).

[167] X. Frazão and L. A. Alexandre. "DropAll: Generalization of Two Convolutional Neural Network Regularization Methods." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 282–289. DOI: 10.1007/978-3-319-11758-4_31. URL: https://doi.org/10.1007/978-3-319-11758-4_31 (cit. on p. 18).

[168] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino. *Curriculum Dropout*. 2017. DOI: 10.48550/ARXIV.1703.06229. URL: https://arxiv.org/abs/1703.06229 (cit. on p. 18).

[169] R. Moradi, R. Berangi, and B. Minaei. "SparseMaps: Convolutional networks with sparse feature maps for tiny image classification." In: *Expert Systems with Applications* 119 (Apr. 2019), pp. 142–154. DOI: 10.1016/j.eswa.2018.10.012. URL: https://doi.org/10.1016/j.eswa.2018.10.012 (cit. on p. 18).

[170] K. Goutam, S. Balasubramanian, D. Gera, and R. R. Sarma. "LayerOut: Freezing Layers in Deep Neural Networks." In: *SN Computer Science* 1.5 (Sept. 2020). DOI: 10.1007/s42979-020-00312-x. URL: https://doi.org/10.1007/s42979-020-00312-x (cit. on p. 18).

[171] L. N. Smith, E. M. Hand, and T. Doster. "Gradual DropIn of Layers to Train Very Deep Neural Networks." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.515. URL: https://doi.org/10.1109/cvpr.2016.515 (cit. on p. 18).

[172] S. Wang and C. Manning. "Fast dropout training." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 2. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 118–126. URL: https://proceedings.mlr.press/v28/wang13a.html (cit. on p. 18).

[173] S. J. Rennie, V. Goel, and S. Thomas. "Annealed dropout training of deep networks." In: *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, Dec. 2014. DOI: 10.1109/slt.2014.7078567. URL: https://doi.org/10.1109/slt.2014.7078567 (cit. on p. 18).

[174] D. P. Kingma, T. Salimans, and M. Welling. "Variational Dropout and the Local Reparameterization Trick." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/bc7316929fe1545bf0b98d114ee3ecb8-Paper.pdf (cit. on p. 18).

[175] T. Moon, H. Choi, H. Lee, and I. Song. "RNNDROP: A novel dropout for RNNS in ASR." In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, Dec. 2015. DOI: 10.1109/asru.2015.7404775. URL: https://doi.org/10.1109/asru.2015.7404775 (cit. on p. 18).

[176] W. Zaremba, I. Sutskever, and O. Vinyals. *Recurrent Neural Network Regularization*. 2014. DOI: 10.48550/ARXIV.1409.2329. URL: https://arxiv.org/abs/1409.2329 (cit. on p. 18).

[177] S. Semeniuta, A. Severyn, and E. Barth. "Recurrent Dropout without Memory Loss." In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 1757–1766. URL: https://aclanthology.org/C16-1165 (cit. on p. 18).

[178] S. Merity, N. S. Keskar, and R. Socher. "Regularizing and Optimizing LSTM Language Models." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SyyGPP0TZ (cit. on p. 18).

[179] H. Wu and X. Gu. "Towards dropout training for convolutional neural networks." In: *Neural Networks* 71 (Nov. 2015), pp. 1–10. DOI: 10.1016/j.neunet.2015.07.007. URL: https://doi.org/10.1016/j.neunet.2015.07.007 (cit. on p. 18).

[180] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. "Efficient object localization using Convolutional Networks." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. DOI: 10.1109/cvpr.2015.7298664. URL: https://doi.org/10.1109/cvpr.2015.7298664 (cit. on p. 18).

[181] S. Lee and C. Lee. "Revisiting spatial dropout for regularizing convolutional neural networks." In: *Multimedia Tools and Applications* 79.45-46 (June 2020), pp. 34195–34207. DOI: 10.1007/s11042-020-09054-7. URL: https://doi.org/10.1007/s11042-020-09054-7 (cit. on p. 18).

[182] Z. Li, B. Gong, and T. Yang. "Improved Dropout for Shallow and Deep Learning." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/7bb060764a818184ebb1cc0d43d382aa-Paper.pdf (cit. on p. 18).

[183] S. Singh, D. Hoiem, and D. Forsyth. "Swapout: Learning an ensemble of deep architectures." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/c51ce410c124a10e0db5e4b97fc2af39-Paper.pdf (cit. on p. 18).

[184] D. Molchanov, A. Ashukha, and D. Vetrov. "Variational Dropout Sparsifies Deep Neural Networks." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2498–2507. URL: https://proceedings.mlr.press/v70/molchanov17a.html (cit. on p. 18).

[185] K. Neklyudov, D. Molchanov, A. Ashukha, and D. P. Vetrov. "Structured Bayesian Pruning via Log-Normal Multiplicative Noise." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/dab49080d80c724aad5ebf158d63df41-Paper.pdf (cit. on p. 18).

[186] A. N. Gomez, I. Zhang, Y. Gal, and G. E. Hinton. "Targeted Dropout." In: *2018 CDNNRIA Workshop at the 32nd Conference on Neural Information Processing Systems*. 2018. URL: https://openreview.net/forum?id=HkghWScuoQ (cit. on p. 18).

[187] A. N. Gomez, I. Zhang, S. R. Kamalakara, D. Madaan, K. Swersky, Y. Gal, and G. E. Hinton. *Learning Sparse Networks Using Targeted Dropout*. 2019. DOI: 10.48550/ARXIV.1905.13678. URL: https://arxiv.org/abs/1905.13678 (cit. on p. 18).

[188] Y. Gal, J. Hron, and A. Kendall. "Concrete Dropout." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/84ddfb34126fc3a48ee38d7044e87276-Paper.pdf (cit. on p. 18).

[189] Y. Li and Y. Gal. "Dropout Inference in Bayesian Neural Networks with Alpha-divergences." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2052–2061. URL: https://proceedings.mlr.press/v70/li17a.html (cit. on p. 18).

[190] K. Saito, Y. Ushiku, T. Harada, and K. Saenko. "Adversarial Dropout Regularization." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HJIoJWZCZ (cit. on p. 18).

[191] S. Park, J. Park, S.-J. Shin, and I.-C. Moon. "Adversarial Dropout for Supervised and Semi-Supervised Learning." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11634. URL: https://doi.org/10.1609/aaai.v32i1.11634 (cit. on p. 18).

[192] S. Park, K. Song, M. Ji, W. Lee, and I.-C. Moon. "Adversarial Dropout for Recurrent Neural Networks." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 4699–4706. DOI: 10.1609/aaai.v33i01.33014699. URL: https://doi.org/10.1609/aaai.v33i01.33014699 (cit. on p. 18).

[193] K. Zolna, D. Arpit, D. Suhubdy, and Y. Bengio. "Fraternal Dropout." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SJyVzQ-C- (cit. on p. 18).

[194] A. Achille and S. Soatto. "Information Dropout: Learning Optimal Representations Through Noisy Computation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12 (Dec. 2018), pp. 2897–2905. DOI: 10.1109/tpami.2017.2784440. URL: https://doi.org/10.1109/tpami.2017.2784440 (cit. on p. 18).

[195] Y. Yeoh, T. Morie, and H. Tamukoh. "An efficient hardware-oriented dropout algorithm." In: *Neurocomputing* 427 (Feb. 2021), pp. 191–200. DOI: 10.1016/j.neucom.2020.11.055. URL: https://doi.org/10.1016/j.neucom.2020.11.055 (cit. on p. 18).

[196] Y. Tang, Z. Liang, H. Shi, P. Fu, and Q. Sun. "Ranked dropout for handwritten digit recognition." In: *Twelfth International Conference on Graphics and Image Processing (ICGIP 2020)*. Ed. by Z. Pan and X. Hei. SPIE, Jan. 2021. DOI: 10.1117/12.2589394. URL: https://doi.org/10.1117/12.2589394 (cit. on p. 18).

[197] J. Hu, Y. Chen, L. Zhang, and Z. Yi. "Surrogate dropout: Learning optimal drop rate through proxy." In: *Knowledge-Based Systems* 206 (Oct. 2020), p. 106340. DOI: 10.1016/j.knosys.2020.106340. URL: https://doi.org/10.1016/j.knosys.2020.106340 (cit. on p. 18).

[198] Z. Ma, A. Sattar, J. Zhou, Q. Chen, and K. Su. "Dropout with Tabu Strategy for Regularizing Deep Neural Networks." In: *The Computer Journal* 63.7 (Aug. 2019), pp. 1031–1038. DOI: 10.1093/comjnl/bxz062. URL: https://doi.org/10.1093/comjnl/bxz062 (cit. on p. 18).

[199] M. T. Hasan, A. Akter, M. N. Shamael, M. A. E. Hossain, H. M. M. Billah, S. Islam, and S. Shatabda. "Adaptive Tabu Dropout for Regularization of Deep Neural Networks." In: *Neural Information Processing*. Springer International Publishing, 2023, pp. 355–366. DOI: 10.1007/978-3-031-30105-6_30. URL: https://doi.org/10.1007/978-3-031-30105-6_30 (cit. on p. 18).

[200] L. Zeng, H. Zhang, Y. Li, M. Li, and S. Wang. "Supervision dropout: guidance learning in deep neural network." In: *Multimedia Tools and Applications* 82.12 (Dec. 2022), pp. 18831–18850. DOI: 10.1007/s11042-022-14274-0. URL: https://doi.org/10.1007/s11042-022-14274-0 (cit. on p. 18).

[201] C. Schreckenberger, C. Bartelt, and H. Stuckenschmidt. “iDropout: Leveraging Deep Taylor Decomposition for the Robustness of Deep Neural Networks.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2019, pp. 113–126. DOI: 10.1007/978-3-030-33246-4_7. URL: https://doi.org/10.1007/978-3-030-33246-4_7 (cit. on p. 18).

[202] J. Sicking, M. Akila, M. Pintz, T. Wirtz, S. Wrobel, and A. Fischer. “Wasserstein dropout.” In: *Machine Learning* (Sept. 2022). DOI: 10.1007/s10994-022-06230-8. URL: https://doi.org/10.1007/s10994-022-06230-8 (cit. on p. 18).

[203] K. Fedyanin, E. Tsymbalov, and M. Panov. “Dropout Strikes Back: Improved Uncertainty Estimation via Diversity Sampling.” In: *Communications in Computer and Information Science*. Springer International Publishing, 2022, pp. 125–137. DOI: 10.1007/978-3-031-15168-2_11. URL: https://doi.org/10.1007/978-3-031-15168-2_11 (cit. on p. 18).

[204] H. Nguyen, H. Pham, S. Nguyen, N. V. Linh, and K. Than. “Adaptive infinite dropout for noisy and sparse data streams.” In: *Machine Learning* 111.8 (Apr. 2022), pp. 3025–3060. DOI: 10.1007/s10994-022-06169-w. URL: https://doi.org/10.1007/s10994-022-06169-w (cit. on p. 18).

[205] H. Salehinejad and S. Valaee. “Ising-dropout: A Regularization Method for Training and Compression of Deep Neural Networks.” In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019. DOI: 10.1109/icassp.2019.8682914. URL: https://doi.org/10.1109/icassp.2019.8682914 (cit. on p. 18).

[206] H. Salehinejad, Z. Wang, and S. Valaee. “Ising Dropout with Node Grouping for Training and Compression of Deep Neural Networks.” In: *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, Nov. 2019. DOI: 10.1109/globalsip45357.2019.8969121. URL: https://doi.org/10.1109/globalsip45357.2019.8969121 (cit. on p. 18).

[207] Y. Li, W. Ma, C. Chen, M. Zhang, Y. Liu, S. Ma, and Y. Yang. “A Survey on Dropout Methods and Experimental Verification in Recommendation.” In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–20. DOI: 10.1109/tkde.2022.3187013. URL: https://doi.org/10.1109/tkde.2022.3187013 (cit. on p. 18).

[208] Y. Tian and Y. Zhang. “A comprehensive survey on regularization strategies in machine learning.” In: *Information Fusion* 80 (Apr. 2022), pp. 146–166. DOI: 10.1016/j.inffus.2021.11.005. URL: https://doi.org/10.1016/j.inffus.2021.11.005 (cit. on p. 18).

[209] F. Chollet et al. *Keras*. 2015. URL: https://keras.io (cit. on pp. 18, 20, 25, 225).

[210] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/ (cit. on pp. 18, 225).

[211] A. Amidi and S. Amidi. *Convolutional Neural Networks cheatsheet*. 2020. URL: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks (visited on 10/13/2023) (cit. on pp. 20, 21).

[212] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. “A Convolutional Neural Network for Modelling Sentences.” In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/p14-1062. URL: https://doi.org/10.3115/v1/p14-1062 (cit. on p. 20).

[213] V. Dumoulin and F. Visin. *A guide to convolution arithmetic for deep learning*. 2016. DOI: 10.48550/ARXIV.1603.07285. URL: https://arxiv.org/abs/1603.07285 (cit. on pp. 20, 22).

[214] C. F. G. dos Santos, T. P. Moreira, D. Colombo, and J. P. Papa. “Does Removing Pooling Layers from Convolutional Neural Networks Improve Results?” In: *SN Computer Science* 1.5 (Aug. 2020). DOI: 10.1007/s42979-020-00295-9. URL: https://doi.org/10.1007/s42979-020-00295-9 (cit. on p. 20).

[215] R. Sunkara and T. Luo. “No More Strided Convolutions or Pooling: A New CNN Building Block for Low-Resolution Images and Small Objects.” In: *Machine Learning and Knowledge Discovery in Databases*. Springer Nature Switzerland, 2023, pp. 443–459. DOI: 10.1007/978-3-031-26409-2_27. URL: https://doi.org/10.1007/978-3-031-26409-2_27 (cit. on p. 20).

[216] J. Ngiam, Z. Chen, D. Chia, P. Koh, Q. Le, and A. Ng. “Tiled convolutional neural networks.” In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/01f78be6f7cad02658508fe4616098a9-Paper.pdf (cit. on p. 21).

[217] M. Sun, G. Zhang, H. Dang, X. Qi, X. Zhou, and Q. Chang. “Accurate Gastric Cancer Segmentation in Digital Pathology Images Using Deformable Convolution and Multi-Scale Embedding Networks.” In: *IEEE Access* 7 (2019), pp. 75530–75541. DOI: 10.1109/access.2019.2918800. URL: https://doi.org/10.1109/access.2019.2918800 (cit. on p. 21).

[218] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. "A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform." In: *inverse problems and theoretical imaging*. Springer Berlin Heidelberg, 1990, pp. 286–297. DOI: 10.1007/978-3-642-75988-8_28. URL: https://doi.org/10.1007/978-3-642-75988-8_28 (cit. on p. 21).

[219] Y. Lin and J. Wu. "A Novel Multichannel Dilated Convolution Neural Network for Human Activity Recognition." In: *Mathematical Problems in Engineering* 2020 (July 2020), pp. 1–10. DOI: 10.1155/2020/5426532. URL: https://doi.org/10.1155/2020/5426532 (cit. on p. 21).

[220] F. Yu, V. Koltun, and T. Funkhouser. "Dilated Residual Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.75. URL: https://doi.org/10.1109/cvpr.2017.75 (cit. on p. 21).

[221] N. Lin, G. Chen, Q. Zhou, and C. Liu. "Dilated Residual Shrinkage Network for SAR Image Despeckling." In: *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*. IEEE, Oct. 2021. DOI: 10.1109/icsip52628.2021.9689024. URL: https://doi.org/10.1109/icsip52628.2021.9689024 (cit. on p. 21).

[222] C. Orhei and R. Vasiu. "An Analysis of Extended and Dilated Filters in Sharpening Algorithms." In: *IEEE Access* 11 (2023), pp. 81449–81465. DOI: 10.1109/access.2023.3301453. URL: https://doi.org/10.1109/access.2023.3301453 (cit. on p. 21).

[223] T. Wang, M. Sun, and K. Hu. "Dilated Deep Residual Network for Image Denoising." In: *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, Nov. 2017. DOI: 10.1109/ictai.2017.00192. URL: https://doi.org/10.1109/ictai.2017.00192 (cit. on p. 21).

[224] N. T. Trung, D.-H. Trinh, N. L. Trung, T. T. T. Quynh, and M.-H. Luu. "Dilated Residual Convolutional Neural Networks for Low-Dose CT Image Denoising." In: *2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, Dec. 2020. DOI: 10.1109/apccas50809.2020.9301693. URL: https://doi.org/10.1109/apccas50809.2020.9301693 (cit. on p. 21).

[225] M. Gholizadeh-Ansari, J. Alirezaie, and P. Babyn. "Low-dose CT Denoising with Dilated Residual Network." In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, July 2018. DOI: 10.1109/embc.2018.8513453. URL: https://doi.org/10.1109/embc.2018.8513453 (cit. on p. 21).

[226] J. Liu, X. Xiong, J. Li, C. Wu, and R. Song. "Dilated Residual Network Based on Dual Expectation Maximization Attention for Semantic Segmentation of Remote Sensing Images." In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, Sept. 2020. DOI: 10.1109/igarss39084.2020.9324423. URL: https://doi.org/10.1109/igarss39084.2020.9324423 (cit. on p. 21).

[227] M. Zhai, X. Xiang, R. Zhang, N. Lv, and A. E. Saddik. "Learning Optical Flow Using Deep Dilated Residual Networks." In: *IEEE Access* 7 (2019), pp. 22566–22578. DOI: 10.1109/access.2019.2898988. URL: https://doi.org/10.1109/access.2019.2898988 (cit. on p. 22).

[228] K. Tan, J. Chen, and D. Wang. "Gated Residual Networks with Dilated Convolutions for Supervised Speech Separation." In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Apr. 2018. DOI: 10.1109/icassp.2018.8461819. URL: https://doi.org/10.1109/icassp.2018.8461819 (cit. on p. 22).

[229] D. Shafique, M. U. Akram, T. Hassan, T. Anwar, and A. A. Salam. "Dilated Convolution and Residual Network based Convolutional Neural Network for Recognition of Disastrous Events." In: *2022 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*. IEEE, Nov. 2022. DOI: 10.1109/rose56499.2022.9977424. URL: https://doi.org/10.1109/rose56499.2022.9977424 (cit. on p. 22).

[230] K. Pooja, R. R. Nidamanuri, and D. Mishra. "Multi-Scale Dilated Residual Convolutional Neural Network for Hyperspectral Image Classification." In: *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE, Sept. 2019. DOI: 10.1109/whispers.2019.8921284. URL: https://doi.org/10.1109/whispers.2019.8921284 (cit. on p. 22).

[231] X. Chen, J. Wu, L. Lin, D. Liang, H. Hu, Q. Zhang, Y. Iwamoto, X.-H. Han, Y.-W. Chen, and R. Tong. "A Dual-Attention Dilated Residual Network for Liver Lesion Classification and Localization on CT Images." In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2019. DOI: 10.1109/icip.2019.8803009. URL: https://doi.org/10.1109/icip.2019.8803009 (cit. on p. 22).

[232] R. Li, Z. Wu, J. Jia, S. Zhao, and H. Meng. "Dilated Residual Network with Multi-head Self-attention for Speech Emotion Recognition." In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019. DOI: 10.1109/icassp.2019.8682154. URL: https://doi.org/10.1109/icassp.2019.8682154 (cit. on p. 22).

[233] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. "Deconvolutional networks." In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, June 2010. DOI: 10.1109/cvpr.2010.5539957. URL: https://doi.org/10.1109/cvpr.2010.5539957 (cit. on p. 22).

[234] M. D. Zeiler, G. W. Taylor, and R. Fergus. "Adaptive deconvolutional networks for mid and high level feature learning." In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126474. URL: https://doi.org/10.1109/iccv.2011.6126474 (cit. on p. 22).

[235] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville. "ReSeg: A Recurrent Neural Network-Based Model for Semantic Segmentation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2016. URL: http://arxiv.org/pdf/1511.07053 (cit. on p. 22).

[236] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. *Generating images with recurrent adversarial networks*. 2016. DOI: 10.48550/ARXIV.1602.05110. URL: https://arxiv.org/abs/1602.05110 (cit. on p. 22).

[237] X. Chen, Y. Wu, T. Lu, Q. Kong, J. Wang, and Y. Wang. "Remote Sensing Image Super-Resolution With Residual Split Attention Mechanism." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16 (2023), pp. 1–13. DOI: 10.1109/jstars.2023.3287894. URL: https://doi.org/10.1109/jstars.2023.3287894 (cit. on p. 22).

[238] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. DOI: 10.1109/cvpr.2015.7298965. URL: https://doi.org/10.1109/cvpr.2015.7298965 (cit. on p. 22).

[239] M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks." In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53. URL: https://doi.org/10.1007/978-3-319-10590-1_53 (cit. on p. 22).

[240] Y. Choi, H. Jang, and J. Baek. "Chest tomosynthesis deblurring using CNN with deconvolution layer for vertebrae segmentation." In: *Medical Physics* (July 2023). DOI: 10.1002/mp.16576. URL: https://doi.org/10.1002/mp.16576 (cit. on p. 22).

[241] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang. "Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study." In: *Neural Computing and Applications* 34.7 (Feb. 2022), pp. 5321–5347. DOI: 10.1007/s00521-022-06953-8. URL: https://doi.org/10.1007/s00521-022-06953-8 (cit. on pp. 22–24).

[242] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. *Striving for Simplicity: The All Convolutional Net*. 2014. DOI: 10.48550/ARXIV.1412.6806. URL: https://arxiv.org/abs/1412.6806 (cit. on p. 23).

[243] A. Krizhevsky. "Learning multiple layers of features from tiny images." MA thesis. Department of Computer Science, University of Toronto, 2009 (cit. on pp. 23, 54, 58, 61, 64–66, 82, 85, 89, 93, 119, 120, 124, 130, 131, 140, 149, 159, 289).

[244] Y.-L. Boureau, J. Ponce, and Y. LeCun. "A Theoretical Analysis of Feature Pooling in Visual Recognition." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 111–118. ISBN: 9781605589077 (cit. on p. 23).

[245] A. Zafar, M. Aamir, N. M. Nawi, A. Arshad, S. Riaz, A. Alruban, A. K. Dutta, and S. Almotairi. "A Comparison of Pooling Methods for Convolutional Neural Networks." In: *Applied Sciences* 12.17 (Aug. 2022), p. 8643. DOI: 10.3390/app12178643. URL: https://doi.org/10.3390/app12178643 (cit. on pp. 23, 24).

[246] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio. "Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks." In: *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2014, pp. 530–546. DOI: 10.1007/978-3-662-44848-9_34. URL: https://doi.org/10.1007/978-3-662-44848-9_34 (cit. on p. 23).

[247] B. Graham. *Fractional Max-Pooling*. 2014. DOI: 10.48550/ARXIV.1412.6071. URL: https://arxiv.org/abs/1412.6071 (cit. on p. 23).

[248] S.-H. Wang, S. C. Satapathy, D. Anderson, S.-X. Chen, and Y.-D. Zhang. "Deep Fractional Max Pooling Neural Network for COVID-19 Recognition." In: *Frontiers in Public Health* 9 (Aug. 2021). DOI: 10.3389/fpubh.2021.726144. URL: https://doi.org/10.3389/fpubh.2021.726144 (cit. on p. 23).

[249] K. Yue, F. Xu, and J. Yu. "Shallow and wide fractional max-pooling network for image classification." In: *Neural Computing and Applications* 31.2 (July 2017), pp. 409–419. DOI: 10.1007/s00521-017-3073-x. URL: https://doi.org/10.1007/s00521-017-3073-x (cit. on p. 23).

[250] Z. Shi, Y. Ye, and Y. Wu. "Rank-based pooling for deep convolutional neural networks." In: *Neural Networks* 83 (Nov. 2016), pp. 21–31. DOI: 10.1016/j.neunet.2016.07.003. URL: https://doi.org/10.1016/j.neunet.2016.07.003 (cit. on p. 23).

[251] Q. Xu, M. Zhang, Z. Gu, and G. Pan. "Overfitting remedy by sparsifying regularization on fully-connected layers of CNNs." In: *Neurocomputing* 328 (Feb. 2019), pp. 69–74. DOI: 10.1016/j.neucom.2018.03.080. URL: https://doi.org/10.1016/j.neucom.2018.03.080 (cit. on p. 23).

[252] Y. Chen, D. Ming, and X. Lv. "Superpixel based land cover classification of VHR satellite image combining multi-scale CNN and scale parameter estimation." In: *Earth Science Informatics* 12.3 (Apr. 2019), pp. 341–363. DOI: 10.1007/s12145-019-00383-2. URL: https://doi.org/10.1007/s12145-019-00383-2 (cit. on p. 23).

[253] C.-Y. Lee, P. Gallagher, and Z. Tu. "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (Apr. 2018), pp. 863–875. DOI: 10.1109/tpami.2017.2703082. URL: https://doi.org/10.1109/tpami.2017.2703082 (cit. on p. 23).

[254] J. Chen, Z. Hua, J. Wang, and S. Cheng. "A Convolutional Neural Network with Dynamic Correlation Pooling." In: *2017 13th International Conference on Computational Intelligence and Security (CIS)*. IEEE, Dec. 2017. DOI: 10.1109/cis.2017.00115. URL: https://doi.org/10.1109/cis.2017.00115 (cit. on p. 24).

[255] A. Jiménez-Sánchez, A. Kazi, S. Albarqouni, S. Kirchhoff, A. Sträter, P. Biberthaler, D. Mateus, and N. Navab. *Weakly-Supervised Localization and Classification of Proximal Femur Fractures*. 2018. DOI: 10.48550/ARXIV.1809.10692. URL: https://arxiv.org/abs/1809.10692 (cit. on p. 24).

[256] B. Navaneeth and M. Suchetha. "A dynamic pooling based convolutional neural network approach to detect chronic kidney disease." In: *Biomedical Signal Processing and Control* 62 (Sept. 2020), p. 102068. DOI: 10.1016/j.bspc.2020.102068. URL: https://doi.org/10.1016/j.bspc.2020.102068 (cit. on p. 24).

[257] F. Bieder, R. Sandkühler, and P. C. Cattin. *Comparison of Methods Generalizing Max- and Average-Pooling*. 2021. DOI: 10.48550/ARXIV.2103.01746. URL: https://arxiv.org/abs/2103.01746 (cit. on p. 24).

[258] A. Stergiou, R. Poppe, and G. Kalliatakis. "Refining activation downsampling with SoftPool." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01019. URL: https://doi.org/10.1109/iccv48922.2021.01019 (cit. on p. 24).

[259] Z. Wei, J. Zhang, L. Liu, F. Zhu, F. Shen, Y. Zhou, S. Liu, Y. Sun, and L. Shao. "Building Detail-Sensitive Semantic Segmentation Networks With Polynomial Pooling." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.00728. URL: https://doi.org/10.1109/cvpr.2019.00728 (cit. on p. 24).

[260] W. Czaja, W. Li, Y. Li, and M. Pekala. *Maximal function pooling with applications*. 2021. DOI: 10.48550/ARXIV.2103.01292. URL: https://arxiv.org/abs/2103.01292 (cit. on p. 24).

[261] A. Kumar. *Ordinal Pooling Networks: For Preserving Information over Shrinking Feature Maps*. 2018. DOI: 10.48550/ARXIV.1804.02702. URL: https://arxiv.org/abs/1804.02702 (cit. on p. 24).

[262] T. Otsuzuki, H. Hayashi, Y. Zheng, and S. Uchida. "Regularized Pooling." In: *Artificial Neural Networks and Machine Learning – ICANN 2020*. Springer International Publishing, 2020, pp. 241–254. DOI: 10.1007/978-3-030-61616-8_20. URL: https://doi.org/10.1007/978-3-030-61616-8_20 (cit. on p. 24).

[263] J. Yang, K. Yu, Y. Gong, and T. Huang. "Linear spatial pyramid matching using sparse coding for image classification." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. DOI: 10.1109/cvpr.2009.5206757. URL: https://doi.org/10.1109/cvpr.2009.5206757 (cit. on p. 24).

[264] T. Kobayashi. "Global Feature Guided Local Pooling." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00346. URL: https://doi.org/10.1109/iccv.2019.00346 (cit. on p. 24).

[265] M. D. Zeiler and R. Fergus. *Stochastic Pooling for Regularization of Deep Convolutional Neural Networks*. 2013. DOI: 10.48550/ARXIV.1301.3557. URL: https://arxiv.org/abs/1301.3557 (cit. on p. 24).

[266] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. Feris. "S3Pool: Pooling with Stochastic Spatial Sampling." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.426. URL: https://doi.org/10.1109/cvpr.2017.426 (cit. on p. 24).

[267] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. DOI: 10.1007/978-3-319-10578-9_23. URL: https://doi.org/10.1007/978-3-319-10578-9_23 (cit. on p. 24).

[268] K. Qi, Q. Guan, C. Yang, F. Peng, S. Shen, and H. Wu. "Concentric Circle Pooling in Deep Convolutional Networks for Remote Sensing Scene Classification." In: *Remote Sensing* 10.6 (June 2018), p. 934. DOI: 10.3390/rs10060934. URL: https://doi.org/10.3390/rs10060934 (cit. on p. 24).

[269] K. Qi, C. Yang, C. Hu, Q. Guan, W. Tian, S. Shen, and F. Peng. "Polycentric Circle Pooling in Deep Convolutional Networks for High-Resolution Remote Sensing Image Recognition." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 632–641. DOI: 10.1109/jstars.2020.2968564. URL: https://doi.org/10.1109/jstars.2020.2968564 (cit. on p. 24).

[270] F. Wang, S. Huang, L. Shi, and W. Fan. "The application of series multi-pooling convolutional neural networks for medical image segmentation." In: *International Journal of Distributed Sensor Networks* 13.12 (Dec. 2017), p. 155014771774889. DOI: 10.1177/1550147717748899. URL: https://doi.org/10.1177/1550147717748899 (cit. on p. 24).

[271] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. "Semantic Segmentation with Second-Order Pooling." In: *Computer Vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 430–443. DOI: 10.1007/978-3-642-33786-4_32. URL: https://doi.org/10.1007/978-3-642-33786-4_32 (cit. on p. 24).

[272] T.-Y. Lin and S. Maji. *Improved Bilinear Pooling with CNNs*. 2017. DOI: 10.48550/ARXIV.1707.06772. URL: https://arxiv.org/abs/1707.06772 (cit. on p. 24).

[273] E. Li, A. Samat, P. Du, W. Liu, and J. Hu. "Improved Bilinear CNN Model for Remote Sensing Scene Classification." In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022), pp. 1–5. DOI: 10.1109/lgrs.2020.3040153. URL: https://doi.org/10.1109/lgrs.2020.3040153 (cit. on p. 24).

[274] F. Saeedan, N. Weber, M. Goesele, and S. Roth. "Detail-Preserving Pooling in Deep Networks." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00949. URL: https://doi.org/10.1109/cvpr.2018.00949 (cit. on p. 24).

[275] Z. Gao, L. Wang, and G. Wu. "LIP: Local Importance-Based Pooling." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00345. URL: https://doi.org/10.1109/iccv.2019.00345 (cit. on p. 24).

[276] N. Murray and F. Perronnin. "Generalized Max Pooling." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2014. DOI: 10.1109/cvpr.2014.317. URL: https://doi.org/10.1109/cvpr.2014.317 (cit. on p. 24).

[277] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. "TI-POOLING: Transformation-Invariant Pooling for Feature Learning in Convolutional Neural Networks." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.38. URL: https://doi.org/10.1109/cvpr.2016.38 (cit. on p. 24).

[278] X. Wei, Y. Zhang, Y. Gong, and N. Zheng. "Kernelized Subspace Pooling for Deep Local Descriptors." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00200. URL: https://doi.org/10.1109/cvpr.2018.00200 (cit. on p. 24).

[279] J. A. C. Vargas, J. Z. Esquivel, and O. Tickoo. "Introducing Region Pooling Learning." In: *Pattern Recognition. ICPR International Workshops and Challenges*. Springer International Publishing, 2021, pp. 714–724. DOI: 10.1007/978-3-030-68763-2_54. URL: https://doi.org/10.1007/978-3-030-68763-2_54 (cit. on p. 24).

[280] Y. Lee, J. Kim, M. Jung, and J. Kim. "Making a More Reliable Classifier via Random Crop Pooling." In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, July 2016, pp. 309–318. DOI: 10.1007/978-3-319-31293-4_25. URL: https://doi.org/10.1007/978-3-319-31293-4_25 (cit. on p. 24).

[281] N. Akhtar and U. Ragavendran. "Interpretation of intelligence in CNN-pooling processes: a methodological survey." In: *Neural Computing and Applications* 32.3 (July 2019), pp. 879–898. DOI: 10.1007/s00521-019-04296-5. URL: https://doi.org/10.1007/s00521-019-04296-5 (cit. on p. 24).

[282] Z. Tao, C. XiaoYu, L. HuiLing, Y. XinYu, L. YunCan, and Z. XiaoMin. "Pooling Operations in Deep Learning: From "Invariable" to "Variable"." In: *BioMed Research International* 2022 (June 2022). Ed. by C. Li, pp. 1–17. DOI: 10.1155/2022/4067581. URL: https://doi.org/10.1155/2022/4067581 (cit. on p. 24).

[283] Y. Bengio. "Practical Recommendations for Gradient-Based Training of Deep Architectures." In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 437–478. DOI: 10.1007/978-3-642-35289-8_26. URL: https://doi.org/10.1007/978-3-642-35289-8_26 (cit. on p. 24).

[284] O. A. M. López, A. M. López, and J. Crossa. "Fundamentals of Artificial Neural Networks and Deep Learning." In: *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer International Publishing, 2022, pp. 379–425. DOI: 10.1007/978-3-030-89010-0_10. URL: https://doi.org/10.1007/978-3-030-89010-0_10 (cit. on pp. 24, 25).

[285] K. Janocha and W. M. Czarnecki. *On Loss Functions for Deep Neural Networks in Classification*. 2017. DOI: 10.48550/ARXIV.1702.05659. URL: https://arxiv.org/abs/1702.05659 (cit. on p. 25).

[286] R. Abdulkadirov, P. Lyakhov, and N. Nagornov. "Survey of Optimization Algorithms in Modern Neural Networks." In: *Mathematics* 11.11 (May 2023), p. 2466. DOI: 10.3390/math11112466. URL: https://doi.org/10.3390/math11112466 (cit. on pp. 26, 27).

[287] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *J. Mach. Learn. Res.* 12 (July 2011), pp. 2121–2159. ISSN: 1532-4435. URL: http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf (cit. on p. 26).

[288] M. D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. DOI: 10.48550/ARXIV.1212.5701. URL: https://arxiv.org/abs/1212.5701 (cit. on p. 26).

[289] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (cit. on p. 26).

[290] Y. Dauphin, H. de Vries, and Y. Bengio. "Equilibrated adaptive learning rates for non-convex optimization." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/430c3626b879b4005d41b8a46172e0c0-Paper.pdf (cit. on p. 26).

[291] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2014. URL: http://arxiv.org/abs/1312.6114 (cit. on pp. xlv, 26, 181).

[292] T. Dozat. "Incorporating Nesterov Momentum into Adam." In: *Proceedings of the 4th International Conference on Learning Representations*. 2016, pp. 1–4. URL: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ (cit. on pp. 26, 190).

[293] H. Xie, J. Ni, J. Zhang, W. Zhang, and J. Huang. "Evading generated-image detectors: A deep dithering approach." In: *Signal Processing* 197 (Aug. 2022), p. 108558. DOI: 10.1016/j.sigpro.2022.108558. URL: https://doi.org/10.1016/j.sigpro.2022.108558 (cit. on pp. 26, 181).

[294] A. Défossez, L. Bottou, F. Bach, and N. Usunier. "A Simple Convergence Proof of Adam and Adagrad." In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. URL: https://openreview.net/forum?id=ZPQhzTSWA7 (cit. on p. 26).

[295] Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang. *A Novel Convergence Analysis for Algorithms of the Adam Family and Beyond*. 2021. DOI: 10.48550/ARXIV.2104.14840. URL: https://arxiv.org/abs/2104.14840 (cit. on p. 26).

[296] Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang. *A Novel Convergence Analysis for Algorithms of the Adam Family*. 2021. DOI: 10.48550/ARXIV.2112.03459. URL: https://arxiv.org/abs/2112.03459 (cit. on p. 26).

[297] S. J. Reddi, S. Kale, and S. Kumar. "On the Convergence of Adam and Beyond." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ (cit. on p. 26).

[298] H. Iiduka. "Theoretical analysis of Adam using hyperparameters close to one without Lipschitz smoothness." In: *Numerical Algorithms* (July 2023). DOI: 10.1007/s11075-023-01575-0. URL: https://doi.org/10.1007/s11075-023-01575-0 (cit. on p. 26).

[299] Y. Zhang, C. Chen, N. Shi, R. Sun, and Z.-Q. Luo. "Adam Can Converge Without Any Modification On Update Rules." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 28386–28399. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/b6260ae5566442da053e5ab5d691067a-Paper-Conference.pdf (cit. on pp. 26, 27).

[300] S. Osowski, P. Bojarczak, and M. Stodolski. "Fast Second Order Learning Algorithm for Feedforward Multilayer Neural Networks and its Applications." In: *Neural Networks* 9.9 (Dec. 1996), pp. 1583–1596. DOI: 10.1016/s0893-6080(96)00029-9. URL: https://doi.org/10.1016/s0893-6080(96)00029-9 (cit. on p. 26).

[301] K. Tyagi, C. Rane, B. Irie, and M. Manry. “Multistage Newton’s Approach for Training Radial Basis Function Neural Networks.” In: *SN Computer Science* 2.5 (July 2021). DOI: 10.1007/s42979-021-00757-8. URL: https://doi.org/10.1007/s42979-021-00757-8 (cit. on p. 26).

[302] A. Likas and A. Stafylopatis. “Training the random neural network using quasi-Newton methods.” In: *European Journal of Operational Research* 126.2 (Oct. 2000), pp. 331–339. DOI: 10.1016/s0377-2217(99)00482-8. URL: https://doi.org/10.1016/s0377-2217(99)00482-8 (cit. on p. 26).

[303] M. Chen. “Stochastic Gradient Descent Combines Second-Order Information for Training Neural Network.” In: *Proceedings of the 2018 1st International Conference on Mathematics and Statistics*. ACM, July 2018. DOI: 10.1145/3274250.3274262. URL: https://doi.org/10.1145/3274250.3274262 (cit. on p. 26).

[304] S. Sun, Z. Cao, H. Zhu, and J. Zhao. *A Survey of Optimization Methods from a Machine Learning Perspective*. 2019. DOI: 10.48550/ARXIV.1906.06821. URL: https://arxiv.org/abs/1906.06821 (cit. on pp. 26, 27).

[305] H. H. Tan and K. H. Lim. “Review of second-order optimization techniques in artificial neural networks backpropagation.” In: *IOP Conference Series: Materials Science and Engineering* 495 (June 2019), p. 012003. DOI: 10.1088/1757-899x/495/1/012003. URL: https://doi.org/10.1088/1757-899x/495/1/012003 (cit. on pp. 26, 27).

[306] M. Reyad, A. M. Sarhan, and M. Arafa. “A modified Adam algorithm for deep neural network optimization.” In: *Neural Computing and Applications* 35.23 (Apr. 2023), pp. 17095–17112. DOI: 10.1007/s00521-023-08568-z. URL: https://doi.org/10.1007/s00521-023-08568-z (cit. on p. 26).

[307] G. Vrbančič and V. Podgorelec. “Efficient ensemble for image-based identification of Pneumonia utilizing deep CNN and SGD with warm restarts.” In: *Expert Systems with Applications* 187 (Jan. 2022), p. 115834. DOI: 10.1016/j.eswa.2021.115834. URL: https://doi.org/10.1016/j.eswa.2021.115834 (cit. on p. 26).

[308] Z. Yu, G. Sun, and J. Lv. “A fractional-order momentum optimization approach of deep neural networks.” In: *Neural Computing and Applications* 34.9 (Jan. 2022), pp. 7091–7111. DOI: 10.1007/s00521-021-06765-2. URL: https://doi.org/10.1007/s00521-021-06765-2 (cit. on p. 26).

[309] B. Heo, S. Chun, S. J. Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J.-W. Ha. “AdamP: Slowing Down the Slowdown for Momentum Optimizers on Scale-invariant Weights.” In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=Iz3zU3M316D (cit. on p. 26).

[310] J. Ma and D. Yarats. “Quasi-hyperbolic momentum and Adam for deep learning.” In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=S1fUpoR5FQ (cit. on p. 26).

[311] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. *Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks*. 2018. DOI: 10.48550/ARXIV.1806.06763. URL: https://arxiv.org/abs/1806.06763 (cit. on p. 26).

[312] J. Chen and Q. Gu. *Padam: Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks*. 2019. URL: https://openreview.net/forum?id=BJll6o09tm (cit. on p. 26).

[313] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. *Training Deep Neural Networks with Partially Adaptive Momentum*. 2020. URL: https://openreview.net/forum?id=HklWsREKwr (cit. on p. 26).

[314] M. Liu, W. Zhang, F. Orabona, and T. Yang. *Adam^{+}: A Stochastic Method with Adaptive Variance Reduction*. 2020. DOI: 10.48550/ARXIV.2011.11985. URL: https://arxiv.org/abs/2011.11985 (cit. on p. 26).

[315] F. Huang, J. Li, and S. Gao. *BiAdam: Fast Adaptive Bilevel Optimization Methods*. 2021. DOI: 10.48550/ARXIV.2106.11396. URL: https://arxiv.org/abs/2106.11396 (cit. on p. 26).

[316] F. Huang, X. Wu, and Z. Hu. *AdaGDA: Faster Adaptive Gradient Descent Ascent Methods for Minimax Optimization*. 2021. DOI: 10.48550/ARXIV.2106.16101. URL: https://arxiv.org/abs/2106.16101 (cit. on p. 26).

[317] L. Luo, Y. Xiong, and Y. Liu. “Adaptive Gradient Methods with Dynamic Bound of Learning Rate.” In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg3g2R9FX (cit. on p. 26).

[318] J. Liu, J. Kong, D. Xu, M. Qi, and Y. Lu. “Convergence analysis of AdaBound with relaxed bound functions for non-convex optimization.” In: *Neural Networks* 145 (Jan. 2022), pp. 300–307. DOI: 10.1016/j.neunet.2021.10.026. URL: https://doi.org/10.1016/j.neunet.2021.10.026 (cit. on p. 26).

[319] Y. Liu and D. Li. "AdaXod: a new adaptive and momental bound algorithm for training deep neural networks." In: *The Journal of Supercomputing* 79.15 (May 2023), pp. 17691–17715. DOI: 10.1007/s11227-023-05338-5. URL: https://doi.org/10.1007/s11227-023-05338-5 (cit. on p. 26).

[320] J. Ding, X. Ren, R. Luo, and X. Sun. *An Adaptive and Momental Bound Method for Stochastic Learning*. 2019. DOI: 10.48550/ARXIV.1910.12249. URL: https://arxiv.org/abs/1910.12249 (cit. on p. 26).

[321] K. Verma and A. Maiti. "WSAGrad: a novel adaptive gradient based method." In: *Applied Intelligence* 53.11 (Oct. 2022), pp. 14383–14399. DOI: 10.1007/s10489-022-04205-9. URL: https://doi.org/10.1007/s10489-022-04205-9 (cit. on p. 26).

[322] G. Ioannou, T. Tagaris, and A. Stafylopatis. "AdaLip: An Adaptive Learning Rate Method per Layer for Stochastic Optimization." In: *Neural Processing Letters* 55.5 (Jan. 2023), pp. 6311–6338. DOI: 10.1007/s11063-022-11140-w. URL: https://doi.org/10.1007/s11063-022-11140-w (cit. on p. 26).

[323] J. Zhuang, T. Tang, Y. Ding, S. C. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan. "AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18795–18806. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf (cit. on p. 26).

[324] X. Liao, S. Sahran, A. Abdullah, and S. A. Shukor. "AdaCB: An Adaptive Gradient Method with Convergence Range Bound of Learning Rate." In: *Applied Sciences* 12.18 (Sept. 2022), p. 9389. DOI: 10.3390/app12189389. URL: https://doi.org/10.3390/app12189389 (cit. on p. 26).

[325] F. Huang, J. Li, and H. Huang. "SUPER-ADAM: Faster and Universal Framework of Adaptive Gradients." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 9074–9085. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/4be5a36cbaca8ab9d2066debfe4e65c1-Paper.pdf (cit. on p. 27).

[326] J. D. Camacho, C. Villaseñor, A. Y. Alanis, C. Lopez-Franco, and N. Arana-Daniel. "KAdam: Using the Kalman Filter to Improve Adam algorithm." In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing, 2019, pp. 429–438. DOI: 10.1007/978-3-030-33904-3_40. URL: https://doi.org/10.1007/978-3-030-33904-3_40 (cit. on p. 27).

[327] J. Camacho, C. Villaseñor, A. Y. Alanis, C. Lopez-Franco, and N. Arana-Daniel. "sKAdam: An improved scalar extension of KAdam for function optimization." In: *Intelligent Data Analysis* 24 (Dec. 2020), pp. 87–104. DOI: 10.3233/ida-200010. URL: https://doi.org/10.3233/ida-200010 (cit. on p. 27).

[328] U. M. Khaire and R. Dhanalakshmi. "High-dimensional microarray dataset classification using an improved adam optimizer (iAdam)." In: *Journal of Ambient Intelligence and Humanized Computing* 11.11 (Mar. 2020), pp. 5187–5204. DOI: 10.1007/s12652-020-01832-3. URL: https://doi.org/10.1007/s12652-020-01832-3 (cit. on pp. 27, 48).

[329] S. Bhakta, U. Nandi, T. Si, S. K. Ghosal, C. Changdar, and R. K. Pal. "DiffMoment: an adaptive optimization technique for convolutional neural network." In: *Applied Intelligence* 53.13 (Dec. 2022), pp. 16844–16858. DOI: 10.1007/s10489-022-04382-7. URL: https://doi.org/10.1007/s10489-022-04382-7 (cit. on p. 27).

[330] Y. Gui, D. Li, and R. Fang. "A fast adaptive algorithm for training deep neural networks." In: *Applied Intelligence* 53.4 (June 2022), pp. 4099–4108. DOI: 10.1007/s10489-022-03629-7. URL: https://doi.org/10.1007/s10489-022-03629-7 (cit. on p. 27).

[331] E. F. Jose, F. Enembreck, and J. P. Barddal. "ADADRIFT: An Adaptive Learning Technique for Long-history Stream-based Recommender Systems." In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2020. DOI: 10.1109/smc42975.2020.9282922. URL: https://doi.org/10.1109/smc42975.2020.9282922 (cit. on p. 27).

[332] C. Gulcehre, M. Moczulski, and Y. Bengio. *ADASECANT: Robust Adaptive Secant Method for Stochastic Gradient*. 2014. DOI: 10.48550/ARXIV.1412.7419. URL: https://arxiv.org/abs/1412.7419 (cit. on p. 27).

[333] C. Gulcehre, J. Sotelo, M. Moczulski, and Y. Bengio. "A robust adaptive stochastic gradient method for deep learning." In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017. DOI: 10.1109/ijcnn.2017.7965845. URL: https://doi.org/10.1109/ijcnn.2017.7965845 (cit. on p. 27).

[334] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney. “AdaHessian: An Adaptive Second Order Optimizer for Machine Learning.” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.12 (May 2021), pp. 10665–10673. ISSN: 2159-5399. DOI: 10.1609/aaai.v35i12.17275. URL: http://dx.doi.org/10.1609/aaai.v35i12.17275 (cit. on p. 27).

[335] R. Elshamy, O. Abu-Elnasr, M. Elhoseny, and S. Elmougy. “Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning.” In: *Scientific Reports* 13.1 (May 2023). DOI: 10.1038/s41598-023-35663-x. URL: https://doi.org/10.1038/s41598-023-35663-x (cit. on p. 27).

[336] D.-S. Shim and J. Shim. “A Modified Stochastic Gradient Descent Optimization Algorithm With Random Learning Rate for Machine Learning and Deep Learning.” In: *International Journal of Control, Automation and Systems* (Aug. 2023). DOI: 10.1007/s12555-022-0947-1. URL: https://doi.org/10.1007/s12555-022-0947-1 (cit. on p. 27).

[337] R. M. Schmidt, F. Schneider, and P. Hennig. “Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers.” In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 9367–9376. URL: https://proceedings.mlr.press/v139/schmidt21a.html (cit. on p. 27).

[338] X. Li, G. Wu, L. Yang, W. Wang, R. Song, and J. Yang. *A Survey of Historical Learning: Learning Models with Learning History*. 2023. DOI: 10.48550/ARXIV.2303.12992. URL: https://arxiv.org/abs/2303.12992 (cit. on p. 27).

[339] G. Habib and S. Qureshi. “Optimization and acceleration of convolutional neural networks: A survey.” In: *Journal of King Saud University - Computer and Information Sciences* 34.7 (July 2022), pp. 4244–4268. DOI: 10.1016/j.jksuci.2020.10.004. URL: https://doi.org/10.1016/j.jksuci.2020.10.004 (cit. on p. 27).

[340] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl. *On Empirical Comparisons of Optimizers for Deep Learning*. 2019. DOI: 10.48550/ARXIV.1910.05446. URL: https://arxiv.org/abs/1910.05446 (cit. on p. 27).

[341] E. Hassan, M. Y. Shams, N. A. Hikal, and S. Elmougy. “The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study.” In: *Multimedia Tools and Applications* 82.11 (Sept. 2022), pp. 16591–16633. DOI: 10.1007/s11042-022-13820-0. URL: https://doi.org/10.1007/s11042-022-13820-0 (cit. on p. 27).

[342] A. Kalra, S. Kumar, and S. S. Walia. “ANN training: A survey of classical & soft computing approaches.” In: *International Journal of Control Theory and Applications* 9.34 (2016). Cited by: 2, p. 83 104. URL: https://serialsjournals.com/abstract/30973_67.pdf (cit. on p. 27).

[343] M. Magris and A. Iosifidis. “Bayesian learning for neural networks: an algorithmic survey.” In: *Artificial Intelligence Review* 56.10 (Mar. 2023), pp. 11773–11823. DOI: 10.1007/s10462-023-10443-1. URL: https://doi.org/10.1007/s10462-023-10443-1 (cit. on p. 27).

[344] S. Ruder. *An overview of gradient descent optimization algorithms*. 2016. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747 (cit. on p. 27).

[345] E. M. Dogo, O. J. Afolabi, and B. Twala. “On the Relative Impact of Optimizers on Convolutional Neural Networks with Varying Depth and Width for Image Classification.” In: *Applied Sciences* 12.23 (Nov. 2022), p. 11976. DOI: 10.3390/app122311976. URL: https://doi.org/10.3390/app122311976 (cit. on p. 27).

[346] A. L. Tarca, R. Romero, and S. Draghici. “Analysis of microarray experiments of gene expression profiling.” In: *American Journal of Obstetrics and Gynecology* 195.2 (Aug. 2006), pp. 373–388. DOI: 10.1016/j.ajog.2006.07.001. URL: https://doi.org/10.1016/j.ajog.2006.07.001 (cit. on pp. 29, 36).

[347] R. Bumgarner. “Overview of DNA Microarrays: Types, Applications, and Their Future.” In: *Current Protocols in Molecular Biology* 101.1 (Jan. 2013). DOI: 10.1002/0471142727.mb2201s101. URL: https://doi.org/10.1002/0471142727.mb2201s101 (cit. on pp. 29, 31–33).

[348] C. Walsh, P. Hu, J. Batt, and C. Santos. “Microarray Meta-Analysis and Cross-Platform Normalization: Integrative Genomics for Robust Biomarker Discovery.” In: *Microarrays* 4.3 (Aug. 2015), pp. 389–406. DOI: 10.3390/microarrays4030389. URL: https://doi.org/10.3390/microarrays4030389 (cit. on pp. 29, 36).

[349] A. A. Ewis, Z. Zhelev, R. Bakalova, S. Fukuoka, Y. Shinohara, M. Ishikawa, and Y. Baba. “A history of microarrays in biomedicine.” In: *Expert Review of Molecular Diagnostics* 5.3 (May 2005), pp. 315–328. DOI: 10.1586/14737159.5.3.315. URL: https://doi.org/10.1586/14737159.5.3.315 (cit. on p. 29).

[350] A. J. V. Asselt and E. A. Ehli. “Whole-Genome Genotyping Using DNA Microarrays for Population Genetics.” In: *Methods in Molecular Biology*. Springer US, 2022, pp. 269–287. DOI: 10.1007/978-1-0716-1920-9_16. URL: https://doi.org/10.1007/978-1-0716-1920-9_16 (cit. on p. 29).

[351] H. M. U. Aslam, H. Riaz, N. Killiny, X.-G. Zhou, L. S. Thomashow, N. T. Peters, and A. K. Chanda. "Microarray Technology for Detection of Plant Diseases." In: *Trends in Plant Disease Assessment*. Springer Nature Singapore, 2022, pp. 203–223. DOI: 10.1007/978-981-19-5896-0_11. URL: https://doi.org/10.1007/978-981-19-5896-0_11 (cit. on p. 29).

[352] L. Anastasiia and B. Ilia. "Prospects of DNA microarray application in management of chronic obstructive pulmonary disease: A systematic review." In: *Frigid Zone Medicine* 3.1 (Jan. 2023), pp. 5–12. DOI: 10.2478/fzm-2023-0002. URL: https://doi.org/10.2478/fzm-2023-0002 (cit. on p. 29).

[353] E. Schaudy, J. Lietard, and M. M. Somoza. "Enzymatic Synthesis of High-Density RNA Microarrays." In: *Current Protocols* 3.2 (Feb. 2023). DOI: 10.1002/cpz1.667. URL: https://doi.org/10.1002/cpz1.667 (cit. on p. 29).

[354] A. Y. Higashi, B. J. Aronow, and G. R. Dressler. "Expression Profiling of Fibroblasts in Chronic and Acute Disease Models Reveals Novel Pathways in Kidney Fibrosis." In: *Journal of the American Society of Nephrology* 30.1 (Dec. 2018), pp. 80–94. DOI: 10.1681/asn.2018060644. URL: https://doi.org/10.1681/asn.2018060644 (cit. on p. 29).

[355] M. Hao, B. Barlogie, G. Tricot, L. Liu, L. Qiu, J. D. Shaughnessy, and F. Zhan. "Gene Expression Profiling Reveals Aberrant T-cell Marker Expression on Tumor Cells of Waldenström's Macroglobulinemia." In: *Clinical Cancer Research* 25.1 (Jan. 2019), pp. 201–209. DOI: 10.1158/1078-0432.ccr-18-1435. URL: https://doi.org/10.1158/1078-0432.ccr-18-1435 (cit. on p. 29).

[356] D. Edsgärd et al. "Identification of spatial expression trends in single-cell gene expression data." In: *Nature Methods* 15.5 (Mar. 2018), pp. 339–342. DOI: 10.1038/nmeth.4634. URL: https://doi.org/10.1038/nmeth.4634 (cit. on p. 29).

[357] J. B. Nielsen et al. "Biobank-driven genomic discovery yields new insight into atrial fibrillation biology." In: *Nature Genetics* 50.9 (July 2018), pp. 1234–1239. DOI: 10.1038/s41588-018-0171-3. URL: https://doi.org/10.1038/s41588-018-0171-3 (cit. on p. 29).

[358] N. H. Lents. *Current and Future Uses of DNA Microarrays in Forensic Science*. Nov. 2011. DOI: 10.1002/9781118062241.ch10. URL: https://doi.org/10.1002/9781118062241.ch10 (cit. on p. 29).

[359] H. Okamura, H. Yamano, T. Tsuda, J. Morihiro, K. Hirayama, and H. Nagano. "Development of a clinical microarray system for genetic analysis screening." In: *Practical Laboratory Medicine* 33 (Jan. 2023), e00306. DOI: 10.1016/j.plabm.2022.e00306. URL: https://doi.org/10.1016/j.plabm.2022.e00306 (cit. on pp. 29, 33).

[360] N. L. of Medicine. *What is a gene?* Mar. 2021. URL: https://medlineplus.gov/genetics/understanding/basics/gene/ (visited on 10/15/2023) (cit. on p. 29).

[361] D. P. Snustad and M. J. Simmons. *Principles of genetics*. en. 7th ed. Nashville, TN: John Wiley & Sons, Nov. 2015 (cit. on pp. 29, 30).

[362] N. L. of Medicine. *What is DNA?* Jan. 2021. URL: https://medlineplus.gov/genetics/understanding/basics/dna/ (visited on 10/15/2023) (cit. on p. 29).

[363] W. N. Hunter, T. Brown, N. N. Anand, and O. Kennard. "Structure of an adenine·cytosine base pair in DNA and its implications for mismatch repair." In: *Nature* 320.6062 (Apr. 1986), pp. 552–555. DOI: 10.1038/320552a0. URL: https://doi.org/10.1038/320552a0 (cit. on p. 30).

[364] T. K. Karakach, R. M. Flight, S. E. Douglas, and P. D. Wentzell. "An introduction to DNA microarrays for gene expression analysis." In: *Chemometrics and Intelligent Laboratory Systems* 104.1 (Nov. 2010), pp. 28–52. DOI: 10.1016/j.chemolab.2010.04.003. URL: https://doi.org/10.1016/j.chemolab.2010.04.003 (cit. on pp. 30, 33, 36).

[365] F. Crick. "On Protein Synthesis." In: *The Symposia of the Society for Experimental Biology* 12 (1958). URL: https://profiles.nlm.nih.gov/101584582X404 (cit. on p. 30).

[366] F. Crick. "Central Dogma of Molecular Biology." In: *Nature* 227.5258 (Aug. 1970), pp. 561–563. DOI: 10.1038/227561a0. URL: https://doi.org/10.1038/227561a0 (cit. on p. 30).

[367] R. Olby. "Francis Crick, DNA, and the Central Dogma." In: *Daedalus* 99.4 (1970), pp. 938–987. ISSN: 00115266. URL: http://www.jstor.org/stable/20023978 (visited on 10/15/2023) (cit. on p. 30).

[368] P. Šustar. "Crick's notion of genetic information and the 'central dogma' of molecular biology." In: *The British Journal for the Philosophy of Science* 58.1 (Mar. 2007), pp. 13–24. DOI: 10.1093/bjps/axl018. URL: https://doi.org/10.1093/bjps/axl018 (cit. on p. 30).

[369] M. Cobb. "60 years ago, Francis Crick changed the logic of biology." In: *PLOS Biology* 15.9 (Sept. 2017), e2003243. DOI: 10.1371/journal.pbio.2003243. URL: https://doi.org/10.1371/journal.pbio.2003243 (cit. on p. 30).

[370] J. D. Watson and F. H. C. Crick. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid." In: *Nature* 171.4356 (Apr. 1953), pp. 737–738. DOI: 10.1038/171737a0. URL: https://doi.org/10.1038/171737a0 (cit. on p. 30).

[371] E. M. Southern. "DNA Microarrays: History and Overview." In: *DNA Arrays*. Humana Press, 2001, pp. 1–15. DOI: 10.1385/1-59259-234-1:1. URL: https://doi.org/10.1385/1-59259-234-1:1 (cit. on pp. 30–33).

[372] T. Lenoir and E. Giannella. "The emergence and diffusion of DNA microarray technology." In: *Journal of Biomedical Discovery and Collaboration* 1.1 (2006), p. 11. DOI: 10.1186/1747-5333-1-11. URL: https://doi.org/10.1186/1747-5333-1-11 (cit. on p. 32).

[373] Z. Wang, M. Gerstein, and M. Snyder. "RNA-Seq: a revolutionary tool for transcriptomics." In: *Nature Reviews Genetics* 10.1 (Jan. 2009), pp. 57–63. DOI: 10.1038/nrg2484. URL: https://doi.org/10.1038/nrg2484 (cit. on pp. 33, 36).

[374] J.-Y. Coppée. "Do DNA microarrays have their future behind them?" In: *Microbes and Infection* 10.9 (July 2008), pp. 1067–1071. DOI: 10.1016/j.micinf.2008.07.003. URL: https://doi.org/10.1016/j.micinf.2008.07.003 (cit. on p. 33).

[375] K. Dichtl, A. Osterman, R. Barry, and J. Wagener. "A novel microarray-based PCR assay for the detection of HSV-1, HSV-2, and VZV skin infections: A retrospective analysis." In: *Journal of Virological Methods* 312 (Feb. 2023), p. 114650. DOI: 10.1016/j.jviromet.2022.114650. URL: https://doi.org/10.1016/j.jviromet.2022.114650 (cit. on p. 33).

[376] G. M. Aparna and K. K. R. Tetala. "Recent Progress in Development and Application of DNA, Protein, Peptide, Glycan, Antibody, and Aptamer Microarrays." In: *Biomolecules* 13.4 (Mar. 2023), p. 602. DOI: 10.3390/biom13040602. URL: https://doi.org/10.3390/biom13040602 (cit. on pp. xlix, 33, 34, 36).

[377] A. G. Bracamonte. "Microarrays towards nanoarrays and the future Next Generation of Sequencing methodologies (NGS)." In: *Sensing and Bio-Sensing Research* 37 (Aug. 2022), p. 100503. DOI: 10.1016/j.sbsr.2022.100503. URL: https://doi.org/10.1016/j.sbsr.2022.100503 (cit. on p. 33).

[378] L. Fahmideh, E. Khodadadi, E. Khodadadi, E. Zeinalzadeh, S. Dao, Ş. Köse, and H. S. K. 7. "Transcriptome Analysis Methods: From the Serial Analysis of Gene Expression and Microarray to Sequencing new Generation Methods." In: *Biointerface Research in Applied Chemistry* (2023). URL: https://biointerfaceresearch.com/wp-content/uploads/2023/02/BRIAC136.543.pdf (cit. on p. 33).

[379] V. Bolón-Canedo and A. Alonso-Betanzos, eds. *Microarray Bioinformatics*. Springer New York, 2019. DOI: 10.1007/978-1-4939-9442-7. URL: https://doi.org/10.1007/978-1-4939-9442-7 (cit. on pp. xlvi, 33, 34, 36).

[380] A. J. Trachtenberg et al. "A Primer on the Current State of Microarray Technologies." In: *Next Generation Microarray Bioinformatics*. Humana Press, Nov. 2011, pp. 3–17. DOI: 10.1007/978-1-61779-400-1_1. URL: https://doi.org/10.1007/978-1-61779-400-1_1 (cit. on pp. 34, 36).

[381] R. Jaksik, M. Iwanaszko, J. Rzeszowska-Wolny, and M. Kimmel. "Microarray experiments and factors which affect their reliability." In: *Biology Direct* 10.1 (Sept. 2015). DOI: 10.1186/s13062-015-0077-2. URL: https://doi.org/10.1186/s13062-015-0077-2 (cit. on pp. xlvi, 34–36).

[382] S. Hamlet, E. Petcu, and S. Ivanovski. "Genomic Microarray Analysis." In: *Handbook of Vascular Biology Techniques*. Springer Netherlands, 2015, pp. 391–405. DOI: 10.1007/978-94-017-9716-0_30. URL: https://doi.org/10.1007/978-94-017-9716-0_30 (cit. on pp. 34, 36).

[383] P. P. Dubey and D. Kumar. "Microarray Technology: Basic Concept, Protocols, and Applications." In: *Springer Protocols Handbooks*. Springer Berlin Heidelberg, Dec. 2012, pp. 261–279. DOI: 10.1007/978-3-642-34410-7_17. URL: https://doi.org/10.1007/978-3-642-34410-7_17 (cit. on pp. 34, 36).

[384] S. C. Sealfon and T. T. Chu. "RNA and DNA Microarrays." In: *Methods in Molecular Biology*. Humana Press, Sept. 2010, pp. 3–34. DOI: 10.1007/978-1-59745-551-0_1. URL: https://doi.org/10.1007/978-1-59745-551-0_1 (cit. on pp. xlvi, xlix, li, 34).

[385] A. C. Pease, D. Solas, E. J. Sullivan, M. T. Cronin, C. P. Holmes, and S. P. Fodor. "Light-generated oligonucleotide arrays for rapid DNA sequence analysis." In: *Proceedings of the National Academy of Sciences* 91.11 (May 1994), pp. 5022–5026. DOI: 10.1073/pnas.91.11.5022. URL: https://doi.org/10.1073/pnas.91.11.5022 (cit. on p. 34).

[386] L. Gautier, L. Cope, B. M. Bolstad, and R. A. Irizarry. "affy—analysis of Affymetrix GeneChip data at the probe level." In: *Bioinformatics* 20.3 (Feb. 2004), pp. 307–315. DOI: 10.1093/bioinformatics/btg405. URL: https://doi.org/10.1093/bioinformatics/btg405 (cit. on p. 35).

[387] K. J. Archer and T. Guennel. "An application for assessing quality of RNA hybridized to Affymetrix GeneChips." In: *Bioinformatics* 22.21 (Aug. 2006), pp. 2699–2701. DOI: 10.1093/bioinformatics/btl459. URL: https://doi.org/10.1093/bioinformatics/btl459 (cit. on p. 35).

[388] C. L. Wilson, S. D. Pepper, Y. Hey, and C. J. Miller. "Amplification protocols introduce systematic but reproducible errors into gene expression studies." In: *BioTechniques* 36.3 (Mar. 2004), pp. 498–506. DOI: 10.2144/04363rn05. URL: https://doi.org/10.2144/04363rn05 (cit. on p. 35).

[389] H. Sudo, A. Mizoguchi, J. Kawauchi, H. Akiyama, and S. Takizawa. "Use of Non-Amplified RNA Samples for Microarray Analysis of Gene Expression." In: *PLoS ONE* 7.2 (Feb. 2012). Ed. by S. Huang, e31397. DOI: 10.1371/journal.pone.0031397. URL: https://doi.org/10.1371/journal.pone.0031397 (cit. on p. 35).

[390] H. Schindler, A. Wiese, J. Auer, and H. Burtscher. "cRNA target preparation for microarrays: Comparison of gene expression profiles generated with different amplification procedures." In: *Analytical Biochemistry* 344.1 (Sept. 2005), pp. 92–101. DOI: 10.1016/j.ab.2005.06.006. URL: https://doi.org/10.1016/j.ab.2005.06.006 (cit. on p. 35).

[391] W. Tong et al. "Evaluation of external RNA controls for the assessment of microarray performance." In: *Nature Biotechnology* 24.9 (Aug. 2006), pp. 1132–1139. DOI: 10.1038/nbt1237. URL: https://doi.org/10.1038/nbt1237 (cit. on pp. xlvii, 35).

[392] M. Dufva. "Introduction to Microarray Technology." In: *Methods in Molecular Biology*. Humana Press, 2009, pp. 1–22. DOI: 10.1007/978-1-59745-538-1_1. URL: https://doi.org/10.1007/978-1-59745-538-1_1 (cit. on p. 35).

[393] B. Bolstad, R. Irizarry, M. Åstrand, and T. Speed. "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias." In: *Bioinformatics* 19.2 (Jan. 2003), pp. 185–193. DOI: 10.1093/bioinformatics/19.2.185. URL: https://doi.org/10.1093/bioinformatics/19.2.185 (cit. on pp. 35, 36).

[394] G. K. Smyth and T. Speed. "Normalization of cDNA microarray data." In: *Methods* 31.4 (Dec. 2003), pp. 265–273. DOI: 10.1016/s1046-2023(03)00155-5. URL: https://doi.org/10.1016/s1046-2023(03)00155-5 (cit. on p. 36).

[395] F. E. Ahmed. "Microarray RNA transcriptional profiling: Part II. Analytical considerations and annotation." In: *Expert Review of Molecular Diagnostics* 6.5 (Sept. 2006), pp. 703–715. DOI: 10.1586/14737159.6.5.703. URL: https://doi.org/10.1586/14737159.6.5.703 (cit. on p. 36).

[396] G. Pandey, L. N. Ramakrishnan, M. Steinbach, and V. Kumar. "Systematic Evaluation of Scaling Methods for Gene Expression Data." In: *2008 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE, 2008. DOI: 10.1109/bibm.2008.33. URL: https://doi.org/10.1109/bibm.2008.33 (cit. on p. 36).

[397] X. Qiu, R. Hu, and Z. Wu. "Evaluation of Bias-Variance Trade-Off for Commonly Used Post-Summarizing Normalization Procedures in Large-Scale Gene Expression Studies." In: *PLoS ONE* 9.6 (June 2014). Ed. by Z. Wei, e99380. DOI: 10.1371/journal.pone.0099380. URL: https://doi.org/10.1371/journal.pone.0099380 (cit. on p. 36).

[398] Y. Ding and D. Wilkins. "The Effect of Normalization on Microarray Data Analysis." In: *DNA and Cell Biology* 23.10 (Oct. 2004), pp. 635–642. DOI: 10.1089/dna.2004.23.635. URL: https://doi.org/10.1089/dna.2004.23.635 (cit. on p. 36).

[399] Y. Larriba, C. Rueda, M. A. Fernández, and S. D. Peddada. "Microarray Data Normalization and Robust Detection of Rhythmic Features." In: *Methods in Molecular Biology*. Springer New York, 2019, pp. 207–225. DOI: 10.1007/978-1-4939-9442-7_9. URL: https://doi.org/10.1007/978-1-4939-9442-7_9 (cit. on p. 36).

[400] T. Park, S.-G. Yi, S.-H. Kang, S. Lee, Y.-S. Lee, and R. Simon. "Evaluation of normalization methods for microarray data." In: *BMC Bioinformatics* 4.1 (2003), p. 33. DOI: 10.1186/1471-2105-4-33. URL: https://doi.org/10.1186/1471-2105-4-33 (cit. on p. 36).

[401] W. Wu, N. Dave, G. C. Tseng, T. Richards, E. P. Xing, and N. Kaminski. "Comparison of normalization methods for CodeLink Bioarray data." In: *BMC Bioinformatics* 6.1 (Dec. 2005). DOI: 10.1186/1471-2105-6-309. URL: https://doi.org/10.1186/1471-2105-6-309 (cit. on p. 36).

[402] G. C. Tseng. "Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variations and assessment of gene effects." In: *Nucleic Acids Research* 29.12 (June 2001), pp. 2549–2557. DOI: 10.1093/nar/29.12.2549. URL: https://doi.org/10.1093/nar/29.12.2549 (cit. on p. 36).

[403] C. Cheadle, M. P. Vawter, W. J. Freed, and K. G. Becker. "Analysis of Microarray Data Using Z Score Transformation." In: *The Journal of Molecular Diagnostics* 5.2 (May 2003), pp. 73–81. DOI: 10.1016/s1525-1578(10)60455-2. URL: https://doi.org/10.1016/s1525-1578(10)60455-2 (cit. on p. 36).

[404] A. E. Teschendorff, F. Marabita, M. Lechner, T. Bartlett, J. Tegner, D. Gomez-Cabrero, and S. Beck. "A beta-mixture quantile normalization method for correcting probe design bias in Illumina Infinium 450 k DNA methylation data." In: *Bioinformatics* 29.2 (Nov. 2012), pp. 189–196. DOI: 10.1093/bioinformatics/bts680. URL: https://doi.org/10.1093/bioinformatics/bts680 (cit. on p. 36).

[405] J. Maksimovic, L. Gordon, and A. Oshlack. "SWAN: Subset-quantile Within Array Normalization for Illumina Infinium HumanMethylation450 BeadChips." In: *Genome Biology* 13.6 (2012), R44. DOI: 10.1186/gb-2012-13-6-r44. URL: https://doi.org/10.1186/gb-2012-13-6-r44 (cit. on p. 36).

[406] J. A. Berger, S. Hautaniemi, A.-K. Järvinen, H. Edgren, S. K. Mitra, and J. Astola. "Optimized LOWESS normalization parameter selection for DNA microarray data." In: *BMC Bioinformatics* 5.1 (2004), p. 194. DOI: 10.1186/1471-2105-5-194. URL: https://doi.org/10.1186/1471-2105-5-194 (cit. on p. 36).

[407] K. Fundel, R. Küffner, T. Aigner, and R. Zimmer. "Normalization and Gene p-Value Estimation: Issues in Microarray Data Processing." In: *Bioinformatics and Biology Insights* 2 (Jan. 2008), BBI.S441. DOI: 10.4137/bbi.s441. URL: https://doi.org/10.4137/bbi.s441 (cit. on p. 36).

[408] A. L. Tarca and J. E. K. Cooke. "A robust neural networks approach for spatial and intensity-dependent normalization of cDNA microarray data." In: *Bioinformatics* 21.11 (Mar. 2005), pp. 2674–2683. DOI: 10.1093/bioinformatics/bti397. URL: https://doi.org/10.1093/bioinformatics/bti397 (cit. on pp. 36, 46).

[409] N. L. Dawes and J. Glassey. "Normalisation of Multicondition cDNA Macroarray Data." In: *Comparative and Functional Genomics* 2007 (2007), pp. 1–12. DOI: 10.1155/2007/90578. URL: https://doi.org/10.1155/2007/90578 (cit. on p. 36).

[410] Q. Meng, D. Catchpoole, D. Skillicorn, and P. J. Kennedy. "DBNorm: normalizing high-density oligonucleotide microarray data based on distributions." In: *BMC Bioinformatics* 18.1 (Nov. 2017). DOI: 10.1186/s12859-017-1912-5. URL: https://doi.org/10.1186/s12859-017-1912-5 (cit. on p. 36).

[411] T. Majtán, G. Bukovská, and J. Timko. "DNA microarrays — techniques and applications in microbial systems." In: *Folia Microbiologica* 49.6 (Nov. 2004). DOI: 10.1007/bf02931546. URL: https://doi.org/10.1007/bf02931546 (cit. on p. 36).

[412] R. Simon. "Analysis of DNA microarray expression data." In: *Best Practice & Research Clinical Haematology* 22.2 (June 2009), pp. 271–282. DOI: 10.1016/j.beha.2009.07.001. URL: https://doi.org/10.1016/j.beha.2009.07.001 (cit. on p. 36).

[413] R. G. Sanz and A. Sánchez-Pla. "Statistical Analysis of Microarray Data." In: *Methods in Molecular Biology*. Springer New York, 2019, pp. 87–121. DOI: 10.1007/978-1-4939-9442-7_5. URL: https://doi.org/10.1007/978-1-4939-9442-7_5 (cit. on p. 36).

[414] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. "Mapping and quantifying mammalian transcriptomes by RNA-Seq." In: *Nature Methods* 5.7 (May 2008), pp. 621–628. DOI: 10.1038/nmeth.1226. URL: https://doi.org/10.1038/nmeth.1226 (cit. on pp. 36, 37).

[415] U. Nagalakshmi, Z. Wang, K. Waern, C. Shou, D. Raha, M. Gerstein, and M. Snyder. "The Transcriptional Landscape of the Yeast Genome Defined by RNA Sequencing." In: *Science* 320.5881 (June 2008), pp. 1344–1349. DOI: 10.1126/science.1158441. URL: https://doi.org/10.1126/science.1158441 (cit. on p. 36).

[416] B. T. Wilhelm, S. Marguerat, S. Watt, F. Schubert, V. Wood, I. Goodhead, C. J. Penkett, J. Rogers, and J. Bähler. "Dynamic repertoire of a eukaryotic transcriptome surveyed at single-nucleotide resolution." In: *Nature* 453.7199 (May 2008), pp. 1239–1243. DOI: 10.1038/nature07002. URL: https://doi.org/10.1038/nature07002 (cit. on p. 36).

[417] J. Minnier, N. D. Pennock, Q. Guo, P. Schedin, and C. A. Harrington. "RNA-Seq and Expression Arrays: Selection Guidelines for Genome-Wide Expression Profiling." In: *Methods in Molecular Biology*. Springer New York, 2018, pp. 7–33. DOI: 10.1007/978-1-4939-7834-2_2. URL: https://doi.org/10.1007/978-1-4939-7834-2_2 (cit. on pp. 36, 37).

[418] P. Bertone et al. "Global Identification of Human Transcribed Sequences with Genome Tiling Arrays." In: *Science* 306.5705 (Dec. 2004), pp. 2242–2246. DOI: 10.1126/science.1103388. URL: https://doi.org/10.1126/science.1103388 (cit. on p. 37).

[419] T. E. Royce, J. S. Rozowsky, P. Bertone, M. Samanta, V. Stolc, S. Weissman, M. Snyder, and M. Gerstein. "Issues in the analysis of oligonucleotide tiling microarrays for transcript mapping." In: *Trends in Genetics* 21.8 (Aug. 2005), pp. 466–475. DOI: 10.1016/j.tig.2005.06.007. URL: https://doi.org/10.1016/j.tig.2005.06.007 (cit. on p. 37).

[420] P. Kapranov, A. T. Willingham, and T. R. Gingeras. "Genome-wide transcription and the implications for genomic organization." In: *Nature Reviews Genetics* 8.6 (May 2007), pp. 413–423. DOI: 10.1038/nrg2083. URL: https://doi.org/10.1038/nrg2083 (cit. on p. 37).

[421] H. Ishida, T. Yagi, M. Tanaka, Y. Tokuda, K. Kamoi, F. Hongo, A. Kawauchi, M. Nakano, T. Miki, and K. Tashiro. "Identification of a novel gene by whole human genome tiling array." In: *Gene* 516.1 (Mar. 2013), pp. 33–38. DOI: 10.1016/j.gene.2012.11.076. URL: https://doi.org/10.1016/j.gene.2012.11.076 (cit. on p. 37).

[422] M. Griffith, J. R. Walker, N. C. Spies, B. J. Ainscough, and O. L. Griffith. "Informatics for RNA Sequencing: A Web Resource for Analysis on the Cloud." In: *PLOS Computational Biology* 11.8 (Aug. 2015). Ed. by F. Ouellette, e1004393. DOI: 10.1371/journal.pcbi.1004393. URL: https://doi.org/10.1371/journal.pcbi.1004393 (cit. on p. 37).

[423] D. Deshpande et al. "RNA-seq data science: From raw data to effective interpretation." In: *Frontiers in Genetics* 14 (Mar. 2023). DOI: 10.3389/fgene.2023.997383. URL: https://doi.org/10.3389/fgene.2023.997383 (cit. on p. 37).

[424] A. Conesa et al. "A survey of best practices for RNA-seq data analysis." In: *Genome Biology* 17.1 (Jan. 2016). DOI: 10.1186/s13059-016-0881-8. URL: https://doi.org/10.1186/s13059-016-0881-8 (cit. on p. 37).

[425] M. Hong, S. Tao, L. Zhang, L.-T. Diao, X. Huang, S. Huang, S.-J. Xie, Z.-D. Xiao, and H. Zhang. "RNA sequencing: new technologies and applications in cancer research." In: *Journal of Hematology &amp; Oncology* 13.1 (Dec. 2020). DOI: 10.1186/s13045-020-01005-x. URL: https://doi.org/10.1186/s13045-020-01005-x (cit. on pp. 37, 38).

[426] J. Costa-Silva, D. Domingues, and F. M. Lopes. "RNA-Seq differential expression analysis: An extended review and a software tool." In: *PLOS ONE* 12.12 (Dec. 2017). Ed. by Z. Wei, e0190152. DOI: 10.1371/journal.pone.0190152. URL: https://doi.org/10.1371/journal.pone.0190152 (cit. on p. 37).

[427] C. M. Koch, S. F. Chiu, M. Akbarpour, A. Bharat, K. M. Ridge, E. T. Bartom, and D. R. Winter. "A Beginner's Guide to Analysis of RNA Sequencing Data." In: *American Journal of Respiratory Cell and Molecular Biology* 59.2 (Aug. 2018), pp. 145–157. DOI: 10.1165/rcmb.2017-0430tr. URL: https://doi.org/10.1165/rcmb.2017-0430tr (cit. on p. 37).

[428] M. Jeon, Z. Xie, J. E. Evangelista, M. L. Wojciechowicz, D. J. B. Clarke, and A. Ma'ayan. "Transforming L1000 profiles to RNA-seq-like profiles with deep learning." In: *BMC Bioinformatics* 23.1 (Sept. 2022). DOI: 10.1186/s12859-022-04895-5. URL: https://doi.org/10.1186/s12859-022-04895-5 (cit. on pp. 38, 39, 42).

[429] M. Amodio, D. Shung, D. B. Burkhardt, P. Wong, M. Simonov, Y. Yamamoto, D. van Dijk, F. P. Wilson, A. Iwasaki, and S. Krishnaswamy. "Generating hard-to-obtain information from easy-to-obtain information: Applications in drug discovery and clinical inference." In: *Patterns* 2.7 (July 2021), p. 100288. DOI: 10.1016/j.patter.2021.100288. URL: https://doi.org/10.1016/j.patter.2021.100288 (cit. on p. 38).

[430] R. Edgar. "Gene Expression Omnibus: NCBI gene expression and hybridization array data repository." In: *Nucleic Acids Research* 30.1 (Jan. 2002), pp. 207–210. DOI: 10.1093/nar/30.1.207. URL: https://doi.org/10.1093/nar/30.1.207 (cit. on pp. xlvii, 38).

[431] G. C. Tseng and W. H. Wong. "Tight Clustering: A Resampling-Based Approach for Identifying Stable and Tight Patterns in Data." In: *Biometrics* 61.1 (Feb. 2005), pp. 10–16. DOI: 10.1111/j.0006-341x.2005.031032.x. URL: https://doi.org/10.1111/j.0006-341x.2005.031032.x (cit. on p. 39).

[432] L. J. Lancashire et al. "An introduction to artificial neural networks in bioinformatics–application to complex microarray and mass spectrometry datasets in cancer studies." In: *Briefings in Bioinformatics* 10.3 (Dec. 2008), pp. 315–329. DOI: 10.1093/bib/bbp012. URL: https://doi.org/10.1093/bib/bbp012 (cit. on p. 41).

[433] N. Erfanian et al. "Deep learning applications in single-cell genomics and transcriptomics data analysis." In: *Biomedicine &amp; Pharmacotherapy* 165 (Sept. 2023), p. 115077. DOI: 10.1016/j.biopha.2023.115077. URL: https://doi.org/10.1016/j.biopha.2023.115077 (cit. on pp. 41, 44).

[434] J. Liu, J. Li, H. Wang, and J. Yan. "Application of deep learning in genomics." In: *Science China Life Sciences* 63.12 (Oct. 2020), pp. 1860–1878. DOI: 10.1007/s11427-020-1804-5. URL: https://doi.org/10.1007/s11427-020-1804-5 (cit. on p. 41).

[435] W. S. Alharbi and M. Rashid. "A review of deep learning applications in human genomics using next-generation sequencing data." In: *Human Genomics* 16.1 (July 2022). DOI: 10.1186/s40246-022-00396-x. URL: https://doi.org/10.1186/s40246-022-00396-x (cit. on p. 41).

[436] C. Caudai, A. Galizia, F. Geraci, L. L. Pera, V. Morea, E. Salerno, A. Via, and T. Colombo. "AI applications in functional genomics." In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 5762–5790. DOI: 10.1016/j.csbj.2021.10.009. URL: https://doi.org/10.1016/j.csbj.2021.10.009 (cit. on p. 41).

[437] M. Mahmud, M. S. Kaiser, T. M. McGinnity, and A. Hussain. "Deep Learning in Mining Biological Data." In: *Cognitive Computation* 13.1 (Jan. 2021), pp. 1–33. DOI: 10.1007/s12559-020-09773-x. URL: https://doi.org/10.1007/s12559-020-09773-x (cit. on p. 41).

[438] R. Lopez, A. Gayoso, and N. Yosef. "Enhancing scientific discoveries in molecular biology with deep generative models." In: *Molecular Systems Biology* 16.9 (Sept. 2020). DOI: 10.15252/msb.20199198. URL: https://doi.org/10.15252/msb.20199198 (cit. on pp. 41, 43).

[439] A. Eetemadi and I. Tagkopoulos. "Genetic Neural Networks: an artificial neural network architecture for capturing gene expression relationships." In: *Bioinformatics* (Nov. 2018). Ed. by B. Berger. DOI: 10.1093/bioinformatics/bty945. URL: https://doi.org/10.1093/bioinformatics/bty945 (cit. on pp. xlvii, xlviii, 41–43).

[440] T. D. de Bernonville, E. A. Stander, G. D. de Bernonville, S. Besseau, and V. Courdavault. "Predicting Monoterpene Indole Alkaloid-Related Genes from Expression Data with Artificial Neural Networks." In: *Methods in Molecular Biology*. Springer US, 2022, pp. 131–140. DOI: 10.1007/978-1-0716-2349-7_10. URL: https://doi.org/10.1007/978-1-0716-2349-7_10 (cit. on p. 41).

[441] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Y. W. Teh and M. Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (cit. on pp. 41, 51).

[442] G. Consortium. "The Genotype-Tissue Expression (GTEx) pilot analysis: Multitissue gene regulation in humans." In: *Science* 348.6235 (May 2015), pp. 648–660. DOI: 10.1126/science.1262110. URL: https://doi.org/10.1126/science.1262110 (cit. on p. 41).

[443] T. Lappalainen et al. "Transcriptome and genome sequencing uncovers functional variation in humans." In: *Nature* 501.7468 (Sept. 2013), pp. 506–511. DOI: 10.1038/nature12531. URL: https://doi.org/10.1038/nature12531 (cit. on p. 41).

[444] V. Kunc and J. Klema. "On Transformative Adaptive Activation Functions in Neural Networks for Gene Expression Inference." In: *bioRxiv* (2019). DOI: 10.1101/587287. eprint: https://www.biorxiv.org/content/early/2019/03/24/587287.full.pdf. URL: https://www.biorxiv.org/content/early/2019/03/24/587287 (cit. on pp. 42, 194, 198, 278).

[445] H. Wang, C. Li, J. Zhang, J. Wang, Y. Ma, and Y. Lian. "A new LSTM-based gene expression prediction model: L-GEPM." In: *Journal of Bioinformatics and Computational Biology* 17.04 (Aug. 2019), p. 1950022. DOI: 10.1142/s0219720019500227. URL: https://doi.org/10.1142/s0219720019500227 (cit. on p. 42).

[446] W. Li, Y. Yin, X. Quan, and H. Zhang. "Gene Expression Value Prediction Based on XGBoost Algorithm." In: *Frontiers in Genetics* 10 (Nov. 2019). DOI: 10.3389/fgene.2019.01077. URL: https://doi.org/10.3389/fgene.2019.01077 (cit. on p. 42).

[447] M. Schuster and K. Paliwal. "Bidirectional recurrent neural networks." In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093. URL: https://doi.org/10.1109/78.650093 (cit. on pp. xlv, 43).

[448] R. Tibshirani. "Regression Shrinkage and Selection Via the Lasso." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (Jan. 1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x. URL: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x (cit. on pp. xlviii, 43).

[449] S. S. Somathilaka, S. Balasubramaniam, D. P. Martins, and X. Li. "Revealing gene regulation-based neural network computing in bacteria." In: *Biophysical Reports* 3.3 (Sept. 2023), p. 100118. ISSN: 2667-0747. DOI: 10.1016/j.bpr.2023.100118. URL: http://dx.doi.org/10.1016/j.bpr.2023.100118 (cit. on p. 43).

[450] R. Li, L. Li, Y. Xu, and J. Yang. "Machine learning meets omics: applications and perspectives." In: *Briefings in Bioinformatics* 23.1 (Nov. 2021). DOI: 10.1093/bib/bbab460. URL: https://doi.org/10.1093/bib/bbab460 (cit. on p. 43).

[451] M. Lee. "Recent Advances in Generative Adversarial Networks for Gene Expression Data: A Comprehensive Review." In: *Mathematics* 11.14 (July 2023), p. 3055. DOI: 10.3390/math11143055. URL: https://doi.org/10.3390/math11143055 (cit. on pp. 43, 182).

[452] B. Yelmen and F. Jay. "An Overview of Deep Generative Models in Functional and Evolutionary Genomics." In: *Annual Review of Biomedical Data Science* 6.1 (Aug. 2023), pp. 173–189. DOI: 10.1146/annurev-biodatasci-020722-115651. URL: https://doi.org/10.1146/annurev-biodatasci-020722-115651 (cit. on pp. 43, 44).

[453] X. Ai, M. C. Smith, and F. A. Feltus. "Generative adversarial networks applied to gene expression analysis: An interdisciplinary perspective." In: *Computational and Systems Oncology* 3.3 (Aug. 2023). DOI: 10.1002/cso2.1050. URL: https://doi.org/10.1002/cso2.1050 (cit. on pp. 43, 44).

[454] H. Yu and J. D. Welch. “MichiGAN: sampling from disentangled representations of single-cell data using generative adversarial networks.” In: *Genome Biology* 22.1 (May 2021). DOI: 10.1186/s13059-021-02373-4. URL: https://doi.org/10.1186/s13059-021-02373-4 (cit. on pp. 43, 44, 183).

[455] M. Massimino, F. Martorana, S. Stella, S. R. Vitale, C. Tomarchio, L. Manzella, and P. Vigneri. “Single-Cell Analysis in the Omics Era: Technologies and Applications in Cancer.” In: *Genes* 14.7 (June 2023), p. 1330. ISSN: 2073-4425. DOI: 10.3390/genes14071330. URL: http://dx.doi.org/10.3390/genes14071330 (cit. on p. 43).

[456] M. Marouf, P. Machart, V. Bansal, C. Kilian, D. S. Magruder, C. F. Krebs, and S. Bonn. “Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks.” In: *Nature Communications* 11.1 (Jan. 2020). DOI: 10.1038/s41467-019-14018-z. URL: https://doi.org/10.1038/s41467-019-14018-z (cit. on p. 44).

[457] S. Lall, S. Ray, and S. Bandyopadhyay. “LSH-GAN enables in-silico generation of cells for small sample high dimensional scRNA-seq data.” In: *Communications Biology* 5.1 (June 2022). DOI: 10.1038/s42003-022-03473-y. URL: https://doi.org/10.1038/s42003-022-03473-y (cit. on p. 44).

[458] R. Viñas, H. Andrés-Terré, P. Liò, and K. Bryson. “Adversarial generation of gene expression data.” In: *Bioinformatics* 38.3 (Jan. 2021). Ed. by P. L. Martelli, pp. 730–737. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btab035. URL: http://dx.doi.org/10.1093/bioinformatics/btab035 (cit. on p. 44).

[459] J. E. Park, D. Eun, H. S. Kim, D. H. Lee, R. W. Jang, and N. Kim. “Generative adversarial network for glioblastoma ensures morphologic variations and improves diagnostic model for isocitrate dehydrogenase mutant type.” In: *Scientific Reports* 11.1 (May 2021). DOI: 10.1038/s41598-021-89477-w. URL: https://doi.org/10.1038/s41598-021-89477-w (cit. on pp. 44, 47, 183).

[460] P. Chaudhari, H. Agrawal, and K. Kotecha. “Data augmentation using MG-GAN for improved cancer classification on gene expression data.” In: *Soft Computing* 24.15 (Dec. 2019), pp. 11381–11391. ISSN: 1433-7479. DOI: 10.1007/s00500-019-04602-2. URL: http://dx.doi.org/10.1007/s00500-019-04602-2 (cit. on p. 44).

[461] Y. Xiao, J. Wu, and Z. Lin. “Cancer diagnosis using generative adversarial networks based on deep learning from imbalanced data.” In: *Computers in Biology and Medicine* 135 (Aug. 2021), p. 104540. DOI: 10.1016/j.compbiomed.2021.104540. URL: https://doi.org/10.1016/j.compbiomed.2021.104540 (cit. on p. 44).

[462] S. Zhu and F. Han. “A Data Enhancement Method for Gene Expression Profile Based on Improved WGAN-GP.” In: *Communications in Computer and Information Science*. Springer Singapore, 2021, pp. 242–254. ISBN: 9789811651885. DOI: 10.1007/978-981-16-5188-5_18. URL: http://dx.doi.org/10.1007/978-981-16-5188-5_18 (cit. on p. 44).

[463] A. Tsourtis, G. Papoutsoglou, and Y. Pantazis. “GAN-Based Training of Semi-Interpretable Generators for Biological Data Interpolation and Augmentation.” In: *Applied Sciences* 12.11 (May 2022), p. 5434. DOI: 10.3390/app12115434. URL: https://doi.org/10.3390/app12115434 (cit. on p. 44).

[464] H. Wang and X. Ma. “Learning discriminative and structural samples for rare cell types with deep generative model.” In: *Briefings in Bioinformatics* 23.5 (Aug. 2022). DOI: 10.1093/bib/bbac317. URL: https://doi.org/10.1093/bib/bbac317 (cit. on p. 44).

[465] M. Oh and L. Zhang. “Generalizing predictions to unseen sequencing profiles via deep generative models.” In: *Scientific Reports* 12.1 (May 2022). DOI: 10.1038/s41598-022-11363-w. URL: https://doi.org/10.1038/s41598-022-11363-w (cit. on p. 44).

[466] F. J. Moreno-Barea, J. M. Jerez, and L. Franco. “GAN-Based Data Augmentation for Prediction Improvement Using Gene Expression Data in Cancer.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2022, pp. 28–42. ISBN: 9783031087578. DOI: 10.1007/978-3-031-08757-8_3. URL: http://dx.doi.org/10.1007/978-3-031-08757-8_3 (cit. on p. 44).

[467] F. Han, S. Zhu, Q. Ling, H. Han, H. Li, X. Guo, and J. Cao. “Gene-CWGAN: a data enhancement method for gene expression profile based on improved CWGAN-GP.” In: *Neural Computing and Applications* 34.19 (June 2022), pp. 16325–16339. ISSN: 1433-3058. DOI: 10.1007/s00521-022-07417-9. URL: http://dx.doi.org/10.1007/s00521-022-07417-9 (cit. on p. 44).

[468] R. Li, J. Wu, G. Li, J. Liu, J. Xuan, and Q. Zhu. “Mdwgan-gp: data augmentation for gene expression data based on multiple discriminator WGAN-GP.” In: *BMC Bioinformatics* 24.1 (Nov. 2023). ISSN: 1471-2105. DOI: 10.1186/s12859-023-05558-9. URL: http://dx.doi.org/10.1186/s12859-023-05558-9 (cit. on p. 44).

[469] S. Fang, F. Han, W.-Y. Liang, and J. Jiang. “An Improved Conditional Generative Adversarial Network for Microarray Data.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 105–114. ISBN: 9783030607999. DOI: 10.1007/978-3-030-60799-9_9. URL: http://dx.doi.org/10.1007/978-3-030-60799-9_9 (cit. on p. 44).

[470] K. Wei, T. Li, F. Huang, J. Chen, and Z. He. "Cancer classification with data augmentation based on generative adversarial networks." In: *Frontiers of Computer Science* 16.2 (Sept. 2021). ISSN: 2095-2236. DOI: 10.1007/s11704-020-0025-x. URL: http://dx.doi.org/10.1007/s11704-020-0025-x (cit. on p. 44).

[471] C. Kwon, S. Park, S. Ko, and J. Ahn. "Increasing prediction accuracy of pathogenic staging by sample augmentation with a GAN." In: *PLOS ONE* 16.4 (Apr. 2021). Ed. by P. Pławiak, e0250458. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0250458. URL: http://dx.doi.org/10.1371/journal.pone.0250458 (cit. on p. 44).

[472] H. Ravaee, M. H. Manshaei, M. Safayani, and J. S. Sartakhti. "Intelligent Phenotype-detection and Gene expression profile Generation with Generative adversarial networks." In: *Journal of Theoretical Biology* (Nov. 2023), p. 111636. ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2023.111636. URL: http://dx.doi.org/10.1016/j.jtbi.2023.111636 (cit. on p. 44).

[473] Y. Chung and H. Lee. "Joint triplet loss with semi-hard constraint for data augmentation and disease prediction using gene expression data." In: *Scientific Reports* 13.1 (Oct. 2023). ISSN: 2045-2322. DOI: 10.1038/s41598-023-45467-8. URL: http://dx.doi.org/10.1038/s41598-023-45467-8 (cit. on p. 44).

[474] X. Chen, R. Roberts, W. Tong, and Z. Liu. "Tox-GAN: An Artificial Intelligence Approach Alternative to Animal Studies—A Case Study With Toxicogenomics." In: *Toxicological Sciences* 186.2 (Dec. 2021), pp. 242–259. ISSN: 1096-0929. DOI: 10.1093/toxsci/kfab157. URL: http://dx.doi.org/10.1093/toxsci/kfab157 (cit. on p. 44).

[475] T. Sun, D. Song, W. V. Li, and J. J. Li. "scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured." In: *Genome Biology* 22.1 (May 2021). DOI: 10.1186/s13059-021-02367-2. URL: https://doi.org/10.1186/s13059-021-02367-2 (cit. on p. 44).

[476] P. Dibaeinia and S. Sinha. "SERGIO: A Single-Cell Expression Simulator Guided by Gene Regulatory Networks." In: *Cell Systems* 11.3 (Sept. 2020), 252–271.e11. DOI: 10.1016/j.cels.2020.08.003. URL: https://doi.org/10.1016/j.cels.2020.08.003 (cit. on p. 44).

[477] C. Wan and D. T. Jones. "Protein function prediction is improved by creating synthetic feature samples with generative adversarial networks." In: *Nature Machine Intelligence* 2.9 (Aug. 2020), pp. 540–550. DOI: 10.1038/s42256-020-0222-1. URL: https://doi.org/10.1038/s42256-020-0222-1 (cit. on p. 44).

[478] O. Méndez-Lucio, B. Baillif, D.-A. Clevert, D. Rouquié, and J. Wichard. "De novo generation of hit-like molecules from gene expression signatures using artificial intelligence." In: *Nature Communications* 11.1 (Jan. 2020). ISSN: 2041-1723. DOI: 10.1038/s41467-019-13807-w. URL: http://dx.doi.org/10.1038/s41467-019-13807-w (cit. on p. 44).

[479] B. Yelmen, A. Decelle, L. Ongaro, D. Marnetto, C. Tallec, F. Montinaro, C. Furtlehner, L. Pagani, and F. Jay. "Creating artificial human genomes using generative neural networks." In: *PLOS Genetics* 17.2 (Feb. 2021). Ed. by S. Mathieson, e1009303. ISSN: 1553-7404. DOI: 10.1371/journal.pgen.1009303. URL: http://dx.doi.org/10.1371/journal.pgen.1009303 (cit. on p. 44).

[480] R. K. Mondol, N. D. Truong, M. Reza, S. Ippolito, E. Ebrahimie, and O. Kavehei. "AFExNet: An Adversarial Autoencoder for Differentiating Breast Cancer Sub-Types and Extracting Biologically Relevant Genes." In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19.4 (July 2022), pp. 2060–2070. DOI: 10.1109/tcbb.2021.3066086. URL: https://doi.org/10.1109/tcbb.2021.3066086 (cit. on p. 44).

[481] G. Eraslan, L. M. Simon, M. Mircea, N. S. Mueller, and F. J. Theis. "Single-cell RNA-seq denoising using a deep count autoencoder." In: *Nature Communications* 10.1 (Jan. 2019). DOI: 10.1038/s41467-018-07931-2. URL: https://doi.org/10.1038/s41467-018-07931-2 (cit. on p. 44).

[482] J. Wang, D. Agarwal, M. Huang, G. Hu, Z. Zhou, C. Ye, and N. R. Zhang. "Data denoising with transfer learning in single-cell transcriptomics." In: *Nature Methods* 16.9 (Aug. 2019), pp. 875–878. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0537-1. URL: http://dx.doi.org/10.1038/s41592-019-0537-1 (cit. on p. 44).

[483] Q. Li. "scTour: a deep learning architecture for robust inference and accurate prediction of cellular dynamics." In: *Genome Biology* 24.1 (June 2023). DOI: 10.1186/s13059-023-02988-9. URL: https://doi.org/10.1186/s13059-023-02988-9 (cit. on p. 44).

[484] L. Chen, Y. Zhai, Q. He, W. Wang, and M. Deng. "Integrating Deep Supervised, Self-Supervised and Unsupervised Learning for Single-Cell RNA-seq Clustering and Annotation." In: *Genes* 11.7 (July 2020), p. 792. DOI: 10.3390/genes11070792. URL: https://doi.org/10.3390/genes11070792 (cit. on p. 44).

[485] D. Pandey and P. O. Perumal. "A scoping review on deep learning for next-generation RNA-Seq. data analysis." In: *Functional &amp; Integrative Genomics* 23.2 (Apr. 2023). DOI: 10.1007/s10142-023-01064-6. URL: https://doi.org/10.1007/s10142-023-01064-6 (cit. on p. 44).

[486] M. Brendel, C. Su, Z. Bai, H. Zhang, O. Elemento, and F. Wang. "Application of Deep Learning on Single-cell RNA Sequencing Data Analysis: A Review." In: *Genomics, Proteomics & Bioinformatics* 20.5 (Oct. 2022), pp. 814–835. ISSN: 1672-0229. DOI: 10.1016/j.gpb.2022.11.011. URL: http://dx.doi.org/10.1016/j.gpb.2022.11.011 (cit. on p. 44).

[487] J. Wang, Q. Zou, and C. Lin. "A comparison of deep learning-based pre-processing and clustering approaches for single-cell RNA sequencing data." In: *Briefings in Bioinformatics* 23.1 (Sept. 2021). DOI: 10.1093/bib/bbab345. URL: https://doi.org/10.1093/bib/bbab345 (cit. on p. 44).

[488] M. Flores et al. "Deep learning tackles single-cell analysis—a survey of deep learning for scRNA-seq analysis." In: *Briefings in Bioinformatics* 23.1 (Dec. 2021). DOI: 10.1093/bib/bbab531. URL: https://doi.org/10.1093/bib/bbab531 (cit. on p. 44).

[489] A. A. Heydari and S. S. Sindi. "Deep learning in spatial transcriptomics: Learning from the next next-generation sequencing." In: *Biophysics Reviews* 4.1 (Feb. 2023). DOI: 10.1063/5.0091135. URL: https://doi.org/10.1063/5.0091135 (cit. on p. 44).

[490] M. Lotfollahi, F. A. Wolf, and F. J. Theis. "scGen predicts single-cell perturbation responses." In: *Nature Methods* 16.8 (July 2019), pp. 715–721. DOI: 10.1038/s41592-019-0494-8. URL: https://doi.org/10.1038/s41592-019-0494-8 (cit. on pp. 44, 46, 181).

[491] M. Lotfollahi, M. Naghipourfar, F. J. Theis, and F. A. Wolf. "Conditional out-of-distribution generation for unpaired data using transfer VAE." In: *Bioinformatics* 36.Supplement_2 (Dec. 2020), pp. i610–i617. DOI: 10.1093/bioinformatics/btaa800. URL: https://doi.org/10.1093/bioinformatics/btaa800 (cit. on pp. 44, 181).

[492] A. Gupta, H. Wang, and M. Ganapathiraju. "Learning structure in gene expression data using deep architectures, with an application to gene clustering." In: *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Nov. 2015. DOI: 10.1109/bibm.2015.7359871. URL: https://doi.org/10.1109/bibm.2015.7359871 (cit. on p. 44).

[493] M. Treppner, A. Salas-Bastos, M. Hess, S. Lenz, T. Vogel, and H. Binder. "Synthetic single cell RNA sequencing data from small pilot studies using deep generative models." In: *Scientific Reports* 11.1 (Apr. 2021). DOI: 10.1038/s41598-021-88875-4. URL: https://doi.org/10.1038/s41598-021-88875-4 (cit. on p. 44).

[494] Y. Pantazis, C. Tselas, K. Lakiotaki, V. Lagani, and I. Tsamardinos. "Latent Feature Representations for Human Gene Expression Data Improve Phenotypic Predictions." In: *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Dec. 2020. DOI: 10.1109/bibm49941.2020.9313286. URL: https://doi.org/10.1109/bibm49941.2020.9313286 (cit. on p. 45).

[495] H. Duan, F. Li, J. Shang, J. Liu, Y. Li, and X. Liu. "scVAEBGM: Clustering Analysis of Single-Cell ATAC-seq Data Using a Deep Generative Model." In: *Interdisciplinary Sciences: Computational Life Sciences* 14.4 (Aug. 2022), pp. 917–928. DOI: 10.1007/s12539-022-00536-w. URL: https://doi.org/10.1007/s12539-022-00536-w (cit. on p. 45).

[496] B. Tran, D. Tran, H. Nguyen, S. Ro, and T. Nguyen. "scCAN: single-cell clustering using autoencoder and network fusion." In: *Scientific Reports* 12.1 (June 2022). DOI: 10.1038/s41598-022-14218-6. URL: https://doi.org/10.1038/s41598-022-14218-6 (cit. on p. 45).

[497] K. Märtens and C. Yau. "BasisVAE: Translation-invariant feature-level clustering with Variational Autoencoders." In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 26–28 Aug 2020, pp. 2928–2937. URL: https://proceedings.mlr.press/v108/martens20b.html (cit. on p. 45).

[498] Y. Zeng, Z. Wei, F. Zhong, Z. Pan, Y. Lu, and Y. Yang. "A parameter-free deep embedded clustering method for single-cell RNA-seq data." In: *Briefings in Bioinformatics* 23.5 (May 2022). DOI: 10.1093/bib/bbac172. URL: https://doi.org/10.1093/bib/bbac172 (cit. on p. 45).

[499] C. H. Grønbech, M. F. Vording, P. N. Timshel, C. K. Sønderby, T. H. Pers, and O. Winther. "scVAE: variational auto-encoders for single-cell gene expression data." In: *Bioinformatics* 36.16 (May 2020). Ed. by B. Berger, pp. 4415–4422. DOI: 10.1093/bioinformatics/btaa293. URL: https://doi.org/10.1093/bioinformatics/btaa293 (cit. on p. 45).

[500] J. Peng, X. Wang, and X. Shang. "Combining gene ontology with deep neural networks to enhance the clustering of single cell RNA-Seq data." In: *BMC Bioinformatics* 20.S8 (June 2019). DOI: 10.1186/s12859-019-2769-6. URL: https://doi.org/10.1186/s12859-019-2769-6 (cit. on p. 45).

[501] S. He, J. Fan, and T. Yu. "G3DC: a Gene-Graph-Guided selective Deep Clustering method for single cell RNA-seq data." In: (Jan. 2023). DOI: 10.1101/2023.01.15.524109. URL: https://doi.org/10.1101/2023.01.15.524109 (cit. on p. 45).

[502] L. Seninge, I. Anastopoulos, H. Ding, and J. Stuart. "VEGA is an interpretable generative model for inferring biological network activity in single-cell transcriptomics." In: *Nature Communications* 12.1 (Sept. 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-26017-0. URL: http://dx.doi.org/10.1038/s41467-021-26017-0 (cit. on p. 45).

[503] J. S. Walbech, S. Kinalis, O. Winther, F. C. Nielsen, and F. O. Bagger. "Interpretable Autoencoders Trained on Single Cell Sequencing Data Can Transfer Directly to Data from Unseen Tissues." In: *Cells* 11.1 (Dec. 2021), p. 85. ISSN: 2073-4409. DOI: 10.3390/cells11010085. URL: http://dx.doi.org/10.3390/cells11010085 (cit. on p. 45).

[504] Z. Wang, H. Wang, J. Zhao, and C. Zheng. "scSemiAAE: a semi-supervised clustering model for single-cell RNA-seq data." In: *BMC Bioinformatics* 24.1 (May 2023). DOI: 10.1186/s12859-023-05339-4. URL: https://doi.org/10.1186/s12859-023-05339-4 (cit. on p. 45).

[505] S. K. Pati, M. K. Gupta, R. Shai, A. Banerjee, and A. Ghosh. "Missing value estimation of microarray data using Sim-GAN." In: *Knowledge and Information Systems* 64.10 (July 2022), pp. 2661–2687. ISSN: 0219-3116. DOI: 10.1007/s10115-022-01718-0. URL: http://dx.doi.org/10.1007/s10115-022-01718-0 (cit. on p. 45).

[506] S. Zhao, L. Zhang, and X. Liu. "AE-TPGG: a novel autoencoder-based approach for single-cell RNA-seq data imputation and dimensionality reduction." In: *Frontiers of Computer Science* 17.3 (Oct. 2022). DOI: 10.1007/s11704-022-2011-y. URL: https://doi.org/10.1007/s11704-022-2011-y (cit. on p. 45).

[507] A. B. Dincer, J. D. Janizek, and S.-I. Lee. "Adversarial deconfounding autoencoder for learning robust gene expression embeddings." In: *Bioinformatics* 36.Supplement_2 (Dec. 2020), pp. i573–i582. DOI: 10.1093/bioinformatics/btaa796. URL: https://doi.org/10.1093/bioinformatics/btaa796 (cit. on p. 45).

[508] M. B. Badsha, R. Li, B. Liu, Y. I. Li, M. Xian, N. E. Banovich, and A. Q. Fu. "Imputation of single-cell gene expression with an autoencoder neural network." In: *Quantitative Biology* 8.1 (Jan. 2020), pp. 78–94. DOI: 10.1007/s40484-019-0192-7. URL: https://doi.org/10.1007/s40484-019-0192-7 (cit. on p. 45).

[509] C. Arisdakessian, O. Poirion, B. Yunits, X. Zhu, and L. X. Garmire. "DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data." In: *Genome Biology* 20.1 (Oct. 2019). DOI: 10.1186/s13059-019-1837-6. URL: https://doi.org/10.1186/s13059-019-1837-6 (cit. on p. 45).

[510] D. Talwar, A. Mongia, D. Sengupta, and A. Majumdar. "AutoImpute: Autoencoder based imputation of single-cell RNA-seq data." In: *Scientific Reports* 8.1 (Nov. 2018). DOI: 10.1038/s41598-018-34688-x. URL: https://doi.org/10.1038/s41598-018-34688-x (cit. on p. 45).

[511] D. Pandey and P. P. Onkara. "Improved downstream functional analysis of single-cell RNA-sequence data using DGAN." In: *Scientific Reports* 13.1 (Jan. 2023). DOI: 10.1038/s41598-023-28952-y. URL: https://doi.org/10.1038/s41598-023-28952-y (cit. on p. 45).

[512] Y. He, H. Yuan, C. Wu, and Z. Xie. "DISC: a highly scalable and accurate inference of gene expression and structure for single-cell transcriptomes using semi-supervised deep learning." In: *Genome Biology* 21.1 (July 2020). DOI: 10.1186/s13059-020-02083-3. URL: https://doi.org/10.1186/s13059-020-02083-3 (cit. on p. 45).

[513] D. Tran, F. C. Harris, B. Tran, N. S. Vo, H. Nguyen, and T. Nguyen. "Single-Cell RNA Sequencing Data Imputation Using Deep Neural Network." In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2021, pp. 403–410. DOI: 10.1007/978-3-030-70416-2_52. URL: https://doi.org/10.1007/978-3-030-70416-2_52 (cit. on p. 45).

[514] R. Viñas, T. Azevedo, E. R. Gamazon, and P. Liò. "Deep Learning Enables Fast and Accurate Imputation of Gene Expression." In: *Frontiers in Genetics* 12 (Apr. 2021). DOI: 10.3389/fgene.2021.624128. URL: https://doi.org/10.3389/fgene.2021.624128 (cit. on p. 45).

[515] Y. Xu, Z. Zhang, L. You, J. Liu, Z. Fan, and X. Zhou. "scIGANs: single-cell RNA-seq imputation using generative adversarial networks." In: *Nucleic Acids Research* 48.15 (June 2020), e85–e85. DOI: 10.1093/nar/gkaa506. URL: https://doi.org/10.1093/nar/gkaa506 (cit. on p. 45).

[516] M. K. Gunady, J. Kancherla, H. C. Bravo, and S. Feizi. "scGAIN: Single Cell RNA-seq Data Imputation using Generative Adversarial Networks." In: (Nov. 2019). DOI: 10.1101/837302. URL: https://doi.org/10.1101/837302 (cit. on p. 45).

[517] Y. Deng, F. Bao, Q. Dai, L. F. Wu, and S. J. Altschuler. "Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning." In: *Nature Methods* 16.4 (Mar. 2019), pp. 311–314. DOI: 10.1038/s41592-019-0353-7. URL: https://doi.org/10.1038/s41592-019-0353-7 (cit. on p. 45).

[518] M. Amodio et al. "Exploring single-cell data with deep multitasking neural networks." In: *Nature Methods* 16.11 (Oct. 2019), pp. 1139–1145. DOI: 10.1038/s41592-019-0576-7. URL: https://doi.org/10.1038/s41592-019-0576-7 (cit. on p. 45).

[519] D. Tran, H. Nguyen, B. Tran, C. L. Vecchia, H. N. Luu, and T. Nguyen. "Fast and precise single-cell data analysis using a hierarchical autoencoder." In: *Nature Communications* 12.1 (Feb. 2021). DOI: 10.1038/s41467-021-21312-2. URL: https://doi.org/10.1038/s41467-021-21312-2 (cit. on p. 45).

[520] M. Huang, Z. Zhang, and N. R. Zhang. "Dimension reduction and denoising of single-cell RNA sequencing data in the presence of observed confounding variables." In: (Aug. 2020). DOI: 10.1101/2020.08.03.234765. URL: http://dx.doi.org/10.1101/2020.08.03.234765 (cit. on p. 45).

[521] X. Li, S. Li, L. Huang, S. Zhang, and K.-c. Wong. "High-throughput single-cell RNA-seq data imputation and characterization with surrogate-assisted automated deep learning." In: *Briefings in Bioinformatics* 23.1 (Sept. 2021). ISSN: 1477-4054. DOI: 10.1093/bib/bbab368. URL: http://dx.doi.org/10.1093/bib/bbab368 (cit. on p. 45).

[522] T. Tian, M. R. Min, and Z. Wei. "Model-based autoencoders for imputing discrete single-cell RNA-seq data." In: *Methods* 192 (Aug. 2021), pp. 112–119. ISSN: 1046-2023. DOI: 10.1016/j.ymeth.2020.09.010. URL: http://dx.doi.org/10.1016/j.ymeth.2020.09.010 (cit. on p. 45).

[523] W. Chi and M. Deng. "Sparsity-Penalized Stacked Denoising Autoencoders for Imputing Single-Cell RNA-seq Data." In: *Genes* 11.5 (May 2020), p. 532. ISSN: 2073-4425. DOI: 10.3390/genes11050532. URL: http://dx.doi.org/10.3390/genes11050532 (cit. on p. 45).

[524] J. Rao, X. Zhou, Y. Lu, H. Zhao, and Y. Yang. "Imputing single-cell RNA-seq data by combining graph convolution and autoencoder neural networks." In: *iScience* 24.5 (May 2021), p. 102393. ISSN: 2589-0042. DOI: 10.1016/j.isci.2021.102393. URL: http://dx.doi.org/10.1016/j.isci.2021.102393 (cit. on p. 45).

[525] L. Xu, Y. Xu, T. Xue, X. Zhang, and J. Li. "AdImpute: An Imputation Method for Single-Cell RNA-Seq Data Based on Semi-Supervised Autoencoders." In: *Frontiers in Genetics* 12 (Sept. 2021). ISSN: 1664-8021. DOI: 10.3389/fgene.2021.739677. URL: http://dx.doi.org/10.3389/fgene.2021.739677 (cit. on p. 45).

[526] C. Xu, L. Cai, and J. Gao. "An efficient scRNA-seq dropout imputation method using graph attention network." In: *BMC Bioinformatics* 22.1 (Dec. 2021). ISSN: 1471-2105. DOI: 10.1186/s12859-021-04493-x. URL: http://dx.doi.org/10.1186/s12859-021-04493-x (cit. on p. 45).

[527] X. Zhang, Z. Chen, R. Bhadani, S. Cao, M. Lu, N. Lytal, Y. Chen, and L. An. "NISC: Neural Network-Imputation for Single-Cell RNA Sequencing and Cell Type Clustering." In: *Frontiers in Genetics* 13 (May 2022). ISSN: 1664-8021. DOI: 10.3389/fgene.2022.847112. URL: http://dx.doi.org/10.3389/fgene.2022.847112 (cit. on p. 45).

[528] L. Huang, M. Song, H. Shen, H. Hong, P. Gong, H.-W. Deng, and C. Zhang. "Deep Learning Methods for Omics Data Imputation." In: *Biology* 12.10 (Oct. 2023), p. 1313. DOI: 10.3390/biology12101313. URL: https://doi.org/10.3390/biology12101313 (cit. on p. 45).

[529] R. Shahbazian and S. Greco. "Generative Adversarial Networks Assist Missing Data Imputation: A Comprehensive Survey and Evaluation." In: *IEEE Access* 11 (2023), pp. 88908–88928. ISSN: 2169-3536. DOI: 10.1109/access.2023.3306721. URL: http://dx.doi.org/10.1109/ACCESS.2023.3306721 (cit. on p. 45).

[530] F. Hausmann, C. Ergen, R. Khatri, M. Marouf, S. Hänzelmann, N. Gagliani, S. Huber, P. Machart, and S. Bonn. "DISCERN: deep single-cell expression reconstruction for improved cell clustering and cell subtype and state detection." In: *Genome Biology* 24.1 (Sept. 2023). DOI: 10.1186/s13059-023-03049-x. URL: https://doi.org/10.1186/s13059-023-03049-x (cit. on p. 45).

[531] M. Kircher, E. Chludzinski, J. Krepel, B. Saremi, A. Beineke, and K. Jung. "Augmentation of Transcriptomic Data for Improved Classification of Patients with Respiratory Diseases of Viral Origin." In: *International Journal of Molecular Sciences* 23.5 (Feb. 2022), p. 2481. DOI: 10.3390/ijms23052481. URL: https://doi.org/10.3390/ijms23052481 (cit. on p. 45).

[532] B. Jahanyar, H. Tabatabaee, and A. Rowhanimanesh. "MS-ACGAN: A modified auxiliary classifier generative adversarial network for schizophrenia's samples augmentation based on microarray gene expression data." In: *Computers in Biology and Medicine* 162 (Aug. 2023), p. 107024. DOI: 10.1016/j.compbiomed.2023.107024. URL: https://doi.org/10.1016/j.compbiomed.2023.107024 (cit. on p. 45).

[533] A. Lacan, M. Sebag, and B. Hanczar. "GAN-based data augmentation for transcriptomics: survey and comparative assessment." In: *Bioinformatics* 39.Supplement_1 (June 2023), pp. i111–i120. DOI: 10.1093/bioinformatics/btad239. URL: https://doi.org/10.1093/bioinformatics/btad239 (cit. on p. 45).

[534] G. He, M. Chen, Y. Bian, and E. Yang. "MTM: a multi-task learning framework to predict individualized tissue gene expression profiles." In: *Bioinformatics* 39.6 (June 2023). Ed. by I. Birol. DOI: 10.1093/bioinformatics/btad363. URL: https://doi.org/10.1093/bioinformatics/btad363 (cit. on p. 45).

[535] S. Kinalis, F. C. Nielsen, O. Winther, and F. O. Bagger. “Deconvolution of autoencoders to learn biological regulatory modules from single cell mRNA sequencing data.” In: *BMC Bioinformatics* 20.1 (July 2019). DOI: 10.1186/s12859-019-2952-9. URL: https://doi.org/10.1186/s12859-019-2952-9 (cit. on p. 45).

[536] L. Alessandri and R. A. Calogero. “Functional-Feature-Based Data Reduction Using Sparsely Connected Autoencoders.” In: *Methods in Molecular Biology*. Springer US, Dec. 2022, pp. 231–240. DOI: 10.1007/978-1-0716-2756-3_11. URL: https://doi.org/10.1007/978-1-0716-2756-3_11 (cit. on p. 45).

[537] J. Ding. “A versatile model for single-cell data analysis.” In: *Nature Computational Science* 1.7 (July 2021), pp. 460–461. DOI: 10.1038/s43588-021-00103-1. URL: https://doi.org/10.1038/s43588-021-00103-1 (cit. on p. 45).

[538] H. Shu, J. Zhou, Q. Lian, H. Li, D. Zhao, J. Zeng, and J. Ma. “Modeling gene regulatory networks using neural network architectures.” In: *Nature Computational Science* 1.7 (July 2021), pp. 491–501. DOI: 10.1038/s43588-021-00099-8. URL: https://doi.org/10.1038/s43588-021-00099-8 (cit. on p. 45).

[539] Y. Chen, Y. Wang, Y. Chen, Y. Cheng, Y. Wei, Y. Li, J. Wang, Y. Wei, T.-F. Chan, and Y. Li. “Deep autoencoder for interpretable tissue-adaptive deconvolution and cell-type-specific gene analysis.” In: *Nature Communications* 13.1 (Nov. 2022). DOI: 10.1038/s41467-022-34550-9. URL: https://doi.org/10.1038/s41467-022-34550-9 (cit. on p. 46).

[540] F. Jackson and T. Lukasiewicz. “Deconvolution of cell-free DNA in cancer liquid biopsy using a deep AutoEncoder.” In: *Proceedings of the 14th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, Sept. 2023. DOI: 10.1145/3584371.3612976. URL: https://doi.org/10.1145/3584371.3612976 (cit. on p. 46).

[541] T. A. Geddes, T. Kim, L. Nan, J. G. Burchfield, J. Y. H. Yang, D. Tao, and P. Yang. “Autoencoder-based cluster ensembles for single-cell RNA-seq data analysis.” In: *BMC Bioinformatics* 20.S19 (Dec. 2019). DOI: 10.1186/s12859-019-3179-5. URL: https://doi.org/10.1186/s12859-019-3179-5 (cit. on p. 46).

[542] W. Pan, F. Long, and J. Pan. “ScInfoVAE: interpretable dimensional reduction of single cell transcription data with variational autoencoders and extended mutual information regularization.” In: *BioData Mining* 16.1 (June 2023). ISSN: 1756-0381. DOI: 10.1186/s13040-023-00333-1. URL: http://dx.doi.org/10.1186/s13040-023-00333-1 (cit. on p. 46).

[543] T. Nguyen, Y. Wei, Y. Nakada, J. Y. Chen, Y. Zhou, G. Walcott, and J. Zhang. “Analysis of cardiac single-cell RNA-sequencing data can be improved by the use of artificial-intelligence-based tools.” In: *Scientific Reports* 13.1 (Apr. 2023). DOI: 10.1038/s41598-023-32293-1. URL: https://doi.org/10.1038/s41598-023-32293-1 (cit. on p. 46).

[544] J. D. Janizek, A. Spiro, S. Celik, B. W. Blue, J. C. Russell, T.-I. Lee, M. Kaeberlin, and S.-I. Lee. “PAUSE: principled feature attribution for unsupervised gene expression analysis.” In: *Genome Biology* 24.1 (Apr. 2023). DOI: 10.1186/s13059-023-02901-4. URL: https://doi.org/10.1186/s13059-023-02901-4 (cit. on p. 46).

[545] J. Ding and A. Regev. “Deep generative model embedding of single-cell RNA-Seq profiles on hyperspheres and hyperbolic spaces.” In: *Nature Communications* 12.1 (May 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-22851-4. URL: http://dx.doi.org/10.1038/s41467-021-22851-4 (cit. on p. 46).

[546] B. Mieth, J. R. F. Hockley, N. Görnitz, M. M.-C. Vidovic, K.-R. Müller, A. Gutteridge, and D. Ziemek. “Using transfer learning from prior reference knowledge to improve the clustering of single-cell RNA-Seq data.” In: *Scientific Reports* 9.1 (Dec. 2019). ISSN: 2045-2322. DOI: 10.1038/s41598-019-56911-z. URL: http://dx.doi.org/10.1038/s41598-019-56911-z (cit. on p. 46).

[547] L. Yu, C. Liu, J. Y. H. Yang, and P. Yang. “Ensemble deep learning of embeddings for clustering multimodal single-cell omics data.” In: *Bioinformatics* 39.6 (June 2023). Ed. by M. Nikolski. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad382. URL: http://dx.doi.org/10.1093/bioinformatics/btad382 (cit. on p. 46).

[548] T. Zhang, A. Amirsoleimani, J. K. Eshraghian, M. R. Azghadi, R. Genov, and Y. Xia. “SSCAE: A Neuromorphic SNN Autoencoder for sc-RNA-seq Dimensionality Reduction.” In: *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2023. DOI: 10.1109/iscas46773.2023.10181994. URL: http://dx.doi.org/10.1109/ISCAS46773.2023.10181994 (cit. on p. 46).

[549] E. Lin, S. Mukherjee, and S. Kannan. “A deep adversarial variational autoencoder model for dimensionality reduction in single-cell RNA sequencing analysis.” In: *BMC Bioinformatics* 21.1 (Feb. 2020). ISSN: 1471-2105. DOI: 10.1186/s12859-020-3401-5. URL: http://dx.doi.org/10.1186/s12859-020-3401-5 (cit. on p. 46).

[550] C. Zhang. “Single-Cell Data Analysis Using MMD Variational Autoencoder for a More Informative Latent Representation.” In: (Apr. 2019). DOI: 10.1101/613414. URL: http://dx.doi.org/10.1101/613414 (cit. on p. 46).

[551] J. C. Kimmel. “Disentangling latent representations of single cell RNA-seq experiments.” In: (Mar. 2020). DOI: 10.1101/2020.03.04.972166. URL: http://dx.doi.org/10.1101/2020.03.04.972166 (cit. on p. 46).

[552] E. Prince and T. C. Hankinson. “HD Spot: Interpretable Deep Learning Classification of Single Cell Transcript Data.” In: (Oct. 2019). DOI: 10.1101/822759. URL: http://dx.doi.org/10.1101/822759 (cit. on p. 46).

[553] S. Rybakov, M. Lotfollahi, F. J. Theis, and F. A. Wolf. “Learning interpretable latent autoencoder representations with annotations of feature sets.” In: (Dec. 2020). DOI: 10.1101/2020.12.02.401182. URL: http://dx.doi.org/10.1101/2020.12.02.401182 (cit. on p. 46).

[554] S. Lukassen, F. W. Ten, L. Adam, R. Eils, and C. Conrad. “Gene set inference from single-cell sequencing data using a hybrid of matrix factorization and variational autoencoders.” In: *Nature Machine Intelligence* 2.12 (Dec. 2020), pp. 800–809. ISSN: 2522-5839. DOI: 10.1038/s42256-020-00269-9. URL: http://dx.doi.org/10.1038/s42256-020-00269-9 (cit. on p. 46).

[555] A. K. Mondal, H. Asnani, P. Singla, and P. AP. “scRAE: Deterministic Regularized Autoencoders With Flexible Priors for Clustering Single-Cell Gene Expression Data.” In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19.5 (Sept. 2022), pp. 2996–3007. ISSN: 2374-0043. DOI: 10.1109/tcbb.2021.3098394. URL: http://dx.doi.org/10.1109/tcbb.2021.3098394 (cit. on p. 46).

[556] B. Yu, C. Chen, R. Qi, R. Zheng, P. J. Skillman-Lawrence, X. Wang, A. Ma, and H. Gu. “scGMAI: a Gaussian mixture model for clustering single-cell RNA-Seq data based on deep autoencoder.” In: *Briefings in Bioinformatics* (Dec. 2020). ISSN: 1477-4054. DOI: 10.1093/bib/bbaa316. URL: http://dx.doi.org/10.1093/bib/bbaa316 (cit. on p. 46).

[557] D. Buterez, I. Bica, I. Tariq, H. Andrés-Terré, and P. Liò. “CellVGAE: an unsupervised scRNA-seq analysis workflow with graph attention networks.” In: *Bioinformatics* 38.5 (Dec. 2021). Ed. by V. Boeva, pp. 1277–1286. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btab804. URL: http://dx.doi.org/10.1093/bioinformatics/btab804 (cit. on p. 46).

[558] H. Cho, B. Berger, and J. Peng. “Generalizable and Scalable Visualization of Single-Cell Data Using Neural Networks.” In: *Cell Systems* 7.2 (Aug. 2018), 185–191.e4. ISSN: 2405-4712. DOI: 10.1016/j.cels.2018.05.017. URL: http://dx.doi.org/10.1016/j.cels.2018.05.017 (cit. on p. 46).

[559] D. Wang and J. Gu. “VASC: Dimension Reduction and Visualization of Single-cell RNA-seq Data by Deep Variational Autoencoder.” In: *Genomics, Proteomics & Bioinformatics* 16.5 (Oct. 2018), pp. 320–331. ISSN: 1672-0229. DOI: 10.1016/j.gpb.2018.08.003. URL: http://dx.doi.org/10.1016/j.gpb.2018.08.003 (cit. on p. 46).

[560] J. Li, W. Jiang, H. Han, J. Liu, B. Liu, and Y. Wang. “ScGSLC: An unsupervised graph similarity learning framework for single-cell RNA-seq data clustering.” In: *Computational Biology and Chemistry* 90 (Feb. 2021), p. 107415. ISSN: 1476-9271. DOI: 10.1016/j.compbiolchem.2020.107415. URL: http://dx.doi.org/10.1016/j.compbiolchem.2020.107415 (cit. on p. 46).

[561] I. Bica, H. Andrés-Terré, A. Cvejic, and P. Liò. “Unsupervised generative and graph representation learning for modelling cell differentiation.” In: *Scientific Reports* 10.1 (June 2020). ISSN: 2045-2322. DOI: 10.1038/s41598-020-66166-8. URL: http://dx.doi.org/10.1038/s41598-020-66166-8 (cit. on p. 46).

[562] S. Zhang, X. Li, Q. Lin, J. Lin, and K.-C. Wong. “Uncovering the key dimensions of high-throughput biomolecular data using deep learning.” In: *Nucleic Acids Research* 48.10 (Mar. 2020), e56–e56. ISSN: 1362-4962. DOI: 10.1093/nar/gkaa191. URL: http://dx.doi.org/10.1093/nar/gkaa191 (cit. on p. 46).

[563] G. Gut, S. G. Stark, G. Rätsch, and N. R. Davidson. “pmVAE: Learning Interpretable Single-Cell Representations with Pathway Modules.” In: (Jan. 2021). DOI: 10.1101/2021.01.28.428664. URL: http://dx.doi.org/10.1101/2021.01.28.428664 (cit. on p. 46).

[564] V. Svensson, A. Gayoso, N. Yosef, and L. Pachter. “Interpretable factor models of single-cell RNA-seq via variational autoencoders.” In: *Bioinformatics* 36.11 (Mar. 2020). Ed. by A. Mathelier, pp. 3418–3421. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btaa169. URL: http://dx.doi.org/10.1093/bioinformatics/btaa169 (cit. on p. 46).

[565] J. Zhao, N. Wang, H. Wang, C. Zheng, and Y. Su. “SCDRHA: A scRNA-Seq Data Dimensionality Reduction Algorithm Based on Hierarchical Autoencoder.” In: *Frontiers in Genetics* 12 (Aug. 2021). ISSN: 1664-8021. DOI: 10.3389/fgene.2021.733906. URL: http://dx.doi.org/10.3389/fgene.2021.733906 (cit. on p. 46).

[566] H. Wang, J. Zhao, Y. Su, and C.-H. Zheng. “scCDG: A Method based on DAE and GCN for scRNA-seq data Analysis.” In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2021), pp. 1–1. ISSN: 2374-0043. DOI: 10.1109/tcbb.2021.3126641. URL: http://dx.doi.org/10.1109/TCBB.2021.3126641 (cit. on p. 46).

[567] M. Ciortan and M. Defrance. “GNN-based embedding for clustering scRNA-seq data.” In: *Bioinformatics* 38.4 (Nov. 2021). Ed. by V. Boeva, pp. 1037–1044. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btab787. URL: http://dx.doi.org/10.1093/bioinformatics/btab787 (cit. on p. 46).

[568] M. Ciortan and M. Defrance. “Contrastive self-supervised clustering of scRNA-seq data.” In: *BMC Bioinformatics* 22.1 (May 2021). ISSN: 1471-2105. DOI: 10.1186/s12859-021-04210-8. URL: http://dx.doi.org/10.1186/s12859-021-04210-8 (cit. on p. 46).

[569] N. Fortelny and C. Bock. “Knowledge-primed neural networks enable biologically interpretable deep learning on single-cell sequencing data.” In: *Genome Biology* 21.1 (Aug. 2020). ISSN: 1474-760X. DOI: 10.1186/s13059-020-02100-5. URL: http://dx.doi.org/10.1186/s13059-020-02100-5 (cit. on p. 46).

[570] M. Bahrami, M. Maitra, C. Nagy, G. Turecki, H. R. Rabiee, and Y. Li. “Deep feature extraction of single-cell transcriptomes by generative adversarial network.” In: *Bioinformatics* 37.10 (Dec. 2020). Ed. by I. Birol, pp. 1345–1351. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btaa976. URL: http://dx.doi.org/10.1093/bioinformatics/btaa976 (cit. on p. 46).

[571] Y. Choi, R. Li, and G. Quon. “siVAE: interpretable deep generative models for single-cell transcriptomes.” In: *Genome Biology* 24.1 (Feb. 2023). ISSN: 1474-760X. DOI: 10.1186/s13059-023-02850-y. URL: http://dx.doi.org/10.1186/s13059-023-02850-y (cit. on p. 46).

[572] F. W. Ten, D. Yuan, N. Jabareen, Y. J. Phua, R. Eils, S. Lukassen, and C. Conrad. “resVAE ensemble: Unsupervised identification of gene sets in multi-modal single-cell sequencing data using deep ensembles.” In: *Frontiers in Cell and Developmental Biology* 11 (Feb. 2023). ISSN: 2296-634X. DOI: 10.3389/fcell.2023.1091047. URL: http://dx.doi.org/10.3389/fcell.2023.1091047 (cit. on p. 46).

[573] X. Zhang, X. Wang, G. V. Shivashankar, and C. Uhler. “Graph-based autoencoder integrates spatial transcriptomics with chromatin images and identifies joint biomarkers for Alzheimer’s disease.” In: *Nature Communications* 13.1 (Dec. 2022). DOI: 10.1038/s41467-022-35233-1. URL: https://doi.org/10.1038/s41467-022-35233-1 (cit. on p. 46).

[574] C. Yan, M. Li, Z. Suo, J. Zhang, J. Wang, G. Zhang, W. Liang, and H. Luo. “Biomarkers Identification of Hepatocellular Carcinoma Based on Multiomics Data Integration and Graph-embedded Deep Neural Network.” In: *Current Bioinformatics* 18.6 (July 2023), pp. 459–471. DOI: 10.2174/1574893618666230227122331. URL: https://doi.org/10.2174/1574893618666230227122331 (cit. on p. 46).

[575] S. Tasaki, C. Gaiteri, S. Mostafavi, and Y. Wang. “Deep learning decodes the principles of differential gene expression.” In: *Nature Machine Intelligence* 2.7 (July 2020), pp. 376–386. ISSN: 2522-5839. DOI: 10.1038/s42256-020-0201-6. URL: http://dx.doi.org/10.1038/s42256-020-0201-6 (cit. on p. 46).

[576] K. D. Yang, A. Belyaeva, S. Venkatachalapathy, K. Damodaran, A. Katcoff, A. Radhakrishnan, G. V. Shivashankar, and C. Uhler. “Multi-domain translation between single-cell imaging and sequencing data using autoencoders.” In: *Nature Communications* 12.1 (Jan. 2021). DOI: 10.1038/s41467-020-20249-2. URL: https://doi.org/10.1038/s41467-020-20249-2 (cit. on p. 46).

[577] Z.-J. Cao and G. Gao. “Multi-omics single-cell data integration and regulatory inference with graph-linked embedding.” In: *Nature Biotechnology* 40.10 (May 2022), pp. 1458–1466. ISSN: 1546-1696. DOI: 10.1038/s41587-022-01284-4. URL: http://dx.doi.org/10.1038/s41587-022-01284-4 (cit. on p. 46).

[578] T. Ashuach, M. I. Gabitto, R. V. Koodli, G.-A. Saldi, M. I. Jordan, and N. Yosef. “MultiVI: deep generative model for the integration of multimodal data.” In: *Nature Methods* 20.8 (June 2023), pp. 1222–1231. DOI: 10.1038/s41592-023-01909-9. URL: https://doi.org/10.1038/s41592-023-01909-9 (cit. on p. 46).

[579] T. Ashuach, D. A. Reidenbach, A. Gayoso, and N. Yosef. “PeakVI: A deep generative model for single-cell chromatin accessibility analysis.” In: *Cell Reports Methods* 2.3 (Mar. 2022), p. 100182. DOI: 10.1016/j.crmeth.2022.100182. URL: https://doi.org/10.1016/j.crmeth.2022.100182 (cit. on p. 46).

[580] R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef. “Deep generative modeling for single-cell transcriptomics.” In: *Nature Methods* 15.12 (Nov. 2018), pp. 1053–1058. DOI: 10.1038/s41592-018-0229-2. URL: https://doi.org/10.1038/s41592-018-0229-2 (cit. on p. 46).

[581] A. Gayoso, Z. Steier, R. Lopez, J. Regier, K. L. Nazor, A. Streets, and N. Yosef. "Joint probabilistic modeling of single-cell multi-omic data with totalVI." In: *Nature Methods* 18.3 (Feb. 2021), pp. 272–282. DOI: 10.1038/s41592-020-01050-x. URL: https://doi.org/10.1038/s41592-020-01050-x (cit. on p. 46).

[582] Y. Cheng, X. Ma, L. Yuan, Z. Sun, and P. Wang. "Evaluating imputation methods for single-cell RNA-seq data." In: *BMC Bioinformatics* 24.1 (July 2023). DOI: 10.1186/s12859-023-05417-7. URL: https://doi.org/10.1186/s12859-023-05417-7 (cit. on p. 46).

[583] X. Zhang and Y. Guo. "OmiTrans: Generative Adversarial Networks Based Omics-to-omics Translation Framework." In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Dec. 2022. DOI: 10.1109/bibm55620.2022.9995537. URL: https://doi.org/10.1109/bibm55620.2022.9995537 (cit. on p. 46).

[584] A. Ma et al. "Single-cell biological network inference using a heterogeneous graph transformer." In: *Nature Communications* 14.1 (Feb. 2023). DOI: 10.1038/s41467-023-36559-0. URL: https://doi.org/10.1038/s41467-023-36559-0 (cit. on p. 46).

[585] Y. Xu, E. Begoli, and R. P. McCord. "sciCAN: single-cell chromatin accessibility and gene expression data integration via cycle-consistent adversarial network." In: *npj Systems Biology and Applications* 8.1 (Sept. 2022). DOI: 10.1038/s41540-022-00245-6. URL: https://doi.org/10.1038/s41540-022-00245-6 (cit. on p. 46).

[586] C. Zuo and L. Chen. "Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data." In: *Briefings in Bioinformatics* 22.4 (Nov. 2020). DOI: 10.1093/bib/bbaa287. URL: https://doi.org/10.1093/bib/bbaa287 (cit. on p. 46).

[587] C. Zuo, H. Dai, and L. Chen. "Deep cross-omics cycle attention model for joint analysis of single-cell multi-omics data." In: *Bioinformatics* 37.22 (May 2021). Ed. by A. Mathelier, pp. 4091–4099. DOI: 10.1093/bioinformatics/btab403. URL: https://doi.org/10.1093/bioinformatics/btab403 (cit. on p. 46).

[588] K. E. Wu, K. E. Yost, H. Y. Chang, and J. Zou. "BABEL enables cross-modality translation between multiomic profiles at single-cell resolution." In: *Proceedings of the National Academy of Sciences* 118.15 (Apr. 2021). DOI: 10.1073/pnas.2023070118. URL: https://doi.org/10.1073/pnas.2023070118 (cit. on p. 46).

[589] M. Lotfollahi et al. "Mapping single-cell data to reference atlases by transfer learning." In: *Nature Biotechnology* 40.1 (Aug. 2021), pp. 121–130. DOI: 10.1038/s41587-021-01001-7. URL: https://doi.org/10.1038/s41587-021-01001-7 (cit. on p. 46).

[590] F. Maseda, Z. Cang, and Q. Nie. "DEEPsc: A Deep Learning-Based Map Connecting Single-Cell Transcriptomics and Spatial Imaging Data." In: *Frontiers in Genetics* 12 (Mar. 2021). DOI: 10.3389/fgene.2021.636743. URL: https://doi.org/10.3389/fgene.2021.636743 (cit. on p. 46).

[591] Y. Xu, P. Das, and R. P. McCord. "SMILE: mutual information learning for integration of single-cell omics data." In: *Bioinformatics* 38.2 (Oct. 2021). Ed. by J. Xu, pp. 476–486. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btab706. URL: http://dx.doi.org/10.1093/bioinformatics/btab706 (cit. on p. 46).

[592] K. T. Ahmed, J. Sun, S. Cheng, J. Yong, and W. Zhang. "Multi-omics data integration by generative adversarial network." In: *Bioinformatics* 38.1 (Aug. 2021). Ed. by P. Robinson, pp. 179–186. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btab608. URL: http://dx.doi.org/10.1093/bioinformatics/btab608 (cit. on p. 46).

[593] T. Wang, W. Shao, Z. Huang, H. Tang, J. Zhang, Z. Ding, and K. Huang. "MOGONET integrates multi-omics data using graph convolutional networks allowing patient classification and biomarker identification." In: *Nature Communications* 12.1 (June 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-23774-w. URL: http://dx.doi.org/10.1038/s41467-021-23774-w (cit. on p. 46).

[594] Y. Zhong, Y. Peng, Y. Lin, D. Chen, H. Zhang, W. Zheng, Y. Chen, and C. Wu. "MODILM: towards better complex diseases classification using a novel multi-omics data integration learning model." In: *BMC Medical Informatics and Decision Making* 23.1 (May 2023). ISSN: 1472-6947. DOI: 10.1186/s12911-023-02173-9. URL: http://dx.doi.org/10.1186/s12911-023-02173-9 (cit. on pp. 46, 48).

[595] J. M. Choi and H. Chae. "moBRCA-net: a breast cancer subtype classification framework based on multi-omics attention neural networks." In: *BMC Bioinformatics* 24.1 (Apr. 2023). ISSN: 1471-2105. DOI: 10.1186/s12859-023-05273-5. URL: http://dx.doi.org/10.1186/s12859-023-05273-5 (cit. on pp. 46, 48).

[596] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija. "Comprehensive Integration of Single-Cell Data." In: *Cell* 177.7 (June 2019), 1888–1902.e21. DOI: 10.1016/j.cell.2019.05.031. URL: https://doi.org/10.1016/j.cell.2019.05.031 (cit. on p. 46).

[597] W. Kopp, A. Akalin, and U. Ohler. “Simultaneous dimensionality reduction and integration for single-cell ATAC-seq data using deep learning.” In: *Nature Machine Intelligence* 4.2 (Feb. 2022), pp. 162–168. DOI: 10.1038/s42256-022-00443-1. URL: https://doi.org/10.1038/s42256-022-00443-1 (cit. on p. 46).

[598] Q. Liu, S. Chen, R. Jiang, and W. H. Wong. “Simultaneous deep generative modelling and clustering of single-cell genomic data.” In: *Nature Machine Intelligence* 3.6 (May 2021), pp. 536–544. DOI: 10.1038/s42256-021-00333-y. URL: https://doi.org/10.1038/s42256-021-00333-y (cit. on p. 46).

[599] U. Shaham, K. P. Stanton, J. Zhao, H. Li, K. Raddassi, R. Montgomery, and Y. Kluger. “Removal of batch effects using distribution-matching residual networks.” In: *Bioinformatics* 33.16 (Apr. 2017). Ed. by J. Wren, pp. 2539–2546. DOI: 10.1093/bioinformatics/btx196. URL: https://doi.org/10.1093/bioinformatics/btx196 (cit. on p. 46).

[600] Y. Wang, T. Liu, and H. Zhao. “ResPAN: a powerful batch correction model for scRNA-seq data through residual adversarial networks.” In: *Bioinformatics* 38.16 (June 2022). Ed. by V. Boeva, pp. 3942–3949. DOI: 10.1093/bioinformatics/btac427. URL: https://doi.org/10.1093/bioinformatics/btac427 (cit. on p. 46).

[601] H. T. N. Tran, K. S. Ang, M. Chevrier, X. Zhang, N. Y. S. Lee, M. Goh, and J. Chen. “A benchmark of batch-effect correction methods for single-cell RNA sequencing data.” In: *Genome Biology* 21.1 (Jan. 2020). DOI: 10.1186/s13059-019-1850-9. URL: https://doi.org/10.1186/s13059-019-1850-9 (cit. on p. 46).

[602] B. Zou, T. Zhang, R. Zhou, X. Jiang, H. Yang, X. Jin, and Y. Bai. “deepMNN: Deep Learning-Based Single-Cell RNA Sequencing Data Batch Correction Using Mutual Nearest Neighbors.” In: *Frontiers in Genetics* 12 (Aug. 2021). DOI: 10.3389/fgene.2021.708981. URL: https://doi.org/10.3389/fgene.2021.708981 (cit. on p. 47).

[603] J. Hu, Y. Zhong, and X. Shang. “Efficient and scalable integration of single-cell data using domain-adversarial and variational approximation.” In: (Apr. 2021). DOI: 10.1101/2021.04.06.438733. URL: http://dx.doi.org/10.1101/2021.04.06.438733 (cit. on p. 47).

[604] L. Xiong, K. Tian, Y. Li, W. Ning, X. Gao, and Q. C. Zhang. “Online single-cell data integration through projecting heterogeneous datasets into a common cell-embedding space.” In: *Nature Communications* 13.1 (Oct. 2022). ISSN: 2041-1723. DOI: 10.1038/s41467-022-33758-z. URL: http://dx.doi.org/10.1038/s41467-022-33758-z (cit. on p. 47).

[605] X. Wang, J. Wang, H. Zhang, S. Huang, and Y. Yin. “HDMC: a novel deep learning-based framework for removing batch effects in single-cell RNA-seq data.” In: *Bioinformatics* 38.5 (Dec. 2021). Ed. by V. Boeva, pp. 1295–1303. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btab821. URL: http://dx.doi.org/10.1093/bioinformatics/btab821 (cit. on p. 47).

[606] S. Ge, H. Wang, A. Alavi, E. Xing, and Z. Bar-joseph. “Supervised Adversarial Alignment of Single-Cell RNA-seq Data.” In: *Journal of Computational Biology* 28.5 (May 2021), pp. 501–513. ISSN: 1557-8666. DOI: 10.1089/cmb.2020.0439. URL: http://dx.doi.org/10.1089/cmb.2020.0439 (cit. on p. 47).

[607] Y. Zhao, H. Cai, Z. Zhang, J. Tang, and Y. Li. “Learning interpretable cellular and gene signature embeddings from single-cell transcriptomic data.” In: *Nature Communications* 12.1 (Sept. 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-25534-2. URL: http://dx.doi.org/10.1038/s41467-021-25534-2 (cit. on p. 47).

[608] D. Wang, S. Hou, L. Zhang, X. Wang, B. Liu, and Z. Zhang. “iMAP: integration of multiple single-cell datasets by adversarial paired transfer networks.” In: *Genome Biology* 22.1 (Feb. 2021). ISSN: 1474-760X. DOI: 10.1186/s13059-021-02280-8. URL: http://dx.doi.org/10.1186/s13059-021-02280-8 (cit. on p. 47).

[609] T. Wang, T. S. Johnson, W. Shao, Z. Lu, B. R. Helm, J. Zhang, and K. Huang. “BERMUDA: a novel deep transfer learning method for single-cell RNA sequencing batch correction reveals hidden high-resolution cellular subtypes.” In: *Genome Biology* 20.1 (Aug. 2019). ISSN: 1474-760X. DOI: 10.1186/s13059-019-1764-6. URL: http://dx.doi.org/10.1186/s13059-019-1764-6 (cit. on p. 47).

[610] W. Yu, A. Mahfouz, and M. J. T. Reinders. “CBA: Cluster-Guided Batch Alignment for Single Cell RNA-seq.” In: *Frontiers in Genetics* 12 (Apr. 2021). ISSN: 1664-8021. DOI: 10.3389/fgene.2021.644211. URL: http://dx.doi.org/10.3389/fgene.2021.644211 (cit. on p. 47).

[611] S. G. Riva, P. Cazzaniga, and A. Tangherloni. “Integration of Multiple scRNA-Seq Datasets on the Autoencoder Latent Space.” In: *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Dec. 2021. DOI: 10.1109/bibm52615.2021.9669807. URL: http://dx.doi.org/10.1109/BIBM52615.2021.9669807 (cit. on p. 47).

[612] D. Gan and J. Li. “SCIBER: a simple method for removing batch effects from single-cell RNA-sequencing data.” In: *Bioinformatics* 39.1 (Dec. 2022). Ed. by V. Boeva. DOI: 10.1093/bioinformatics/btac819. URL: https://doi.org/10.1093/bioinformatics/btac819 (cit. on p. 47).

[613] T. Fei and T. Yu. “scBatch: batch-effect correction of RNA-seq data through sample distance matrix adjustment.” In: *Bioinformatics* 36.10 (Feb. 2020). Ed. by J. Wren, pp. 3115–3123. DOI: 10.1093/bioinformatics/btaa097. URL: https://doi.org/10.1093/bioinformatics/btaa097 (cit. on p. 47).

[614] H. Lahmer, A. E. Oueslati, and Z. Lachiri. “Classification of DNA Microarrays Using Deep Learning to identify Cell Cycle Regulated Genes.” In: *2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. IEEE, Sept. 2020. DOI: 10.1109/atsip49331.2020.9231888. URL: https://doi.org/10.1109/atsip49331.2020.9231888 (cit. on p. 47).

[615] H. Lahmer, A. E. Oueslati, and Z. Lachiri. “DNA Microarray Analysis Using Machine Learning to Recognize Cell Cycle Regulated Genes.” In: *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*. IEEE, July 2019. DOI: 10.1109/iccad46983.2019.9037868. URL: https://doi.org/10.1109/iccad46983.2019.9037868 (cit. on p. 47).

[616] O. H. Purba, E. A. Sarwoko, Khadijah, Suhartono, and A. Wibowo. “Classification of liver cancer with microrna data using the deep neural network (DNN) method.” In: *Journal of Physics: Conference Series* 1524.1 (Apr. 2020), p. 012129. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1524/1/012129. URL: http://dx.doi.org/10.1088/1742-6596/1524/1/012129 (cit. on pp. 47, 49).

[617] B. Schmauch et al. “A deep learning model to predict RNA-Seq expression of tumours from whole slide images.” In: *Nature Communications* 11.1 (Aug. 2020). ISSN: 2041-1723. DOI: 10.1038/s41467-020-17678-4. URL: http://dx.doi.org/10.1038/s41467-020-17678-4 (cit. on p. 47).

[618] A. Levy-Jurgenson, X. Tekpli, V. N. Kristensen, and Z. Yakhini. “Spatial transcriptomics inferred from pathology whole-slide images links tumor heterogeneity to survival in breast and lung cancer.” In: *Scientific Reports* 10.1 (Nov. 2020). ISSN: 2045-2322. DOI: 10.1038/s41598-020-75708-z. URL: http://dx.doi.org/10.1038/s41598-020-75708-z (cit. on p. 47).

[619] A. Alsaafin, A. Safarpoor, M. Sikaroudi, J. D. Hipp, and H. R. Tizhoosh. “Learning to predict RNA sequence expressions from whole slide images with applications for search and classification.” In: *Communications Biology* 6.1 (Mar. 2023). ISSN: 2399-3642. DOI: 10.1038/s42003-023-04583-x. URL: http://dx.doi.org/10.1038/s42003-023-04583-x (cit. on p. 47).

[620] B. Yuan, D. Yang, B. E. G. Rothberg, H. Chang, and T. Xu. “Unsupervised and supervised learning with neural network for human transcriptome analysis and cancer diagnosis.” In: *Scientific Reports* 10.1 (Nov. 2020). ISSN: 2045-2322. DOI: 10.1038/s41598-020-75715-0. URL: http://dx.doi.org/10.1038/s41598-020-75715-0 (cit. on p. 47).

[621] J. Hong, L. D. Hachem, and M. G. Fehlings. “A deep learning model to classify neoplastic state and tissue origin from transcriptomic data.” In: *Scientific Reports* 12.1 (June 2022). ISSN: 2045-2322. DOI: 10.1038/s41598-022-13665-5. URL: http://dx.doi.org/10.1038/s41598-022-13665-5 (cit. on p. 47).

[622] B. Azarkhalili, A. Saberi, H. Chitsaz, and A. Sharifi-Zarchi. “DeePathology: Deep Multi-Task Learning for Inferring Molecular Pathology from Cancer Transcriptome.” In: *Scientific Reports* 9.1 (Nov. 2019). ISSN: 2045-2322. DOI: 10.1038/s41598-019-52937-5. URL: http://dx.doi.org/10.1038/s41598-019-52937-5 (cit. on p. 47).

[623] M. Yap et al. “Verifying explainability of a deep learning tissue classifier trained on RNA-seq data.” In: *Scientific Reports* 11.1 (Jan. 2021). ISSN: 2045-2322. DOI: 10.1038/s41598-021-81773-9. URL: http://dx.doi.org/10.1038/s41598-021-81773-9 (cit. on p. 47).

[624] D. S. Watson. “Interpretable machine learning for genomics.” In: *Human Genetics* 141.9 (Oct. 2021), pp. 1499–1513. ISSN: 1432-1203. DOI: 10.1007/s00439-021-02387-9. URL: http://dx.doi.org/10.1007/s00439-021-02387-9 (cit. on p. 47).

[625] M. Wysocka, O. Wysocki, M. Zufferey, D. Landers, and A. Freitas. “A systematic review of biologically-informed deep learning models for cancer: fundamental trends for encoding and interpreting oncology data.” In: *BMC Bioinformatics* 24.1 (May 2023). ISSN: 1471-2105. DOI: 10.1186/s12859-023-05262-8. URL: http://dx.doi.org/10.1186/s12859-023-05262-8 (cit. on p. 47).

[626] S. MacDonald et al. “Generalising uncertainty improves accuracy and safety of deep learning analytics applied to oncology.” In: *Scientific Reports* 13.1 (May 2023). ISSN: 2045-2322. DOI: 10.1038/s41598-023-31126-5. URL: http://dx.doi.org/10.1038/s41598-023-31126-5 (cit. on p. 47).

[627] C. Luchini, A. Pea, and A. Scarpa. “Artificial intelligence in oncology: current applications and future perspectives.” In: *British Journal of Cancer* 126.1 (Nov. 2021), pp. 4–9. ISSN: 1532-1827. DOI: 10.1038/s41416-021-01633-1. URL: http://dx.doi.org/10.1038/s41416-021-01633-1 (cit. on p. 47).

[628] R. Zhang, G.-B. Huang, N. Sundararajan, and P. Saratchandran. "Multicategory Classification Using An Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis." In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4.3 (July 2007), pp. 485–495. DOI: 10.1109/tcbb.2007.1012. URL: https://doi.org/10.1109/tcbb.2007.1012 (cit. on pp. 47, 176).

[629] M. Shi, X. Li, M. Li, and Y. Si. "Attention-based generative adversarial networks improve prognostic outcome prediction of cancer from multimodal data." In: *Briefings in Bioinformatics* 24.6 (Sept. 2023). DOI: 10.1093/bib/bbad329. URL: https://doi.org/10.1093/bib/bbad329 (cit. on p. 48).

[630] D. Duroux, C. Wohlfart, K. Van Steen, A. Vladimirova, and M. King. "Graph-based multi-modality integration for prediction of cancer subtype and severity." In: *Scientific Reports* 13.1 (Nov. 2023). ISSN: 2045-2322. DOI: 10.1038/s41598-023-46392-6. URL: http://dx.doi.org/10.1038/s41598-023-46392-6 (cit. on p. 48).

[631] F. Yan, L. Jiang, F. Ye, J. Ping, T. Y. Bowley, S. A. Ness, C.-I. Li, D. Marchetti, J. Tang, and Y. Guo. "Deep neural network based tissue deconvolution of circulating tumor cell RNA." In: *Journal of Translational Medicine* 21.1 (Nov. 2023). ISSN: 1479-5876. DOI: 10.1186/s12967-023-04663-w. URL: http://dx.doi.org/10.1186/s12967-023-04663-w (cit. on p. 48).

[632] F. Gao, W. Wang, M. Tan, L. Zhu, Y. Zhang, E. Fessler, L. Vermeulen, and X. Wang. "DeepCC: a novel deep learning-based framework for cancer molecular subtype classification." In: *Oncogenesis* 8.9 (Aug. 2019). ISSN: 2157-9024. DOI: 10.1038/s41389-019-0157-8. URL: http://dx.doi.org/10.1038/s41389-019-0157-8 (cit. on p. 48).

[633] H. A. Elmarakeby et al. "Biologically informed deep neural network for prostate cancer discovery." In: *Nature* 598.7880 (Sept. 2021), pp. 348–352. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03922-4. URL: http://dx.doi.org/10.1038/s41586-021-03922-4 (cit. on p. 48).

[634] W. Jiao et al. "A deep learning system accurately classifies primary and metastatic cancers using passenger mutation patterns." In: *Nature Communications* 11.1 (Feb. 2020). ISSN: 2041-1723. DOI: 10.1038/s41467-019-13825-8. URL: http://dx.doi.org/10.1038/s41467-019-13825-8 (cit. on p. 48).

[635] R. Mahdi-Esferizi, B. Haji Molla Hoseyni, A. Mehrpanah, Y. Golzade, A. Najafi, F. Elahian, A. Zadeh Shirazi, G. A. Gomez, and S. Tahmasebian. "DeeP4med: deep learning for P4 medicine to predict normal and cancer transcriptome in multiple human tissues." In: *BMC Bioinformatics* 24.1 (July 2023). ISSN: 1471-2105. DOI: 10.1186/s12859-023-05400-2. URL: http://dx.doi.org/10.1186/s12859-023-05400-2 (cit. on p. 48).

[636] R. Lupat, R. Perera, S. Loi, and J. Li. "Moanna: Multi-Omics Autoencoder-Based Neural Network Algorithm for Predicting Breast Cancer Subtypes." In: *IEEE Access* 11 (2023), pp. 10912–10924. ISSN: 2169-3536. DOI: 10.1109/access.2023.3240515. URL: http://dx.doi.org/10.1109/ACCESS.2023.3240515 (cit. on p. 48).

[637] R. Qi, C.-H. Zheng, C.-M. Ji, N. Yu, J.-C. Ni, and Y.-T. Wang. "Cell Classification Based on Stacked Autoencoder for Single-Cell RNA Sequencing." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2022, pp. 245–259. ISBN: 9783031138294. DOI: 10.1007/978-3-031-13829-4_20. URL: http://dx.doi.org/10.1007/978-3-031-13829-4_20 (cit. on p. 48).

[638] C. Guttà, C. Morhard, and M. Rehm. "Applying a GAN-based classifier to improve transcriptome-based prognostication in breast cancer." In: *PLOS Computational Biology* 19.4 (Apr. 2023). Ed. by Z. Zhang, e1011035. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1011035. URL: http://dx.doi.org/10.1371/journal.pcbi.1011035 (cit. on p. 48).

[639] A. K. Dwivedi. "Artificial neural network model for effective cancer classification using microarray gene expression data." In: *Neural Computing and Applications* 29.12 (Nov. 2016), pp. 1545–1554. ISSN: 1433-3058. DOI: 10.1007/s00521-016-2701-1. URL: http://dx.doi.org/10.1007/s00521-016-2701-1 (cit. on p. 48).

[640] U. Ravindran and C. Gunavathi. "A survey on gene expression data analysis using deep learning methods for cancer diagnosis." In: *Progress in Biophysics and Molecular Biology* 177 (Jan. 2023), pp. 1–13. DOI: 10.1016/j.pbiomolbio.2022.08.004. URL: https://doi.org/10.1016/j.pbiomolbio.2022.08.004 (cit. on p. 48).

[641] F. Azuaje. "Artificial intelligence for precision oncology: beyond patient stratification." In: *npj Precision Oncology* 3.1 (Feb. 2019). ISSN: 2397-768X. DOI: 10.1038/s41698-019-0078-1. URL: http://dx.doi.org/10.1038/s41698-019-0078-1 (cit. on p. 48).

[642] I. El Naqa, A. Karolak, Y. Luo, L. Folio, A. A. Tarhini, D. Rollison, and K. Parodi. "Translation of AI into oncology clinical practice." In: *Oncogene* 42.42 (Sept. 2023), pp. 3089–3097. ISSN: 1476-5594. DOI: 10.1038/s41388-023-02826-z. URL: http://dx.doi.org/10.1038/s41388-023-02826-z (cit. on p. 48).

[643] E. Capobianco. "High-dimensional role of AI and machine learning in cancer research." In: *British Journal of Cancer* 126.4 (Jan. 2022), pp. 523–532. ISSN: 1532-1827. DOI: 10.1038/s41416-021-01689-z. URL: http://dx.doi.org/10.1038/s41416-021-01689-z (cit. on p. 48).

[644] K. Badal, C. M. Lee, and L. J. Esserman. "Guiding principles for the responsible development of artificial intelligence tools for healthcare." In: *Communications Medicine* 3.1 (Apr. 2023). ISSN: 2730-664X. DOI: 10.1038/s43856-023-00279-9. URL: http://dx.doi.org/10.1038/s43856-023-00279-9 (cit. on p. 48).

[645] J. Guo, J. Hu, Y. Zheng, S. Zhao, and J. Ma. "Artificial intelligence: opportunities and challenges in the clinical applications of triple-negative breast cancer." In: *British Journal of Cancer* 128.12 (Mar. 2023), pp. 2141–2149. ISSN: 1532-1827. DOI: 10.1038/s41416-023-02215-z. URL: http://dx.doi.org/10.1038/s41416-023-02215-z (cit. on p. 48).

[646] A. Danilevsky and N. Shomron. "Deep Learning Applied on Next Generation Sequencing Data Analysis." In: *Methods in Molecular Biology*. Springer US, 2021, pp. 169–182. DOI: 10.1007/978-1-0716-1103-6_9. URL: https://doi.org/10.1007/978-1-0716-1103-6_9 (cit. on p. 48).

[647] G. Pei, R. Hu, Y. Dai, Z. Zhao, and P. Jia. "Decoding whole-genome mutational signatures in 37 human pan-cancers by denoising sparse autoencoder neural network." In: *Oncogene* 39.27 (June 2020), pp. 5031–5041. ISSN: 1476-5594. DOI: 10.1038/s41388-020-1343-z. URL: http://dx.doi.org/10.1038/s41388-020-1343-z (cit. on p. 48).

[648] S. Hayou, A. Doucet, and J. Rousseau. "On the Impact of the Activation function on Deep Neural Networks Training." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2672–2680. URL: https://proceedings.mlr.press/v97/hayou19a.html (cit. on pp. 49, 117).

[649] A. Nader and D. Azar. "Evolution of Activation Functions: An Empirical Investigation." In: *ACM Transactions on Evolutionary Learning and Optimization* 1.2 (June 2021), pp. 1–36. DOI: 10.1145/3464384. URL: https://doi.org/10.1145/3464384 (cit. on p. 49).

[650] G. Bingham and R. Miikkulainen. "Discovering Parametric Activation Functions." In: *Neural Networks* 148 (Apr. 2022), pp. 48–65. DOI: 10.1016/j.neunet.2022.01.001. URL: https://doi.org/10.1016/j.neunet.2022.01.001 (cit. on p. 49).

[651] M. Basirat and P. M. Roth. *The Quest for the Golden Activation Function*. 2018. DOI: 10.48550/ARXIV.1808.00783. URL: https://arxiv.org/abs/1808.00783 (cit. on pp. 49, 73, 89).

[652] Basirat, Mina, Jammer, Alexandra, and Roth, Peter M. "The Quest for the Golden Activation Function." In: *Proceedings of ARW & OAGM Workshop 2019*. Verlag der Technischen Universität Graz, 2019. DOI: 10.3217/978-3-85125-663-5-41. URL: https://openlib.tugraz.at/download.php?id=5d09dba127371&location=medra (cit. on p. 49).

[653] H. A. Mayer and R. Schwaiger. "Differentiation of neuron types by evolving activation function templates for artificial neural networks." In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. IEEE, 2002. DOI: 10.1109/ijcnn.2002.1007787. URL: https://doi.org/10.1109/ijcnn.2002.1007787 (cit. on pp. 49, 162).

[654] A. Hagg, M. Mensing, and A. Asteroth. "Evolving parsimonious networks by mixing activation functions." In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, July 2017. DOI: 10.1145/3071178.3071275. URL: https://doi.org/10.1145/3071178.3071275 (cit. on p. 49).

[655] K. Knezevic, J. Fulir, D. Jakobovic, S. Picek, and M. Durasevic. "NeuroSCA: Evolving Activation Functions for Side-Channel Analysis." In: *IEEE Access* 11 (2023), pp. 284–299. DOI: 10.1109/access.2022.3232064. URL: https://doi.org/10.1109/access.2022.3232064 (cit. on p. 49).

[656] Y. Liu and X. Yao. "Evolutionary design of artificial neural networks with different nodes." In: *Proceedings of IEEE International Conference on Evolutionary Computation*. IEEE, 1996. DOI: 10.1109/icec.1996.542681. URL: https://doi.org/10.1109/icec.1996.542681 (cit. on p. 49).

[657] P. Cui and K. C. Wiese. "EvoDNN - Evolving Weights, Biases, and Activation Functions in a Deep Neural Network." In: *2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, Aug. 2022. DOI: 10.1109/cibcb55180.2022.9863054. URL: https://doi.org/10.1109/cibcb55180.2022.9863054 (cit. on p. 49).

[658] P. Cui, B. Shabash, and K. C. Wiese. "EvoDNN - An Evolutionary Deep Neural Network with Heterogeneous Activation Functions." In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, June 2019. DOI: 10.1109/cec.2019.8789964. URL: https://doi.org/10.1109/cec.2019.8789964 (cit. on p. 49).

[659] K. Vijayaprabakaran and K. Sathiyamurthy. "Towards activation function search for long short-term model network: A differential evolution based approach." In: *Journal of King Saud University - Computer and Information Sciences* 34.6 (June 2022), pp. 2637–2650. DOI: 10.1016/j.jksuci.2020.04.015. URL: https://doi.org/10.1016/j.jksuci.2020.04.015 (cit. on pp. 49, 98, 201).

[660] D. O'Neill, B. Xue, and M. Zhang. "Co-evolution of Novel Tree-Like ANNs and Activation Functions: An Observational Study." In: *AI 2018: Advances in Artificial Intelligence*. Springer International Publishing, 2018, pp. 616–629. DOI: 10.1007/978-3-030-03991-2_56. URL: https://doi.org/10.1007/978-3-030-03991-2_56 (cit. on p. 49).

[661] M. Sipper. "Neural Networks with À La Carte Selection of Activation Functions." In: *SN Computer Science* 2.6 (Sept. 2021). ISSN: 2661-8907. DOI: 10.1007/s42979-021-00885-1. URL: http://dx.doi.org/10.1007/s42979-021-00885-1 (cit. on pp. 49, 75).

[662] M. Salimi, M. Loni, and M. Sirjani. "Learning Activation Functions for Adversarial Attack Resilience in CNNs." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2023, pp. 203–214. ISBN: 9783031425059. DOI: 10.1007/978-3-031-42505-9_18. URL: http://dx.doi.org/10.1007/978-3-031-42505-9_18 (cit. on p. 49).

[663] M. Salimi, M. Loni, M. Sirjani, A. Cicchetti, and S. Abbaspour Asadollah. "SARAF: Searching for Adversarial Robust Activation Functions." In: *Proceedings of the 2023 6th International Conference on Machine Vision and Applications*. ICMVA 2023. ACM, Mar. 2023. DOI: 10.1145/3589572.3589598. URL: http://dx.doi.org/10.1145/3589572.3589598 (cit. on p. 49).

[664] Y. Li, T. Geng, S. Stein, A. Li, and H. Yu. "GAAF: Searching Activation Functions for Binary Neural Networks Through Genetic Algorithm." In: *Tsinghua Science and Technology* 28.1 (Feb. 2023), pp. 207–220. ISSN: 1007-0214. DOI: 10.26599/tst.2021.9010084. URL: http://dx.doi.org/10.26599/TST.2021.9010084 (cit. on p. 49).

[665] J. Chen. *Combinatorially Generated Piecewise Activation Functions*. 2016. DOI: 10.48550/ARXIV.1605.05216. URL: https://arxiv.org/abs/1605.05216 (cit. on p. 49).

[666] K. Vijayaprabakaran and K. Sathiyamurthy. "Neuroevolution based hierarchical activation function for long short-term model network." In: *Journal of Ambient Intelligence and Humanized Computing* 12.12 (Jan. 2021), pp. 10757–10768. ISSN: 1868-5145. DOI: 10.1007/s12652-020-02889-w. URL: http://dx.doi.org/10.1007/s12652-020-02889-w (cit. on pp. 49, 101).

[667] Y. Pan, Y. Wang, P. Zhou, Y. Yan, and D. Guo. "Activation functions selection for BP neural network model of ground surface roughness." In: *Journal of Intelligent Manufacturing* 31.8 (Jan. 2020), pp. 1825–1836. DOI: 10.1007/s10845-020-01538-5. URL: https://doi.org/10.1007/s10845-020-01538-5 (cit. on pp. 49, 99, 100).

[668] P. Ramachandran, B. Zoph, and Q. V. Le. *Searching for Activation Functions*. 2017. DOI: 10.48550/ARXIV.1710.05941. URL: https://arxiv.org/abs/1710.05941 (cit. on pp. 49, 58, 129, 203, 279).

[669] A. Marchisio, M. A. Hanif, S. Rehman, M. Martina, and M. Shafique. *A Methodology for Automatic Selection of Activation Functions to Design Hybrid Deep Neural Networks*. 2018. DOI: 10.48550/ARXIV.1811.03980. URL: https://arxiv.org/abs/1811.03980 (cit. on p. 49).

[670] R. P. Tripathi, M. Tiwari, A. Dhawan, A. Sharma, and S. K. Jha. "A Survey on Efficient Realization of Activation Functions of Artificial Neural Network." In: *2021 Asian Conference on Innovation in Technology (ASIANCON)*. IEEE, Aug. 2021. DOI: 10.1109/asiancon51346.2021.9544754. URL: https://doi.org/10.1109/asiancon51346.2021.9544754 (cit. on p. 49).

[671] S. Bouguezzi, H. Faiedh, and C. Souani. "Hardware Implementation of Tanh Exponential Activation Function using FPGA." In: *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*. IEEE, Mar. 2021. DOI: 10.1109/ssd52085.2021.9429506. URL: https://doi.org/10.1109/ssd52085.2021.9429506 (cit. on p. 49).

[672] L. Li, S. Zhang, and J. Wu. "An Efficient Hardware Architecture for Activation Function in Deep Learning Processor." In: *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, June 2018. DOI: 10.1109/icivc.2018.8492754. URL: https://doi.org/10.1109/icivc.2018.8492754 (cit. on p. 49).

[673] C.-H. Tsai, Y.-T. Chih, W. H. Wong, and C.-Y. Lee. "A Hardware-Efficient Sigmoid Function With Adjustable Precision for a Neural Network System." In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62.11 (Nov. 2015), pp. 1073–1077. DOI: 10.1109/tcsii.2015.2456531. URL: https://doi.org/10.1109/tcsii.2015.2456531 (cit. on p. 49).

[674] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi. "Efficient hardware implementation of the hyperbolic tangent sigmoid function." In: *2009 IEEE International Symposium on Circuits and Systems*. IEEE, May 2009. DOI: 10.1109/iscas.2009.5118213. URL: https://doi.org/10.1109/iscas.2009.5118213 (cit. on p. 49).

[675] R. Pogiri, S. Ari, and K. K. Mahapatra. "Design and FPGA Implementation of the LUT based Sigmoid Function for DNN Applications." In: *2022 IEEE International Symposium on Smart Electronic Systems (iSES)*. IEEE, Dec. 2022. DOI: 10.1109/ises54909.2022.00090. URL: https://doi.org/10.1109/ises54909.2022.00090 (cit. on p. 49).

[676] F. M. Shakiba and M. Zhou. "Novel Analog Implementation of a Hyperbolic Tangent Neuron in Artificial Neural Networks." In: *IEEE Transactions on Industrial Electronics* 68.11 (Nov. 2021), pp. 10856–10867. DOI: 10.1109/tie.2020.3034856. URL: https://doi.org/10.1109/tie.2020.3034856 (cit. on p. 49).

[677] Y. Xie, A. N. J. Raj, Z. Hu, S. Huang, Z. Fan, and M. Joler. "A Twofold Lookup Table Architecture for Efficient Approximation of Activation Functions." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.12 (Dec. 2020), pp. 2540–2550. DOI: 10.1109/tvlsi.2020.3015391. URL: https://doi.org/10.1109/tvlsi.2020.3015391 (cit. on p. 49).

[678] P. Priyanka, G. K. Nisarga, and S. Raghuram. "CMOS Implementations of Rectified Linear Activation Function." In: *VLSI Design and Test*. Springer Singapore, 2019, pp. 121–129. ISBN: 9789811359507. DOI: 10.1007/978-981-13-5950-7_11. URL: http://dx.doi.org/10.1007/978-981-13-5950-7_11 (cit. on p. 49).

[679] S.-Y. Lin and J.-C. Chiang. "Low-area architecture design of multi-mode activation functions with controllable maximum absolute error for neural network applications." In: *Microprocessors and Microsystems* 103 (Nov. 2023), p. 104952. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2023.104952. URL: http://dx.doi.org/10.1016/j.micpro.2023.104952 (cit. on pp. 49, 131).

[680] V. Shatravin, D. Shashev, and S. Shidlovskiy. "Sigmoid Activation Implementation for Neural Networks Hardware Accelerators Based on Reconfigurable Computing Environments for Low-Power Intelligent Systems." In: *Applied Sciences* 12.10 (May 2022), p. 5216. ISSN: 2076-3417. DOI: 10.3390/app12105216. URL: http://dx.doi.org/10.3390/app12105216 (cit. on p. 49).

[681] L. Derczynski. *Power Consumption Variation over Activation Functions*. 2020. DOI: 10.48550/ARXIV.2006.07237. URL: https://arxiv.org/abs/2006.07237 (cit. on p. 49).

[682] A. D. Jagtap and G. E. Karniadakis. *How important are activation functions in regression and classification? A survey, performance comparison, and future directions*. 2022. DOI: 10.48550/ARXIV.2209.02681. URL: https://arxiv.org/abs/2209.02681 (cit. on p. 49).

[683] G. K. Pandey and S. Srivastava. "ResNet-18 comparative analysis of various activation functions for image classification." In: *2023 International Conference on Inventive Computation Technologies (ICICT)*. IEEE, Apr. 2023. DOI: 10.1109/icict57646.2023.10134464. URL: https://doi.org/10.1109/icict57646.2023.10134464 (cit. on pp. 49, 55).

[684] M. M. Noel, S. Bharadwaj, V. Muthiah-Nakarajan, P. Dutta, and G. B. Amali. *Biologically Inspired Oscillating Activation Functions Can Bridge the Performance Gap between Biological and Artificial Neurons*. 2021. DOI: 10.48550/ARXIV.2111.04020. URL: https://arxiv.org/abs/2111.04020 (cit. on pp. 49, 50).

[685] S. Eger, P. Youssef, and I. Gurevych. "Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/d18-1472. URL: https://doi.org/10.18653/v1/d18-1472 (cit. on pp. 49, 54).

[686] L. Nanni, S. Brahnam, M. Paci, and S. Ghidoni. "Comparison of Different Convolutional Neural Network Activation Functions and Methods for Building Ensembles for Small to Midsize Medical Data Sets." In: *Sensors* 22.16 (Aug. 2022), p. 6129. DOI: 10.3390/s22166129. URL: https://doi.org/10.3390/s22166129 (cit. on p. 49).

[687] K. Adem. "Impact of activation functions and number of layers on detection of exudates using circular Hough transform and convolutional neural networks." In: *Expert Systems with Applications* 203 (Oct. 2022), p. 117583. DOI: 10.1016/j.eswa.2022.117583. URL: https://doi.org/10.1016/j.eswa.2022.117583 (cit. on p. 49).

[688] K. Adu, Y. Yu, J. Cai, I. Asare, and J. Quahin. "The influence of the activation function in a capsule network for brain tumor type classification." In: *International Journal of Imaging Systems and Technology* 32.1 (Aug. 2021), pp. 123–143. DOI: 10.1002/ima.22638. URL: https://doi.org/10.1002/ima.22638 (cit. on pp. 49, 136, 203, 289).

[689] Z. A. Haq and Z. A. Jaffery. "Impact of activation functions and number of layers on the classification of fruits using CNN." In: *Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development, INDIACom 2021*. IEEE, 2021, pp. 227–231. DOI: 10.1109/INDIACom51348.2021.00040. URL: https://ieeexplore.ieee.org/document/9441246 (cit. on p. 49).

[690] A. Nguyen, K. Pham, D. Ngo, T. Ngo, and L. Pham. "An Analysis of State-of-the-art Activation Functions For Supervised Deep Neural Network." In: *2021 International Conference on System Science and Engineering (ICSSE)*. IEEE, Aug. 2021. DOI: 10.1109/icsse52999.2021.9538437. URL: https://doi.org/10.1109/icsse52999.2021.9538437 (cit. on p. 49).

[691] A. Mishra, P. Chandra, U. Ghose, and S. S. Sodhi. "Bi-modal derivative adaptive activation function sigmoidal feedforward artificial neural networks." In: *Applied Soft Computing* 61 (Dec. 2017), pp. 983–994. DOI: 10.1016/j.asoc.2017.09.002. URL: https://doi.org/10.1016/j.asoc.2017.09.002 (cit. on pp. 49, 126).

[692] D. E. Ratnawati, Marjono, Widodo, and S. Anam. "Comparison of activation function on extreme learning machine (ELM) performance for classifying the active compound." In: *SYMPOSIUM ON BIOMATHEMATICS 2019 (SYMOMATH 2019)*. AIP Publishing, 2020. DOI: 10.1063/5.0023872. URL: https://doi.org/10.1063/5.0023872 (cit. on pp. 49, 176).

[693] A. Dureja and P. Pahwa. "Analysis of Nonlinear Activation Functions for Classification Tasks Using Convolutional Neural Networks." In: *Lecture Notes in Electrical Engineering*. Springer Singapore, 2019, pp. 1179–1190. DOI: 10.1007/978-981-13-6772-4_103. URL: https://doi.org/10.1007/978-981-13-6772-4_103 (cit. on p. 49).

[694] V. M. Vargas, D. Guijo-Rubio, P. A. Gutiérrez, and C. Hervás-Martínez. "ReLU-Based Activations: Analysis and Experimental Study for Deep Learning." In: *Advances in Artificial Intelligence*. Springer International Publishing, 2021, pp. 33–43. DOI: 10.1007/978-3-030-85713-4_4. URL: https://doi.org/10.1007/978-3-030-85713-4_4 (cit. on p. 49).

[695] Y. Singh, M. Saini, and Savita. "Impact and Performance Analysis of Various Activation Functions for Classification Problems." In: *2023 IEEE International Conference on Contemporary Computing and Communications (InC4)*. IEEE, Apr. 2023. DOI: 10.1109/inc457730.2023.10263129. URL: http://dx.doi.org/10.1109/InC457730.2023.10263129 (cit. on p. 49).

[696] D. K.-H. Lai, E. S.-W. Cheng, B. P.-H. So, Y.-J. Mao, S. M.-Y. Cheung, D. S. K. Cheung, D. W.-C. Wong, and J. C.-W. Cheung. "Transformer Models and Convolutional Networks with Different Activation Functions for Swallow Classification Using Depth Video Data." In: *Mathematics* 11.14 (July 2023), p. 3081. DOI: 10.3390/math11143081. URL: https://doi.org/10.3390/math11143081 (cit. on p. 49).

[697] E. Papavasileiou and B. Jansen. "The importance of the activation function in NeuroEvolution with FS-NEAT and FD-NEAT." In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Nov. 2017. DOI: 10.1109/ssci.2017.8285328. URL: https://doi.org/10.1109/ssci.2017.8285328 (cit. on p. 49).

[698] W. H. Kang, J. Alam, and A. Fathan. "Investigation on activation functions for robust end-to-end spoofing attack detection system." In: *2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge*. ISCA, Sept. 2021. DOI: 10.21437/asvspoof.2021-13. URL: https://doi.org/10.21437/asvspoof.2021-13 (cit. on pp. 49, 111).

[699] V. M. Vargas, P. A. Gutiérrez, J. Barbero-Gómez, and C. Hervás-Martínez. "Activation Functions for Convolutional Neural Networks: Proposals and Experimental Study." In: *IEEE Transactions on Neural Networks and Learning Systems* 34.3 (Mar. 2023), pp. 1478–1488. DOI: 10.1109/tnnls.2021.3105444. URL: https://doi.org/10.1109/tnnls.2021.3105444 (cit. on pp. 49, 143, 145, 203).

[700] A. D. Jagtap and G. E. Karniadakis. "How Important Are Activation Functions in Regression and Classification? A Survey, Performance Comparison, and Future Directions." In: *Journal of Machine Learning for Modeling and Computing* 4.1 (2023), pp. 21–75. DOI: 10.1615/jmachlearnmodelcomput.2023047367. URL: https://doi.org/10.1615/jmachlearnmodelcomput.2023047367 (cit. on p. 49).

[701] R. H. K. Emanuel, P. D. Docherty, H. Lunt, and K. Möller. "The effect of activation functions on accuracy, convergence speed, and misclassification confidence in CNN text classification: a comprehensive exploration." In: *The Journal of Supercomputing* 80.1 (June 2023), pp. 292–312. ISSN: 1573-0484. DOI: 10.1007/s11227-023-05441-7. URL: http://dx.doi.org/10.1007/s11227-023-05441-7 (cit. on p. 49).

[702] Z. Zhang, X. Li, Y. Yang, and Z. Shi. "Enhancing Deep Learning Models for Image Classification using Hybrid Activation Functions." In: (Nov. 2023). DOI: 10.21203/rs.3.rs-3574353/v1. URL: http://dx.doi.org/10.21203/rs.3.rs-3574353/v1 (cit. on p. 49).

[703] B. Singh, S. Patel, A. Vijayvargiya, and R. Kumar. "Analyzing the impact of activation functions on the performance of the data-driven gait model." In: *Results in Engineering* 18 (June 2023), p. 101029. ISSN: 2590-1230. DOI: 10.1016/j.rineng.2023.101029. URL: http://dx.doi.org/10.1016/j.rineng.2023.101029 (cit. on pp. 49, 75).

[704] K.-C. Lin, C.-H. Hu, and K.-C. Wang. "Innovative deep energy method for piezoelectricity problems." In: *Applied Mathematical Modelling* 126 (Feb. 2024), pp. 405–419. ISSN: 0307-904X. DOI: 10.1016/j.apm.2023.11.006. URL: http://dx.doi.org/10.1016/j.apm.2023.11.006 (cit. on p. 49).

[705] N. T. Koh, A. Sharma, J. Xiao, X. Peng, and W. L. Woo. "Solar Irradiance Forecast using Long Short-Term Memory: A Comparative Analysis of Different Activation Functions." In: *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2022. DOI: 10.1109/ssci51031.2022.10022163. URL: http://dx.doi.org/10.1109/SSCI51031.2022.10022163 (cit. on p. 49).

[706] S. H. Bhojani and N. Bhatt. "Performance Analysis of Activation Functions for Wheat Crop Yield Prediction." In: *IOP Conference Series: Materials Science and Engineering* 1042.1 (Jan. 2021), p. 012015. ISSN: 1757-899X. DOI: 10.1088/1757-899x/1042/1/012015. URL: http://dx.doi.org/10.1088/1757-899X/1042/1/012015 (cit. on p. 49).

[707] L. A. Hurley, J. G. Restrepo, and S. E. Shaheen. *Tuning the activation function to optimize the forecast horizon of a reservoir computer*. 2023. DOI: 10.48550/ARXIV.2312.13151. URL: https://arxiv.org/abs/2312.13151 (cit. on pp. 49, 94, 177).

[708] F. Makhrus. "The effect of amplitude modification in S-shaped activation functions on neural network regression." In: *Neural Network World* 33.4 (2023), pp. 245–269. ISSN: 2336-4335. DOI: 10.14311/nnw.2023.33.015. URL: http://dx.doi.org/10.14311/nnw.2023.33.015 (cit. on p. 49).

[709] A. Mishra, P. Chandra, and U. Ghose. "A Non-monotonic Activation Function for Neural Networks Validated on Benchmark Tasks." In: *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*. Springer International Publishing, 2021, pp. 319–327. ISBN: 9783030682910. DOI: 10.1007/978-3-030-68291-0_25. URL: http://dx.doi.org/10.1007/978-3-030-68291-0_25 (cit. on pp. 49, 94).

[710] M. H. Essai Ali, A. B. Abdel-Raman, and E. A. Badry. "Developing Novel Activation Functions Based Deep Learning LSTM for Classification." In: *IEEE Access* 10 (2022), pp. 97259–97275. ISSN: 2169-3536. DOI: 10.1109/access.2022.3205774. URL: http://dx.doi.org/10.1109/ACCESS.2022.3205774 (cit. on pp. 49, 96).

[711] D. J. Rumala, E. M. Yuniarno, R. F. Rachmadi, S. M. S. Nugroho, H. P. A. Tjahyaningtijas, Y. Adrianto, A. D. Sensusiati, and I. K. E. Purnama. "Activation Functions Evaluation to Improve Performance of Convolutional Neural Network in Brain Disease Classification Based on Magnetic Resonance Images." In: *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*. IEEE, Nov. 2020. DOI: 10.1109/cenim51130.2020.9297862. URL: http://dx.doi.org/10.1109/CENIM51130.2020.9297862 (cit. on p. 49).

[712] L. Suciningtyas, R. Alfatikarani, M. B. F. Alan, F. M. Maimunir, and H. P. A. Tjahyaningtijas. "Activation Function Comparison On Potato Leaf Disease Classification Performance." In: *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*. IEEE, Oct. 2023. DOI: 10.1109/icvee59738.2023.10348283. URL: http://dx.doi.org/10.1109/ICVEE59738.2023.10348283 (cit. on p. 49).

[713] M. A. Mercioni and S. Holban. "A Brief Review of the Most Recent Activation Functions for Neural Networks." In: *2023 17th International Conference on Engineering of Modern Electric Systems (EMES)*. IEEE, June 2023. DOI: 10.1109/emes58375.2023.10171705. URL: http://dx.doi.org/10.1109/EMES58375.2023.10171705 (cit. on p. 49).

[714] H. Wang, L. Lu, S. S. null, and G. Huang. "Learning Specialized Activation Functions for Physics-Informed Neural Networks." In: *Communications in Computational Physics* 34.4 (June 2023), pp. 869–906. ISSN: 1991-7120. DOI: 10.4208/cicp.oa-2023-0058. URL: http://dx.doi.org/10.4208/cicp.OA-2023-0058 (cit. on p. 49).

[715] O. Pantalé. "Comparing Activation Functions in Machine Learning for Finite Element Simulations in Thermomechanical Forming." In: *Algorithms* 16.12 (Nov. 2023), p. 537. ISSN: 1999-4893. DOI: 10.3390/a16120537. URL: http://dx.doi.org/10.3390/a16120537 (cit. on p. 49).

[716] D. V. Dung, N. D. Song, P. S. Palar, and L. R. Zuhal. "On The Choice of Activation Functions in Physics-Informed Neural Network for Solving Incompressible Fluid Flows." In: *AIAA SCITECH 2023 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2023. DOI: 10.2514/6.2023-1803. URL: http://dx.doi.org/10.2514/6.2023-1803 (cit. on p. 49).

[717] X. Liu, J. Zhou, and H. Qian. "Comparison and Evaluation of Activation Functions in Term of Gradient Instability in Deep Neural Networks." In: *2019 Chinese Control And Decision Conference (CCDC)*. IEEE, June 2019. DOI: 10.1109/ccdc.2019.8832578. URL: http://dx.doi.org/10.1109/CCDC.2019.8832578 (cit. on p. 49).

[718] K. Ingole and N. Patil. "Performance Analysis of Various Activation Function on a Shallow Neural Network." In: *International Journal of Emerging Technologies and Innovative Research* 7.6 (June 2020), pp. 269–276. ISSN: 2349-5162. URL: http://www.jetir.org/papers/JETIR2006041.pdf (cit. on p. 49).

[719] L. Nanni, A. Lumini, S. Ghidoni, and G. Maguolo. “Comparisons among different stochastic selections of activation layers for convolutional neural networks for health care.” In: *Cognitive and Soft Computing Techniques for the Analysis of Healthcare Data*. Elsevier, 2022, pp. 151–164. DOI: 10.1016/b978-0-323-85751-2.00003-7. URL: http://dx.doi.org/10.1016/b978-0-323-85751-2.00003-7 (cit. on p. 49).

[720] R. Zhang, Y. Zhu, Z. Ge, H. Mu, D. Qi, and H. Ni. “Transfer Learning for Leaf Small Dataset Using Improved ResNet50 Network with Mixed Activation Functions.” In: *Forests* 13.12 (Dec. 2022), p. 2072. ISSN: 1999-4907. DOI: 10.3390/f13122072. URL: http://dx.doi.org/10.3390/f13122072 (cit. on p. 49).

[721] A. Chaturvedi, N. Apoorva, M. S. Awasthi, S. Jyoti, D. P. Akarsha, S. Brunda, and C. S. Soumya. “Analyzing the Performance of Novel Activation Functions on Deep Learning Architectures.” In: *Lecture Notes in Electrical Engineering*. Springer Nature Singapore, Dec. 2022, pp. 903–915. ISBN: 9789811954825. DOI: 10.1007/978-981-19-5482-5_76. URL: http://dx.doi.org/10.1007/978-981-19-5482-5_76 (cit. on p. 49).

[722] D. Pedamonti. *Comparison of non-linear activation functions for deep neural networks on MNIST classification task*. 2018. DOI: 10.48550/ARXIV.1804.02763. URL: https://arxiv.org/abs/1804.02763 (cit. on p. 49).

[723] Y. Bai. “RELU-Function and Derived Function Review.” In: *SHS Web of Conferences* 144 (2022). Ed. by A. Luqman, Q. Zhang, and W. Liu, p. 02006. ISSN: 2261-2424. DOI: 10.1051/shsconf/202214402006. URL: http://dx.doi.org/10.1051/shsconf/202214402006 (cit. on p. 49).

[724] F. Kamalov, A. Nazir, M. Safaraliev, A. K. Cherukuri, and R. Zgheib. “Comparative analysis of activation functions in neural networks.” In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, Nov. 2021. DOI: 10.1109/icecs53924.2021.9665646. URL: http://dx.doi.org/10.1109/ICECS53924.2021.9665646 (cit. on p. 49).

[725] M. M. Lau and K. Hann Lim. “Review of Adaptive Activation Function in Deep Neural Network.” In: *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*. IEEE, Dec. 2018. DOI: 10.1109/iecbes.2018.8626714. URL: http://dx.doi.org/10.1109/IECBES.2018.8626714 (cit. on p. 49).

[726] A. Dubowski. *Activation function impact on Sparse Neural Networks*. 2020. DOI: 10.48550/ARXIV.2010.05943. URL: https://arxiv.org/abs/2010.05943 (cit. on p. 49).

[727] I. A. Kandhro, M. Uddin, S. Hussain, T. J. Chaudhery, M. Shorfuzzaman, H. Meshref, M. Albalhaq, R. Alsaqour, and O. I. Khalaf. “Impact of Activation, Optimization, and Regularization Methods on the Facial Expression Model Using CNN.” In: *Computational Intelligence and Neuroscience* 2022 (June 2022). Ed. by M. Z. Asghar, pp. 1–9. ISSN: 1687-5265. DOI: 10.1155/2022/3098604. URL: http://dx.doi.org/10.1155/2022/3098604 (cit. on p. 49).

[728] E. C. Seyrek and M. Uysal. “A comparative analysis of various activation functions and optimizers in a convolutional neural network for hyperspectral image classification.” In: *Multimedia Tools and Applications* (Nov. 2023). ISSN: 1573-7721. DOI: 10.1007/s11042-023-17546-5. URL: http://dx.doi.org/10.1007/s11042-023-17546-5 (cit. on p. 49).

[729] C. Bircanoglu and N. Arica. “A comparison of activation functions in artificial neural networks.” In: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, May 2018. DOI: 10.1109/siu.2018.8404724. URL: http://dx.doi.org/10.1109/SIU.2018.8404724 (cit. on p. 49).

[730] P. Tatraiya, H. Priyadarshi, K. Singh, D. Mishra, and A. Shrivastava. “Applicative Analysis Of Activation Functions For Pneumonia Detection Using Convolutional Neural Networks.” In: *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*. IEEE, May 2023. DOI: 10.1109/globconet56651.2023.10149937. URL: http://dx.doi.org/10.1109/GlobConET56651.2023.10149937 (cit. on p. 49).

[731] W. Qiu, C. He, G. Zheng, Q. Yi, and G. Chen. “Activation Function Dependence of Data-Driven Spectra Prediction of Nanostructures.” In: *Advanced Theory and Simulations* 6.5 (Feb. 2023). ISSN: 2513-0390. DOI: 10.1002/adts.202200867. URL: http://dx.doi.org/10.1002/adts.202200867 (cit. on p. 49).

[732] V. K. Kayala and P. Kodali. “Performance Analysis of Activation Functions on Convolutional Neural Networks Using Cloud GPU.” In: *Advances in Communications, Signal Processing, and VLSI*. Springer Singapore, 2021, pp. 35–48. ISBN: 9789813340589. DOI: 10.1007/978-981-33-4058-9_4. URL: http://dx.doi.org/10.1007/978-981-33-4058-9_4 (cit. on p. 49).

[733] L. Xu and C. L. Philip Chen. “Comparison and Combination of Activation Functions in Broad Learning System.” In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2020. DOI: 10.1109/smc42975.2020.9282871. URL: http://dx.doi.org/10.1109/SMC42975.2020.9282871 (cit. on pp. 49, 62, 71, 84, 85, 98, 99).

[734] A. Salam, A. E. Hibaoui, and A. Saif. "A comparison of activation functions in multilayer neural network for predicting the production and consumption of electricity power." In: *International Journal of Electrical and Computer Engineering (IJECE)* 11.1 (Feb. 2021), p. 163. ISSN: 2088-8708. DOI: 10.11591/ijece.v11i1.pp163-170. URL: http://dx.doi.org/10.11591/ijece.v11i1.pp163-170 (cit. on p. 49).

[735] A. S. Lutakamale and Y. Z. Manyesela. "The Influence of Non-learnable Activation Functions on the Positioning Performance of Deep Learning-Based Fingerprinting Models Trained with Small CSI Sample Sizes." In: *Transactions of the Indian National Academy of Engineering* 7.3 (July 2022), pp. 1059–1067. ISSN: 2662-5423. DOI: 10.1007/s41403-022-00347-x. URL: http://dx.doi.org/10.1007/s41403-022-00347-x (cit. on p. 49).

[736] A. Nandi, N. D. Jana, and S. Das. "Improving the Performance of Neural Networks with an Ensemble of Activation Functions." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2020. DOI: 10.1109/ijcnn48605.2020.9207277. URL: http://dx.doi.org/10.1109/IJCNN48605.2020.9207277 (cit. on p. 49).

[737] J. M. Locke, D. Paradice, and R. K. Rainer. "Mitigating bias through random activation function selection." In: *Neural Computing and Applications* (Nov. 2023). ISSN: 1433-3058. DOI: 10.1007/s00521-023-09178-5. URL: http://dx.doi.org/10.1007/s00521-023-09178-5 (cit. on p. 49).

[738] K. Wong, R. Dornberger, and T. Hanne. "An analysis of weight initialization methods in connection with different activation functions for feedforward neural networks." In: *Evolutionary Intelligence* (Nov. 2022). DOI: 10.1007/s12065-022-00795-y. URL: https://doi.org/10.1007/s12065-022-00795-y (cit. on pp. 49, 62).

[739] S. Saha, A. Mathur, A. Pandey, and H. Arun Kumar. "DiffAct: A Unifying Framework for Activation Functions." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2021. DOI: 10.1109/ijcnn52387.2021.9534391. URL: http://dx.doi.org/10.1109/IJCNN52387.2021.9534391 (cit. on p. 49).

[740] A. Farzad, H. Mashayekhi, and H. Hassanpour. "A comparative performance analysis of different activation functions in LSTM networks for classification." In: *Neural Computing and Applications* 31.7 (Oct. 2017), pp. 2507–2521. DOI: 10.1007/s00521-017-3210-6. URL: https://doi.org/10.1007/s00521-017-3210-6 (cit. on pp. 49, 55).

[741] M. Badiger and J. A. Mathew. "Retrospective Review of Activation Functions in Artificial Neural Networks." In: *Proceedings of Third International Conference on Communication, Computing and Electronics Systems*. Springer Singapore, 2022, pp. 905–919. DOI: 10.1007/978-981-16-8862-1_59. URL: https://doi.org/10.1007/978-981-16-8862-1_59 (cit. on p. 49).

[742] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. DOI: 10.48550/ARXIV.1811.03378. URL: https://arxiv.org/abs/1811.03378 (cit. on pp. 49, 141, 142).

[743] W. Duch and N. Jankowski. "Taxonomy of neural transfer functions." In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000. DOI: 10.1109/ijcnn.2000.861353. URL: http://dx.doi.org/10.1109/IJCNN.2000.861353 (cit. on p. 49).

[744] B. Raitani. "Survey on recent activation functions with emphasis on oscillating activation functions." In: (June 2022). DOI: 10.31224/2429. URL: http://dx.doi.org/10.31224/2429 (cit. on p. 49).

[745] L. Datta. *A Survey on Activation Functions and their relation with Xavier and He Normal Initialization*. 2020. DOI: 10.48550/ARXIV.2004.06632. URL: https://arxiv.org/abs/2004.06632 (cit. on p. 49).

[746] A. D. Rasamoelina, F. Adjailia, and P. Sincak. "A Review of Activation Function for Artificial Neural Network." In: *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, Jan. 2020. DOI: 10.1109/sami48414.2020.9108717. URL: http://dx.doi.org/10.1109/SAMI48414.2020.9108717 (cit. on p. 49).

[747] M. A. Mercioni and S. Holban. "The Most Used Activation Functions: Classic Versus Current." In: *2020 International Conference on Development and Application Systems (DAS)*. IEEE, May 2020. DOI: 10.1109/das49615.2020.9108942. URL: http://dx.doi.org/10.1109/DAS49615.2020.9108942 (cit. on p. 49).

[748] K. Liu. "Analysis of Features of Different Activation Functions." In: *2021 2nd International Conference on Computing and Data Science (CDS)*. IEEE, Jan. 2021. DOI: 10.1109/cds52072.2021.00078. URL: http://dx.doi.org/10.1109/CDS52072.2021.00078 (cit. on p. 49).

[749] S. Serhat Kiliçarslan, K. Adem, and M. Çelik. "An overview of the activation functions used in deep learning algorithms." In: *Journal of New Results in Science* 10.3 (Dec. 2021), pp. 75–88. ISSN: 1304-7981. DOI: 10.54187/jnrs.1011739. URL: http://dx.doi.org/10.54187/jnrs.1011739 (cit. on p. 49).

[750] M. Çelebi and M. Ceylan. "The New Activation Function for Complex Valued Neural Networks: Complex Swish Function." In: *4th International Symposium on Innovative Approaches in Engineering and Natural Sciences Proceedings*. SETSCI, July 2019. DOI: 10.36287/setsci.4.6.050. URL: https://doi.org/10.36287/setsci.4.6.050 (cit. on p. 49).

[751] Y. Zhang, Q. Hua, H. Wang, Z. Ji, and Y. Wang. "Gaussian-type activation function with learnable parameters in complex-valued convolutional neural network and its application for PolSAR classification." In: *Neurocomputing* 518 (Jan. 2023), pp. 95–110. DOI: 10.1016/j.neucom.2022.10.082. URL: https://doi.org/10.1016/j.neucom.2022.10.082 (cit. on p. 49).

[752] S. Scardapane, S. V. Vaerenbergh, A. Hussain, and A. Uncini. "Complex-Valued Neural Networks With Nonparametric Activation Functions." In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.2 (Apr. 2020), pp. 140–150. DOI: 10.1109/tetci.2018.2872600. URL: https://doi.org/10.1109/tetci.2018.2872600 (cit. on p. 49).

[753] B. N. Örnek, S. B. Aydemir, T. Düzenli, and B. Özak. "Some remarks on activation function design in complex extreme learning using Schwarz lemma." In: *Neurocomputing* 492 (July 2022), pp. 23–33. DOI: 10.1016/j.neucom.2022.04.010. URL: https://doi.org/10.1016/j.neucom.2022.04.010 (cit. on p. 49).

[754] Q. Hua, Y. Zhang, Y. Jiang, and H. Mu. "Gaussian-type activation function for complex-valued CNN and its application in polar-SAR image classification." In: *Journal of Applied Remote Sensing* 15.02 (May 2021). DOI: 10.1117/1.jrs.15.026510. URL: https://doi.org/10.1117/1.jrs.15.026510 (cit. on p. 49).

[755] T. Kim and T. Adali. "Fully Complex Multi-Layer Perceptron Network for Nonlinear Signal Processing." In: *The Journal of VLSI Signal Processing* 32.1/2 (2002), pp. 29–43. DOI: 10.1023/a:1016359216961. URL: https://doi.org/10.1023/a:1016359216961 (cit. on p. 49).

[756] R. Savitha, S. Suresh, N. Sundararajan, and P. Saratchandran. "A new learning algorithm with logarithmic performance index for complex-valued neural networks." In: *Neurocomputing* 72.16-18 (Oct. 2009), pp. 3771–3781. DOI: 10.1016/j.neucom.2009.06.004. URL: https://doi.org/10.1016/j.neucom.2009.06.004 (cit. on p. 49).

[757] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew. "Incremental extreme learning machine with fully complex hidden nodes." In: *Neurocomputing* 71.4-6 (Jan. 2008), pp. 576–583. DOI: 10.1016/j.neucom.2007.07.025. URL: https://doi.org/10.1016/j.neucom.2007.07.025 (cit. on p. 49).

[758] R. Savitha, S. Suresh, N. Sundararajan, and H. Kim. "A fully complex-valued radial basis function classifier for real-valued classification problems." In: *Neurocomputing* 78.1 (Feb. 2012), pp. 104–110. DOI: 10.1016/j.neucom.2011.05.036. URL: https://doi.org/10.1016/j.neucom.2011.05.036 (cit. on p. 49).

[759] J. Hu, H. Tan, and C. Zeng. "Global exponential stability of delayed complex-valued neural networks with discontinuous activation functions." In: *Neurocomputing* 416 (Nov. 2020), pp. 1–11. DOI: 10.1016/j.neucom.2020.02.006. URL: https://doi.org/10.1016/j.neucom.2020.02.006 (cit. on p. 49).

[760] M. Tan and D. Xu. "Multiple μ-stability analysis for memristor-based complex-valued neural networks with nonmonotonic piecewise nonlinear activation functions and unbounded time-varying delays." In: *Neurocomputing* 275 (Jan. 2018), pp. 2681–2701. DOI: 10.1016/j.neucom.2017.11.047. URL: https://doi.org/10.1016/j.neucom.2017.11.047 (cit. on p. 49).

[761] Y. Kuroe, M. Yoshid, and T. Mori. "On Activation Functions for Complex-Valued Neural Networks — Existence of Energy Functions." In: *Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003*. Springer Berlin Heidelberg, 2003, pp. 985–992. DOI: 10.1007/3-540-44989-2_117. URL: https://doi.org/10.1007/3-540-44989-2_117 (cit. on p. 49).

[762] N. Özdemir, B. B. İskender, and N. Y. Özgür. "Complex valued neural network with Möbius activation function." In: *Communications in Nonlinear Science and Numerical Simulation* 16.12 (Dec. 2011), pp. 4698–4703. DOI: 10.1016/j.cnsns.2011.03.005. URL: https://doi.org/10.1016/j.cnsns.2011.03.005 (cit. on p. 49).

[763] J. Gao, B. Deng, Y. Qin, H. Wang, and X. Li. "Enhanced Radar Imaging Using a Complex-Valued Convolutional Neural Network." In: *IEEE Geoscience and Remote Sensing Letters* 16.1 (Jan. 2019), pp. 35–39. ISSN: 1558-0571. DOI: 10.1109/lgrs.2018.2866567. URL: http://dx.doi.org/10.1109/LGRS.2018.2866567 (cit. on p. 49).

[764] C. Lee, H. Hasegawa, and S. Gao. "Complex-Valued Neural Networks: A Comprehensive Survey." In: *IEEE/CAA Journal of Automatica Sinica* 9.8 (Aug. 2022), pp. 1406–1426. DOI: 10.1109/jas.2022.105743. URL: https://doi.org/10.1109/jas.2022.105743 (cit. on p. 49).

[765] N. Vieira. "Bicomplex Neural Networks with Hypergeometric Activation Functions." In: *Advances in Applied Clifford Algebras* 33.2 (Mar. 2023). ISSN: 1661-4909. DOI: 10.1007/s00006-023-01268-w. URL: http://dx.doi.org/10.1007/s00006-023-01268-w (cit. on p. 49).

[766] D. García-Retuerta, R. Casado-Vara, A. Martin-del Rey, F. De la Prieta, J. Prieto, and J. M. Corchado. "Quaternion Neural Networks: State-of-the-Art and Research Challenges." In: *Intelligent Data Engineering and Automated Learning – IDEAL 2020*. Springer International Publishing, 2020, pp. 456–467. ISBN: 9783030623654. DOI: 10.1007/978-3-030-62365-4_43. URL: http://dx.doi.org/10.1007/978-3-030-62365-4_43 (cit. on p. 49).

[767] X. Zhu, Y. Xu, H. Xu, and C. Chen. "Quaternion Convolutional Neural Networks." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018. URL: https://openaccess.thecvf.com/content_ECCV_2018/papers/Xuanyu_Zhu_Quaternion_Convolutional_Neural_ECCV_2018_paper.pdf (cit. on p. 49).

[768] C.-A. Popa. "Scaled Conjugate Gradient Learning for Quaternion-Valued Neural Networks." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 243–252. ISBN: 9783319466750. DOI: 10.1007/978-3-319-46675-0_27. URL: http://dx.doi.org/10.1007/978-3-319-46675-0_27 (cit. on p. 49).

[769] C.-A. Popa. "Learning Algorithms for Quaternion-Valued Neural Networks." In: *Neural Processing Letters* 47.3 (Sept. 2017), pp. 949–973. ISSN: 1573-773X. DOI: 10.1007/s11063-017-9716-1. URL: http://dx.doi.org/10.1007/s11063-017-9716-1 (cit. on p. 49).

[770] S. Yu, H. Li, X. Chen, and D. Lin. "Multistability analysis of quaternion-valued neural networks with cosine activation functions." In: *Applied Mathematics and Computation* 445 (May 2023), p. 127849. ISSN: 0096-3003. DOI: 10.1016/j.amc.2023.127849. URL: http://dx.doi.org/10.1016/j.amc.2023.127849 (cit. on p. 49).

[771] Z. Xu, B. Tang, X. Zhang, J. F. Leong, J. Pan, S. Hooda, E. Zamburg, and A. V.-Y. Thean. "Reconfigurable nonlinear photonic activation function for photonic neural network based on non-volatile opto-resistive RAM switch." In: *Light: Science & Applications* 11.1 (Oct. 2022). ISSN: 2047-7538. DOI: 10.1038/s41377-022-00976-5. URL: http://dx.doi.org/10.1038/s41377-022-00976-5 (cit. on p. 49).

[772] P. V. de Campos Souza. "Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature." In: *Applied Soft Computing* 92 (July 2020), p. 106275. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2020.106275. URL: http://dx.doi.org/10.1016/j.asoc.2020.106275 (cit. on p. 49).

[773] R. Das, S. Sen, and U. Maulik. "A Survey on Fuzzy Deep Neural Networks." In: *ACM Computing Surveys* 53.3 (May 2020), pp. 1–25. ISSN: 1557-7341. DOI: 10.1145/3369798. URL: http://dx.doi.org/10.1145/3369798 (cit. on p. 49).

[774] S. L. Bangare. "Classification of optimal brain tissue using dynamic region growing and fuzzy min-max neural network in brain magnetic resonance images." In: *Neuroscience Informatics* 2.3 (Sept. 2022), p. 100019. ISSN: 2772-5286. DOI: 10.1016/j.neuri.2021.100019. URL: http://dx.doi.org/10.1016/j.neuri.2021.100019 (cit. on p. 49).

[775] M. Malcangi and G. Nano. "Biofeedback: e-health prediction based on evolving fuzzy neural network and wearable technologies." In: *Evolving Systems* 12.3 (Mar. 2021), pp. 645–653. ISSN: 1868-6486. DOI: 10.1007/s12530-021-09374-5. URL: http://dx.doi.org/10.1007/s12530-021-09374-5 (cit. on p. 49).

[776] J. Fei, Z. Wang, X. Liang, Z. Feng, and Y. Xue. "Fractional Sliding-Mode Control for Microgyroscope Based on Multilayer Recurrent Fuzzy Neural Network." In: *IEEE Transactions on Fuzzy Systems* 30.6 (June 2022), pp. 1712–1721. ISSN: 1941-0034. DOI: 10.1109/tfuzz.2021.3064704. URL: http://dx.doi.org/10.1109/TFUZZ.2021.3064704 (cit. on p. 49).

[777] X.-h. Liu, D. Zhang, J. Zhang, T. Zhang, and H. Zhu. "A path planning method based on the particle swarm optimization trained fuzzy neural network algorithm." In: *Cluster Computing* 24.3 (Jan. 2021), pp. 1901–1915. ISSN: 1573-7543. DOI: 10.1007/s10586-021-03235-1. URL: http://dx.doi.org/10.1007/s10586-021-03235-1 (cit. on p. 49).

[778] J. A. Duersch, T. A. Catanach, and N. Das. *Adaptive n-ary Activation Functions for Probabilistic Boolean Logic*. 2022. DOI: 10.48550/ARXIV.2203.08977. URL: https://arxiv.org/abs/2203.08977 (cit. on p. 49).

[779] L. Parisi, D. Neagu, R. Ma, and F. Campean. "Quantum ReLU activation for Convolutional Neural Networks to improve diagnosis of Parkinson's disease and COVID-19." In: *Expert Systems with Applications* 187 (Jan. 2022), p. 115892. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.115892. URL: http://dx.doi.org/10.1016/j.eswa.2021.115892 (cit. on p. 49).

[780] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. "Efficient Neural Network Robustness Certification with General Activation Functions." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf (cit. on p. 50).

[781] A. Dinu and M. Cirstea. "A Digital Neural Network FPGA Direct Hardware Implementation Algorithm." In: *2007 IEEE International Symposium on Industrial Electronics*. IEEE, June 2007. DOI: 10.1109/isie.2007.4374572. URL: https://doi.org/10.1109/isie.2007.4374572 (cit. on p. 50).

[782] A. Dinu, M. N. Cirstea, and S. E. Cirstea. "Direct Neural-Network Hardware-Implementation Algorithm." In: *IEEE Transactions on Industrial Electronics* 57.5 (May 2010), pp. 1845–1848. DOI: 10.1109/tie.2009.2033097. URL: https://doi.org/10.1109/tie.2009.2033097 (cit. on p. 50).

[783] J. Chen and Z. Pan. *Saturated Non-Monotonic Activation Functions*. 2023. DOI: 10.48550/ARXIV.2305.07537. URL: https://arxiv.org/abs/2305.07537 (cit. on p. 50).

[784] G. S. da S. Gomes, T. B. Ludermir, and L. M. M. R. Lima. "Comparison of new activation functions in neural network for forecasting financial time series." In: *Neural Computing and Applications* 20.3 (June 2010), pp. 417–439. ISSN: 1433-3058. DOI: 10.1007/s00521-010-0407-3. URL: http://dx.doi.org/10.1007/s00521-010-0407-3 (cit. on pp. 51, 56, 58).

[785] T. E. Simos and C. Tsitouras. "Efficiently inaccurate approximation of hyperbolic tangent used as transfer function in artificial neural networks." In: *Neural Computing and Applications* 33.16 (Mar. 2021), pp. 10227–10233. DOI: 10.1007/s00521-021-05787-0. URL: https://doi.org/10.1007/s00521-021-05787-0 (cit. on p. 51).

[786] G. Dudek. "Data-Driven Learning of Feedforward Neural Networks with Different Activation Functions." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 66–77. ISBN: 9783030879860. DOI: 10.1007/978-3-030-87986-0_6. URL: http://dx.doi.org/10.1007/978-3-030-87986-0_6 (cit. on pp. 51, 99).

[787] D. W. Edwards and I. Dinc. "LRTanH: Substitution for the Activation Function Derivative during Back Propagation." In: *2019 SoutheastCon*. IEEE, Apr. 2019. DOI: 10.1109/southeastcon42311.2019.9020655. URL: http://dx.doi.org/10.1109/SoutheastCon42311.2019.9020655 (cit. on p. 51).

[788] K. Sunat, C. Lursinsap, and C.-H. H. Chu. "The p-recursive piecewise polynomial sigmoid generators and first-order algorithms for multilayer tanh-like neurons." In: *Neural Computing and Applications* 16.1 (Apr. 2006), pp. 33–47. ISSN: 1433-3058. DOI: 10.1007/s00521-006-0046-x. URL: http://dx.doi.org/10.1007/s00521-006-0046-x (cit. on p. 51).

[789] H. Kwan. "Simple sigmoid-like activation function suitable for digital hardware implementation." In: *Electronics Letters* 28.15 (1992), p. 1379. ISSN: 0013-5194. DOI: 10.1049/el:19920877. URL: http://dx.doi.org/10.1049/el:19920877 (cit. on p. 51).

[790] M. Zhang, S. Vassiliadis, and J. Delgado-Frias. "Sigmoid generators for neural computing using piecewise approximations." In: *IEEE Transactions on Computers* 45.9 (1996), pp. 1045–1049. ISSN: 0018-9340. DOI: 10.1109/12.537127. URL: http://dx.doi.org/10.1109/12.537127 (cit. on p. 51).

[791] B. Xu, R. Huang, and M. Li. *Revise Saturated Activation Functions*. 2016. DOI: 10.48550/ARXIV.1602.05980. URL: https://arxiv.org/abs/1602.05980 (cit. on pp. 51, 54, 212).

[792] D. B. Mulindwa and S. Du. "An n-Sigmoid Activation Function to Improve the Squeeze-and-Excitation for 2D and 3D Deep Networks." In: *Electronics* 12.4 (Feb. 2023), p. 911. ISSN: 2079-9292. DOI: 10.3390/electronics12040911. URL: http://dx.doi.org/10.3390/electronics12040911 (cit. on p. 51).

[793] H. Arai and H. Imamura. "Spin-wave coupled spin torque oscillators for artificial neural network." In: *Journal of Applied Physics* 124.15 (Oct. 2018). ISSN: 1089-7550. DOI: 10.1063/1.5040020. URL: http://dx.doi.org/10.1063/1.5040020 (cit. on pp. 51, 200).

[794] J. Han and C. Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning." In: *From Natural to Artificial Neural Computation*. Springer Berlin Heidelberg, 1995, pp. 195–201. ISBN: 9783540492887. DOI: 10.1007/3-540-59497-3_175. URL: http://dx.doi.org/10.1007/3-540-59497-3_175 (cit. on pp. 52, 200).

[795] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791. URL: https://doi.org/10.1109/5.726791 (cit. on pp. 52, 126, 195, 200).

[796] S. S. Sodhi and P. Chandra. "Bi-modal derivative activation function for sigmoidal feedforward networks." In: *Neurocomputing* 143 (Nov. 2014), pp. 182–196. DOI: 10.1016/j.neucom.2014.06.007. URL: https://doi.org/10.1016/j.neucom.2014.06.007 (cit. on p. 52).

[797] T. T. Sivri, N. P. Akman, and A. Berkol. "Multiclass Classification Using Arctangent Activation Function and Its Variations." In: *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, June 2022. DOI: 10.1109/ecai54874.2022.9847486. URL: https://doi.org/10.1109/ecai54874.2022.9847486 (cit. on p. 52).

[798] J. Kamruzzaman and S. Aziz. “A note on activation function in multilayer feedforward learning.” In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. IJCNN-02. IEEE, 2002. DOI: 10.1109/ijcnn.2002.1005526. URL: http://dx.doi.org/10.1109/IJCNN.2002.1005526 (cit. on pp. 52, 94).

[799] T. Tümer-Sivri, N. Pervan-Akman, and A. Berkol. “The Impact of Irrationals on the Range of Arctan Activation Function for Deep Learning Models.” In: (May 2023). DOI: 10.20944/preprints202305.1245.v1. URL: http://dx.doi.org/10.20944/preprints202305.1245.v1 (cit. on pp. 52, 61).

[800] Y. Koçak and G. Üstündağ Şiray. “New activation functions for single layer feedforward neural network.” In: *Expert Systems with Applications* 164 (Feb. 2021), p. 113977. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2020.113977. URL: http://dx.doi.org/10.1016/j.eswa.2020.113977 (cit. on pp. 53, 55, 60, 61, 82).

[801] Y. Qin, X. Wang, and J. Zou. “The Optimized Deep Belief Networks With Improved Logistic Sigmoid Units and Their Application in Fault Diagnosis for Planetary Gearboxes of Wind Turbines.” In: *IEEE Transactions on Industrial Electronics* 66.5 (May 2019), pp. 3814–3824. DOI: 10.1109/tie.2018.2856205. URL: https://doi.org/10.1109/tie.2018.2856205 (cit. on pp. 53, 212).

[802] M. Roodschild, J. Gotay Sardiñas, and A. Will. “A new approach for the vanishing gradient problem on sigmoid activation.” In: *Progress in Artificial Intelligence* 9.4 (Oct. 2020), pp. 351–360. ISSN: 2192-6360. DOI: 10.1007/s13748-020-00218-y. URL: http://dx.doi.org/10.1007/s13748-020-00218-y (cit. on p. 53).

[803] D. Li and Y. Zhou. “Soft-Root-Sign: A New Bounded Neural Activation Function.” In: *Pattern Recognition and Computer Vision*. Springer International Publishing, 2020, pp. 310–319. DOI: 10.1007/978-3-030-60636-7_26. URL: https://doi.org/10.1007/978-3-030-60636-7_26 (cit. on p. 54).

[804] I. Ohn and Y. Kim. “Smooth Function Approximation by Deep Neural Networks with General Activation Functions.” In: *Entropy* 21.7 (June 2019), p. 627. ISSN: 1099-4300. DOI: 10.3390/e21070627. URL: http://dx.doi.org/10.3390/e21070627 (cit. on p. 54).

[805] M. Klimek and M. Perelstein. “Neural network-based approach to phase space integration.” In: *SciPost Physics* 9.4 (Oct. 2020). ISSN: 2542-4653. DOI: 10.21468/scipostphys.9.4.053. URL: http://dx.doi.org/10.21468/SciPostPhys.9.4.053 (cit. on p. 54).

[806] S. Kong and M. Takatsuka. “Hexpo: A vanishing-proof activation function.” In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017. DOI: 10.1109/ijcnn.2017.7966168. URL: https://doi.org/10.1109/ijcnn.2017.7966168 (cit. on pp. 54, 55, 212).

[807] H. Hazimeh, N. Ponomareva, P. Mol, Z. Tan, and R. Mazumder. “The tree ensemble layer: differentiability meets conditional computation.” In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020. URL: http://proceedings.mlr.press/v119/hazimeh20a/hazimeh20a.pdf (cit. on p. 55).

[808] D. L. Elliott. *A better Activation Function for Artificial Neural Networks*. Tech. rep. 1993. URL: https://www.researchgate.net/publication/277299531 (cit. on p. 55).

[809] H. Burhani, W. Feng, and G. Hu. “Denoising AutoEncoder in Neural Networks with Modified Elliott Activation Function and Sparsity-Favoring Cost Function.” In: *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*. IEEE, July 2015. DOI: 10.1109/acit-csi.2015.67. URL: http://dx.doi.org/10.1109/ACIT-CSI.2015.67 (cit. on pp. 55, 93).

[810] M. Kaytan, İ. B. Aydilek, C. Yeroglu, and A. Karci. “Sigmoid-Gumbel: Yeni Bir Hibrit Aktivasyon Fonksiyonu.” In: *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi* 11.1 (Mar. 2022), pp. 29–45. ISSN: 2147-3129. DOI: 10.17798/bitlisfen.990508. URL: http://dx.doi.org/10.17798/bitlisfen.990508 (cit. on p. 56).

[811] A. Kumar and S. S. Sodhi. “Image Segmentation on Convolutional Neural Network (CNN) using Some New Activation Functions.” In: *Communications in Mathematics and Applications* 14.2 (Sept. 2023), pp. 949–968. ISSN: 0975-8607. DOI: 10.26713/cma.v14i2.2152. URL: http://dx.doi.org/10.26713/cma.v14i2.2152 (cit. on p. 56).

[812] C. Közkurt, S. Kiliçarslan, S. Baş, and A. Elen. “SechSig and TanhSig: two novel non-monotonic activation functions.” In: *Soft Computing* 27.24 (Oct. 2023), pp. 18451–18467. ISSN: 1433-7479. DOI: 10.1007/s00500-023-09279-2. URL: http://dx.doi.org/10.1007/s00500-023-09279-2 (cit. on p. 57).

[813] C. Cai, Y. Xu, D. Ke, and K. Su. “Deep Neural Networks with Multistate Activation Functions.” In: *Computational Intelligence and Neuroscience* 2015 (2015), pp. 1–10. ISSN: 1687-5273. DOI: 10.1155/2015/721367. URL: http://dx.doi.org/10.1155/2015/721367 (cit. on pp. 57, 219).

[814] W. Duch and N. Jankowski. *Survey of Neural Transfer Functions*. 1999 (cit. on pp. 57, 58).

[815] M. C. N. Guevraa, V. G. S. Cruz, O. O. V. Vergara, M. Nandayapa, H. d. J. D. Ochoa, and H. J. A. Sossa. "Study of the Effect of Combining Activation Functions in a Convolutional Neural Network." In: *IEEE Latin America Transactions* 19.5 (May 2021), pp. 844–852. ISSN: 1548-0992. DOI: 10.1109/tla.2021.9448319. URL: http://dx.doi.org/10.1109/TLA.2021.9448319 (cit. on pp. 58, 84, 115, 290).

[816] S. Elfwing, E. Uchibe, and K. Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning." In: *Neural Networks* 107 (Nov. 2018), pp. 3–11. DOI: 10.1016/j.neunet.2017.12.012. URL: https://doi.org/10.1016/j.neunet.2017.12.012 (cit. on pp. 58, 63, 129).

[817] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. DOI: 10.1109/cvpr.2009.5206848. URL: https://doi.org/10.1109/cvpr.2009.5206848 (cit. on pp. 58, 65, 140).

[818] M. Tanaka. "Weighted sigmoid gate unit for an activation function of deep neural network." In: *Pattern Recognition Letters* 135 (July 2020), pp. 354–359. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2020.05.017. URL: http://dx.doi.org/10.1016/j.patrec.2020.05.017 (cit. on p. 59).

[819] D. Hendrycks and K. Gimpel. *Gaussian Error Linear Units (GELUs)*. 2016. DOI: 10.48550/ARXIV.1606.08415. URL: https://arxiv.org/abs/1606.08415 (cit. on p. 59).

[820] M. Lee. *GELU Activation Function in Deep Learning: A Comprehensive Mathematical Analysis and Performance*. 2023. DOI: 10.48550/ARXIV.2305.12073. URL: https://arxiv.org/abs/2305.12073 (cit. on p. 59).

[821] J. Kang, R. Liu, Y. Li, Q. Liu, P. Wang, Q. Zhang, and D. Zhou. "An Improved 3D Human Pose Estimation Model Based on Temporal Convolution with Gaussian Error Linear Units." In: *2022 8th International Conference on Virtual Reality (ICVR)*. IEEE, May 2022. DOI: 10.1109/icvr55215.2022.9848068. URL: https://doi.org/10.1109/icvr55215.2022.9848068 (cit. on pp. 59, 111).

[822] C. Yu and Z. Su. *Symmetrical Gaussian Error Linear Units (SGELUs)*. 2019. DOI: 10.48550/ARXIV.1911.03925. URL: https://arxiv.org/abs/1911.03925 (cit. on pp. 59, 60, 195, 201).

[823] Z. Wu, H. Yu, L. Zhang, and Y. Sui. "The Adaptive Quadratic Linear Unit (AQuLU): Adaptive Non Monotonic Piecewise Activation Function." In: *Tehnicki vjesnik - Technical Gazette* 30.5 (Oct. 2023). ISSN: 1848-6339. DOI: 10.17559/tv-20230614000735. URL: http://dx.doi.org/10.17559/TV-20230614000735 (cit. on pp. 60, 61, 131).

[824] A. Vagerwal. *Deeper Learning with CoLU Activation*. 2021. DOI: 10.48550/ARXIV.2112.12078. URL: https://arxiv.org/abs/2112.12078 (cit. on p. 60).

[825] M. Kaytan, İ. B. Aydilek, and C. Yeroğlu. "Gish: a novel activation function for image classification." In: *Neural Computing and Applications* 35.34 (Sept. 2023), pp. 24259–24281. ISSN: 1433-3058. DOI: 10.1007/s00521-023-09035-5. URL: http://dx.doi.org/10.1007/s00521-023-09035-5 (cit. on p. 61).

[826] H. Zhu, H. Zeng, J. Liu, and X. Zhang. "Logish: A new nonlinear nonmonotonic activation function for convolutional neural network." In: *Neurocomputing* 458 (Oct. 2021), pp. 490–499. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.06.067. URL: http://dx.doi.org/10.1016/j.neucom.2021.06.067 (cit. on pp. 61, 62, 196, 200).

[827] P. Naveen. "Phish: A Novel Hyper-Optimizable Activation Function." In: (Jan. 2022). DOI: 10.36227/techrxiv.17283824.v2. URL: http://dx.doi.org/10.36227/techrxiv.17283824.v2 (cit. on p. 62).

[828] S. Jianlin. *A brief discussion on the design of activation functions in neural networks*. 2017. URL: https://spaces.ac.cn/archives/4647 (visited on 01/28/2024) (cit. on p. 62).

[829] M. A. Mercioni, A. M. Tat, and S. Holban. "Improving the Accuracy of Deep Neural Networks Through Developing New Activation Functions." In: *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, Sept. 2020. DOI: 10.1109/iccp51029.2020.9266162. URL: http://dx.doi.org/10.1109/ICCP51029.2020.9266162 (cit. on pp. 62, 131).

[830] S. Verma, A. Chug, and A. P. Singh. "Revisiting activation functions: empirical evaluation for image understanding and classification." In: *Multimedia Tools and Applications* (July 2023). ISSN: 1573-7721. DOI: 10.1007/s11042-023-16159-2. URL: http://dx.doi.org/10.1007/s11042-023-16159-2 (cit. on pp. 63, 64).

[831] S. Hayou, A. Doucet, and J. Rousseau. *On the Selection of Initialization and Activation Function for Deep Neural Networks*. 2018. DOI: 10.48550/ARXIV.1805.08266. URL: https://arxiv.org/abs/1805.08266 (cit. on p. 63).

[832] A. N. S. Njikam and H. Zhao. “A novel activation function for multilayer feed-forward neural networks.” In: *Applied Intelligence* 45.1 (Jan. 2016), pp. 75–82. DOI: 10.1007/s10489-015-0744-0. URL: https://doi.org/10.1007/s10489-015-0744-0 (cit. on p. 64).

[833] S. K. Roy, S. Manna, S. R. Dubey, and B. B. Chaudhuri. “LiSHT: Non-parametric Linearly Scaled Hyperbolic Tangent Activation Function for Neural Networks.” In: *Communications in Computer and Information Science*. Springer Nature Switzerland, 2023, pp. 462–476. DOI: 10.1007/978-3-031-31407-0_35. URL: https://doi.org/10.1007/978-3-031-31407-0_35 (cit. on p. 64).

[834] A. Go, R. Bhayani, and L. Huang. *Twitter Sentiment Classification using Distant Supervision*. Tech. rep. 2009. URL: https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf (cit. on p. 64).

[835] T. Sahni, C. Chandak, N. R. Chedeti, and M. Singh. “Efficient Twitter sentiment classification using subjective distant supervision.” In: *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, Jan. 2017. DOI: 10.1109/comsnets.2017.7945451. URL: https://doi.org/10.1109/comsnets.2017.7945451 (cit. on p. 64).

[836] I. Vallés-Pérez, E. Soria-Olivas, M. Martínez-Sober, A. J. Serrano-López, J. Vila-Francés, and J. Gómez-Sanchís. “Empirical study of the modulus as activation function in computer vision applications.” In: *Engineering Applications of Artificial Intelligence* 120 (Apr. 2023), p. 105863. DOI: 10.1016/j.engappai.2023.105863. URL: https://doi.org/10.1016/j.engappai.2023.105863 (cit. on pp. 64, 76, 77, 213).

[837] D. Misra. “Mish: A Self Regularized Non-Monotonic Activation Function.” In: *The 31st British Machine Vision Conference*. BMVC, Sept. 2020. URL: https://www.bmvc2020-conference.com/conference/papers/paper_0928.html (cit. on pp. 64, 65).

[838] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. “Densely Connected Convolutional Networks.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.243. URL: https://doi.org/10.1109/cvpr.2017.243 (cit. on pp. 65, 149, 152, 171, 177).

[839] J. Hu, L. Shen, and G. Sun. “Squeeze-and-Excitation Networks.” In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00745. URL: https://doi.org/10.1109/cvpr.2018.00745 (cit. on p. 65).

[840] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. DOI: 10.48550/ARXIV.2004.10934. URL: https://arxiv.org/abs/2004.10934 (cit. on p. 65).

[841] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. “Scaled-YOLOv4: Scaling Cross Stage Partial Network.” In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.01283. URL: https://doi.org/10.1109/cvpr46437.2021.01283 (cit. on pp. 65, 66).

[842] X. Wang, H. Ren, and A. Wang. “Smish: A Novel Activation Function for Deep Learning Methods.” In: *Electronics* 11.4 (Feb. 2022), p. 540. ISSN: 2079-9292. DOI: 10.3390/electronics11040540. URL: http://dx.doi.org/10.3390/electronics11040540 (cit. on p. 65).

[843] X. Liu and X. Di. “TanhExp: A smooth activation function with high convergence speed for lightweight neural networks.” In: *IET Computer Vision* 15.2 (Feb. 2021), pp. 136–150. ISSN: 1751-9640. DOI: 10.1049/cvi2.12020. URL: http://dx.doi.org/10.1049/cvi2.12020 (cit. on p. 65).

[844] M. A. Mercioni and S. Holban. “TeLU: A New Activation Function for Deep Learning.” In: *2020 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, Nov. 2020. DOI: 10.1109/isetc50328.2020.9301084. URL: http://dx.doi.org/10.1109/ISETC50328.2020.9301084 (cit. on pp. 65, 115).

[845] S. Nag, M. Bhattacharyya, A. Mukherjee, and R. Kundu. “Serf: Towards better training of deep neural networks using log-Softplus ERror activation Function.” In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00529. URL: http://dx.doi.org/10.1109/WACV56688.2023.00529 (cit. on p. 65).

[846] D. Elliott, S. Frank, K. Sima’an, and L. Specia. “Multi30K: Multilingual English-German Image Descriptions.” In: *Proceedings of the 5th Workshop on Vision and Language*. Association for Computational Linguistics, 2016. DOI: 10.18653/v1/w16-3210. URL: http://dx.doi.org/10.18653/v1/W16-3210 (cit. on p. 65).

[847] E. Chai, W. Yu, T. Cui, J. Ren, and S. Ding. “An Efficient Asymmetric Nonlinear Activation Function for Deep Neural Networks.” In: *Symmetry* 14.5 (May 2022), p. 1027. DOI: 10.3390/sym14051027. URL: https://doi.org/10.3390/sym14051027 (cit. on pp. 65, 66).

[848] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. “Focal Loss for Dense Object Detection.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (Feb. 2020), pp. 318–327. DOI: 10.1109/tpami.2018.2858826. URL: https://doi.org/10.1109/tpami.2018.2858826 (cit. on p. 66).

[849] K. Douge, A. Berrahou, Y. T. Alaoui, and M. T. Alaoui. “A Self-gated Activation Function SINSIG Based on the Sine Trigonometric for Neural Network Models.” In: *Machine Learning for Networking*. Springer International Publishing, 2021, pp. 237–244. DOI: 10.1007/978-3-030-70866-5_15. URL: https://doi.org/10.1007/978-3-030-70866-5_15 (cit. on p. 66).

[850] K. He, X. Zhang, S. Ren, and J. Sun. “Identity Mappings in Deep Residual Networks.” In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 630–645. DOI: 10.1007/978-3-319-46493-0_38. URL: https://doi.org/10.1007/978-3-319-46493-0_38 (cit. on p. 66).

[851] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. 2016. DOI: 10.48550/ARXIV.1602.07360. URL: https://arxiv.org/abs/1602.07360 (cit. on p. 66).

[852] X. Zhang, X. Zhou, M. Lin, and J. Sun. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.” In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00716. URL: https://doi.org/10.1109/cvpr.2018.00716 (cit. on p. 66).

[853] B. Ahmed, M. A. Haque, M. A. Iquebal, S. Jaiswal, U. B. Angadi, D. Kumar, and A. Rai. “DeepAProt: Deep learning based abiotic stress protein sequence classification and identification tool in cereals.” In: *Frontiers in Plant Science* 13 (Jan. 2023). ISSN: 1664-462X. DOI: 10.3389/fpls.2022.1008756. URL: http://dx.doi.org/10.3389/fpls.2022.1008756 (cit. on p. 66).

[854] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. “Language Modeling with Gated Convolutional Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 933–941. URL: https://proceedings.mlr.press/v70/dauphin17a.html (cit. on pp. 66, 67).

[855] N. Shazeer. *GLU Variants Improve Transformer*. 2020. DOI: 10.48550/ARXIV.2002.05202. URL: https://arxiv.org/abs/2002.05202 (cit. on pp. 66, 67).

[856] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. “Convolutional Sequence to Sequence Learning.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1243–1252. URL: https://proceedings.mlr.press/v70/gehring17a.html (cit. on p. 66).

[857] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. “Conditional Image Generation with PixelCNN Decoders.” In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805. ISBN: 9781510838819. URL: https://dl.acm.org/doi/10.5555/3157382.3157633 (cit. on p. 66).

[858] J. Bridle. “Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters.” In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf (cit. on p. 67).

[859] T. Pearce, A. Brintrup, and J. Zhu. *Understanding Softmax Confidence and Uncertainty*. 2021. DOI: 10.48550/ARXIV.2106.04972. URL: https://arxiv.org/abs/2106.04972 (cit. on p. 67).

[860] M. Bhuvaneshwari and E. G. M. Kanaga. “Convolutional Neural Network for Addiction Detection using Improved Activation Function.” In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, Apr. 2021. DOI: 10.1109/iccmc51019.2021.9418022. URL: http://dx.doi.org/10.1109/ICCMC51019.2021.9418022 (cit. on pp. 67, 132).

[861] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines.” In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Ed. by J. Fürnkranz and T. Joachims. Omnipress, 2010, pp. 807–814. URL: http://www.cs.toronto.edu/~fritz/absps/reluICML.pdf (cit. on pp. 68, 70, 108, 213).

[862] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Networks.” In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a.html (cit. on pp. 68, 95).

[863] K. Hara, D. Saito, and H. Shouno. "Analysis of function of rectified linear unit used in deep learning." In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015. DOI: 10.1109/ijcnn.2015.7280578. URL: https://doi.org/10.1109/ijcnn.2015.7280578 (cit. on p. 68).

[864] M. Telgarsky. "Neural Networks and Rational Functions." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3387–3393. URL: https://proceedings.mlr.press/v70/telgarsky17a.html (cit. on p. 68).

[865] A. Aleksandrov and K. Völlinger. "Formalizing Piecewise Affine Activation Functions of Neural Networks in Coq." In: *NASA Formal Methods*. Springer Nature Switzerland, 2023, pp. 62–78. ISBN: 9783031331701. DOI: 10.1007/978-3-031-33170-1_4. URL: http://dx.doi.org/10.1007/978-3-031-33170-1_4 (cit. on p. 68).

[866] D. Mishkin, N. Sergievskiy, and J. Matas. "Systematic evaluation of convolution neural network advances on the Imagenet." In: *Computer Vision and Image Understanding* 161 (Aug. 2017), pp. 11–19. DOI: 10.1016/j.cviu.2017.05.007. URL: https://doi.org/10.1016/j.cviu.2017.05.007 (cit. on p. 68).

[867] W. Wang, X. Yang, B. C. Ooi, D. Zhang, and Y. Zhuang. "Effective deep learning-based multi-modal retrieval." In: *The VLDB Journal* 25.1 (July 2015), pp. 79–101. DOI: 10.1007/s00778-015-0391-4. URL: https://doi.org/10.1007/s00778-015-0391-4 (cit. on p. 68).

[868] D. Mishkin and J. Matas. "All you need is a good init." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016. URL: http://arxiv.org/abs/1511.06422 (cit. on pp. 68, 69, 120, 155).

[869] A. L. Maas, A. Y. Hannun, and A. Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2013. URL: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf (cit. on pp. 68, 69, 105, 212).

[870] B. Graham. *Spatially-sparse convolutional neural networks*. 2014. DOI: 10.48550/ARXIV.1409.6070. URL: https://arxiv.org/abs/1409.6070 (cit. on pp. 68, 69, 212).

[871] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.123. URL: https://doi.org/10.1109/iccv.2015.123 (cit. on pp. 68, 105, 120, 124, 214).

[872] B. Xu, N. Wang, T. Chen, and M. Li. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. DOI: 10.48550/ARXIV.1505.00853. URL: https://arxiv.org/abs/1505.00853 (cit. on pp. 68, 69, 213).

[873] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan. "Deep Learning with S-Shaped Rectified Linear Activation Units." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (Feb. 2016). DOI: 10.1609/aaai.v30i1.10287. URL: https://doi.org/10.1609/aaai.v30i1.10287 (cit. on pp. 68, 147, 149, 155, 217).

[874] D. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016. URL: http://arxiv.org/abs/1511.07289 (cit. on pp. 68, 83, 110, 213).

[875] R. Parhi and R. D. Nowak. "The Role of Neural Network Activation Functions." In: *IEEE Signal Processing Letters* 27 (2020), pp. 1779–1783. DOI: 10.1109/lsp.2020.3027517. URL: https://doi.org/10.1109/lsp.2020.3027517 (cit. on p. 69).

[876] B. H. Nayef, S. N. H. S. Abdullah, R. Sulaiman, and Z. A. A. Alyasseri. "Optimized leaky ReLU for handwritten Arabic character recognition using convolution neural networks." In: *Multimedia Tools and Applications* 81.2 (Oct. 2021), pp. 2065–2094. DOI: 10.1007/s11042-021-11593-6. URL: https://doi.org/10.1007/s11042-021-11593-6 (cit. on pp. 69, 212).

[877] M. K. Elakkiya and Dejey. "Novel deep learning models with novel integrated activation functions for autism screening: AutiNet and MinAutiNet." In: *Expert Systems with Applications* 238 (Mar. 2024), p. 122102. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.122102. URL: http://dx.doi.org/10.1016/j.eswa.2023.122102 (cit. on pp. 70, 78).

[878] M. K. Elakkiya and Dejey. "Stacked autoencoder with novel integrated activation functions for the diagnosis of autism spectrum disorder." In: *Neural Computing and Applications* 35.23 (Apr. 2023), pp. 17043–17075. ISSN: 1433-3058. DOI: 10.1007/s00521-023-08565-2. URL: http://dx.doi.org/10.1007/s00521-023-08565-2 (cit. on p. 70).

[879] J. Seo, J. Lee, and K. Kim. "Activation functions of deep neural networks for polar decoding applications." In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, Oct. 2017. DOI: 10.1109/pimrc.2017.8292678. URL: https://doi.org/10.1109/pimrc.2017.8292678 (cit. on pp. 70, 200).

[880] A. Gupta and S. Ahuja. *Parametric Variational Linear Units (PVLUs) in Deep Convolutional Networks*. 2021. DOI: 10.48550/ARXIV.2110.12246. URL: https://arxiv.org/abs/2110.12246 (cit. on pp. 70, 71, 126).

[881] W. Rodrigues. *SineReLU — An Alternative to the ReLU Activation Function*. 2018. URL: https://wilder-rodrigues.medium.com/sinerelu-an-alternative-to-the-relu-activation-function-e46a6199997d (visited on 01/28/2024) (cit. on p. 70).

[882] K. Yamamichi and X.-H. Han. "Lightweight Multi-Scale Context Aggregation Deraining Network With Artifact-Attenuating Pooling and Activation Functions." In: *IEEE Access* 9 (2021), pp. 146948–146958. ISSN: 2169-3536. DOI: 10.1109/access.2021.3122450. URL: http://dx.doi.org/10.1109/ACCESS.2021.3122450 (cit. on p. 71).

[883] J. Cao, Y. Pang, X. Li, and J. Liang. "Randomly translational activation inspired by the input distributions of ReLU." In: *Neurocomputing* 275 (Jan. 2018), pp. 859–868. DOI: 10.1016/j.neucom.2017.09.031. URL: https://doi.org/10.1016/j.neucom.2017.09.031 (cit. on pp. 71, 108, 213, 214).

[884] Y. Liu, J. Zhang, C. Gao, J. Qu, and L. Ji. "Natural-Logarithm-Rectified Activation Function in Convolutional Neural Networks." In: *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. IEEE, Dec. 2019. DOI: 10.1109/iccc47050.2019.9064398. URL: https://doi.org/10.1109/iccc47050.2019.9064398 (cit. on p. 71).

[885] H. Zhao, F. Liu, L. Li, and C. Luo. "A novel softplus linear unit for deep convolutional neural networks." In: *Applied Intelligence* 48.7 (Sept. 2017), pp. 1707–1720. DOI: 10.1007/s10489-017-1028-7. URL: https://doi.org/10.1007/s10489-017-1028-7 (cit. on p. 72).

[886] C. Xu, J. Huang, S.-p. Wang, and A.-q. Hu. "A Novel Parameterized Activation Function in Visual Geometry Group." In: *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*. IEEE, Sept. 2018. DOI: 10.1109/icdsba.2018.00079. URL: https://doi.org/10.1109/icdsba.2018.00079 (cit. on pp. 72, 213).

[887] I. E. Jaafari, A. Ellahyani, and S. Charfi. "Parametric rectified nonlinear unit (PRenu) for convolution neural networks." In: *Signal, Image and Video Processing* 15.2 (July 2020), pp. 241–246. DOI: 10.1007/s11760-020-01746-9. URL: https://doi.org/10.1007/s11760-020-01746-9 (cit. on p. 72).

[888] S. S. Liew, M. Khalil-Hani, and R. Bakhteri. "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems." In: *Neurocomputing* 216 (Dec. 2016), pp. 718–734. DOI: 10.1016/j.neucom.2016.08.037. URL: https://doi.org/10.1016/j.neucom.2016.08.037 (cit. on pp. 73, 75, 76, 97, 149, 213).

[889] E. Pishchik. "Trainable Activations for Image Classification." In: (Jan. 2023). DOI: 10.20944/preprints202301.0463.v1. URL: https://doi.org/10.20944/preprints202301.0463.v1 (cit. on pp. xlvi, 73, 85, 110, 117, 119, 137, 143, 156, 158, 201, 217).

[890] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. 2016. DOI: 10.48550/ARXIV.1602.02830. URL: https://arxiv.org/abs/1602.02830 (cit. on pp. 73, 89).

[891] R. Avenash and P. Viswanath. "Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function." In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019. DOI: 10.5220/0007469604130420. URL: http://dx.doi.org/10.5220/0007469604130420 (cit. on pp. 73, 148, 149).

[892] S. M. Waseem, A. V. Suraj, and S. K. Roy. "Accelerating the Activation Function Selection for Hybrid Deep Neural Networks – FPGA Implementation." In: *2021 IEEE Region 10 Symposium (TENSYMP)*. IEEE, Aug. 2021. DOI: 10.1109/tensymp52854.2021.9551000. URL: https://doi.org/10.1109/tensymp52854.2021.9551000 (cit. on p. 73).

[893] D. Lupu and I. Necoara. "Exact Representation and Efficient Approximations of Linear Model Predictive Control Laws Via Hardtanh Type Deep Neural Networks." In: (2023). DOI: 10.2139/ssrn.4516044. URL: http://dx.doi.org/10.2139/ssrn.4516044 (cit. on p. 73).

[894] Z. Liu, H. Zhang, Z. Su, and X. Zhu. "Adaptive Binarization Method for Binary Neural Network." In: *2021 40th Chinese Control Conference (CCC)*. IEEE, July 2021. DOI: 10.23919/ccc52363.2021.9549344. URL: http://dx.doi.org/10.23919/CCC52363.2021.9549344 (cit. on pp. 73, 111, 202).

[895] H. Kim, J. Park, C. Lee, and J.-J. Kim. "Improving Accuracy of Binary Neural Networks using Unbalanced Activation Distribution." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00777. URL: http://dx.doi.org/10.1109/CVPR46437.2021.00777 (cit. on pp. 74, 202).

[896] A. Howard et al. "Searching for MobileNetV3." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00140. URL: http://dx.doi.org/10.1109/ICCV.2019.00140 (cit. on p. 74).

[897] K. R. Konda, R. Memisevic, and D. Krueger. "Zero-bias autoencoders and the benefits of co-adapting features." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. URL: http://arxiv.org/abs/1402.3337 (cit. on p. 75).

[898] R. Goroshin and Y. LeCun. *Saturating Auto-Encoders*. 2013. DOI: 10.48550/ARXIV.1301.3577. URL: https://arxiv.org/abs/1301.3577 (cit. on p. 75).

[899] N. A. Golilarz and H. Demirel. "Thresholding neural network (TNN) based noise reduction with a new improved thresholding function." In: *Computational Research Progress in Applied Science & Engineering* 3.2 (2017), pp. 81–84. URL: https://jms.procedia.org/archive/CRPASE_169/CRPASE_procedia_2017_3_2_3.pdf (cit. on p. 75).

[900] R. Zhang, J. Yi, H. Tang, J. Xiang, and Y. Ren. "Fault Diagnosis Method of Waterproof Valves in Engineering Mixing Machinery Based on a New Adaptive Feature Extraction Model." In: *Energies* 15.8 (Apr. 2022), p. 2796. ISSN: 1996-1073. DOI: 10.3390/en15082796. URL: http://dx.doi.org/10.3390/en15082796 (cit. on p. 75).

[901] H. Hu. "vReLU Activation Functions for Artificial Neural Networks." In: *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, July 2018. DOI: 10.1109/fskd.2018.8687140. URL: https://doi.org/10.1109/fskd.2018.8687140 (cit. on p. 76).

[902] H. Hu. "Symmetric Rectified Linear Units for Fully Connected Deep Models." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 291–298. ISBN: 9783319992471. DOI: 10.1007/978-3-319-99247-1_26. URL: http://dx.doi.org/10.1007/978-3-319-99247-1_26 (cit. on p. 76).

[903] O. I. Berngardt. *Improving Classification Neural Networks by using Absolute activation function (MNIST/LeNET-5 example)*. 2023. DOI: 10.48550/ARXIV.2304.11758. URL: https://arxiv.org/abs/2304.11758 (cit. on p. 76).

[904] T. Y. Pan, G. Yang, J. Zhao, and J. Ding. "Smoothing piecewise linear activation functions based on mollified square root functions." In: *Mathematical Foundations of Computing* (2023). ISSN: 2577-8838. DOI: 10.3934/mfc.2023032. URL: http://dx.doi.org/10.3934/mfc.2023032 (cit. on pp. xlv, xlvi, lii, 76, 77, 151, 164–167).

[905] X. Bresson, S. Esedoḡlu, P. Vandergheynst, J.-P. Thiran, and S. Osher. "Fast Global Minimization of the Active Contour/Snake Model." In: *Journal of Mathematical Imaging and Vision* 28.2 (July 2007), pp. 151–167. ISSN: 1573-7683. DOI: 10.1007/s10851-007-0002-0. URL: http://dx.doi.org/10.1007/s10851-007-0002-0 (cit. on pp. 76, 164).

[906] A. A. Alkhouly, A. Mohammed, and H. A. Hefny. "Improving the Performance of Deep Neural Networks Using Two Proposed Activation Functions." In: *IEEE Access* 9 (2021), pp. 82249–82271. ISSN: 2169-3536. DOI: 10.1109/access.2021.3085855. URL: http://dx.doi.org/10.1109/ACCESS.2021.3085855 (cit. on pp. 76, 86).

[907] Q. Zhao and L. D. Griffin. *Suppressing the Unusual: towards Robust CNNs using Symmetric Activation Functions*. 2016. DOI: 10.48550/ARXIV.1603.05145. URL: https://arxiv.org/abs/1603.05145 (cit. on p. 76).

[908] C.-H. Shan, X.-R. Guo, and J. Ou. "Deep Leaky Single-peaked Triangle Neural Networks." In: *International Journal of Control, Automation and Systems* 17.10 (Aug. 2019), pp. 2693–2701. ISSN: 2005-4092. DOI: 10.1007/s12555-018-0796-0. URL: http://dx.doi.org/10.1007/s12555-018-0796-0 (cit. on p. 77).

[909] A. S. D. Desabathula and S. Eluri. "A novel Leaky Rectified Triangle Linear Unit based Deep Convolutional Neural Network for facial emotion recognition." In: *Multimedia Tools and Applications* 82.12 (Nov. 2022), pp. 18669–18689. ISSN: 1573-7721. DOI: 10.1007/s11042-022-14186-z. URL: http://dx.doi.org/10.1007/s11042-022-14186-z (cit. on p. 77).

[910] G. Lin and W. Shen. "Research on convolutional neural network based on improved Relu piecewise activation function." In: *Procedia Computer Science* 131 (2018), pp. 977–984. DOI: 10.1016/j.procs.2018.04.239. URL: https://doi.org/10.1016/j.procs.2018.04.239 (cit. on pp. 77, 213).

[911] J. Li, H. Feng, and D.-X. Zhou. "SignReLU neural network and its approximation ability." In: *Journal of Computational and Applied Mathematics* 440 (Apr. 2024), p. 115551. ISSN: 0377-0427. DOI: 10.1016/j.cam.2023.115551. URL: http://dx.doi.org/10.1016/j.cam.2023.115551 (cit. on p. 77).

[912] J. Li, H. Feng, and D.-X. Zhou. *A new activation for neural networks and its approximation*. 2022. URL: https://arxiv.org/abs/2210.10264v1 (cit. on pp. xlvi, 77).

[913] W. Shang, K. Sohn, D. Almeida, and H. Lee. "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2217–2225. URL: https://proceedings.mlr.press/v48/shang16.html (cit. on p. 78).

[914] S. Zagoruyko and N. Komodakis. *DiracNets: Training Very Deep Neural Networks Without Skip-Connections*. 2017. DOI: 10.48550/ARXIV.1706.00388. URL: https://arxiv.org/abs/1706.00388 (cit. on p. 78).

[915] L. Eidnes and A. Nøkland. *Shifting Mean Activation Towards Zero with Bipolar Activation Functions*. 2017. DOI: 10.48550/ARXIV.1709.04054. URL: https://arxiv.org/abs/1709.04054 (cit. on p. 78).

[916] F. Godin, J. Degrave, J. Dambre, and W. D. Neve. "Dual Rectified Linear Units (DReLUs): A replacement for tanh activation functions in Quasi-Recurrent Neural Networks." In: *Pattern Recognition Letters* 116 (Dec. 2018), pp. 8–14. DOI: 10.1016/j.patrec.2018.09.006. URL: https://doi.org/10.1016/j.patrec.2018.09.006 (cit. on pp. 79, 85).

[917] A. Chernodub and D. Nowicki. *Norm-preserving Orthogonal Permutation Linear Unit Activation Functions (OPLU)*. 2016. DOI: 10.48550/ARXIV.1604.02313. URL: https://arxiv.org/abs/1604.02313 (cit. on p. 79).

[918] X. Jiang, Y. Pang, X. Li, J. Pan, and Y. Xie. "Deep neural networks with Elastic Rectified Linear Units for object recognition." In: *Neurocomputing* 275 (Jan. 2018), pp. 1132–1139. DOI: 10.1016/j.neucom.2017.09.056. URL: https://doi.org/10.1016/j.neucom.2017.09.056 (cit. on pp. 79, 113, 213, 215).

[919] Y. Berradi. "Symmetric Power Activation Functions for Deep Neural Networks." In: *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*. ACM, May 2018. DOI: 10.1145/3230905.3230956. URL: https://doi.org/10.1145/3230905.3230956 (cit. on p. 80).

[920] B. Li, S. Tang, and H. Yu. "PowerNet: Efficient Representations of Polynomials and Smooth Functions by Deep Neural Networks with Rectified Power Units." In: (2019). DOI: 10.48550/ARXIV.1909.05136. URL: https://arxiv.org/abs/1909.05136 (cit. on p. 80).

[921] T. J. Heeringa, L. Spek, F. Schwenninger, and C. Brune. *Embeddings between Barron spaces with higher order activation functions*. 2023. DOI: 10.48550/ARXIV.2305.15839. URL: https://arxiv.org/abs/2305.15839 (cit. on p. 80).

[922] D. R. So, W. Mańke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le. *Primer: Searching for Efficient Transformers for Language Modeling*. 2021. DOI: 10.48550/ARXIV.2109.08668. URL: https://arxiv.org/abs/2109.08668 (cit. on p. 80).

[923] S. Saha, N. Nagaraj, A. Mathur, R. Yedida, and S. H. R. "Evolution of novel activation functions in neural network training for astronomy data: habitability classification of exoplanets." In: *The European Physical Journal Special Topics* 229.16 (Nov. 2020), pp. 2629–2738. DOI: 10.1140/epjst/e2020-000098-9. URL: https://doi.org/10.1140/epjst/e2020-000098-9 (cit. on pp. 80, 94).

[924] I. Mediratta, S. Saha, and S. Mathur. "LipARELU: ARELU Networks aided by Lipschitz Acceleration." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2021. DOI: 10.1109/ijcnn52387.2021.9533853. URL: http://dx.doi.org/10.1109/IJCNN52387.2021.9533853 (cit. on p. 80).

[925] K. Nasiri and K. Ghiasi-Shirazi. "PowerLinear Activation Functions with application to the first layer of CNNs." In: *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*. IEEE, Oct. 2021. DOI: 10.1109/iccke54056.2021.9721450. URL: http://dx.doi.org/10.1109/ICCKE54056.2021.9721450 (cit. on p. 81).

[926] S. R. Dubey and S. Chakraborty. "Average biased ReLU based CNN descriptor for improved face retrieval." In: *Multimedia Tools and Applications* 80.15 (Jan. 2021), pp. 23181–23206. DOI: 10.1007/s11042-020-10269-x. URL: https://doi.org/10.1007/s11042-020-10269-x (cit. on pp. 81, 196, 200).

[927] C. Shan, A. Li, and X. Chen. "Deep delay rectified neural networks." In: *The Journal of Supercomputing* 79.1 (July 2022), pp. 880–896. ISSN: 1573-0484. DOI: 10.1007/s11227-022-04704-z. URL: http://dx.doi.org/10.1007/s11227-022-04704-z (cit. on pp. 81, 82, 200).

[928] D. Macêdo, C. Zanchettin, A. L. Oliveira, and T. Ludermir. "Enhancing batch normalized convolutional networks using displaced rectifier linear units: A systematic comparative study." In: *Expert Systems with Applications* 124 (June 2019), pp. 271–281. DOI: 10.1016/j.eswa.2019.01.066. URL: https://doi.org/10.1016/j.eswa.2019.01.066 (cit. on pp. 82, 201).

[929] H. Yang, A. Alsadoon, P. W. C. Prasad, T. Al-Dala'in, T. A. Rashid, A. Maag, and O. H. Alsadoon. "Deep learning neural networks for emotion classification from text: enhanced leaky rectified linear unit activation and weighted loss." In: *Multimedia Tools and Applications* 81.11 (Feb. 2022), pp. 15439–15468. ISSN: 1573-7721. DOI: 10.1007/s11042-022-12629-1. URL: http://dx.doi.org/10.1007/s11042-022-12629-1 (cit. on p. 82).

[930] H. H. Chieng, N. Wahid, O. Pauline, and S. R. K. Perla. "Flatten-T Swish: a thresholded ReLU-Swish-like activation function for deep learning." In: *International Journal of Advances in Intelligent Informatics* 4.2 (July 2018), p. 76. DOI: 10.26555/ijain.v4i2.249. URL: https://doi.org/10.26555/ijain.v4i2.249 (cit. on pp. 82, 201).

[931] O. Sharma. "A Novel Activation Function in Convolutional Neural Network for Image Classification in Deep Learning." In: *Data Science and Analytics*. Springer Singapore, 2020, pp. 120–130. DOI: 10.1007/978-981-15-5827-6_10. URL: https://doi.org/10.1007/978-981-15-5827-6_10 (cit. on p. 83).

[932] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, and S. Peng. "Rectified Exponential Units for Convolutional Neural Networks." In: *IEEE Access* 7 (2019), pp. 101633–101640. DOI: 10.1109/access.2019.2928442. URL: https://doi.org/10.1109/access.2019.2928442 (cit. on pp. 83, 108, 214).

[933] M.-I. Georgescu, R. T. Ionescu, N.-C. Ristea, and N. Sebe. "Nonlinear neurons with human-like apical dendrite activations." In: *Applied Intelligence* 53.21 (Aug. 2023), pp. 25984–26007. ISSN: 1573-7497. DOI: 10.1007/s10489-023-04921-w. URL: http://dx.doi.org/10.1007/s10489-023-04921-w (cit. on pp. 83, 84).

[934] M. N. Qureshi and M. Sarosh Umar. "SaRa: A Novel Activation Function with Application to Melanoma Image Classification." In: *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*. IEEE, Dec. 2022. DOI: 10.1109/icacrs55517.2022.10029161. URL: http://dx.doi.org/10.1109/ICACRS55517.2022.10029161 (cit. on p. 84).

[935] H. Fu, H. Shi, Y. Xu, and J. Shao. "Research on Gas Outburst Prediction Model Based on Multiple Strategy Fusion Improved Snake Optimization Algorithm With Temporal Convolutional Network." In: *IEEE Access* 10 (2022), pp. 117973–117984. ISSN: 2169-3536. DOI: 10.1109/access.2022.3220765. URL: http://dx.doi.org/10.1109/ACCESS.2022.3220765 (cit. on p. 85).

[936] Z. Hu, H. Huang, Q. Ran, and M. Yuan. "Improving Convolutional Neural Network Expression via Difference Exponentially Linear Units." In: *Journal of Physics: Conference Series* 1651.1 (Nov. 2020), p. 012163. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1651/1/012163. URL: http://dx.doi.org/10.1088/1742-6596/1651/1/012163 (cit. on p. 85).

[937] H.-S. Feng and C.-H. Yang. "PolyLU: A Simple and Robust Polynomial-Based Linear Unit Activation Function for Deep Learning." In: *IEEE Access* 11 (2023), pp. 101347–101358. ISSN: 2169-3536. DOI: 10.1109/access.2023.3315308. URL: http://dx.doi.org/10.1109/ACCESS.2023.3315308 (cit. on p. 85).

[938] Kaggle. *Dogs vs. Cats*. 2013. URL: https://www.kaggle.com/c/dogs-vs-cats/data (visited on 01/12/2024) (cit. on p. 85).

[939] J. Elson, J. R. Douceur, J. Howell, and J. Saul. "Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization." In: *Proceedings of the 14th ACM conference on Computer and communications security*. CCS07. ACM, Oct. 2007. DOI: 10.1145/1315245.1315291. URL: http://dx.doi.org/10.1145/1315245.1315291 (cit. on p. 85).

[940] B. Duan, Y. Yang, and X. Dai. "Activation by Switch Unit of Opposite First Powers." In: *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*. IEEE, Dec. 2022. DOI: 10.1109/iccc56324.2022.10065932. URL: http://dx.doi.org/10.1109/ICCC56324.2022.10065932 (cit. on pp. 85, 125).

[941] Y. Li, P. L. K. Ding, and B. Li. *Training Neural Networks by Using Power Linear Units (PoLUs)*. 2018. DOI: 10.48550/ARXIV.1802.00212. URL: https://arxiv.org/abs/1802.00212 (cit. on p. 86).

[942] M. Zhu, W. Min, Q. Wang, S. Zou, and X. Chen. "PFLU and FPFLU: Two novel non-monotonic activation functions in convolutional neural networks." In: *Neurocomputing* 429 (Mar. 2021), pp. 110–117. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.11.068. URL: http://dx.doi.org/10.1016/j.neucom.2020.11.068 (cit. on p. 86).

[943] C. Zhang, Y. Xu, and Z. Sheng. "Elastic Adaptively Parametric Compounded Units for Convolutional Neural Network." In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* 27.4 (July 2023), pp. 576–584. ISSN: 1343-0130. DOI: 10.20965/jaciii.2023.p0576. URL: http://dx.doi.org/10.20965/jaciii.2023.p0576 (cit. on pp. 86, 87).

[944] M. Basirat and P. M. Roth. “L*ReLU: Piece-wise Linear Activation Functions for Deep Fine-grained Visual Categorization.” In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020. DOI: 10.1109/wacv45572.2020.9093485. URL: https://doi.org/10.1109/wacv45572.2020.9093485 (cit. on p. 87).

[945] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. “Self-Normalizing Neural Networks.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/5d44ee6f2c3f71b73125876103c8f6c4-Paper.pdf (cit. on pp. 87, 117, 213).

[946] Z. Chen, Z. WeiQin, L. Deng, G. Li, and Y. Xie. *Redefining The Self-Normalization Property*. 2021. URL: https://openreview.net/forum?id=gfwf0skyzSx (cit. on pp. 88, 214).

[947] G. Zhang and H. Li. *Effectiveness of Scaled Exponentially-Regularized Linear Units (SERLUs)*. 2018. DOI: 10.48550/ARXIV.1807.10117. URL: https://arxiv.org/abs/1807.10117 (cit. on p. 88).

[948] S. Hermawan and R. Mandala. “Improving Accuracy using The ASERLU layer in CNN-BiLSTM Architecture on Sentiment Analysis.” In: *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* 5.5 (Oct. 2021), pp. 1001–1007. ISSN: 2580-0760. DOI: 10.29207/resti.v5i5.3534. URL: http://dx.doi.org/10.29207/resti.v5i5.3534 (cit. on p. 88).

[949] S. Kiliçarslan and M. Celik. “RSigELU: A nonlinear activation function for deep neural networks.” In: *Expert Systems with Applications* 174 (July 2021), p. 114805. DOI: 10.1016/j.eswa.2021.114805. URL: https://doi.org/10.1016/j.eswa.2021.114805 (cit. on pp. 88, 89, 212, 214).

[950] H. Xiao, K. Rasul, and R. Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. DOI: 10.48550/ARXIV.1708.07747. URL: https://arxiv.org/abs/1708.07747 (cit. on pp. 89, 140, 160, 289).

[951] S. Kiliçarslan. “A novel nonlinear hybrid HardSReLUE activation function in transfer learning architectures for hemorrhage classification.” In: *Multimedia Tools and Applications* 82.4 (Dec. 2022), pp. 6345–6365. ISSN: 1573-7721. DOI: 10.1007/s11042-022-14313-w. URL: http://dx.doi.org/10.1007/s11042-022-14313-w (cit. on p. 89).

[952] Y. Wang, Y. Li, Y. Song, and X. Rong. “The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition.” In: *Applied Sciences* 10.5 (Mar. 2020), p. 1897. ISSN: 2076-3417. DOI: 10.3390/app10051897. URL: http://dx.doi.org/10.3390/app10051897 (cit. on p. 90).

[953] A. Wuraola, N. Patel, and S. K. Nguang. “Efficient activation functions for embedded inference engines.” In: *Neurocomputing* 442 (June 2021), pp. 73–88. DOI: 10.1016/j.neucom.2021.02.030. URL: https://doi.org/10.1016/j.neucom.2021.02.030 (cit. on pp. 90–92).

[954] A. Wuraola and N. Patel. “SQNL: A New Computationally Efficient Activation Function.” In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2018. DOI: 10.1109/ijcnn.2018.8489043. URL: https://doi.org/10.1109/ijcnn.2018.8489043 (cit. on pp. 90–92).

[955] J. Bilski and A. I. Galushkin. “A New Proposition of the Activation Function for Significant Improvement of Neural Networks Performance.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 35–45. ISBN: 9783319393780. DOI: 10.1007/978-3-319-39378-0_4. URL: http://dx.doi.org/10.1007/978-3-319-39378-0_4 (cit. on pp. 90, 92, 217).

[956] D. Dua and E. Karra Taniskidou. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (cit. on pp. 91, 171).

[957] B. Carlile, G. Delamarter, P. Kinney, A. Marti, and B. Whitney. *Improving Deep Learning by Inverse Square Root Linear Units (ISRLUs)*. 2017. DOI: 10.48550/ARXIV.1710.09967. URL: https://arxiv.org/abs/1710.09967 (cit. on pp. 92, 93).

[958] X. Yang, Y. Chen, and H. Liang. “Square Root Based Activation Function in Neural Networks.” In: *2018 International Conference on Audio, Language and Image Processing (ICALIP)*. IEEE, July 2018. DOI: 10.1109/icalip.2018.8455590. URL: https://doi.org/10.1109/icalip.2018.8455590 (cit. on p. 93).

[959] M. Khachumov, Y. Emelyanova, and V. Khachumov. “Parabola-Based Artificial Neural Network Activation Functions.” In: *2023 International Russian Automation Conference (RusAutoCon)*. IEEE, Sept. 2023. DOI: 10.1109/rusautocon58002.2023.10272855. URL: http://dx.doi.org/10.1109/RusAutoCon58002.2023.10272855 (cit. on p. 93).

[960] Mate Labs. *Secret Sauce behind the beauty of Deep Learning: Beginners guide to Activation Functions*. 2017. URL: https://towardsdatascience.com/secret-sauce-behind-the-beauty-of-deep-learning-beginners-guide-to-activation-functions-a8e23a57d046 (visited on 01/28/2024) (cit. on p. 94).

[961] A. Mishra, P. Chandra, and U. Ghose. “A New Activation Function Validated on Function Approximation Tasks.” In: *Proceedings of 3rd International Conference on Computing Informatics and Networks*. Springer Singapore, 2021, pp. 311–321. ISBN: 9789811597121. DOI: 10.1007/978-981-15-9712-1_26. URL: http://dx.doi.org/10.1007/978-981-15-9712-1_26 (cit. on p. 94).

[962] S. Saha, A. Mathur, K. Bora, S. Basak, and S. Agrawal. “A New Activation Function for Artificial Neural Net Based Habitability Classification.” In: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Sept. 2018. DOI: 10.1109/icacci.2018.8554460. URL: http://dx.doi.org/10.1109/ICACCI.2018.8554460 (cit. on p. 94).

[963] J. Bilski. “The Backpropagation learning with logarithmic transfer function.” In: *Proc. 5th Conf. On Neural Networks and Soft Computing*. 2000, pp. 71–76 (cit. on p. 94).

[964] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. *Mastering Diverse Domains through World Models*. 2023. DOI: 10.48550/ARXIV.2301.04104. URL: https://arxiv.org/abs/2301.04104 (cit. on p. 94).

[965] J. Kiseľák, Y. Lu, J. Švihra, P. Szépe, and M. Stehlík. “"SPOCU": scaled polynomial constant unit activation function.” In: *Neural Computing and Applications* 33.8 (July 2020), pp. 3385–3401. ISSN: 1433-3058. DOI: 10.1007/s00521-020-05182-1. URL: http://dx.doi.org/10.1007/s00521-020-05182-1 (cit. on p. 95).

[966] J. Kiseľák, Y. Lu, J. Švihra, P. Szépe, and M. Stehlík. “Correction to: "SPOCU": scaled polynomial constant unit activation function.” In: *Neural Computing and Applications* 33.5 (Nov. 2020), pp. 1749–1750. ISSN: 1433-3058. DOI: 10.1007/s00521-020-05412-6. URL: http://dx.doi.org/10.1007/s00521-020-05412-6 (cit. on p. 95).

[967] S.-Y. Hwang and J.-J. Kim. “A Universal Activation Function for Deep Learning.” In: *Computers, Materials & Continua* 75.2 (2023), pp. 3553–3569. ISSN: 1546-2226. DOI: 10.32604/cmc.2023.037028. URL: http://dx.doi.org/10.32604/cmc.2023.037028 (cit. on p. 95).

[968] B. Yuen, M. T. Hoang, X. Dong, and T. Lu. “Universal activation function for machine learning.” In: *Scientific Reports* 11.1 (Sept. 2021). ISSN: 2045-2322. DOI: 10.1038/s41598-021-96723-8. URL: http://dx.doi.org/10.1038/s41598-021-96723-8 (cit. on pp. 95, 143).

[969] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. “Incorporating Second-Order Functional Knowledge for Better Option Pricing.” In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/44968aece94f667e4095002d140b5896-Paper.pdf (cit. on p. 95).

[970] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li. “Improving deep neural networks using softplus units.” In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015. DOI: 10.1109/ijcnn.2015.7280459. URL: https://doi.org/10.1109/ijcnn.2015.7280459 (cit. on p. 95).

[971] Q. Liu and S. Furber. “Noisy Softplus: A Biology Inspired Activation Function.” In: *Neural Information Processing*. Springer International Publishing, 2016, pp. 405–412. DOI: 10.1007/978-3-319-46681-1_49. URL: https://doi.org/10.1007/978-3-319-46681-1_49 (cit. on p. 95).

[972] K. Sun, J. Yu, L. Zhang, and Z. Dong. “A Convolutional Neural Network Model Based on Improved Softplus Activation Function.” In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, July 2019, pp. 1326–1335. DOI: 10.1007/978-3-030-25128-4_164. URL: https://doi.org/10.1007/978-3-030-25128-4_164 (cit. on pp. 96, 194, 201).

[973] A. Ciuparu, A. Nagy-Dăbâcan, and R. C. Mureşan. “Soft++, a multi-parametric non-saturating non-linearity that improves convergence in deep neural architectures.” In: *Neurocomputing* 384 (Apr. 2020), pp. 376–388. DOI: 10.1016/j.neucom.2019.12.014. URL: https://doi.org/10.1016/j.neucom.2019.12.014 (cit. on pp. 96, 214).

[974] Y. Chen, Y. Mai, J. Xiao, and L. Zhang. “Improving the Antinoise Ability of DNNs via a Bio-Inspired Noise Adaptive Activation Function Rand Softplus.” In: *Neural Computation* 31.6 (June 2019), pp. 1215–1233. DOI: 10.1162/neco_a_01192. URL: https://doi.org/10.1162/neco_a_01192 (cit. on p. 96).

[975] G. S. da S. Gomes and T. B. Ludermir. “Optimization of the weights and asymmetric activation function family of neural network for time series forecasting.” In: *Expert Systems with Applications* 40.16 (Nov. 2013), pp. 6438–6446. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2013.05.053. URL: http://dx.doi.org/10.1016/j.eswa.2013.05.053 (cit. on p. 96).

[976] F. J. Aranda-Ordaz. “On two families of transformations to additivity for binary response data.” In: *Biometrika* 68.2 (1981), pp. 357–363. ISSN: 1464-3510. DOI: 10.1093/biomet/68.2.357. URL: http://dx.doi.org/10.1093/biomet/68.2.357 (cit. on p. 96).

[977] J.-C. Li, W. W. Y. Ng, D. S. Yeung, and P. P. K. Chan. “Bi-firing deep neural networks.” In: *International Journal of Machine Learning and Cybernetics* 5.1 (Sept. 2013), pp. 73–83. DOI: 10.1007/s13042-013-0198-9. URL: https://doi.org/10.1007/s13042-013-0198-9 (cit. on p. 97).

[978] P. Liu, Z. Zeng, and J. Wang. “Multistability analysis of a general class of recurrent neural networks with non-monotonic activation functions and time-varying delays.” In: *Neural Networks* 79 (July 2016), pp. 117–127. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2016.03.010. URL: http://dx.doi.org/10.1016/j.neunet.2016.03.010 (cit. on pp. 97, 98).

[979] L. Parisi. *m-arcsinh: An Efficient and Reliable Function for SVM and MLP in scikit-learn*. 2020. DOI: 10.48550/ARXIV.2009.07530. URL: https://arxiv.org/abs/2009.07530 (cit. on p. 98).

[980] L. Parisi, R. Ma, N. RaviChandran, and M. Lanzillotta. “hyper-sinh: An accurate and reliable function from shallow to deep learning in TensorFlow and Keras.” In: *Machine Learning with Applications* 6 (Dec. 2021), p. 100112. ISSN: 2666-8270. DOI: 10.1016/j.mlwa.2021.100112. URL: http://dx.doi.org/10.1016/j.mlwa.2021.100112 (cit. on p. 98).

[981] L. Parisi, A. Zaernia, R. Ma, and M. Youseffi. “Hyper-sinh-Convolutional Neural Network for Early Detection of Parkinson’s Disease from Spiral Drawings.” In: *WSEAS TRANSACTIONS ON COMPUTER RESEARCH* 9 (Mar. 2021), pp. 1–7. ISSN: 1991-8755. DOI: 10.37394/232018.2021.9.1. URL: http://dx.doi.org/10.37394/232018.2021.9.1 (cit. on p. 98).

[982] K. Hara and K. Nakayamma. “Comparison of activation functions in multilayer neural network for pattern classification.” In: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN’94)*. ICNN-94. IEEE, 1994. DOI: 10.1109/icnn.1994.374710. URL: http://dx.doi.org/10.1109/ICNN.1994.374710 (cit. on pp. 99–101).

[983] M. S. Gashler and S. C. Ashmore. “Training Deep Fourier Neural Networks to Fit Time-Series Data.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 48–55. ISBN: 9783319093307. DOI: 10.1007/978-3-319-09330-7_7. URL: http://dx.doi.org/10.1007/978-3-319-09330-7_7 (cit. on p. 99).

[984] Y. Zhang, L. Qu, J. Liu, D. Guo, and M. Li. “Sine neural network (SNN) with double-stage weights and structure determination (DS-WASD).” In: *Soft Computing* 20.1 (Oct. 2014), pp. 211–221. ISSN: 1433-7479. DOI: 10.1007/s00500-014-1491-6. URL: http://dx.doi.org/10.1007/s00500-014-1491-6 (cit. on p. 99).

[985] J. Sopena. “Neural networks with periodic and monotonic activation functions: a comparative study in classification problems.” In: *9th International Conference on Artificial Neural Networks: ICANN ’99*. IEE, 1999. DOI: 10.1049/cp:19991129. URL: http://dx.doi.org/10.1049/cp:19991129 (cit. on p. 99).

[986] S. A. Faroughi, R. Soltanmohammadi, P. Datta, S. K. Mahjour, and S. Faroughi. “Physics-Informed Neural Networks with Periodic Activation Functions for Solute Transport in Heterogeneous Porous Media.” In: *Mathematics* 12.1 (Dec. 2023), p. 63. ISSN: 2227-7390. DOI: 10.3390/math12010063. URL: http://dx.doi.org/10.3390/math12010063 (cit. on p. 99).

[987] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. “Implicit Neural Representations with Periodic Activation Functions.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf (cit. on p. 99).

[988] W. X. Cheng, P. Suganthan, and R. Katuwal. “Time series classification using diversified Ensemble Deep Random Vector Functional Link and Resnet features.” In: *Applied Soft Computing* 112 (Nov. 2021), p. 107826. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2021.107826. URL: http://dx.doi.org/10.1016/j.asoc.2021.107826 (cit. on pp. 99, 174).

[989] A. Daskin. “A Simple Quantum Neural Net with a Periodic Activation Function.” In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2018. DOI: 10.1109/smc.2018.00491. URL: http://dx.doi.org/10.1109/SMC.2018.00491 (cit. on p. 99).

[990] K.-H. Chan, S.-K. Im, W. Ke, and N.-L. Lei. “SinP[N]: A Fast Convergence Activation Function for Convolutional Neural Networks.” In: *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, Dec. 2018. DOI: 10.1109/ucc-companion.2018.00082. URL: http://dx.doi.org/10.1109/UCC-Companion.2018.00082 (cit. on p. 99).

[991] M. M. Noel, A. L, A. Trivedi, and P. Dutta. *Growing Cosine Unit: A Novel Oscillatory Activation Function That Can Speedup Training and Reduce Parameters in Convolutional Neural Networks*. 2021. DOI: 10.48550/ARXIV.2108.12943. URL: https://arxiv.org/abs/2108.12943 (cit. on p. 99).

[992] J. Sharma. *Evaluating CNN with Oscillatory Activation Function*. 2022. DOI: 10.48550/ARXIV.2211.06878. URL: https://arxiv.org/abs/2211.06878 (cit. on p. 99).

[993] P. Sharma, A. R. Sahoo, S. Sinha, and S. Bharadwaj. “NFT artwork generation using oscillatory activation functions in GANs.” In: (Mar. 2022). DOI: 10.31224/2225. URL: http://dx.doi.org/10.31224/2225 (cit. on p. 99).

[994] J. U. Rahman, F. Makhdoom, and D. Lu. *Amplifying Sine Unit: An Oscillatory Activation Function for Deep Neural Networks to Recover Nonlinear Oscillations Efficiently*. 2023. DOI: 10.48550/ARXIV.2304.09759. URL: https://arxiv.org/abs/2304.09759 (cit. on p. 100).

[995] J. U. Rahman, F. Makhdoom, and D. Lu. *ASU-CNN: An Efficient Deep Architecture for Image Classification and Feature Visualizations*. 2023. DOI: 10.48550/ARXIV.2305.19146. URL: https://arxiv.org/abs/2305.19146 (cit. on p. 100).

[996] M. Ö. Efe. "Novel Neuronal Activation Functions for Feedforward Neural Networks." In: *Neural Processing Letters* 28.2 (Aug. 2008), pp. 63–79. ISSN: 1573-773X. DOI: 10.1007/s11063-008-9082-0. URL: http://dx.doi.org/10.1007/s11063-008-9082-0 (cit. on pp. 100, 101, 152).

[997] M. Gustineli. *A survey on recently proposed activation functions for Deep Learning*. 2022. DOI: 10.48550/ARXIV.2204.02921. URL: https://arxiv.org/abs/2204.02921 (cit. on pp. 100–102).

[998] H. Abdel-Nabi, G. Al-Naymat, M. Z. Ali, and A. Awajan. "HcLSH: A Novel Non-Linear Monotonic Activation Function for Deep Learning Methods." In: *IEEE Access* 11 (2023), pp. 47794–47815. ISSN: 2169-3536. DOI: 10.1109/access.2023.3276298. URL: http://dx.doi.org/10.1109/ACCESS.2023.3276298 (cit. on p. 100).

[999] T. Kalaiselvi, S. T. Padmapriya, K. Somasundaram, and S. Praveenkumar. "E-Tanh: a novel activation function for image processing neural network models." In: *Neural Computing and Applications* 34.19 (June 2022), pp. 16563–16575. ISSN: 1433-3058. DOI: 10.1007/s00521-022-07245-x. URL: http://dx.doi.org/10.1007/s00521-022-07245-x (cit. on pp. 101, 200).

[1000] M. Afshari Nia, F. Panahi, and M. Ehteram. "Convolutional Neural Network- ANN- E (Tanh): A New Deep Learning Model for Predicting Rainfall." In: *Water Resources Management* 37.4 (Feb. 2023), pp. 1785–1810. ISSN: 1573-1650. DOI: 10.1007/s11269-023-03454-8. URL: http://dx.doi.org/10.1007/s11269-023-03454-8 (cit. on p. 101).

[1001] G. Chen, Q. Wang, X. Li, and Y. Zhang. "Target detection based on a new triple activation function." In: *Systems Science & Control Engineering* 10.1 (June 2022), pp. 629–635. ISSN: 2164-2583. DOI: 10.1080/21642583.2022.2091060. URL: http://dx.doi.org/10.1080/21642583.2022.2091060 (cit. on p. 101).

[1002] Z. Wang, H. Liu, F. Liu, and D. Gao. "Why KDAC? A general activation function for knowledge discovery." In: *Neurocomputing* 501 (Aug. 2022), pp. 343–358. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.06.019. URL: http://dx.doi.org/10.1016/j.neucom.2022.06.019 (cit. on p. 102).

[1003] C. Xiao, P. Zhong, and C. Zheng. "Enhancing Adversarial Defense by k-Winners-Take-All." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=Skgvy64tvr (cit. on p. 103).

[1004] F. Kayim and A. Yilmaz. "Time Series Forecasting With Volatility Activation Function." In: *IEEE Access* 10 (2022), pp. 104000–104010. ISSN: 2169-3536. DOI: 10.1109/access.2022.3211312. URL: http://dx.doi.org/10.1109/ACCESS.2022.3211312 (cit. on p. 103).

[1005] N. Xin, J. Su, and M. M. Hasan. "Multivariate Time Series Spatial Extreme Clustering with Voformer-Ec Neural Networks." In: (2023). DOI: 10.2139/ssrn.4502409. URL: http://dx.doi.org/10.2139/ssrn.4502409 (cit. on p. 103).

[1006] S. Reid and K. Ferens. "A Hybrid Chaotic Activation Function for Artificial Neural Networks." In: *Advances in Artificial Intelligence and Applied Cognitive Computing*. Springer International Publishing, 2021, pp. 1097–1105. ISBN: 9783030702960. DOI: 10.1007/978-3-030-70296-0_87. URL: http://dx.doi.org/10.1007/978-3-030-70296-0_87 (cit. on pp. 103, 104).

[1007] A. N. M. E. Kabir, A. F. M. N. uddin, M. Asaduzzaman, M. F. Hasan, M. I. Hasan, and M. Shahjahan. "Fusion of Chaotic Activation Functions in training neural network." In: *2012 7th International Conference on Electrical and Computer Engineering*. IEEE, Dec. 2012. DOI: 10.1109/icece.2012.6471592. URL: http://dx.doi.org/10.1109/ICECE.2012.6471592 (cit. on p. 104).

[1008] H. Abbasi, M. Yaghoobi, M. Teshnehlab, and A. Sharifi. "Cascade chaotic neural network (CCNN): a new model." In: *Neural Computing and Applications* 34.11 (Jan. 2022), pp. 8897–8917. ISSN: 1433-3058. DOI: 10.1007/s00521-022-06912-3. URL: http://dx.doi.org/10.1007/s00521-022-06912-3 (cit. on p. 104).

[1009] Y. Wu, M. Zhao, and X. Ding. "Beyond weights adaptation: a new neuron model with trainable activation function and its supervised learning." In: *Proceedings of International Conference on Neural Networks (ICNN'97)*. ICNN-97. IEEE, 1997. DOI: 10.1109/icnn.1997.616194. URL: http://dx.doi.org/10.1109/ICNN.1997.616194 (cit. on pp. 105, 156).

[1010] Y. Wu and M. Zhao. "A neuron model with trainable activation function (TAF) and its MFNN supervised learning." In: *Science in China Series F Information Sciences* 44.5 (Oct. 2001), pp. 366–375. ISSN: 1862-2836. DOI: 10.1007/bf02714739. URL: http://dx.doi.org/10.1007/BF02714739 (cit. on p. 105).

[1011] S. Flennerhag et al. "Breaking the Activation Function Bottleneck through Adaptive Parameterization." In: *CoRR* abs/1805.08574 (2018). arXiv: 1805.08574. URL: http://arxiv.org/abs/1805.08574 (cit. on p. 105).

[1012] L. Vecci et al. "Learning and Approximation Capabilities of Adaptive Spline Activation Function Neural Networks." In: *Neural Networks* 11.2 (1998), pp. 259–270. DOI: 10.1016/S0893-6080(97)00118-4. URL: https://doi.org/10.1016/S0893-6080(97)00118-4 (cit. on pp. 105, 162).

[1013] M. Dushkoff and R. Ptucha. "Adaptive Activation Functions for Deep Networks." In: *Electronic Imaging* 2016.19 (Feb. 2016), pp. 1–5. DOI: 10.2352/issn.2470-1173.2016.19.coimg-149. URL: https://doi.org/10.2352/issn.2470-1173.2016.19.coimg-149 (cit. on pp. 105, 158, 219).

[1014] L. Hou et al. "ConvNets with Smooth Adaptive Activation Functions for Regression." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Singh and J. Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 430–439. URL: http://proceedings.mlr.press/v54/hou17a.html (cit. on pp. 105, 146, 217).

[1015] S. Scardapane et al. "Learning Activation Functions from Data Using Cubic Spline Interpolation." In: *Neural Advances in Processing Nonlinear Dynamic Signals*. Springer International Publishing, July 2018, pp. 73–83. DOI: 10.1007/978-3-319-95098-3_7. URL: https://doi.org/10.1007/978-3-319-95098-3_7 (cit. on pp. 105, 162).

[1016] Y.-D. Zhang, C. Pan, J. Sun, and C. Tang. "Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU." In: *Journal of Computational Science* 28 (Sept. 2018), pp. 1–10. DOI: 10.1016/j.jocs.2018.07.003. URL: https://doi.org/10.1016/j.jocs.2018.07.003 (cit. on p. 105).

[1017] F. Leofante, P. Henriksen, and A. Lomuscio. "Verification-friendly Networks: the Case for Parametric ReLUs." In: *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, June 2023. DOI: 10.1109/ijcnn54540.2023.10191169. URL: http://dx.doi.org/10.1109/IJCNN54540.2023.10191169 (cit. on p. 105).

[1018] S. Dai, S. Mahloujifar, and P. Mittal. "Parameterizing Activation Functions for Adversarial Robustness." In: *2022 IEEE Security and Privacy Workshops (SPW)*. IEEE, May 2022. DOI: 10.1109/spw54247.2022.9833884. URL: http://dx.doi.org/10.1109/SPW54247.2022.9833884 (cit. on pp. 106, 117, 129, 200).

[1019] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi. "A Comprehensive Overhaul of Feature Distillation." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00201. URL: http://dx.doi.org/10.1109/ICCV.2019.00201 (cit. on p. 106).

[1020] N. Ma, X. Zhang, and J. Sun. "Funnel Activation for Visual Recognition." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 351–368. ISBN: 9783030586218. DOI: 10.1007/978-3-030-58621-8_21. URL: http://dx.doi.org/10.1007/978-3-030-58621-8_21 (cit. on p. 106).

[1021] J. Gao, J. Yi, and Y. L. Murphey. "Self-Supervised Learning for Driving Maneuver Prediction from Multivariate Temporal Signals." In: *2020 Chinese Automation Congress (CAC)*. IEEE, Nov. 2020. DOI: 10.1109/cac51589.2020.9327088. URL: http://dx.doi.org/10.1109/CAC51589.2020.9327088 (cit. on p. 106).

[1022] S. Bogoi and A. Udrea. "A Lightweight Deep Learning Approach for Liver Segmentation." In: *Mathematics* 11.1 (Dec. 2022), p. 95. ISSN: 2227-7390. DOI: 10.3390/math11010095. URL: http://dx.doi.org/10.3390/math11010095 (cit. on p. 106).

[1023] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng. "ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 143–159. ISBN: 9783030585686. DOI: 10.1007/978-3-030-58568-6_9. URL: http://dx.doi.org/10.1007/978-3-030-58568-6_9 (cit. on pp. 107, 135, 201).

[1024] K. Biswas, S. Kumar, S. Banerjee, and A. Kumar Pandey. "SAU: Smooth Activation Function Using Convolution with Approximate Identities." In: *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 313–329. ISBN: 9783031198038. DOI: 10.1007/978-3-031-19803-8_19. URL: http://dx.doi.org/10.1007/978-3-031-19803-8_19 (cit. on p. 107).

[1025] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "Smooth Maximum Unit: Smooth Activation Function for Deep Networks using Smoothing Maximum Technique." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.00087. URL: http://dx.doi.org/10.1109/CVPR52688.2022.00087 (cit. on pp. 107, 165).

[1026] A. Maniatopoulos and N. Mitianoudis. "Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function." In: *Information* 12.12 (Dec. 2021), p. 513. ISSN: 2078-2489. DOI: 10.3390/info12120513. URL: http://dx.doi.org/10.3390/info12120513 (cit. on pp. 107, 201).

[1027] D. Kim, J. Kim, and J. Kim. “Elastic exponential linear units for convolutional neural networks.” In: *Neurocomputing* 406 (Sept. 2020), pp. 253–266. DOI: 10.1016/j.neucom.2020.03.051. URL: https://doi.org/10.1016/j.neucom.2020.03.051 (cit. on pp. 108, 125, 217).

[1028] K. Shridhar, J. Lee, H. Hayashi, P. Mehta, B. K. Iwana, S. Kang, S. Uchida, S. Ahmed, and A. Dengel. *ProbAct: A Probabilistic Activation Function for Deep Neural Networks*. 2019. DOI: 10.48550/ARXIV.1905.10761. URL: https://arxiv.org/abs/1905.10761 (cit. on p. 108).

[1029] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio. “Noisy Activation Functions.” In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 3059–3068. URL: https://proceedings.mlr.press/v48/gulcehre16.html (cit. on pp. 108, 138, 158).

[1030] S. Reid, K. Ferens, and W. Kinsner. “Adaptive Chaotic Injection to Reduce Overfitting in Artificial Neural Networks.” In: *2022 IEEE 21st International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*. IEEE, Dec. 2022. DOI: 10.1109/iccicc57084.2022.10101500. URL: http://dx.doi.org/10.1109/ICCICC57084.2022.10101500 (cit. on pp. 108, 109).

[1031] Y. Jiang, J. Xie, and D. Zhang. “An Adaptive Offset Activation Function for CNN Image Classification Tasks.” In: *Electronics* 11.22 (Nov. 2022), p. 3799. DOI: 10.3390/electronics11223799. URL: https://doi.org/10.3390/electronics11223799 (cit. on pp. 109, 200).

[1032] X. Hu, P. Niu, J. Wang, and X. Zhang. “A Dynamic Rectified Linear Activation Units.” In: *IEEE Access* 7 (2019), pp. 180409–180416. DOI: 10.1109/access.2019.2959036. URL: https://doi.org/10.1109/access.2019.2959036 (cit. on pp. 109, 110, 169, 212).

[1033] J. Si, S. L. Harris, and E. Yfantis. “A Dynamic ReLU on Neural Network.” In: *2018 IEEE 13th Dallas Circuits and Systems Conference (DCAS)*. IEEE, Nov. 2018. DOI: 10.1109/dcas.2018.8620116. URL: https://doi.org/10.1109/dcas.2018.8620116 (cit. on pp. 109, 110, 169, 200).

[1034] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu. *Dynamic ReLU*. 2020. DOI: 10.48550/ARXIV.2003.10027. URL: https://arxiv.org/abs/2003.10027 (cit. on pp. 109, 110, 169, 218).

[1035] S. Qiu, X. Xu, and B. Cai. “FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks.” In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8546022. URL: https://doi.org/10.1109/icpr.2018.8546022 (cit. on pp. 110, 201).

[1036] W. Yu, C. Si, P. Zhou, M. Luo, Y. Zhou, J. Feng, S. Yan, and X. Wang. “MetaFormer Baselines for Vision.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.2 (Feb. 2024), pp. 896–912. ISSN: 1939-3539. DOI: 10.1109/tpami.2023.3329173. URL: http://dx.doi.org/10.1109/TPAMI.2023.3329173 (cit. on pp. 110, 111).

[1037] D. Chen, J. Li, and K. Xu. *AReLU: Attention-based Rectified Linear Unit*. 2020. DOI: 10.48550/ARXIV.2006.13858. URL: https://arxiv.org/abs/2006.13858 (cit. on pp. 111, 214).

[1038] P. Gupta, H. Siebert, M. P. Heinrich, and K. T. Rajamani. “DA-AR-Net: an attentive activation based Deformable auto-encoder for group-wise registration.” In: *Medical Imaging 2021: Image Processing*. Ed. by B. A. Landman and I. Išgum. SPIE, Feb. 2021. DOI: 10.1117/12.2582176. URL: https://doi.org/10.1117/12.2582176 (cit. on p. 111).

[1039] S. Balaji, T. Kavya, and N. Sebastian. “Learn-Able Parameter Guided Activation Functions.” In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, Aug. 2020, pp. 583–597. DOI: 10.1007/978-3-030-55180-3_43. URL: https://doi.org/10.1007/978-3-030-55180-3_43 (cit. on pp. 111, 112, 215).

[1040] M. Varshney and P. Singh. “Optimizing nonlinear activation function for convolutional neural networks.” In: *Signal, Image and Video Processing* 15.6 (Feb. 2021), pp. 1323–1330. DOI: 10.1007/s11760-021-01863-z. URL: https://doi.org/10.1007/s11760-021-01863-z (cit. on pp. 112, 113, 215).

[1041] J. Inturrisi, S. Y. Khoo, A. Kouzani, and R. Pagliarella. *Piecewise Linear Units Improve Deep Neural Networks*. 2021. DOI: 10.48550/ARXIV.2108.00700. URL: https://arxiv.org/abs/2108.00700 (cit. on pp. 112, 215).

[1042] Z. Tang, L. Luo, H. Peng, and S. Li. “A joint residual network with paired ReLUs activation for image super-resolution.” In: *Neurocomputing* 273 (Jan. 2018), pp. 37–46. DOI: 10.1016/j.neucom.2017.07.061. URL: https://doi.org/10.1016/j.neucom.2017.07.061 (cit. on pp. 113, 201).

[1043] A. Rozsa and T. E. Boult. *Improved Adversarial Robustness by Reducing Open Space Risk via Tent Activations*. 2019. DOI: 10.48550/ARXIV.1908.02435. URL: https://arxiv.org/abs/1908.02435 (cit. on pp. 113, 114).

[1044] J. Wang, J. Xu, and J. Zhu. “CNNs with Compact Activation Function.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2022, pp. 319–327. ISBN: 9783031087547. DOI: 10.1007/978-3-031-08754-7_40. URL: http://dx.doi.org/10.1007/978-3-031-08754-7_40 (cit. on p. 114).

[1045] Q. Hong, J. W. Siegel, Q. Tan, and J. Xu. *On the Activation Function Dependence of the Spectral Bias of Neural Networks*. 2022. DOI: 10.48550/ARXIV.2208.04924. URL: https://arxiv.org/abs/2208.04924 (cit. on p. 114).

[1046] Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, and M. A. Ayidzoe. "RMAF: Relu-Memristor-Like Activation Function for Deep Learning." In: *IEEE Access* 8 (2020), pp. 72727–72741. DOI: 10.1109/access.2020.2987829. URL: https://doi.org/10.1109/access.2020.2987829 (cit. on pp. 114, 201).

[1047] A. Gupta and R. Duggal. "P-TELU: Parametric Tan Hyperbolic Linear Unit Activation for Deep Neural Networks." In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, Oct. 2017. DOI: 10.1109/iccvw.2017.119. URL: https://doi.org/10.1109/iccvw.2017.119 (cit. on pp. 114, 115, 215, 219).

[1048] M. A. Mercioni and S. Holban. "Developing Novel Activation Functions in Time Series Anomaly Detection with LSTM Autoencoder." In: *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, May 2021. DOI: 10.1109/saci51354.2021.9465604. URL: http://dx.doi.org/10.1109/SACI51354.2021.9465604 (cit. on p. 115).

[1049] S.-L. Shen, N. Zhang, A. Zhou, and Z.-Y. Yin. "Enhancement of neural networks with an alternative activation function tanhLU." In: *Expert Systems with Applications* 199 (Aug. 2022), p. 117181. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.117181. URL: http://dx.doi.org/10.1016/j.eswa.2022.117181 (cit. on pp. 115, 214).

[1050] T. Zhang, J. Yang, W.-a. Song, and C.-f. Song. "Research on Improved Activation Function TReLU." In: *Journal of Chinese Computer Systems* 40.1, 58 (2019), pp. 58–63. URL: http://xwxt.sict.ac.cn/EN/Y2019/V40/I1/58 (cit. on pp. 116, 215).

[1051] M. Nakhua, D. Bavishi, S. Tikoo, and S. Khedkar. "TReLU: A Novel Activation Function for Modern Day Intrusion Detection System Using Deep Neural Networks." In: *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, July 2023. DOI: 10.1109/icccnt56998.2023.10306887. URL: http://dx.doi.org/10.1109/ICCCNT56998.2023.10306887 (cit. on p. 116).

[1052] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen. "ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis." In: *Neurocomputing* 363 (Oct. 2019), pp. 88–98. DOI: 10.1016/j.neucom.2019.07.017. URL: https://doi.org/10.1016/j.neucom.2019.07.017 (cit. on p. 116).

[1053] L. B. Godfrey. "An Evaluation of Parametric Activation Functions for Deep Learning." In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, Oct. 2019. DOI: 10.1109/smc.2019.8913972. URL: https://doi.org/10.1109/smc.2019.8913972 (cit. on pp. 117, 215).

[1054] L. Zhang, T. Yang, R. Jin, and X. He. "O(logT) Projections for Stochastic Optimization of Smooth and Strongly Convex Functions." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1121–1129. URL: https://proceedings.mlr.press/v28/zhang13e.html (cit. on p. 117).

[1055] T. Yang. "Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/dc912a253d1e9ba40e2c597ed2376640-Paper.pdf (cit. on p. 117).

[1056] E. H. Bergou, Y. Diouane, V. Kunc, V. Kungurtsev, and C. W. Royer. "A Subsampling Line-Search Method with Second-Order Results." In: *INFORMS Journal on Optimization* 4.4 (Oct. 2022), pp. 403–425. DOI: 10.1287/ijoo.2022.0072. URL: https://doi.org/10.1287/ijoo.2022.0072 (cit. on p. 117).

[1057] M. Mahdavi, L. Zhang, and R. Jin. "Mixed Optimization for Smooth Functions." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/f73b76ce8949fe29bf2a537cfa420e8f-Paper.pdf (cit. on p. 117).

[1058] M. A. Mercioni and S. Holban. "Soft Clipping Mish - A Novel Activation Function for Deep Learning." In: *2021 4th International Conference on Information and Computer Technologies (ICICT)*. IEEE, Mar. 2021. DOI: 10.1109/icict52872.2021.00010. URL: http://dx.doi.org/10.1109/ICICT52872.2021.00010 (cit. on p. 118).

[1059] M. A. Mercioni and S. Holban. "Prediction of Machine Temperature System Failure Using a Novel Activation Function." In: *2022 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, Nov. 2022. DOI: 10.1109/isetc56213.2022.10010046. URL: http://dx.doi.org/10.1109/ISETC56213.2022.10010046 (cit. on p. 118).

[1060] M.-A. Mercioni and S. Holban. "Weather Forecasting Modeling Using Soft-Clipping Swish Activation Function." In: *2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, May 2022. DOI: 10.1109/saci55618.2022.9919575. URL: http://dx.doi.org/10.1109/SACI55618.2022.9919575 (cit. on p. 118).

[1061] M. A. Mercioni and S. Holban. "Soft-Clipping Swish: A Novel Activation Function for Deep Learning." In: *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, May 2021. DOI: 10.1109/saci51354.2021.9465622. URL: http://dx.doi.org/10.1109/SACI51354.2021.9465622 (cit. on p. 118).

[1062] M. A. Mercioni and S. Holban. "P-Swish: Activation Function with Learnable Parameters Based on Swish Activation Function in Deep Learning." In: *2020 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, Nov. 2020. DOI: 10.1109/isetc50328.2020.9301059. URL: http://dx.doi.org/10.1109/ISETC50328.2020.9301059 (cit. on p. 118).

[1063] L. Shi and X. Xie. "Image Segmentation Method for Maize Ear Using Self-defined Activation Function." In: *Procedia Computer Science* 208 (2022), pp. 162–169. ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.10.024. URL: http://dx.doi.org/10.1016/j.procs.2022.10.024 (cit. on p. 118).

[1064] L. Trottier, P. Giguère, and B. Chaib-draa. *Parametric Exponential Linear Unit for Deep Convolutional Neural Networks*. 2016. DOI: 10.48550/ARXIV.1605.09332. URL: https://arxiv.org/abs/1605.09332 (cit. on pp. 118, 119, 216).

[1065] S. Qian et al. "Adaptive activation functions in convolutional neural networks." In: *Neurocomputing* 272 (Jan. 2018), pp. 204–212. DOI: 10.1016/j.neucom.2017.06.070. URL: https://doi.org/10.1016/j.neucom.2017.06.070 (cit. on pp. 119, 141, 142).

[1066] B. Çatalbaş and Ö. Morgül. "Deep learning with ExtendeD Exponential Linear Unit (DELU)." In: *Neural Computing and Applications* 35.30 (Aug. 2023), pp. 22705–22724. ISSN: 1433-3058. DOI: 10.1007/s00521-023-08932-z. URL: http://dx.doi.org/10.1007/s00521-023-08932-z (cit. on p. 119).

[1067] Z. Qiumei, T. Dan, and W. Fenghua. "Improved Convolutional Neural Network Based on Fast Exponentially Linear Unit Activation Function." In: *IEEE Access* 7 (2019), pp. 151359–151367. DOI: 10.1109/access.2019.2948112. URL: https://doi.org/10.1109/access.2019.2948112 (cit. on pp. 120, 216).

[1068] N. N. Schraudolph. "A Fast, Compact Approximation of the Exponential Function." In: *Neural Computation* 11.4 (May 1999), pp. 853–862. DOI: 10.1162/089976699300016467. URL: https://doi.org/10.1162/089976699300016467 (cit. on p. 120).

[1069] K. Adem. "P+FELU: Flexible and trainable fast exponential linear unit for deep learning architectures." In: *Neural Computing and Applications* 34.24 (July 2022), pp. 21729–21740. ISSN: 1433-3058. DOI: 10.1007/s00521-022-07625-3. URL: http://dx.doi.org/10.1007/s00521-022-07625-3 (cit. on pp. 120, 197, 202).

[1070] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming. "Improving deep neural network with Multiple Parametric Exponential Linear Units." In: *Neurocomputing* 301 (Aug. 2018), pp. 11–24. DOI: 10.1016/j.neucom.2018.01.084. URL: https://doi.org/10.1016/j.neucom.2018.01.084 (cit. on pp. 120, 125, 216).

[1071] C.-H. Pham, C. Tor-Díez, H. Meunier, N. Bednarek, R. Fablet, N. Passat, and F. Rousseau. "Multiscale brain MRI super-resolution using deep 3D convolutional networks." In: *Computerized Medical Imaging and Graphics* 77 (Oct. 2019), p. 101647. DOI: 10.1016/j.compmedimag.2019.101647. URL: https://doi.org/10.1016/j.compmedimag.2019.101647 (cit. on p. 120).

[1072] M. Lin et al. "Network In Network." In: *CoRR* abs/1312.4400 (2013). arXiv: 1312.4400. URL: http://arxiv.org/abs/1312.4400 (cit. on pp. 120, 168).

[1073] R. Jie, J. Gao, A. Vasnev, and M.-n. Tran. "Regularized Flexible Activation Function Combination for Deep Neural Networks." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. DOI: 10.1109/icpr48806.2021.9412370. URL: http://dx.doi.org/10.1109/ICPR48806.2021.9412370 (cit. on pp. 121, 135, 278, 279).

[1074] L. B. Godfrey and M. S. Gashler. "A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks." In: *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2015, pp. 481–486. ISBN: 978-9-8975-8164-9. URL: https://ieeexplore.ieee.org/document/7526959/ (cit. on p. 121).

[1075] J. T. Barron. *Continuously Differentiable Exponential Linear Units*. 2017. DOI: 10.48550/ARXIV.1704.07483. URL: https://arxiv.org/abs/1704.07483 (cit. on pp. 121, 216).

[1076] A. Rajanand and P. Singh. *ErfReLU: Adaptive Activation Function for Deep Neural Network*. 2023. DOI: 10.48550/ARXIV.2306.01822. URL: https://arxiv.org/abs/2306.01822 (cit. on p. 122).

[1077] K. Pratama and D.-K. Kang. “Trainable activation function with differentiable negative side and adaptable rectified point.” In: *Applied Intelligence* 51.3 (Oct. 2020), pp. 1784–1801. DOI: 10.1007/s10489-020-01885-z. URL: https://doi.org/10.1007/s10489-020-01885-z (cit. on pp. 122, 216).

[1078] B. Grelsson and M. Felsberg. “Improved Learning in Convolutional Neural Networks with Shifted Exponential Linear Units (ShELUs).” In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8545104. URL: https://doi.org/10.1109/icpr.2018.8545104 (cit. on pp. 123, 194, 197, 201, 289).

[1079] I. Javid, R. Ghazali, I. Syed, N. A. Husaini, and M. Zulqarnain. “Developing Novel T-Swish Activation Function in Deep Learning.” In: *2022 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, Oct. 2022. DOI: 10.1109/icit56493.2022.9989151. URL: http://dx.doi.org/10.1109/ICIT56493.2022.9989151 (cit. on pp. 123, 124, 216).

[1080] A. M. Atto, S. Galichet, D. Pastor, and N. Méger. “On joint parameterizations of linear and nonlinear functionals in neural networks.” In: *Neural Networks* 160 (Mar. 2023), pp. 12–21. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.12.019. URL: http://dx.doi.org/10.1016/j.neunet.2022.12.019 (cit. on p. 124).

[1081] A. M. Atto, D. Pastor, and G. Mercier. “Smooth sigmoid wavelet shrinkage for non-parametric estimation.” In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, Mar. 2008. DOI: 10.1109/icassp.2008.4518347. URL: http://dx.doi.org/10.1109/ICASSP.2008.4518347 (cit. on p. 124).

[1082] Q. Cheng, H. Li, Q. Wu, L. Ma, and K. N. Ngan. “Parametric Deformable Exponential Linear Units for deep neural networks.” In: *Neural Networks* 125 (May 2020), pp. 281–289. DOI: 10.1016/j.neunet.2020.02.012. URL: https://doi.org/10.1016/j.neunet.2020.02.012 (cit. on pp. 124, 194, 216).

[1083] B. Duan, Y. Yang, and X. Dai. “Feature Activation through First Power Linear Unit with Sign.” In: *Electronics* 11.13 (June 2022), p. 1980. ISSN: 2079-9292. DOI: 10.3390/electronics11131980. URL: http://dx.doi.org/10.3390/electronics11131980 (cit. on p. 125).

[1084] T. Yamada and T. Yabuta. “Neural network controller using autotuning method for nonlinear functions.” In: *IEEE Transactions on Neural Networks* 3.4 (July 1992), pp. 595–601. DOI: 10.1109/72.143373. URL: https://doi.org/10.1109/72.143373 (cit. on pp. 126, 127, 202).

[1085] C.-T. Chen and W.-D. Chang. “A feedforward neural network with function shape autotuning.” In: *Neural Networks* 9.4 (June 1996), pp. 627–641. DOI: 10.1016/0893-6080(96)00006-8. URL: https://doi.org/10.1016/0893-6080(96)00006-8 (cit. on pp. 126, 194, 202).

[1086] E. Trentin. “Networks with trainable amplitude of activation functions.” In: *Neural Networks* 14.4-5 (May 2001), pp. 471–493. DOI: 10.1016/s0893-6080(01)00028-4. URL: https://doi.org/10.1016/s0893-6080(01)00028-4 (cit. on pp. 126, 194, 202, 221, 278, 279).

[1087] S. L. Goh and D. P. Mandic. “Recurrent neural networks with trainable amplitude of activation functions.” In: *Neural Networks* 16.8 (Oct. 2003), pp. 1095–1100. DOI: 10.1016/s0893-6080(03)00139-4. URL: https://doi.org/10.1016/s0893-6080(03)00139-4 (cit. on p. 126).

[1088] T. Yamada and T. Yabuta. “Remarks on a neural network controller which uses an auto-tuning method for nonlinear functions.” In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. IEEE, 1992. DOI: 10.1109/ijcnn.1992.226893. URL: http://dx.doi.org/10.1109/IJCNN.1992.226893 (cit. on p. 127).

[1089] N. M. Nawi, R. Ransing, M. N. M. Salleh, R. Ghazali, and N. A. Hamid. “The effect of gain variation in improving learning speed of back propagation neural network algorithm on classification problems.” In: *Symposium on Progress in Information & Communication Technology*. 2009 (cit. on pp. 127, 202).

[1090] S. K. Sharma and P. Chandra. “An Adaptive Sigmoidal Activation Function Cascading Neural Networks.” In: *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*. Springer Berlin Heidelberg, 2011, pp. 105–116. ISBN: 9783642196447. DOI: 10.1007/978-3-642-19644-7_12. URL: http://dx.doi.org/10.1007/978-3-642-19644-7_12 (cit. on p. 127).

[1091] M. A. Mercioni, A. Tiron, and S. Holban. “Dynamic Modification of Activation Function using the Backpropagation Algorithm in the Artificial Neural Networks.” In: *International Journal of Advanced Computer Science and Applications* 10.4 (2019). ISSN: 2158-107X. DOI: 10.14569/ijacsa.2019.0100406. URL: http://dx.doi.org/10.14569/IJACSA.2019.0100406 (cit. on p. 127).

[1092] Y. Bai et al. “The performance of the backpropagation algorithm with varying slope of the activation function.” In: *Chaos, Solitons & Fractals* 40.1 (Apr. 2009), pp. 69–77. DOI: 10.1016/j.chaos.2007.07.033. URL: https://doi.org/10.1016/j.chaos.2007.07.033 (cit. on pp. 127, 194, 202, 279).

[1093] C.-C. Yu et al. "An adaptive activation function for multilayer feedforward neural networks." In: *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM '02. Proceedings*. IEEE, 2002. DOI: 10.1109/tencon.2002.1181357. URL: https://doi.org/10.1109/tencon.2002.1181357 (cit. on p. 127).

[1094] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. *TanhSoft – a family of activation functions combining Tanh and Softplus*. 2020. DOI: 10.48550/ARXIV.2009.03863. URL: https://arxiv.org/abs/2009.03863 (cit. on p. 127).

[1095] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "TanhSoft—Dynamic Trainable Activation Functions for Faster Learning and Better Performance." In: *IEEE Access* 9 (2021), pp. 120613–120623. ISSN: 2169-3536. DOI: 10.1109/access.2021.3105355. URL: http://dx.doi.org/10.1109/ACCESS.2021.3105355 (cit. on pp. liii, 127).

[1096] Y. Ying, N. Zhang, P. Shan, L. Miao, P. Sun, and S. Peng. "PSigmoid: Improving squeeze-and-excitation block with parametric sigmoid." In: *Applied Intelligence* 51.10 (Mar. 2021), pp. 7427–7439. ISSN: 1573-7497. DOI: 10.1007/s10489-021-02247-z. URL: http://dx.doi.org/10.1007/s10489-021-02247-z (cit. on pp. 128, 203).

[1097] Y. Özbay and G. Tezel. "A new method for classification of ECG arrhythmias using neural network with adaptive activation function." In: *Digital Signal Processing* 20.4 (July 2010), pp. 1040–1049. ISSN: 1051-2004. DOI: 10.1016/j.dsp.2009.10.016. URL: http://dx.doi.org/10.1016/j.dsp.2009.10.016 (cit. on pp. 128, 152, 153).

[1098] P. Chandra and Y. Singh. "An activation function adapting training algorithm for sigmoidal feedforward networks." In: *Neurocomputing* 61 (Oct. 2004), pp. 429–437. DOI: 10.1016/j.neucom.2004.04.001. URL: https://doi.org/10.1016/j.neucom.2004.04.001 (cit. on p. 128).

[1099] Y. Singh and P. Chandra. "A class +1 sigmoidal activation functions for FFANNs." In: *Journal of Economic Dynamics and Control* 28.1 (Oct. 2003), pp. 183–187. ISSN: 0165-1889. DOI: 10.1016/s0165-1889(02)00157-4. URL: http://dx.doi.org/10.1016/S0165-1889(02)00157-4 (cit. on p. 128).

[1100] P. Chandra and Y. Singh. "A case for the self-adaptation of activation functions in FFANNs." In: *Neurocomputing* 56 (Jan. 2004), pp. 447–454. DOI: 10.1016/j.neucom.2003.08.005. URL: https://doi.org/10.1016/j.neucom.2003.08.005 (cit. on p. 128).

[1101] P. Chandra. "Sigmoidal Function Classes for Feedforward Artificial Neural Networks." In: *Neural Processing Letters* 18.3 (Dec. 2003), pp. 205–215. DOI: 10.1023/b:nepl.0000011137.04221.96. URL: https://doi.org/10.1023/b:nepl.0000011137.04221.96 (cit. on p. 128).

[1102] T. Zhang, S. Liu, Y. Wei, and H. Zhang. "A novel feature adaptive extraction method based on deep learning for bearing fault diagnosis." In: *Measurement* 185 (Nov. 2021), p. 110030. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2021.110030. URL: http://dx.doi.org/10.1016/j.measurement.2021.110030 (cit. on pp. 128, 212).

[1103] N. E. Protonotarios, A. S. Fokas, G. A. Kastis, and N. Dikaios. "Sigmoid and Beyond: Algebraic Activation Functions for Artificial Neural Networks Based on Solutions of a Riccati Equation." In: *IT Professional* 24.5 (Sept. 2022), pp. 30–36. ISSN: 1941-045X. DOI: 10.1109/mitp.2022.3204904. URL: http://dx.doi.org/10.1109/MITP.2022.3204904 (cit. on p. 128).

[1104] N. Ma, X. Zhang, M. Liu, and J. Sun. "Activate or Not: Learning Customized Activation." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00794. URL: http://dx.doi.org/10.1109/CVPR46437.2021.00794 (cit. on pp. 129, 130).

[1105] Y. Bodyanskiy and S. Kostiuk. "Adaptive hybrid activation function for deep neural networks." In: *System research and information technologies* 1 (Apr. 2022), pp. 87–96. ISSN: 1681-6048. DOI: 10.20535/srit.2308-8893.2022.1.07. URL: http://dx.doi.org/10.20535/SRIT.2308-8893.2022.1.07 (cit. on pp. 129, 203).

[1106] E. Alcaide. *E-swish: Adjusting Activations to Different Network Depths*. 2018. DOI: 10.48550/ARXIV.1801.07145. URL: https://arxiv.org/abs/1801.07145 (cit. on pp. 129, 130, 195, 200).

[1107] R. Zhang, K. Zheng, P. Shi, Y. Mei, H. Li, and T. Qiu. "Traffic Sign Detection Based on the Improved YOLOv5." In: *Applied Sciences* 13.17 (Aug. 2023), p. 9748. ISSN: 2076-3417. DOI: 10.3390/app13179748. URL: http://dx.doi.org/10.3390/app13179748 (cit. on p. 130).

[1108] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors." In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2023. DOI: 10.1109/cvpr52729.2023.00721. URL: http://dx.doi.org/10.1109/CVPR52729.2023.00721 (cit. on p. 130).

[1109] Y. Ye, Q. Liu, L. Li, Z. Zhang, L. Xu, J. Chu, and B. Wen. "Improving insulator fault detection with effective-YOLOv7 network." In: *Journal of Electronic Imaging* 32.06 (Nov. 2023). ISSN: 1017-9909. DOI: 10.1117/1.jei.32.6.063021. URL: http://dx.doi.org/10.1117/1.JEI.32.6.063021 (cit. on p. 130).

[1110] S. Kan, W. Fang, J. Wu, and V. S. Sheng. "Real-Time Domestic Garbage Detection Method Based on Improved YOLOv5." In: *Communications in Computer and Information Science*. Springer International Publishing, 2022, pp. 62–74. ISBN: 9783031067679. DOI: 10.1007/978-3-031-06767-9_5. URL: http://dx.doi.org/10.1007/978-3-031-06767-9_5 (cit. on p. 130).

[1111] G. Tu, J. Qin, and N. Xiong. "Algorithm of Computer Mainboard Quality Detection for Real-Time Based on QD-YOLO." In: *Electronics* 11.15 (Aug. 2022), p. 2424. ISSN: 2079-9292. DOI: 10.3390/electronics11152424. URL: http://dx.doi.org/10.3390/electronics11152424 (cit. on p. 130).

[1112] X. Xi, Y. Wu, C. Xia, and S. He. "Feature fusion for object detection at one map." In: *Image and Vision Computing* 123 (July 2022), p. 104466. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2022.104466. URL: http://dx.doi.org/10.1016/j.imavis.2022.104466 (cit. on p. 130).

[1113] H. Li, L. Wang, and S. Cheng. "HARNU-Net: Hierarchical Attention Residual Nested U-Net for Change Detection in Remote Sensing Images." In: *Sensors* 22.12 (June 2022), p. 4626. ISSN: 1424-8220. DOI: 10.3390/s22124626. URL: http://dx.doi.org/10.3390/s22124626 (cit. on p. 130).

[1114] J. Wu, J. Li, R. Li, X. Xi, D. Gui, and J. Yin. "A Fast Maritime Target Identification Algorithm for Offshore Ship Detection." In: *Applied Sciences* 12.10 (May 2022), p. 4938. ISSN: 2076-3417. DOI: 10.3390/app12104938. URL: http://dx.doi.org/10.3390/app12104938 (cit. on p. 130).

[1115] Q. Niu, Y. Wang, S. Yuan, K. Li, and X. Wang. "An Indoor Pool Drowning Risk Detection Method Based on Improved YOLOv4." In: *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, Dec. 2022. DOI: 10.1109/imcec55388.2022.10020040. URL: http://dx.doi.org/10.1109/IMCEC55388.2022.10020040 (cit. on p. 130).

[1116] B. Zhang, W. Chen, X. Wang, and C. Zhao. "A Lightweight Detection Method of Smartphone Assembly Parts." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 330–342. ISBN: 9783031204975. DOI: 10.1007/978-3-031-20497-5_27. URL: http://dx.doi.org/10.1007/978-3-031-20497-5_27 (cit. on p. 130).

[1117] K.-Y. Cao, X. Cui, and J.-C. Piao. "Smaller Target Detection Algorithms Based on YOLOv5 in Safety Helmet Wearing Detection." In: *2022 4th International Conference on Robotics and Computer Vision (ICRCV)*. IEEE, Sept. 2022. DOI: 10.1109/icrcv55858.2022.9953233. URL: http://dx.doi.org/10.1109/ICRCV55858.2022.9953233 (cit. on p. 130).

[1118] Y. Jia, J. Zhao, and L. Yu. "AADH-YOLOv5: improved YOLOv5 based on adaptive activate decoupled head for garbage detection." In: *Journal of Electronic Imaging* 32.04 (July 2023). ISSN: 1017-9909. DOI: 10.1117/1.jei.32.4.043017. URL: http://dx.doi.org/10.1117/1.JEI.32.4.043017 (cit. on p. 130).

[1119] H. Xu, W. Zheng, F. Liu, P. Li, and R. Wang. "Unmanned Aerial Vehicle Perspective Small Target Recognition Algorithm Based on Improved YOLOv5." In: *Remote Sensing* 15.14 (July 2023), p. 3583. ISSN: 2072-4292. DOI: 10.3390/rs15143583. URL: http://dx.doi.org/10.3390/rs15143583 (cit. on p. 130).

[1120] J. He, J. Duan, Z. Yang, J. Ou, X. Ou, S. Yu, M. Xie, Y. Luo, H. Wang, and Q. Jiang. "Method for Segmentation of Banana Crown Based on Improved DeepLabv3+." In: *Agronomy* 13.7 (July 2023), p. 1838. ISSN: 2073-4395. DOI: 10.3390/agronomy13071838. URL: http://dx.doi.org/10.3390/agronomy13071838 (cit. on p. 130).

[1121] H. Qin, J. Pan, J. Li, and F. Huang. "Fault Diagnosis Method of Rolling Bearing Based on CBAM_ResNet and ACON Activation Function." In: *Applied Sciences* 13.13 (June 2023), p. 7593. ISSN: 2076-3417. DOI: 10.3390/app13137593. URL: http://dx.doi.org/10.3390/app13137593 (cit. on p. 130).

[1122] N. Chen, Y. Li, Z. Yang, Z. Lu, S. Wang, and J. Wang. "LODNU: lightweight object detection network in UAV vision." In: *The Journal of Supercomputing* 79.9 (Feb. 2023), pp. 10117–10138. ISSN: 1573-0484. DOI: 10.1007/s11227-023-05065-x. URL: http://dx.doi.org/10.1007/s11227-023-05065-x (cit. on p. 130).

[1123] Y. Zhao, L. Lu, W. Yang, Q. Li, and X. Zhang. "Lightweight Tennis Ball Detection Algorithm Based on Robomaster EP." In: *Applied Sciences* 13.6 (Mar. 2023), p. 3461. ISSN: 2076-3417. DOI: 10.3390/app13063461. URL: http://dx.doi.org/10.3390/app13063461 (cit. on p. 130).

[1124] J. Liu, X. Wang, S. Wu, L. Wan, and F. Xie. "Wind turbine fault detection based on deep residual networks." In: *Expert Systems with Applications* 213 (Mar. 2023), p. 119102. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.119102. URL: http://dx.doi.org/10.1016/j.eswa.2022.119102 (cit. on p. 130).

[1125] Z. Li, X. Yang, K. Shen, F. Jiang, J. Jiang, H. Ren, and Y. Li. "PSGU: Parametric self-circulation gating unit for deep neural networks." In: *Journal of Visual Communication and Image Representation* 80 (Oct. 2021), p. 103294. ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2021.103294. URL: http://dx.doi.org/10.1016/j.jvcir.2021.103294 (cit. on pp. 130, 131).

[1126] B. Zheng and Z. Wang. "PATS: A New Neural Network Activation Function with Parameter." In: *2020 5th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, May 2020. DOI: 10.1109/iccs49078.2020.9118471. URL: http://dx.doi.org/10.1109/ICCCS49078.2020.9118471 (cit. on p. 131).

[1127] A. Paul, R. Bandyopadhyay, J. H. Yoon, Z. W. Geem, and R. Sarkar. "SinLU: Sinu-Sigmoidal Linear Unit." In: *Mathematics* 10.3 (Jan. 2022), p. 337. ISSN: 2227-7390. DOI: 10.3390/math10030337. URL: http://dx.doi.org/10.3390/math10030337 (cit. on pp. 132, 219).

[1128] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "ErfAct and Pserf: Non-monotonic Smooth Trainable Activation Functions." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.6 (June 2022), pp. 6097–6105. ISSN: 2159-5399. DOI: 10.1609/aaai.v36i6.20557. URL: http://dx.doi.org/10.1609/aaai.v36i6.20557 (cit. on p. 132).

[1129] M. Abdool and T. Dear. *Swim: A General-Purpose, High-Performing, and Efficient Activation Function for Locomotion Control Tasks*. 2023. DOI: 10.48550/ARXIV.2303.02640. URL: https://arxiv.org/abs/2303.02640 (cit. on p. 132).

[1130] V. Terziyan, D. Malyk, M. Golovianko, and V. Branytskyi. "Hyper-flexible Convolutional Neural Networks based on Generalized Lehmer and Power Means." In: *Neural Networks* 155 (Nov. 2022), pp. 177–203. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.08.017. URL: http://dx.doi.org/10.1016/j.neunet.2022.08.017 (cit. on pp. 133, 144).

[1131] Q. Jiang, L. Zhu, C. Shu, and V. Sekar. "Multilayer perceptron neural network activated by adaptive Gaussian radial basis function and its application to predict lid-driven cavity flow." In: *Acta Mechanica Sinica* 37.12 (Dec. 2021), pp. 1757–1772. ISSN: 1614-3116. DOI: 10.1007/s10409-021-01144-5. URL: http://dx.doi.org/10.1007/s10409-021-01144-5 (cit. on p. 134).

[1132] F. Duan, F. Chapeau-Blondeau, and D. Abbott. "Optimized injection of noise in activation functions to improve generalization of neural networks." In: *Chaos, Solitons & Fractals* 178 (Jan. 2024), p. 114363. ISSN: 0960-0779. DOI: 10.1016/j.chaos.2023.114363. URL: http://dx.doi.org/10.1016/j.chaos.2023.114363 (cit. on pp. 134, 135).

[1133] H. H. Chieng, N. Wahid, and P. Ong. "Parametric Flatten-T swish: an adaptive nonlinear activation function for deep learning." In: *Journal of Information and Communication Technology* 20 (2020). DOI: 10.32890/jict.20.1.2021.9267. URL: https://doi.org/10.32890/jict.20.1.2021.9267 (cit. on pp. 134, 203).

[1134] A. Mondal and V. K. Shrivastava. "A novel Parametric Flatten-p Mish activation function based deep CNN model for brain tumor classification." In: *Computers in Biology and Medicine* 150 (Nov. 2022), p. 106183. ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2022.106183. URL: http://dx.doi.org/10.1016/j.compbiomed.2022.106183 (cit. on p. 134).

[1135] B. Khagi and G.-R. Kwon. "A novel scaled-gamma-tanh (SGT) activation function in 3D CNN applied for MRI classification." In: *Scientific Reports* 12.1 (Sept. 2022). ISSN: 2045-2322. DOI: 10.1038/s41598-022-19020-y. URL: http://dx.doi.org/10.1038/s41598-022-19020-y (cit. on p. 135).

[1136] R. Ding, H. Liu, and X. Zhou. "IE-Net: Information-Enhanced Binary Neural Networks for Accurate Classification." In: *Electronics* 11.6 (Mar. 2022), p. 937. ISSN: 2079-9292. DOI: 10.3390/electronics11060937. URL: http://dx.doi.org/10.3390/electronics11060937 (cit. on p. 135).

[1137] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks." In: *Journal of Computational Physics* 404 (Mar. 2020), p. 109136. DOI: 10.1016/j.jcp.2019.109136. URL: https://doi.org/10.1016/j.jcp.2019.109136 (cit. on pp. 135, 136, 198, 202, 278).

[1138] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks." In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2239 (July 2020), p. 20200334. DOI: 10.1098/rspa.2020.0334. URL: https://doi.org/10.1098/rspa.2020.0334 (cit. on pp. 135, 194, 278).

[1139] D. S. Kapoor and A. K. Kohli. "Adaptive-Slope Squashing-Function-Based ANN for CSI Estimation and Symbol Detection in SFBC-OFDM System." In: *Arabian Journal for Science and Engineering* 46.10 (Jan. 2021), pp. 9451–9464. DOI: 10.1007/s13369-020-05207-w. URL: https://doi.org/10.1007/s13369-020-05207-w (cit. on pp. 136, 203, 279).

[1140] S. S. Husain, E.-J. Ong, and M. Bober. "ACTNET: End-to-End Learning of Feature Activations and Multi-stream Aggregation for Effective Instance Image Retrieval." In: *International Journal of Computer Vision* 129.5 (Feb. 2021), pp. 1432–1450. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01444-0. URL: http://dx.doi.org/10.1007/s11263-021-01444-0 (cit. on pp. 136, 137, 152, 203).

[1141] X.-M. Zhou, L.-F. Li, X.-Z. Zheng, and M. Luo. *LAU: A novel two-parameter learnable Logmoid Activation Unit*. 2023. URL: https://openreview.net/forum?id=uwBUzlm0GS (cit. on p. 137).

[1142] X.-M. Zhou, L.-F. Li, X.-Z. Zheng, and M. Luo. *A two-parameter learnable Logmoid Activation Unit*. 2023. URL: https://openreview.net/forum?id=LcXWYmA8Ek (cit. on p. 137).

[1143] F. Farhadi, V. P. Nia, and A. Lodi. *Activation Adaptation in Neural Networks*. 2019. DOI: 10.48550/ARXIV.1901.09849. URL: https://arxiv.org/abs/1901.09849 (cit. on pp. 137, 184, 189).

[1144] Y. Zhou, D. Li, S. Huo, and S.-Y. Kung. "Shape autotuning activation function." In: *Expert Systems with Applications* 171 (June 2021), p. 114534. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2020.114534. URL: http://dx.doi.org/10.1016/j.eswa.2020.114534 (cit. on p. 138).

[1145] J. Z. Zamora-Esquivel, J. A. Cruz Vargas, and P. Lopez-Meyer. "Fractional Adaptation of Activation Functions In Neural Networks." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. DOI: 10.1109/icpr48806.2021.9413338. URL: http://dx.doi.org/10.1109/ICPR48806.2021.9413338 (cit. on pp. 139, 140).

[1146] J. Zamora-Esquivel, A. D. Rhodes, and L. Nachman. "Fractional Adaptive Linear Units." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.8 (June 2022), pp. 8988–8996. ISSN: 2159-5399. DOI: 10.1609/aaai.v36i8.20882. URL: http://dx.doi.org/10.1609/aaai.v36i8.20882 (cit. on pp. 139, 140).

[1147] M. S. Job, P. H. Bhateja, M. Gupta, K. Bingi, and B. R. Prusty. "Fractional Rectified Linear Unit Activation Function and Its Variants." In: *Mathematical Problems in Engineering* 2022 (June 2022). Ed. by X. Li, pp. 1–15. ISSN: 1024-123X. DOI: 10.1155/2022/1860779. URL: http://dx.doi.org/10.1155/2022/1860779 (cit. on pp. 139–143).

[1148] B. Ramadevi, V. R. Kasi, and K. Bingi. "Fractional ordering of activation functions for neural networks: A case study on Texas wind turbine." In: *Engineering Applications of Artificial Intelligence* 127 (Jan. 2024), p. 107308. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.107308. URL: http://dx.doi.org/10.1016/j.engappai.2023.107308 (cit. on p. 139).

[1149] J. Zamora-Esquivel, A. Cruz Vargas, R. Camacho Perez, P. Lopez Meyer, H. Cordourier, and O. Tickoo. "Adaptive Activation Functions Using Fractional Calculus." In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, Oct. 2019. DOI: 10.1109/iccvw.2019.00250. URL: http://dx.doi.org/10.1109/ICCVW.2019.00250 (cit. on p. 139).

[1150] M. D. Ortigueira. *Fractional Calculus for Scientists and Engineers*. Springer Netherlands, 2011. ISBN: 9789400707474. DOI: 10.1007/978-94-007-0747-4. URL: http://dx.doi.org/10.1007/978-94-007-0747-4 (cit. on p. 139).

[1151] Y. Bodyanskiy and S. Kostiuk. "Learnable Extended Activation Function for Deep Neural Networks." In: *International Journal of Computing* (Oct. 2023), pp. 311–318. ISSN: 1727-6209. DOI: 10.47839/ijc.22.3.3225. URL: http://dx.doi.org/10.47839/ijc.22.3.3225 (cit. on p. 144).

[1152] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. "Extreme Learning Machine for Regression and Multiclass Classification." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.2 (Apr. 2012), pp. 513–529. ISSN: 1941-0492. DOI: 10.1109/tsmcb.2011.2168604. URL: http://dx.doi.org/10.1109/TSMCB.2011.2168604 (cit. on pp. 144, 175).

[1153] R. Siouda, M. Nemissi, and H. Seridi. "Diverse activation functions based-hybrid RBF-ELM neural network for medical classification." In: *Evolutionary Intelligence* (July 2022). ISSN: 1864-5917. DOI: 10.1007/s12065-022-00758-3. URL: http://dx.doi.org/10.1007/s12065-022-00758-3 (cit. on pp. 144, 175, 176).

[1154] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "EIS - Efficient and Trainable Activation Functions for Better Accuracy and Performance." In: *Artificial Neural Networks and Machine Learning – ICANN 2021*. Springer International Publishing, 2021, pp. 260–272. ISBN: 9783030863401. DOI: 10.1007/978-3-030-86340-1_21. URL: http://dx.doi.org/10.1007/978-3-030-86340-1_21 (cit. on pp. 144, 145).

[1155] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. *EIS – a family of activation functions combining Exponential, ISRU, and Softplus*. 2020. DOI: 10.48550/ARXIV.2009.13501. URL: https://arxiv.org/abs/2009.13501 (cit. on pp. 144, 145).

[1156] T. A. E. Ferreira, M. Mattheakis, and P. Protopapas. *A New Artificial Neuron Proposal with Trainable Simultaneous Local and Global Activation Function*. 2021. DOI: 10.48550/ARXIV.2101.06100. URL: https://arxiv.org/abs/2101.06100 (cit. on pp. 145, 146).

[1157] J. H. De Medeiros Delgado and T. A. E. Ferreira. "Autoencoder performance analysis with adaptive and trainable activation function to compress images." In: *2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, Nov. 2022. DOI: 10.1109/la-cci54402.2022.9981644. URL: http://dx.doi.org/10.1109/LA-CCI54402.2022.9981644 (cit. on p. 146).

[1158] S. Xu and M. Zhang. "Justification of a neuron-adaptive activation function." In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000. DOI: 10.1109/ijcnn.2000.861351. URL: https://doi.org/10.1109/ijcnn.2000.861351 (cit. on pp. 146, 194, 217).

[1159] S. Xu and M. Zhang. "Data Mining — An Adaptive Neural Network Model for Financial Analysis." In: *Third International Conference on Information Technology and Applications (ICITA'05)*. IEEE, 2005. DOI: 10.1109/icita.2005.109. URL: https://doi.org/10.1109/icita.2005.109 (cit. on p. 146).

[1160] S. Xu and M. Zhang. "A New Adaptive Neural Network Model for Financial Data Mining." In: *Advances in Neural Networks – ISNN 2007*. Springer Berlin Heidelberg, 2007, pp. 1265–1273. DOI: 10.1007/978-3-540-72383-7_147. URL: https://doi.org/10.1007/978-3-540-72383-7_147 (cit. on p. 146).

[1161] G. Tezel and Y. Özbay. "A New Neural Network with Adaptive Activation Function for Classification of ECG Arrhythmias." In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 1–8. DOI: 10.1007/978-3-540-74819-9_1. URL: https://doi.org/10.1007/978-3-540-74819-9_1 (cit. on pp. 146, 203, 217).

[1162] S. Xu and M. Zhang. "An Adaptive Activation Function for Higher Order Neural Networks." In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 356–362. DOI: 10.1007/3-540-36187-1_31. URL: https://doi.org/10.1007/3-540-36187-1_31 (cit. on p. 146).

[1163] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. *Learning Activation Functions to Improve Deep Neural Networks*. 2014. DOI: 10.48550/ARXIV.1412.6830. URL: https://arxiv.org/abs/1412.6830 (cit. on pp. 146, 147).

[1164] S. Aziznejad, H. Gupta, J. Campos, and M. Unser. "Deep Neural Networks With Trainable Activations and Controlled Lipschitz Constant." In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 4688–4699. DOI: 10.1109/tsp.2020.3014611. URL: https://doi.org/10.1109/tsp.2020.3014611 (cit. on p. 147).

[1165] M. Tavakoli, F. Agostinelli, and P. Baldi. "SPLASH: Learnable activation functions for improving accuracy and adversarial robustness." In: *Neural Networks* 140 (Aug. 2021), pp. 1–12. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2021.02.023. URL: http://dx.doi.org/10.1016/j.neunet.2021.02.023 (cit. on p. 147).

[1166] H. Li, W. Ouyang, and X. Wang. "Multi-Bias Non-linear Activation in Deep Neural Networks." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 221–229. URL: https://proceedings.mlr.press/v48/lia16.html (cit. on pp. 147, 203).

[1167] G. Maguolo, L. Nanni, and S. Ghidoni. "Ensemble of convolutional neural networks trained with different activation functions." In: *Expert Systems with Applications* 166 (Mar. 2021), p. 114048. DOI: 10.1016/j.eswa.2020.114048. URL: https://doi.org/10.1016/j.eswa.2020.114048 (cit. on p. 148).

[1168] M. Fakhfakh and L. Chaari. *Bayesian optimization for sparse neural networks with trainable activation functions*. 2023. DOI: 10.48550/ARXIV.2304.04455. URL: https://arxiv.org/abs/2304.04455 (cit. on p. 148).

[1169] L. Nanni, A. Lumini, S. Ghidoni, and G. Maguolo. "Stochastic Selection of Activation Layers for Convolutional Neural Networks." In: *Sensors* 20.6 (Mar. 2020), p. 1626. ISSN: 1424-8220. DOI: 10.3390/s20061626. URL: http://dx.doi.org/10.3390/s20061626 (cit. on p. 148).

[1170] B. Prach and C. H. Lampert. *1-Lipschitz Neural Networks are more expressive with N-Activations*. 2023. DOI: 10.48550/ARXIV.2311.06103. URL: https://arxiv.org/abs/2311.06103 (cit. on p. 149).

[1171] V. S. Bawa and V. Kumar. "Linearized sigmoidal activation: A novel activation function with tractable non-linear characteristics to boost representation capability." In: *Expert Systems with Applications* 120 (Apr. 2019), pp. 346–356. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.11.042. URL: http://dx.doi.org/10.1016/j.eswa.2018.11.042 (cit. on p. 150).

[1172] S. Curci, D. C. Mocanu, and M. Pechenizkiyi. *Truly Sparse Neural Networks at Scale*. 2021. DOI: 10.48550/ARXIV.2102.01732. URL: https://arxiv.org/abs/2102.01732 (cit. on pp. 150, 217, 289).

[1173] A. Nicolae. *PLU: The Piecewise Linear Unit Activation Function*. 2018. DOI: 10.48550/ARXIV.1809.09534. URL: https://arxiv.org/abs/1809.09534 (cit. on pp. 150, 217).

[1174] R. Mo, K. Xu, L. Liu, L. Liu, and D. Wang. "Adaptive Linear Unit for Accurate Binary Neural Networks." In: *2022 16th IEEE International Conference on Signal Processing (ICSP)*. IEEE, Oct. 2022. DOI: 10.1109/icsp56322.2022.9965306. URL: http://dx.doi.org/10.1109/ICSP56322.2022.9965306 (cit. on pp. 151, 218).

[1175] T. Mao, Z. Shi, and D.-X. Zhou. "Approximating functions with multi-features by deep convolutional neural networks." In: *Analysis and Applications* 21.01 (Nov. 2022), pp. 93–125. ISSN: 1793-6861. DOI: 10.1142/s0219530522400085. URL: http://dx.doi.org/10.1142/S0219530522400085 (cit. on p. 151).

[1176] N. Patwardhan, M. Ingalhalikar, and R. Walambe. *ARiA: Utilizing Richard's Curve for Controlling the Non-monotonicity of the Activation Function in Deep Neural Nets*. 2018. DOI: 10.48550/ARXIV.1805.08878. URL: https://arxiv.org/abs/1805.08878 (cit. on pp. 151, 152).

[1177] F. J. Richards. "A Flexible Growth Function for Empirical Use." In: *Journal of Experimental Botany* 10.2 (1959), pp. 290–301. DOI: 10.1093/jxb/10.2.290. URL: https://doi.org/10.1093/jxb/10.2.290 (cit. on p. 151).

[1178] S. Sarkar, S. Agrawal, T. Baker, P. K. R. Maddikunta, and T. R. Gadekallu. "Catalysis of neural activation functions: Adaptive feed-forward training for big data applications." In: *Applied Intelligence* 52.12 (Mar. 2022), pp. 13364–13383. ISSN: 1573-7497. DOI: 10.1007/s10489-021-03082-y. URL: http://dx.doi.org/10.1007/s10489-021-03082-y (cit. on pp. 152, 153).

[1179] S. Gu, W. Li, L. V. Gool, and R. Timofte. "Fast Image Restoration With Multi-Bin Trainable Linear Units." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00429. URL: https://doi.org/10.1109/iccv.2019.00429 (cit. on pp. 153, 218).

[1180] X. Gao, Y. Li, W. Li, L. Duan, L. Van Gool, L. Benini, and M. Magno. "Learning continuous piecewise non-linear activation functions for deep neural networks." In: *2023 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, July 2023. DOI: 10.1109/icme55011.2023.00315. URL: http://dx.doi.org/10.1109/ICME55011.2023.00315 (cit. on pp. 153, 154).

[1181] Y. Zhou, Z. Zhu, and Z. Zhong. "Learning specialized activation functions with the Piecewise Linear Unit." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01188. URL: http://dx.doi.org/10.1109/ICCV48922.2021.01188 (cit. on p. 153).

[1182] Z. Zhu, Y. Zhou, Y. Dong, and Z. Zhong. "PWLU: Learning Specialized Activation Functions with the Piecewise Linear Unit." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), pp. 1–19. ISSN: 1939-3539. DOI: 10.1109/tpami.2023.3286109. URL: http://dx.doi.org/10.1109/TPAMI.2023.3286109 (cit. on p. 153).

[1183] Z. Zhu and Y. Dong. "Non-uniform Piecewise Linear Activation Functions in Deep Neural Networks." In: *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2022. DOI: 10.1109/icpr56361.2022.9956345. URL: http://dx.doi.org/10.1109/ICPR56361.2022.9956345 (cit. on p. 153).

[1184] J. Zhang, S. Zhu, N. Lu, and S. Wen. "Multistability of state-dependent switching neural networks with discontinuous nonmonotonic piecewise linear activation functions." In: *Neurocomputing* 437 (May 2021), pp. 300–311. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.01.046. URL: http://dx.doi.org/10.1016/j.neucom.2021.01.046 (cit. on p. 153).

[1185] A. Goujon, A. Etemadi, and M. Unser. "On the number of regions of piecewise linear neural networks." In: *Journal of Computational and Applied Mathematics* 441 (May 2024), p. 115667. ISSN: 0377-0427. DOI: 10.1016/j.cam.2023.115667. URL: http://dx.doi.org/10.1016/j.cam.2023.115667 (cit. on p. 153).

[1186] M. Wang, B. Liu, and H. Foroosh. "Look-Up Table Unit Activation Function for Deep Convolutional Neural Networks." In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2018. DOI: 10.1109/wacv.2018.00139. URL: https://doi.org/10.1109/wacv.2018.00139 (cit. on pp. 154–157, 160, 218, 219).

[1187] F. Piazza, A. Uncini, and M. Zenobi. "Neural networks with digital LUT activation functions." In: *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*. IEEE, 1993. DOI: 10.1109/ijcnn.1993.716806. URL: https://doi.org/10.1109/ijcnn.1993.716806 (cit. on p. 154).

[1188] S. Fiori. "Hybrid independent component analysis by adaptive LUT activation function neurons." In: *Neural Networks* 15.1 (Jan. 2002), pp. 85–94. DOI: 10.1016/s0893-6080(01)00105-8. URL: https://doi.org/10.1016/s0893-6080(01)00105-8 (cit. on p. 154).

[1189] M. Kang and D. Palmer-Brown. "An Adaptive Function Neural Network (ADFUNN) Classifier." In: *2005 International Conference on Neural Networks and Brain*. IEEE, 2005. DOI: 10.1109/icnnb.2005.1614681. URL: https://doi.org/10.1109/icnnb.2005.1614681 (cit. on p. 154).

[1190] M. Kang and D. Palmer-Brown. "A Multi-layer ADaptive FUnction Neural Network (MADFUNN) for Letter Image Recognition." In: *2007 International Joint Conference on Neural Networks*. IEEE, Aug. 2007. DOI: 10.1109/ijcnn.2007.4371406. URL: https://doi.org/10.1109/ijcnn.2007.4371406 (cit. on p. 154).

[1191] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. "Maxout Networks." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1319–1327. URL: https://proceedings.mlr.press/v28/goodfellow13.html (cit. on pp. 155, 218).

[1192] M. S. Hanif and M. Bilal. “Competitive residual neural network for image classification.” In: *ICT Express* 6.1 (Mar. 2020), pp. 28–37. DOI: 10.1016/j.icte.2019.06.001. URL: https://doi.org/10.1016/j.icte.2019.06.001 (cit. on p. 155).

[1193] G. Castaneda, P. Morris, and T. M. Khoshgoftaar. “Evaluation of maxout activations in deep learning across several big data domains.” In: *Journal of Big Data* 6.1 (Aug. 2019). DOI: 10.1186/s40537-019-0233-0. URL: https://doi.org/10.1186/s40537-019-0233-0 (cit. on p. 155).

[1194] L. R. Sütfeld, F. Brieger, H. Finger, S. Füllhase, and G. Pipa. “Adaptive Blending Units: Trainable Activation Functions for Deep Neural Networks.” In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2020, pp. 37–50. DOI: 10.1007/978-3-030-52243-8_4. URL: https://doi.org/10.1007/978-3-030-52243-8_4 (cit. on pp. 156, 157, 218).

[1195] F. Manessi and A. Rozza. “Learning Combinations of Activation Functions.” In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8545362. URL: https://doi.org/10.1109/icpr.2018.8545362 (cit. on pp. 156, 157, 210, 218).

[1196] D. Klabjan and M. Harmon. “Activation Ensembles for Deep Neural Networks.” In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2019. DOI: 10.1109/bigdata47090.2019.9006069. URL: https://doi.org/10.1109/bigdata47090.2019.9006069 (cit. on pp. 156, 157, 210, 219).

[1197] P. M. Baggenstoss. “Trainable Compound Activation Functions for Machine Learning.” In: *2022 30th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2022. DOI: 10.23919/eusipco55093.2022.9909774. URL: http://dx.doi.org/10.23919/EUSIPCO55093.2022.9909774 (cit. on pp. 157, 218).

[1198] P. M. Baggenstoss. “Improved Auto-Encoding Using Deterministic Projected Belief Networks and Compound Activation Functions.” In: *2023 31st European Signal Processing Conference (EUSIPCO)*. IEEE, Sept. 2023. DOI: 10.23919/eusipco58844.2023.10290080. URL: http://dx.doi.org/10.23919/EUSIPCO58844.2023.10290080 (cit. on pp. 157, 218).

[1199] Z. Liao. *Trainable Activation Function in Image Classification*. 2020. DOI: 10.48550/ARXIV.2004.13271. URL: https://arxiv.org/abs/2004.13271 (cit. on pp. 157, 158, 160, 220).

[1200] A. Ismail et al. “Predictions of bridge scour: Application of a feed-forward neural network with an adaptive activation function.” In: *Engineering Applications of Artificial Intelligence* 26.5-6 (May 2013), pp. 1540–1549. DOI: 10.1016/j.engappai.2012.12.011. URL: https://doi.org/10.1016/j.engappai.2012.12.011 (cit. on p. 158).

[1201] A. D. Jagtap, Y. Shin, K. Kawaguchi, and G. E. Karniadakis. “Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions.” In: *Neurocomputing* 468 (Jan. 2022), pp. 165–180. DOI: 10.1016/j.neucom.2021.10.036. URL: https://doi.org/10.1016/j.neucom.2021.10.036 (cit. on p. 158).

[1202] M. Goyal, R. Goyal, and B. Lall. *Learning Activation Functions: A new paradigm for understanding Neural Networks*. 2019. DOI: 10.48550/ARXIV.1906.09529. URL: https://arxiv.org/abs/1906.09529 (cit. on pp. 159, 219).

[1203] F. Piazza et al. “Artificial Neural Networks With Adaptive Polynomial Activation Function.” In: *Proceedings of the International Joint Conference on Neural Networks.IJCNN*. 1992 (cit. on p. 159).

[1204] K.-C. J. Chen and J.-W. Liang. “A Two-stage Training Mechanism for the CNN with Trainable Activation Function.” In: *2020 International SoC Design Conference (ISOCC)*. IEEE, Oct. 2020. DOI: 10.1109/isocc50952.2020.9333116. URL: http://dx.doi.org/10.1109/ISOCC50952.2020.9333116 (cit. on p. 159).

[1205] T. L. Fonseca and L. Goliatt. “Extreme Learning Machine Based Model Improved with Adaptive Activation Functions.” In: *Computational Intelligence in Information Systems*. Springer International Publishing, 2021, pp. 119–128. ISBN: 9783030681333. DOI: 10.1007/978-3-030-68133-3_12. URL: http://dx.doi.org/10.1007/978-3-030-68133-3_12 (cit. on p. 159).

[1206] M. Deepthi, G. N. V. R. Vikram, and P. Venkatappareddy. “Development of a novel activation function based on Chebyshev polynomials: an aid for classification and denoising of images.” In: *The Journal of Supercomputing* (June 2023). DOI: 10.1007/s11227-023-05466-y. URL: https://doi.org/10.1007/s11227-023-05466-y (cit. on pp. 159, 219).

[1207] P. Venkatappareddy, J. Culli, S. Srivastava, and B. Lall. “A Legendre polynomial based activation function: An aid for modeling of max pooling.” In: *Digital Signal Processing* 115 (Aug. 2021), p. 103093. DOI: 10.1016/j.dsp.2021.103093. URL: https://doi.org/10.1016/j.dsp.2021.103093 (cit. on pp. 159, 160, 219).

[1208] V. S. Lokhande, S. Tasneeyapant, A. Venkatesh, S. N. Ravi, and V. Singh. “Generating Accurate Pseudo-Labels in Semi-Supervised Learning and Avoiding Overconfident Predictions via Hermite Polynomial Activations.” In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.01145. URL: http://dx.doi.org/10.1109/cvpr42600.2020.01145 (cit. on p. 160).

[1209] A. Molina, P. Schramowski, and K. Kersting. “Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks.” In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BJlBSkHtDS (cit. on pp. 161, 219).

[1210] C. Brezinski. “Padé Approximations.” In: *Computational Aspects of Linear Control*. Springer US, 2002, pp. 87–134. DOI: 10.1007/978-1-4613-0261-2_4. URL: https://doi.org/10.1007/978-1-4613-0261-2_4 (cit. on p. 161).

[1211] C. Brezinski. “Extrapolation algorithms and Padé approximations: a historical survey.” In: *Applied Numerical Mathematics* 20.3 (Mar. 1996), pp. 299–318. DOI: 10.1016/0168-9274(95)00110-7. URL: https://doi.org/10.1016/0168-9274(95)00110-7 (cit. on p. 161).

[1212] Q. Delfosse, P. Schramowski, M. Mundt, A. Molina, and K. Kersting. *Adaptive Rational Activations to Boost Deep Reinforcement Learning*. 2021. DOI: 10.48550/ARXIV.2102.09407. URL: https://arxiv.org/abs/2102.09407 (cit. on p. 161).

[1213] N. Boulle, Y. Nakatsukasa, and A. Townsend. “Rational neural networks.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14243–14253. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/a3f390d88e4c41f2747bfa2f1b5f87db-Paper.pdf (cit. on pp. 161, 162).

[1214] Z. Chen, F. Chen, R. Lai, X. Zhang, and C.-T. Lu. “Rational Neural Networks for Approximating Graph Convolution Operator on Jump Discontinuities.” In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2018. DOI: 10.1109/icdm.2018.00021. URL: https://doi.org/10.1109/icdm.2018.00021 (cit. on p. 161).

[1215] M. Trimmel, M. Zanfir, R. Hartley, and C. Sminchisescu. “ERA: Enhanced Rational Activations.” In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 722–738. DOI: 10.1007/978-3-031-20044-1_41. URL: https://doi.org/10.1007/978-3-031-20044-1_41 (cit. on pp. 161, 162).

[1216] K. Biswas, S. Banerjee, and A. K. Pandey. *Orthogonal-Padé Activation Functions: Trainable Activation functions for smooth and faster convergence in deep networks*. 2021. DOI: 10.48550/ARXIV.2106.09693. URL: https://arxiv.org/abs/2106.09693 (cit. on pp. 162, 163).

[1217] S. Guarnieri et al. “Multilayer feedforward networks with adaptive spline activation function.” In: *IEEE Transactions on Neural Networks* 10.3 (May 1999), pp. 672–683. DOI: 10.1109/72.761726. URL: https://doi.org/10.1109/72.761726 (cit. on p. 162).

[1218] M. Solazzi and A. Uncini. “Artificial neural networks with adaptive multidimensional spline activation functions.” In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000. DOI: 10.1109/ijcnn.2000.861352. URL: https://doi.org/10.1109/ijcnn.2000.861352 (cit. on p. 162).

[1219] P. Campolucci, F. Capperelli, S. Guarnieri, F. Piazza, and A. Uncini. “Neural networks with adaptive spline activation function.” In: *Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96)*. IEEE, 1996. DOI: 10.1109/melcon.1996.551220. URL: https://doi.org/10.1109/melcon.1996.551220 (cit. on p. 162).

[1220] P. Bohra, J. Campos, H. Gupta, S. Aziznejad, and M. Unser. “Learning Activation Functions in Deep (Spline) Neural Networks.” In: *IEEE Open Journal of Signal Processing* 1 (2020), pp. 295–309. DOI: 10.1109/ojsp.2020.3039379. URL: https://doi.org/10.1109/ojsp.2020.3039379 (cit. on pp. 162, 163).

[1221] S. Lane, M. Flax, D. Handelman, and J. Gelfand. “Multi-Layer Perceptrons with B-Spline Receptive Field Functions.” In: *Advances in Neural Information Processing Systems*. Ed. by R. Lippmann, J. Moody, and D. Touretzky. Vol. 3. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper_files/paper/1990/file/94f6d7e04a4d452035300f18b984988c-Paper.pdf (cit. on p. 162).

[1222] A. Vaicaitis and J. Dooley. “Segmented Spline Curve Neural Network for Low Latency Digital Predistortion of RF Power Amplifiers.” In: *IEEE Transactions on Microwave Theory and Techniques* 70.11 (Nov. 2022), pp. 4910–4915. DOI: 10.1109/tmtt.2022.3210034. URL: https://doi.org/10.1109/tmtt.2022.3210034 (cit. on p. 162).

[1223] R. G. Kumar and Y. Kumaraswamy. “Spline Activated Neural Network for Classifying Cardiac Arrhythmia.” In: *International Journal of Soft Computing* 9.6 (2014), pp. 377–385. DOI: 10.36478/ijscomp.2014.377.385. URL: https://medwelljournals.com/abstract/?doi=ijscomp.2014.377.385 (cit. on p. 162).

[1224] H. A. Mayer and R. Schwaiger. “Evolution of Cubic Spline Activation Functions for Artificial Neural Networks.” In: *Progress in Artificial Intelligence*. Springer Berlin Heidelberg, 2001, pp. 63–73. DOI: 10.1007/3-540-45329-6_10. URL: https://doi.org/10.1007/3-540-45329-6_10 (cit. on p. 162).

[1225] M. Solazzi, A. Uncini, and F. Piazza. "Neural equalizer with adaptive multidimensional spline activation functions." In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*. IEEE, 2000. DOI: 10.1109/icassp.2000.860155. URL: https://doi.org/10.1109/icassp.2000.860155 (cit. on p. 162).

[1226] S. Neumayer, A. Goujon, P. Bohra, and M. Unser. "Approximation of Lipschitz Functions Using Deep Spline Neural Networks." In: *SIAM Journal on Mathematics of Data Science* 5.2 (May 2023), pp. 306–322. ISSN: 2577-0187. DOI: 10.1137/22m1504573. URL: http://dx.doi.org/10.1137/22M1504573 (cit. on pp. 162, 163).

[1227] S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer, and M. Unser. *Improving Lipschitz-Constrained Neural Networks by Learning Activation Functions*. 2022. DOI: 10.48550/ARXIV.2210.16222. URL: https://arxiv.org/abs/2210.16222 (cit. on p. 162).

[1228] S. Aziznejad and M. Unser. "Deep Spline Networks with Control of Lipschitz Regularity." In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019. DOI: 10.1109/icassp.2019.8682547. URL: http://dx.doi.org/10.1109/ICASSP.2019.8682547 (cit. on p. 162).

[1229] D. Fakhoury, E. Fakhoury, and H. Speleers. "ExSpliNet: An interpretable and expressive spline-based neural network." In: *Neural Networks* 152 (Aug. 2022), pp. 332–346. DOI: 10.1016/j.neunet.2022.04.029. URL: https://doi.org/10.1016/j.neunet.2022.04.029 (cit. on p. 163).

[1230] A. Uncini. "Sound Synthesis by Flexible Activation Function Recurrent Neural Networks." In: *Neural Nets*. Springer Berlin Heidelberg, 2002, pp. 168–177. DOI: 10.1007/3-540-45808-5_19. URL: https://doi.org/10.1007/3-540-45808-5_19 (cit. on p. 163).

[1231] N. Kuzuya and T. Nagao. "Designing B-spline-based Highly Efficient Neural Networks for IoT Applications on Edge Platforms." In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2022. DOI: 10.1109/smc53654.2022.9945478. URL: https://doi.org/10.1109/smc53654.2022.9945478 (cit. on p. 163).

[1232] E. López-Rubio, F. Ortega-Zamorano, E. Domínguez, and J. Muñoz-Pérez. "Piecewise Polynomial Activation Functions for Feedforward Neural Networks." In: *Neural Processing Letters* 50.1 (Jan. 2019), pp. 121–147. DOI: 10.1007/s11063-018-09974-4. URL: https://doi.org/10.1007/s11063-018-09974-4 (cit. on p. 163).

[1233] Q. Su, x. Liao, and L. Carin. "A Probabilistic Framework for Nonlinearities in Stochastic Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/35936504a37d53e03abdfbc7318d9ec7-Paper.pdf (cit. on p. 164).

[1234] J. T. Barron. *Squareplus: A Softplus-Like Algebraic Rectifier*. 2021. DOI: 10.48550/ARXIV.2112.11687. URL: https://arxiv.org/abs/2112.11687 (cit. on pp. lii, 164).

[1235] J.-R. Chang and Y.-S. Chen. *Batch-normalized Maxout Network in Network*. 2015. DOI: 10.48550/ARXIV.1511.02583. URL: https://arxiv.org/abs/1511.02583 (cit. on p. 168).

[1236] M. Wang, B. Liu, and H. Foroosh. "Wide Hidden Expansion Layer for Deep Convolutional Neural Networks." In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020. DOI: 10.1109/wacv45572.2020.9093436. URL: https://doi.org/10.1109/wacv45572.2020.9093436 (cit. on pp. 168, 169).

[1237] C. Eisenach, Z. Wang, and H. Liu. "Nonparametrically Learning Activation Functions in Deep Neural Nets." In: *International Conference on Learning Representations Workshops*. 2017. URL: https://openreview.net/forum?id=H1wgawqxl (cit. on pp. l, 168).

[1238] C. J. Vercellino and W. Y. Wang. "Hyperactivations for Activation Function Exploration." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA, 2017. URL: http://metalearning.ml/2017/papers/metalearn17_vercellino.pdf (cit. on pp. 168, 169).

[1239] S. Zhang, Q. Liu, X. Wu, and W. Chen. "A Self-Adaptive and Multiple Activation Function Neural Network for Facial Expression Recognition." In: *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*. EITCE 2021. ACM, Oct. 2021. DOI: 10.1145/3501409.3501605. URL: http://dx.doi.org/10.1145/3501409.3501605 (cit. on p. 168).

[1240] I. Castelli and E. Trentin. "Semi-unsupervised Weighted Maximum-Likelihood Estimation of Joint Densities for the Co-training of Adaptive Activation Functions." In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 62–71. DOI: 10.1007/978-3-642-28258-4_7. URL: https://doi.org/10.1007/978-3-642-28258-4_7 (cit. on p. 168).

[1241] I. Castelli and E. Trentin. "Supervised and Unsupervised Co-training of Adaptive Activation Functions in Neural Nets." In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 52–61. DOI: 10.1007/978-3-642-28258-4_6. URL: https://doi.org/10.1007/978-3-642-28258-4_6 (cit. on p. 168).

[1242] I. Castelli and E. Trentin. "Combination of supervised and unsupervised learning for training the activation functions of neural networks." In: *Pattern Recognition Letters* 37 (Feb. 2014), pp. 178–191. DOI: 10.1016/j.patrec.2013.06.013. URL: https://doi.org/10.1016/j.patrec.2013.06.013 (cit. on p. 168).

[1243] A. Apicella, F. Isgrò, and R. Prevete. "A simple and efficient architecture for trainable activation functions." In: *Neurocomputing* 370 (Dec. 2019), pp. 1–15. DOI: 10.1016/j.neucom.2019.08.065. URL: https://doi.org/10.1016/j.neucom.2019.08.065 (cit. on pp. 168, 220).

[1244] F. u. A. A. Minhas and A. Asif. *Learning Neural Activations*. 2019. DOI: 10.48550/ARXIV.1912.12187. URL: https://arxiv.org/abs/1912.12187 (cit. on p. 168).

[1245] H. Klopries and A. Schwung. "Flexible Activation Bag: Learning Activation Functions in Autoencoder Networks." In: *2023 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, Apr. 2023. DOI: 10.1109/icit58465.2023.10143113. URL: http://dx.doi.org/10.1109/ICIT58465.2023.10143113 (cit. on p. 169).

[1246] E. Jang, S. Gu, and B. Poole. "Categorical Reparameterization with Gumbel-Softmax." In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rkE3y85ee (cit. on p. 169).

[1247] Ö. F. Ertuğrul. "A novel type of activation function in artificial neural networks: Trained activation function." In: *Neural Networks* 99 (Mar. 2018), pp. 148–157. DOI: 10.1016/j.neunet.2018.01.007. URL: https://doi.org/10.1016/j.neunet.2018.01.007 (cit. on pp. 169, 170, 179, 199, 203, 204).

[1248] S. Scardapane, S. V. Vaerenbergh, S. Totaro, and A. Uncini. "Kafnets: Kernel-based non-parametric activation functions for neural networks." In: *Neural Networks* 110 (Feb. 2019), pp. 19–32. DOI: 10.1016/j.neunet.2018.11.002. URL: https://doi.org/10.1016/j.neunet.2018.11.002 (cit. on pp. 170, 171, 210, 219).

[1249] S. Kiliçarslan and M. Celik. "KAF+RSigELU: a nonlinear and kernel-based activation function for deep neural networks." In: *Neural Computing and Applications* 34.16 (Apr. 2022), pp. 13909–13923. DOI: 10.1007/s00521-022-07211-7. URL: https://doi.org/10.1007/s00521-022-07211-7 (cit. on p. 171).

[1250] W. Zhang, Z. Han, X. Chen, B. Liu, H. Jia, and Y. Tang. "Fully Kernected Neural Networks." In: *Journal of Mathematics* 2023 (June 2023). Ed. by Q. Wu, pp. 1–9. DOI: 10.1155/2023/1539436. URL: https://doi.org/10.1155/2023/1539436 (cit. on p. 171).

[1251] S. Brad. "Enhancing Creativity in Deep Learning Models with SAVE-Inspired Activation Functions." In: *Towards AI-Aided Invention and Innovation*. Springer Nature Switzerland, 2023, pp. 147–171. ISBN: 9783031425325. DOI: 10.1007/978-3-031-42532-5_12. URL: http://dx.doi.org/10.1007/978-3-031-42532-5_12 (cit. on pp. 171, 172).

[1252] S. Brad and E. Ştetco. "An Interactive Artificial Intelligence System for Inventive Problem-Solving." In: *Systematic Innovation Partnerships with Artificial Intelligence and Information Technology*. Springer International Publishing, 2022, pp. 165–177. ISBN: 9783031172885. DOI: 10.1007/978-3-031-17288-5_15. URL: http://dx.doi.org/10.1007/978-3-031-17288-5_15 (cit. on p. 171).

[1253] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber. "Multi-column deep neural network for traffic sign classification." In: *Neural Networks* 32 (Aug. 2012), pp. 333–338. DOI: 10.1016/j.neunet.2012.02.023. URL: https://doi.org/10.1016/j.neunet.2012.02.023 (cit. on p. 171).

[1254] S. Kum, C. Oh, and J. Nam. "Melody Extraction on Vocal Segments Using Multi-Column Deep Neural Networks." In: *ISMIR 2016*. 2016 (cit. on p. 171).

[1255] K. T. Phan, T. H. Maul, and T. T. Vu. "A parallel circuit approach for improving the speed and generalization properties of neural networks." In: *2015 11th International Conference on Natural Computation (ICNC)*. IEEE, Aug. 2015. DOI: 10.1109/icnc.2015.7377956. URL: https://doi.org/10.1109/icnc.2015.7377956 (cit. on pp. 171, 222).

[1256] K. T. Phan, T. H. Maul, and T. T. Vu. "An Empirical Study on Improving the Speed and Generalization of Neural Networks Using a Parallel Circuit Approach." In: *International Journal of Parallel Programming* 45.4 (May 2016), pp. 780–796. DOI: 10.1007/s10766-016-0435-4. URL: https://doi.org/10.1007/s10766-016-0435-4 (cit. on pp. 171, 222).

[1257] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone. "A review on weight initialization strategies for neural networks." In: *Artificial Intelligence Review* 55.1 (June 2021), pp. 291–322. ISSN: 1573-7462. DOI: 10.1007/s10462-021-10033-z. URL: http://dx.doi.org/10.1007/s10462-021-10033-z (cit. on p. 173).

[1258] A. Daniely, R. Frostig, and Y. Singer. "Toward Deeper Understanding of Neural Networks: The Power of Initialization and a Dual View on Expressivity." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/abea47ba24142ed16b7d8fbf2c740e0d-Paper.pdf (cit. on p. 173).

[1259] C. A. R. de Sousa. "An overview on weight initialization methods for feedforward neural networks." In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2016. DOI: 10.1109/ijcnn.2016.7727180. URL: http://dx.doi.org/10.1109/IJCNN.2016.7727180 (cit. on p. 173).

[1260] M. Skorski, A. Temperoni, and M. Theobald. "Revisiting Weight Initialization of Deep Neural Networks." In: *Proceedings of The 13th Asian Conference on Machine Learning*. Ed. by V. N. Balasubramanian and I. Tsang. Vol. 157. Proceedings of Machine Learning Research. PMLR, 17–19 Nov 2021, pp. 1192–1207. URL: https://proceedings.mlr.press/v157/skorski21a.html (cit. on p. 173).

[1261] H. F. Langroudi, C. Merkel, H. Syed, and D. Kudithipudi. "Exploiting Randomness in Deep Learning Algorithms." In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2019. DOI: 10.1109/ijcnn.2019.8852192. URL: https://doi.org/10.1109/ijcnn.2019.8852192 (cit. on p. 173).

[1262] K. He, Y. Wang, and J. Hopcroft. "A Powerful Generative Model Using Random Weights for the Deep Image Representation." In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 631–639. ISBN: 9781510838819 (cit. on pp. 173, 179, 185, 186).

[1263] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang. "A Survey of Accelerator Architectures for Deep Neural Networks." In: *Engineering* 6.3 (Mar. 2020), pp. 264–274. ISSN: 2095-8099. DOI: 10.1016/j.eng.2020.01.007. URL: http://dx.doi.org/10.1016/j.eng.2020.01.007 (cit. on p. 173).

[1264] Z. Li, Y. Wang, T. Zhi, and T. Chen. "A survey of neural network accelerators." In: *Frontiers of Computer Science* 11.5 (May 2017), pp. 746–761. ISSN: 2095-2236. DOI: 10.1007/s11704-016-6159-1. URL: http://dx.doi.org/10.1007/s11704-016-6159-1 (cit. on p. 173).

[1265] S. Mittal. "A survey of FPGA-based accelerators for convolutional neural networks." In: *Neural Computing and Applications* 32.4 (Oct. 2018), pp. 1109–1139. ISSN: 1433-3058. DOI: 10.1007/s00521-018-3761-1. URL: http://dx.doi.org/10.1007/s00521-018-3761-1 (cit. on p. 173).

[1266] R. Ribani and M. Marengoni. "A Survey of Transfer Learning for Convolutional Neural Networks." In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*. IEEE, Oct. 2019. DOI: 10.1109/sibgrapi-t.2019.00010. URL: http://dx.doi.org/10.1109/SIBGRAPI-T.2019.00010 (cit. on p. 173).

[1267] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. *A Comprehensive Survey on Transfer Learning*. 2019. DOI: 10.48550/ARXIV.1911.02685. URL: https://arxiv.org/abs/1911.02685 (cit. on p. 173).

[1268] L. Chato and E. Regentova. "Survey of Transfer Learning Approaches in the Machine Learning of Digital Health Sensing Data." In: *Journal of Personalized Medicine* 13.12 (Dec. 2023), p. 1703. ISSN: 2075-4426. DOI: 10.3390/jpm13121703. URL: http://dx.doi.org/10.3390/jpm13121703 (cit. on p. 173).

[1269] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes. "RMDL: Random Multimodel Deep Learning for Classification." In: *Proceedings of the 2nd International Conference on Information System and Data Mining*. ACM, Apr. 2018. DOI: 10.1145/3206098.3206111. URL: https://doi.org/10.1145/3206098.3206111 (cit. on pp. 173, 179).

[1270] L. Zhang and P. Suganthan. "A survey of randomized algorithms for training neural networks." In: *Information Sciences* 364-365 (Oct. 2016), pp. 146–155. DOI: 10.1016/j.ins.2016.01.039. URL: https://doi.org/10.1016/j.ins.2016.01.039 (cit. on pp. 173, 174).

[1271] E. Ragusa, C. Gianoglio, R. Zunino, and P. Gastaldo. "An approximate randomization-based neural network with dedicated digital architecture for energy-constrained devices." In: *Neural Computing and Applications* 35.9 (Nov. 2022), pp. 6753–6766. DOI: 10.1007/s00521-022-08034-2. URL: https://doi.org/10.1007/s00521-022-08034-2 (cit. on p. 173).

[1272] C. Rössert, P. Dean, and J. Porrill. "At the Edge of Chaos: How Cerebellar Granular Layer Network Dynamics Can Provide the Basis for Temporal Filters." In: *PLOS Computational Biology* 11.10 (Oct. 2015). Ed. by L. J. Graham, e1004515. DOI: 10.1371/journal.pcbi.1004515. URL: https://doi.org/10.1371/journal.pcbi.1004515 (cit. on pp. 173, 177).

[1273] K. Tokuda, N. Fujiwara, A. Sudo, and Y. Katori. "Chaos may enhance expressivity in cerebellar granular layer." In: *Neural Networks* 136 (Apr. 2021), pp. 72–86. DOI: 10.1016/j.neunet.2020.12.020. URL: https://doi.org/10.1016/j.neunet.2020.12.020 (cit. on pp. 173, 177).

[1274] B. Swiderski, S. Osowski, P. Olszewski, L. Gielata, M. Slowinska, and I. Lugowska. "Random Deep Neural Network for Melanoma Recognition." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2021. DOI: 10.1109/ijcnn52387.2021.9533468. URL: http://dx.doi.org/10.1109/IJCNN52387.2021.9533468 (cit. on p. 173).

[1275] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari. "What's Hidden in a Randomly Weighted Neural Network?" In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.01191. URL: https://doi.org/10.1109/cvpr42600.2020.01191 (cit. on pp. 174, 176).

[1276] E. Cambria et al. "Extreme Learning Machines [Trends & Controversies]." In: *IEEE Intelligent Systems* 28.6 (Nov. 2013), pp. 30–59. DOI: 10.1109/mis.2013.140. URL: https://doi.org/10.1109/mis.2013.140 (cit. on p. 174).

[1277] A. Rosenfeld and J. K. Tsotsos. "Intriguing Properties of Randomly Weighted Networks: Generalizing While Learning Next to Nothing." In: *2019 16th Conference on Computer and Robot Vision (CRV)*. IEEE, May 2019. DOI: 10.1109/crv.2019.00010. URL: https://doi.org/10.1109/crv.2019.00010 (cit. on pp. 174, 176).

[1278] G. Huang, G.-B. Huang, S. Song, and K. You. "Trends in extreme learning machines: A review." In: *Neural Networks* 61 (Jan. 2015), pp. 32–48. DOI: 10.1016/j.neunet.2014.10.001. URL: https://doi.org/10.1016/j.neunet.2014.10.001 (cit. on pp. 174–176).

[1279] W. Cao, X. Wang, Z. Ming, and J. Gao. "A review on neural networks with random weights." In: *Neurocomputing* 275 (Jan. 2018), pp. 278–287. DOI: 10.1016/j.neucom.2017.08.040. URL: https://doi.org/10.1016/j.neucom.2017.08.040 (cit. on p. 174).

[1280] A. M. Durán-Rosal, A. Durán-Fernández, F. Fernández-Navarro, and M. Carbonero-Ruz. "A multi-class classification model with parametrized target outputs for randomized-based feedforward neural networks." In: *Applied Soft Computing* 133 (Jan. 2023), p. 109914. DOI: 10.1016/j.asoc.2022.109914. URL: https://doi.org/10.1016/j.asoc.2022.109914 (cit. on p. 174).

[1281] Z. Zhang, F. Feng, and T. Huang. "FNNS: An Effective Feedforward Neural Network Scheme with Random Weights for Processing Large-Scale Datasets." In: *Applied Sciences* 12.23 (Dec. 2022), p. 12478. DOI: 10.3390/app122312478. URL: https://doi.org/10.3390/app122312478 (cit. on pp. 174, 178).

[1282] G. Dudek. "Generating random weights and biases in feedforward neural networks with random hidden nodes." In: *Information Sciences* 481 (May 2019), pp. 33–56. DOI: 10.1016/j.ins.2018.12.063. URL: https://doi.org/10.1016/j.ins.2018.12.063 (cit. on p. 174).

[1283] B. Widrow, A. Greenblatt, Y. Kim, and D. Park. "The No-Prop algorithm: A new learning algorithm for multilayer neural networks." In: *Neural Networks* 37 (Jan. 2013), pp. 182–188. DOI: 10.1016/j.neunet.2012.09.020. URL: https://doi.org/10.1016/j.neunet.2012.09.020 (cit. on pp. 174, 176).

[1284] J. Brauer. "Important Milestones in the Study of Neural Networks with Random Weights." In: (Nov. 2021). DOI: 10.36227/techrxiv.16903219.v1. URL: https://doi.org/10.36227/techrxiv.16903219.v1 (cit. on p. 174).

[1285] S. Scardapane and D. Wang. "Randomness in neural networks: an overview." In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7.2 (Feb. 2017), e1200. DOI: 10.1002/widm.1200. URL: https://doi.org/10.1002/widm.1200 (cit. on pp. 174, 177, 179).

[1286] C. Gallicchio and S. Scardapane. "Deep Randomized Neural Networks." In: *Recent Trends in Learning From Data*. Springer International Publishing, 2020, pp. 43–68. DOI: 10.1007/978-3-030-43883-8_3. URL: https://doi.org/10.1007/978-3-030-43883-8_3 (cit. on p. 174).

[1287] M. Lukoševičius and H. Jaeger. "Reservoir computing approaches to recurrent neural network training." In: *Computer Science Review* 3.3 (Aug. 2009), pp. 127–149. DOI: 10.1016/j.cosrev.2009.03.005. URL: https://doi.org/10.1016/j.cosrev.2009.03.005 (cit. on pp. 174, 177).

[1288] J. Tapson and A. van Schaik. "Learning the pseudoinverse solution to network weights." In: *Neural Networks* 45 (Sept. 2013), pp. 94–100. DOI: 10.1016/j.neunet.2013.02.008. URL: https://doi.org/10.1016/j.neunet.2013.02.008 (cit. on p. 174).

[1289] M. Rigotti. "Internal representation of task rules by recurrent dynamics: the importance of the diversity of neural responses." In: *Frontiers in Computational Neuroscience* 4 (2010). DOI: 10.3389/fncom.2010.00024. URL: https://doi.org/10.3389/fncom.2010.00024 (cit. on p. 174).

[1290] A. Goudarzi and C. Teuscher. "Reservoir Computing." In: *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication*. ACM, Sept. 2016. DOI: 10.1145/2967446.2967448. URL: https://doi.org/10.1145/2967446.2967448 (cit. on pp. 174, 177–179).

[1291] Y.-H. Pao and Y. Takefuji. "Functional-link net computing: theory, system architecture, and functionalities." In: *Computer* 25.5 (May 1992), pp. 76–79. DOI: 10.1109/2.144401. URL: https://doi.org/10.1109/2.144401 (cit. on p. 174).

[1292] B. Igelnik and Y.-H. Pao. "Stochastic choice of basis functions in adaptive function approximation and the functional-link net." In: *IEEE Transactions on Neural Networks* 6.6 (1995), pp. 1320–1329. DOI: 10.1109/72.471375. URL: https://doi.org/10.1109/72.471375 (cit. on p. 174).

[1293] Y.-H. Pao, G.-H. Park, and D. J. Sobajic. "Learning and generalization characteristics of the random vector functional-link net." In: *Neurocomputing* 6.2 (Apr. 1994), pp. 163–180. DOI: 10.1016/0925-2312(94)90053-1. URL: https://doi.org/10.1016/0925-2312(94)90053-1 (cit. on p. 174).

[1294] Y.-H. Pao and S. M. Phillips. "The functional link net and learning optimal control." In: *Neurocomputing* 9.2 (Oct. 1995), pp. 149–164. DOI: 10.1016/0925-2312(95)00066-f. URL: https://doi.org/10.1016/0925-2312(95)00066-f (cit. on p. 174).

[1295] L. Zhang and P. Suganthan. "A comprehensive evaluation of random vector functional link networks." In: *Information Sciences* 367-368 (Nov. 2016), pp. 1094–1105. DOI: 10.1016/j.ins.2015.09.025. URL: https://doi.org/10.1016/j.ins.2015.09.025 (cit. on p. 174).

[1296] Q. Tian, W. Zhao, Y. Wei, and L. Pang. "Thermal Environment Prediction for Metro Stations Based on an RVFL Neural Network." In: *Algorithms* 11.4 (Apr. 2018), p. 49. DOI: 10.3390/a11040049. URL: https://doi.org/10.3390/a11040049 (cit. on p. 174).

[1297] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini. "A semi-supervised random vector functional-link network based on the transductive framework." In: *Information Sciences* 364-365 (Oct. 2016), pp. 156–166. DOI: 10.1016/j.ins.2015.07.060. URL: https://doi.org/10.1016/j.ins.2015.07.060 (cit. on p. 174).

[1298] Q. Shi, P. N. Suganthan, and J. D. Ser. "Jointly optimized ensemble deep random vector functional link network for semi-supervised classification." In: *Engineering Applications of Artificial Intelligence* 115 (Oct. 2022), p. 105214. DOI: 10.1016/j.engappai.2022.105214. URL: https://doi.org/10.1016/j.engappai.2022.105214 (cit. on p. 174).

[1299] J. He, X. Li, P. Liu, L. Wang, H. Zhou, J. Wang, and R. Tang. "Ensemble deep random vector functional link for self-supervised direction-of-arrival estimation." In: *Engineering Applications of Artificial Intelligence* 120 (Apr. 2023), p. 105831. DOI: 10.1016/j.engappai.2023.105831. URL: https://doi.org/10.1016/j.engappai.2023.105831 (cit. on p. 174).

[1300] S. Scardapane, D. Wang, M. Panella, and A. Uncini. "Distributed learning for Random Vector Functional-Link networks." In: *Information Sciences* 301 (Apr. 2015), pp. 271–284. DOI: 10.1016/j.ins.2015.01.007. URL: https://doi.org/10.1016/j.ins.2015.01.007 (cit. on p. 174).

[1301] D. Husmeier and J. G. Taylor. "Neural Networks for Predicting Conditional Probability Densities: Improved Training Scheme Combining EM and RVFL." In: *Neural Networks* 11.1 (Jan. 1998), pp. 89–116. DOI: 10.1016/s0893-6080(97)00089-0. URL: https://doi.org/10.1016/s0893-6080(97)00089-0 (cit. on p. 174).

[1302] B. B. Hazarika and D. Gupta. "Random vector functional link with ϵ-insensitive Huber loss function for biomedical data classification." In: *Computer Methods and Programs in Biomedicine* 215 (Mar. 2022), p. 106622. DOI: 10.1016/j.cmpb.2022.106622. URL: https://doi.org/10.1016/j.cmpb.2022.106622 (cit. on p. 174).

[1303] Y. Zhang, J. Wu, Z. Cai, B. Du, and P. S. Yu. "An unsupervised parameter learning model for RVFL neural network." In: *Neural Networks* 112 (Apr. 2019), pp. 85–97. DOI: 10.1016/j.neunet.2019.01.007. URL: https://doi.org/10.1016/j.neunet.2019.01.007 (cit. on p. 174).

[1304] I. Majumder, P. K. Dash, and R. Bisoi. "Short-term solar power prediction using multi-kernel-based random vector functional link with water cycle algorithm-based parameter optimization." In: *Neural Computing and Applications* 32.12 (June 2019), pp. 8011–8029. DOI: 10.1007/s00521-019-04290-x. URL: https://doi.org/10.1007/s00521-019-04290-x (cit. on p. 174).

[1305] X. Qiu, P. N. Suganthan, and G. A. Amaratunga. "Ensemble incremental learning Random Vector Functional Link network for short-term electric load forecasting." In: *Knowledge-Based Systems* 145 (Apr. 2018), pp. 182–196. DOI: 10.1016/j.knosys.2018.01.015. URL: https://doi.org/10.1016/j.knosys.2018.01.015 (cit. on p. 174).

[1306] B. B. Hazarika and D. Gupta. "Modelling and forecasting of COVID-19 spread using wavelet-coupled random vector functional link networks." In: *Applied Soft Computing* 96 (Nov. 2020), p. 106626. DOI: 10.1016/j.asoc.2020.106626. URL: https://doi.org/10.1016/j.asoc.2020.106626 (cit. on p. 174).

[1307] L. Zhang and P. N. Suganthan. "Visual Tracking With Convolutional Random Vector Functional Link Network." In: *IEEE Transactions on Cybernetics* 47.10 (Oct. 2017), pp. 3243–3253. DOI: 10.1109/tcyb.2016.2588526. URL: https://doi.org/10.1109/tcyb.2016.2588526 (cit. on p. 174).

[1308] K.-K. Xu, H.-X. Li, and H.-D. Yang. "Kernel-Based Random Vector Functional-Link Network for Fast Learning of Spatiotemporal Dynamic Processes." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.5 (May 2019), pp. 1016–1026. DOI: 10.1109/tsmc.2017.2694018. URL: https://doi.org/10.1109/tsmc.2017.2694018 (cit. on p. 174).

[1309] W. Li, D. Wang, and T. Chai. "Multisource Data Ensemble Modeling for Clinker Free Lime Content Estimate in Rotary Kiln Sintering Processes." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.2 (Feb. 2015), pp. 303–314. DOI: 10.1109/tsmc.2014.2332305. URL: https://doi.org/10.1109/tsmc.2014.2332305 (cit. on p. 174).

[1310] P. Zhou, Y. Lv, H. Wang, and T. Chai. "Data-Driven Robust RVFLNs Modeling of a Blast Furnace Iron-Making Process Using Cauchy Distribution Weighted M-Estimation." In: *IEEE Transactions on Industrial Electronics* 64.9 (Sept. 2017), pp. 7141–7151. DOI: 10.1109/tie.2017.2686369. URL: https://doi.org/10.1109/tie.2017.2686369 (cit. on p. 174).

[1311] G. H. Park and Y. H. Pao. "Unconstrained word-based approach for off-line script recognition using density-based random-vector functional-link net." In: *Neurocomputing* 31.1-4 (Mar. 2000), pp. 45–65. DOI: 10.1016/s0925-2312(99)00149-6. URL: https://doi.org/10.1016/s0925-2312(99)00149-6 (cit. on p. 174).

[1312] M. Pratama, P. P. Angelov, E. Lughofer, and M. J. Er. "Parsimonious random vector functional link network for data streams." In: *Information Sciences* 430-431 (Mar. 2018), pp. 519–537. DOI: 10.1016/j.ins.2017.11.050. URL: https://doi.org/10.1016/j.ins.2017.11.050 (cit. on p. 174).

[1313] M. Alhamdoosh and D. Wang. "Fast decorrelated neural network ensembles with random weights." In: *Information Sciences* 264 (Apr. 2014), pp. 104–117. DOI: 10.1016/j.ins.2013.12.016. URL: https://doi.org/10.1016/j.ins.2013.12.016 (cit. on pp. 174, 175).

[1314] F. Cao, D. Wang, H. Zhu, and Y. Wang. "An iterative learning algorithm for feedforward neural networks with random weights." In: *Information Sciences* 328 (Jan. 2016), pp. 546–557. DOI: 10.1016/j.ins.2015.09.002. URL: https://doi.org/10.1016/j.ins.2015.09.002 (cit. on p. 175).

[1315] W. Schmidt, M. Kraaijveld, and R. Duin. "Feedforward neural networks with random weights." In: *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*. IEEE Comput. Soc. Press, 1992. DOI: 10.1109/icpr.1992.201708. URL: http://dx.doi.org/10.1109/ICPR.1992.201708 (cit. on p. 175).

[1316] D. Wang and M. Li. "Stochastic Configuration Networks: Fundamentals and Algorithms." In: *IEEE Transactions on Cybernetics* 47.10 (Oct. 2017), pp. 3466–3479. DOI: 10.1109/tcyb.2017.2734043. URL: https://doi.org/10.1109/tcyb.2017.2734043 (cit. on p. 175).

[1317] Z. Tian and H. Zhang. "Stochastic configuration networks with fast implementations." In: *Review of Scientific Instruments* 92.12 (Dec. 2021), p. 125109. DOI: 10.1063/5.0077044. URL: https://doi.org/10.1063/5.0077044 (cit. on p. 175).

[1318] W. Wang and D. Wang. "Prediction of component concentrations in sodium aluminate liquor using stochastic configuration networks." In: *Neural Computing and Applications* 32.17 (Feb. 2020), pp. 13625–13638. DOI: 10.1007/s00521-020-04771-4. URL: https://doi.org/10.1007/s00521-020-04771-4 (cit. on p. 175).

[1319] W. Li, Z. Zeng, H. Qu, and C. Sun. "A Novel Fiber Intrusion Signal Recognition Method for OFPS Based on SCN With Dropout." In: *Journal of Lightwave Technology* 37.20 (Oct. 2019), pp. 5221–5230. DOI: 10.1109/jlt.2019.2930624. URL: https://doi.org/10.1109/jlt.2019.2930624 (cit. on p. 175).

[1320] Y. Zhao, X. Deng, and S. Li. "A nonlinear industrial soft sensor modeling method based on locality preserving stochastic configuration network with utilizing unlabeled samples." In: *ISA Transactions* (Apr. 2023). DOI: 10.1016/j.isatra.2023.04.012. URL: https://doi.org/10.1016/j.isatra.2023.04.012 (cit. on p. 175).

[1321] K. Li, W. Wang, and S. Lin. "Soft measurement of ammonia nitrogen concentration based on GA-SCN." In: *2018 IEEE Symposium on Product Compliance Engineering - Asia (ISPCE-CN)*. IEEE, Dec. 2018. DOI: 10.1109/ispce-cn.2018.8805767. URL: https://doi.org/10.1109/ispce-cn.2018.8805767 (cit. on p. 175).

[1322] W. Wang, K. Li, and G. Guo. "Seawater Ammonia Nitrogen Concentration Modelling via RS-SCN." In: *2019 Chinese Automation Congress (CAC)*. IEEE, Nov. 2019. DOI: 10.1109/cac48633.2019.8996654. URL: https://doi.org/10.1109/cac48633.2019.8996654 (cit. on p. 175).

[1323] S. Li, X. Deng, P. Wang, and Y. Cao. "A Pruning Regularization Stochastic Configuration Network Algorithm Based on Node Contribution and Its Application in Soft Sensor Development." In: *2022 34th Chinese Control and Decision Conference (CCDC)*. IEEE, Aug. 2022. DOI: 10.1109/ccdc55256.2022.10033864. URL: https://doi.org/10.1109/ccdc55256.2022.10033864 (cit. on p. 175).

[1324] Q. Zhang, W. Li, H. Li, and J. Wang. "Self-blast state detection of glass insulators based on stochastic configuration networks and a feedback transfer learning mechanism." In: *Information Sciences* 522 (June 2020), pp. 259–274. DOI: 10.1016/j.ins.2020.02.058. URL: https://doi.org/10.1016/j.ins.2020.02.058 (cit. on p. 175).

[1325] W. Li, Q. Zhang, D. Wang, W. Sun, and Q. Li. "Stochastic configuration networks for self-blast state recognition of glass insulators with adaptive depth and multi-scale representation." In: *Information Sciences* 604 (Aug. 2022), pp. 61–79. DOI: 10.1016/j.ins.2022.04.061. URL: https://doi.org/10.1016/j.ins.2022.04.061 (cit. on p. 175).

[1326] W. Li, Y. Deng, M. Ding, D. Wang, W. Sun, and Q. Li. "Industrial data classification using stochastic configuration networks with self-attention learning features." In: *Neural Computing and Applications* 34.24 (Aug. 2022), pp. 22047–22069. DOI: 10.1007/s00521-022-07657-9. URL: https://doi.org/10.1007/s00521-022-07657-9 (cit. on p. 175).

[1327] Y. Han, X. Song, K. Li, and X. Yan. "Hybrid modeling for submergence depth of the pumping well using stochastic configuration networks with random sampling." In: *Journal of Petroleum Science and Engineering* 208 (Jan. 2022), p. 109423. DOI: 10.1016/j.petrol.2021.109423. URL: https://doi.org/10.1016/j.petrol.2021.109423 (cit. on p. 175).

[1328] F. Chen, W. Tian, L. Zhang, J. Li, C. Ding, D. Chen, W. Wang, F. Wu, and B. Wang. "Fault Diagnosis of Power Transformer Based on Time-Shift Multiscale Bubble Entropy and Stochastic Configuration Network." In: *Entropy* 24.8 (Aug. 2022), p. 1135. DOI: 10.3390/e24081135. URL: https://doi.org/10.3390/e24081135 (cit. on p. 175).

[1329] C. Huang, M. Li, and D. Wang. "Stochastic configuration network ensembles with selective base models." In: *Neural Networks* 137 (May 2021), pp. 106–118. DOI: 10.1016/j.neunet.2021.01.011. URL: https://doi.org/10.1016/j.neunet.2021.01.011 (cit. on p. 175).

[1330] C. Zhang, S. Ding, and W. Du. "Broad stochastic configuration network for regression." In: *Knowledge-Based Systems* 243 (May 2022), p. 108403. DOI: 10.1016/j.knosys.2022.108403. URL: https://doi.org/10.1016/j.knosys.2022.108403 (cit. on p. 175).

[1331] D. Wang and C. Cui. "Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics." In: *Information Sciences* 417 (Nov. 2017), pp. 55–71. DOI: 10.1016/j.ins.2017.07.003. URL: https://doi.org/10.1016/j.ins.2017.07.003 (cit. on p. 175).

[1332] J. Liu, Y. Liu, and Q. Zhang. "A Gradient-Based Particle-Bat Algorithm for Stochastic Configuration Network." In: *Applied Sciences* 13.5 (Feb. 2023), p. 2878. DOI: 10.3390/app13052878. URL: https://doi.org/10.3390/app13052878 (cit. on p. 175).

[1333] J. Lu and J. Ding. "Mixed-Distribution-Based Robust Stochastic Configuration Networks for Prediction Interval Construction." In: *IEEE Transactions on Industrial Informatics* 16.8 (Aug. 2020), pp. 5099–5109. DOI: 10.1109/tii.2019.2954351. URL: https://doi.org/10.1109/tii.2019.2954351 (cit. on p. 175).

[1334] W. Dai, C. Ning, S. Pei, S. Zhu, and X. Wang. "Orthogonal stochastic configuration networks with adaptive construction parameter for data analytics." In: *Industrial Artificial Intelligence* 1.1 (Mar. 2023). DOI: 10.1007/s44244-023-00004-4. URL: https://doi.org/10.1007/s44244-023-00004-4 (cit. on p. 175).

[1335] J. Guo, A. Yan, and J. Tang. "A robust transfer deep stochastic configuration network for industrial data modeling." In: *Industrial Artificial Intelligence* 1.1 (Mar. 2023). DOI: 10.1007/s44244-023-00003-5. URL: https://doi.org/10.1007/s44244-023-00003-5 (cit. on p. 175).

[1336] M. Li, C. Huang, and D. Wang. "Robust stochastic configuration networks with maximum correntropy criterion for uncertain data regression." In: *Information Sciences* 473 (Jan. 2019), pp. 73–86. DOI: 10.1016/j.ins.2018.09.026. URL: https://doi.org/10.1016/j.ins.2018.09.026 (cit. on p. 175).

[1337] D. Wang and M. Li. "Robust stochastic configuration networks with kernel density estimation for uncertain data regression." In: *Information Sciences* 412-413 (Oct. 2017), pp. 210–222. DOI: 10.1016/j.ins.2017.05.047. URL: https://doi.org/10.1016/j.ins.2017.05.047 (cit. on p. 175).

[1338] H. Wu, A. Zhang, Y. Han, J. Nan, and K. Li. "Fast stochastic configuration network based on an improved sparrow search algorithm for fire flame recognition." In: *Knowledge-Based Systems* 245 (June 2022), p. 108626. DOI: 10.1016/j.knosys.2022.108626. URL: https://doi.org/10.1016/j.knosys.2022.108626 (cit. on p. 175).

[1339] W. Cao, Z. Xie, J. Li, Z. Xu, Z. Ming, and X. Wang. "Bidirectional stochastic configuration network for regression problems." In: *Neural Networks* 140 (Aug. 2021), pp. 237–246. DOI: 10.1016/j.neunet.2021.03.016. URL: https://doi.org/10.1016/j.neunet.2021.03.016 (cit. on p. 175).

[1340] C. Zhang and S. Ding. "A stochastic configuration network based on chaotic sparrow search algorithm." In: *Knowledge-Based Systems* 220 (May 2021), p. 106924. DOI: 10.1016/j.knosys.2021.106924. URL: https://doi.org/10.1016/j.knosys.2021.106924 (cit. on p. 175).

[1341] D. Wang and M. Li. “Deep Stochastic Configuration Networks with Universal Approximation Property.” In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2018. DOI: 10.1109/ijcnn.2018.8489695. URL: https://doi.org/10.1109/ijcnn.2018.8489695 (cit. on p. 175).

[1342] C. Zhang, S. Ding, and L. Ding. “An AdaBoost Based - Deep Stochastic Configuration Network.” In: *IFIP Advances in Information and Communication Technology*. Springer International Publishing, 2022, pp. 3–14. DOI: 10.1007/978-3-031-03948-5_1. URL: https://doi.org/10.1007/978-3-031-03948-5_1 (cit. on p. 175).

[1343] H. Zheng, D. Wang, and W. Zhou. “A modified Bayesian neural network integrating stochastic configuration network and ensemble learning strategy.” In: *2021 8th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*. IEEE, Dec. 2021. DOI: 10.1109/iccss53909.2021.9721995. URL: https://doi.org/10.1109/iccss53909.2021.9721995 (cit. on p. 175).

[1344] Y. Gao, F. Luan, J. Pan, X. Li, and Y. He. “FPGA-Based Implementation of Stochastic Configuration Networks for Regression Prediction.” In: *Sensors* 20.15 (July 2020), p. 4191. DOI: 10.3390/s20154191. URL: https://doi.org/10.3390/s20154191 (cit. on p. 175).

[1345] P. A. Alaba, S. I. Popoola, L. Olatomiwa, M. B. Akanle, O. S. Ohunakin, E. Adetiba, O. D. Alex, A. A. Atayero, and W. M. A. W. Daud. “Towards a more efficient and cost-sensitive extreme learning machine: A state-of-the-art review of recent trend.” In: *Neurocomputing* 350 (July 2019), pp. 70–90. DOI: 10.1016/j.neucom.2019.03.086. URL: https://doi.org/10.1016/j.neucom.2019.03.086 (cit. on pp. 175, 176).

[1346] G.-B. Huang, L. Chen, and C.-K. Siew. “Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes.” In: *IEEE Transactions on Neural Networks* 17.4 (July 2006), pp. 879–892. DOI: 10.1109/tnn.2006.875977. URL: https://doi.org/10.1109/tnn.2006.875977 (cit. on p. 175).

[1347] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu. “Universal Approximation of Extreme Learning Machine With Adaptive Growth of Hidden Nodes.” In: *IEEE Transactions on Neural Networks and Learning Systems* 23.2 (Feb. 2012), pp. 365–371. DOI: 10.1109/tnnls.2011.2178124. URL: https://doi.org/10.1109/tnnls.2011.2178124 (cit. on p. 175).

[1348] G. Huang, S. Song, J. N. D. Gupta, and C. Wu. “Semi-Supervised and Unsupervised Extreme Learning Machines.” In: *IEEE Transactions on Cybernetics* 44.12 (Dec. 2014), pp. 2405–2417. DOI: 10.1109/tcyb.2014.2307349. URL: https://doi.org/10.1109/tcyb.2014.2307349 (cit. on p. 175).

[1349] E. Malar, A. Kandaswamy, D. Chakravarthy, and A. G. Dharan. “A novel approach for detection and classification of mammographic microcalcifications using wavelet analysis and extreme learning machine.” In: *Computers in Biology and Medicine* 42.9 (Sept. 2012), pp. 898–905. DOI: 10.1016/j.compbiomed.2012.07.001. URL: https://doi.org/10.1016/j.compbiomed.2012.07.001 (cit. on p. 175).

[1350] W. Jun, W. Shitong, and F.-l. Chung. “Positive and negative fuzzy rule system, extreme learning machine and image classification.” In: *International Journal of Machine Learning and Cybernetics* 2.4 (June 2011), pp. 261–271. DOI: 10.1007/s13042-011-0024-1. URL: https://doi.org/10.1007/s13042-011-0024-1 (cit. on p. 175).

[1351] W. Zong and G.-B. Huang. “Face recognition based on extreme learning machine.” In: *Neurocomputing* 74.16 (Sept. 2011), pp. 2541–2551. DOI: 10.1016/j.neucom.2010.12.041. URL: https://doi.org/10.1016/j.neucom.2010.12.041 (cit. on p. 175).

[1352] A. Baradarani, Q. J. Wu, and M. Ahmadi. “An efficient illumination invariant face recognition framework via illumination enhancement and DD-DTWT filtering.” In: *Pattern Recognition* 46.1 (Jan. 2013), pp. 57–72. DOI: 10.1016/j.patcog.2012.06.007. URL: https://doi.org/10.1016/j.patcog.2012.06.007 (cit. on p. 175).

[1353] K. Choi, K.-A. Toh, and H. Byun. “Incremental face recognition for large-scale social network services.” In: *Pattern Recognition* 45.8 (Aug. 2012), pp. 2868–2883. DOI: 10.1016/j.patcog.2012.02.002. URL: https://doi.org/10.1016/j.patcog.2012.02.002 (cit. on p. 175).

[1354] B. He, D. Xu, R. Nian, M. van Heeswijk, Q. Yu, Y. Miche, and A. Lendasse. “Fast Face Recognition Via Sparse Coding and Extreme Learning Machine.” In: *Cognitive Computation* (July 2013). DOI: 10.1007/s12559-013-9224-1. URL: https://doi.org/10.1007/s12559-013-9224-1 (cit. on p. 175).

[1355] I. Marqués and M. Graña. “Fusion of lattice independent and linear features improving face identification.” In: *Neurocomputing* 114 (Aug. 2013), pp. 80–85. DOI: 10.1016/j.neucom.2012.06.045. URL: https://doi.org/10.1016/j.neucom.2012.06.045 (cit. on p. 175).

[1356] R. Nian, B. He, and A. Lendasse. “3D object recognition based on a geometrical topology model and extreme learning machine.” In: *Neural Computing and Applications* 22.3-4 (Mar. 2012), pp. 427–433. DOI: 10.1007/s00521-012-0892-7. URL: https://doi.org/10.1007/s00521-012-0892-7 (cit. on p. 175).

[1357] Y. Chen, Z. Zhao, S. Wang, and Z. Chen. "Extreme learning machine-based device displacement free activity recognition model." In: *Soft Computing* 16.9 (Feb. 2012), pp. 1617–1625. DOI: 10.1007/s00500-012-0822-8. URL: https://doi.org/10.1007/s00500-012-0822-8 (cit. on p. 175).

[1358] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. M. J. Wu. "Human action recognition using extreme learning machine based on visual vocabularies." In: *Neurocomputing* 73.10-12 (June 2010), pp. 1906–1917. DOI: 10.1016/j.neucom.2010.01.020. URL: https://doi.org/10.1016/j.neucom.2010.01.020 (cit. on p. 175).

[1359] R. Minhas, A. A. Mohammed, and Q. M. J. Wu. "Incremental Learning in Human Action Recognition Based on Snippets." In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.11 (Nov. 2012), pp. 1529–1541. DOI: 10.1109/tcsvt.2011.2177182. URL: https://doi.org/10.1109/tcsvt.2011.2177182 (cit. on p. 175).

[1360] D. Lu, Y. Yu, and H. Liu. "Gesture recognition using data glove: An extreme learning machine method." In: *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, Dec. 2016. DOI: 10.1109/robio.2016.7866514. URL: http://dx.doi.org/10.1109/ROBIO.2016.7866514 (cit. on p. 175).

[1361] F. Peng, C. Chen, D. Lv, N. Zhang, X. Wang, X. Zhang, and Z. Wang. "Gesture Recognition by Ensemble Extreme Learning Machine Based on Surface Electromyography Signals." In: *Frontiers in Human Neuroscience* 16 (June 2022). ISSN: 1662-5161. DOI: 10.3389/fnhum.2022.911204. URL: http://dx.doi.org/10.3389/fnhum.2022.911204 (cit. on p. 175).

[1362] S. Das, T. P. Sahu, and R. R. Janghel. "Stock market forecasting using intrinsic time-scale decomposition in fusion with cluster based modified CSA optimized ELM." In: *Journal of King Saud University - Computer and Information Sciences* 34.10 (Nov. 2022), pp. 8777–8793. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.10.004. URL: http://dx.doi.org/10.1016/j.jksuci.2021.10.004 (cit. on p. 175).

[1363] F. Chen and T. Ou. "Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry." In: *Expert Systems with Applications* 38.3 (Mar. 2011), pp. 1336–1345. DOI: 10.1016/j.eswa.2010.07.014. URL: https://doi.org/10.1016/j.eswa.2010.07.014 (cit. on p. 175).

[1364] Y. Feng, W. Yao-nan, and Y. Yi-min. "Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace." In: *International Journal of Computers Communications & Control* 7.3 (Sept. 2014), p. 459. DOI: 10.15837/ijccc.2012.3.1387. URL: https://doi.org/10.15837/ijccc.2012.3.1387 (cit. on p. 175).

[1365] A. Lemme, A. Freire, G. Barreto, and J. Steil. "Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot iCub." In: *Neurocomputing* 112 (July 2013), pp. 179–188. DOI: 10.1016/j.neucom.2012.12.040. URL: https://doi.org/10.1016/j.neucom.2012.12.040 (cit. on p. 175).

[1366] E. M. Kan, M. H. Lim, Y. S. Ong, A. H. Tan, and S. P. Yeo. "Extreme learning machine terrain-based navigation for unmanned aerial vehicles." In: *Neural Computing and Applications* 22.3-4 (Feb. 2012), pp. 469–477. DOI: 10.1007/s00521-012-0866-9. URL: https://doi.org/10.1007/s00521-012-0866-9 (cit. on p. 175).

[1367] W. Sun and M. Liu. "Wind speed forecasting using FEEMD echo state networks with RELM in Hebei, China." In: *Energy Conversion and Management* 114 (Apr. 2016), pp. 197–208. DOI: 10.1016/j.enconman.2016.02.022. URL: https://doi.org/10.1016/j.enconman.2016.02.022 (cit. on p. 175).

[1368] H. Fu, Z. Niu, C. Zhang, H. Yu, J. Ma, J. Chen, Y. Chen, and J. Liu. "ASELM: Adaptive semi-supervised ELM with application in question subjectivity identification." In: *Neurocomputing* 207 (Sept. 2016), pp. 599–609. DOI: 10.1016/j.neucom.2016.05.041. URL: https://doi.org/10.1016/j.neucom.2016.05.041 (cit. on pp. 175, 176).

[1369] B. B. Hazarika, D. Gupta, and M. Berlin. "A coiflet LDMR and coiflet OB-ELM for river suspended sediment load prediction." In: *International Journal of Environmental Science and Technology* 18.9 (Oct. 2020), pp. 2675–2692. DOI: 10.1007/s13762-020-02967-8. URL: https://doi.org/10.1007/s13762-020-02967-8 (cit. on p. 176).

[1370] B. B. Hazarika, D. Gupta, and M. Berlin. "Modeling suspended sediment load in a river using extreme learning machine and twin support vector regression with wavelet conjunction." In: *Environmental Earth Sciences* 79.10 (May 2020). DOI: 10.1007/s12665-020-08949-w. URL: https://doi.org/10.1007/s12665-020-08949-w (cit. on p. 176).

[1371] S. Haidong, J. Hongkai, L. Xingqiu, and W. Shuaipeng. "Intelligent fault diagnosis of rolling bearing using deep wavelet auto-encoder with extreme learning machine." In: *Knowledge-Based Systems* 140 (Jan. 2018), pp. 1–14. DOI: 10.1016/j.knosys.2017.10.024. URL: https://doi.org/10.1016/j.knosys.2017.10.024 (cit. on p. 176).

[1372] X. Li and W. Jin. “A method for diagnosing rolling bearing faults based on SDAE-ADHKELM.” In: *Measurement Science and Technology* 34.2 (Nov. 2022), p. 025004. DOI: 10.1088/1361-6501/ac9709. URL: https://doi.org/10.1088/1361-6501/ac9709 (cit. on p. 176).

[1373] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (cit. on pp. 176, 282).

[1374] X. W. Jiang, T. H. Yan, J. J. Zhu, B. He, W. H. Li, H. P. Du, and S. S. Sun. “Densely Connected Deep Extreme Learning Machine Algorithm.” In: *Cognitive Computation* 12.5 (Aug. 2020), pp. 979–990. DOI: 10.1007/s12559-020-09752-2. URL: https://doi.org/10.1007/s12559-020-09752-2 (cit. on p. 176).

[1375] L. Zhang and D. Zhang. “Domain Adaptation Extreme Learning Machines for Drift Compensation in E-Nose Systems.” In: *IEEE Transactions on Instrumentation and Measurement* 64.7 (July 2015), pp. 1790–1801. DOI: 10.1109/tim.2014.2367775. URL: https://doi.org/10.1109/tim.2014.2367775 (cit. on p. 176).

[1376] H. Dai, J. Cao, T. Wang, M. Deng, and Z. Yang. “Multilayer one-class extreme learning machine.” In: *Neural Networks* 115 (July 2019), pp. 11–22. DOI: 10.1016/j.neunet.2019.03.004. URL: https://doi.org/10.1016/j.neunet.2019.03.004 (cit. on p. 176).

[1377] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie. “Extreme learning machine: algorithm, theory and applications.” In: *Artificial Intelligence Review* 44.1 (Apr. 2013), pp. 103–115. DOI: 10.1007/s10462-013-9405-z. URL: https://doi.org/10.1007/s10462-013-9405-z (cit. on p. 176).

[1378] G.-B. Huang, N. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan. “On-Line Sequential Extreme Learning Machine.” In: vol. 2005. Jan. 2005, pp. 232–237 (cit. on p. 176).

[1379] S. Zhang, W. Tan, and Y. Li. “A Survey of Online Sequential Extreme Learning Machine.” In: *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, Apr. 2018. DOI: 10.1109/codit.2018.8394791. URL: https://doi.org/10.1109/codit.2018.8394791 (cit. on p. 176).

[1380] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. “A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks.” In: *IEEE Transactions on Neural Networks* 17.6 (Nov. 2006), pp. 1411–1423. DOI: 10.1109/tnn.2006.880583. URL: https://doi.org/10.1109/tnn.2006.880583 (cit. on p. 176).

[1381] W. Cao, Z. Ming, Z. Xu, J. Zhang, and Q. Wang. “Online Sequential Extreme Learning Machine With Dynamic Forgetting Factor.” In: *IEEE Access* 7 (2019), pp. 179746–179757. DOI: 10.1109/access.2019.2959032. URL: https://doi.org/10.1109/access.2019.2959032 (cit. on p. 176).

[1382] W. Cao, J. Gao, Z. Ming, S. Cai, and Z. Shan. “Fuzziness-based online sequential extreme learning machine for classification problems.” In: *Soft Computing* 22.11 (Feb. 2018), pp. 3487–3494. DOI: 10.1007/s00500-018-3021-4. URL: https://doi.org/10.1007/s00500-018-3021-4 (cit. on p. 176).

[1383] G.-B. Huang and L. Chen. “Convex incremental extreme learning machine.” In: *Neurocomputing* 70.16-18 (Oct. 2007), pp. 3056–3062. DOI: 10.1016/j.neucom.2007.02.009. URL: https://doi.org/10.1016/j.neucom.2007.02.009 (cit. on p. 176).

[1384] G. Feng, Z. Qian, and X. Zhang. “Evolutionary selection extreme learning machine optimization for regression.” In: *Soft Computing* 16.9 (Feb. 2012), pp. 1485–1491. DOI: 10.1007/s00500-012-0823-7. URL: https://doi.org/10.1007/s00500-012-0823-7 (cit. on p. 176).

[1385] G. Feng, G.-B. Huang, Q. Lin, and R. Gay. “Error Minimized Extreme Learning Machine With Growth of Hidden Nodes and Incremental Learning.” In: *IEEE Transactions on Neural Networks* 20.8 (Aug. 2009), pp. 1352–1357. DOI: 10.1109/tnn.2009.2024147. URL: https://doi.org/10.1109/tnn.2009.2024147 (cit. on p. 176).

[1386] E. Soria-Olivas, J. Gómez-Sanchis, J. D. Martin, J. Vila-Francés, M. Martinez, J. R. Magdalena, and A. J. Serrano. “BELM: Bayesian Extreme Learning Machine.” In: *IEEE Transactions on Neural Networks* 22.3 (Mar. 2011), pp. 505–509. DOI: 10.1109/tnn.2010.2103956. URL: https://doi.org/10.1109/tnn.2010.2103956 (cit. on p. 176).

[1387] Y. Wang, Z. Yu, T. Sivanagaraja, and K. C. Veluvolu. “Fast and accurate online sequential learning of respiratory motion with random convolution nodes for radiotherapy applications.” In: *Applied Soft Computing* 95 (Oct. 2020), p. 106528. DOI: 10.1016/j.asoc.2020.106528. URL: https://doi.org/10.1016/j.asoc.2020.106528 (cit. on p. 176).

[1388] H. Zhou, J. Lan, R. Liu, and J. Yosinski. “Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask.” In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019 (cit. on p. 176).

[1389] J. Frankle and M. Carbin. “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.” In: *ICLR*. OpenReview.net, 2019. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2019.html#FrankleC19 (cit. on p. 176).

[1390] E. Hoffer, I. Hubara, and D. Soudry. “Fix your classifier: the marginal value of training the last weight layer.” In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=S1Dh8Tg0- (cit. on p. 177).

[1391] R. Pascanu, T. Mikolov, and Y. Bengio. “On the difficulty of training recurrent neural networks.” In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1310–1318. URL: https://proceedings.mlr.press/v28/pascanu13.html (cit. on p. 177).

[1392] B. Schrauwen, D. Verstraeten, and J. Campenhout. “An overview of reservoir computing: Theory, applications and implementations.” In: Jan. 2007, pp. 471–482 (cit. on p. 177).

[1393] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. “Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach.” In: *Physical Review Letters* 120.2 (Jan. 2018). DOI: 10.1103/physrevlett.120.024102. URL: https://doi.org/10.1103/physrevlett.120.024102 (cit. on p. 177).

[1394] H. Jaeger and H. Haas. “Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication.” In: *Science* 304.5667 (Apr. 2004), pp. 78–80. DOI: 10.1126/science.1091277. URL: https://doi.org/10.1126/science.1091277 (cit. on p. 177).

[1395] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert. “Optimization and applications of echo state networks with leaky- integrator neurons.” In: *Neural Networks* 20.3 (Apr. 2007), pp. 335–352. DOI: 10.1016/j.neunet.2007.04.016. URL: https://doi.org/10.1016/j.neunet.2007.04.016 (cit. on pp. 177, 178).

[1396] N. Rodriguez, E. Izquierdo, and Y.-Y. Ahn. “Optimal modularity and memory capacity of neural reservoirs.” In: *Network Neuroscience* 3.2 (Jan. 2019), pp. 551–566. DOI: 10.1162/netn_a_00082. URL: https://doi.org/10.1162/netn_a_00082 (cit. on p. 177).

[1397] T. Strauss, W. Wustlich, and R. Labahn. “Design Strategies for Weight Matrices of Echo State Networks.” In: *Neural Computation* 24.12 (Dec. 2012), pp. 3246–3276. DOI: 10.1162/neco_a_00374. URL: https://doi.org/10.1162/neco_a_00374 (cit. on p. 177).

[1398] I. B. Yildiz, H. Jaeger, and S. J. Kiebel. “Re-visiting the echo state property.” In: *Neural Networks* 35 (Nov. 2012), pp. 1–9. DOI: 10.1016/j.neunet.2012.07.005. URL: https://doi.org/10.1016/j.neunet.2012.07.005 (cit. on p. 177).

[1399] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. “Training Recurrent Networks by Evolino.” In: *Neural Computation* 19.3 (Mar. 2007), pp. 757–779. DOI: 10.1162/neco.2007.19.3.757. URL: https://doi.org/10.1162/neco.2007.19.3.757 (cit. on p. 177).

[1400] W. Maass. “Liquid State Machines: Motivation, Theory, and Applications.” In: *Computability in Context*. IMPERIAL COLLEGE PRESS, Feb. 2011, pp. 275–296. DOI: 10.1142/9781848162778_0008. URL: https://doi.org/10.1142/9781848162778_0008 (cit. on p. 177).

[1401] W. Maass, P. Joshi, and E. D. Sontag. “Computational Aspects of Feedback in Neural Circuits.” In: *PLoS Computational Biology* 3.1 (Jan. 2007). Ed. by R. Kotter, e165. DOI: 10.1371/journal.pcbi.0020165. URL: https://doi.org/10.1371/journal.pcbi.0020165 (cit. on p. 177).

[1402] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. V. Campenhout. “Isolated word recognition with the Liquid State Machine: a case study.” In: *Information Processing Letters* 95.6 (Sept. 2005), pp. 521–528. DOI: 10.1016/j.ipl.2005.05.019. URL: https://doi.org/10.1016/j.ipl.2005.05.019 (cit. on p. 177).

[1403] W. Maass, T. Natschläger, and H. Markram. “Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations.” In: *Neural Computation* 14.11 (Nov. 2002), pp. 2531–2560. DOI: 10.1162/089976602760407955. URL: https://doi.org/10.1162/089976602760407955 (cit. on p. 177).

[1404] T. Miconi. “Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks.” In: *eLife* 6 (Feb. 2017). DOI: 10.7554/elife.20899. URL: https://doi.org/10.7554/elife.20899 (cit. on p. 177).

[1405] T. Yamazaki and S. Tanaka. “The cerebellum as a liquid state machine.” In: *Neural Networks* 20.3 (Apr. 2007), pp. 290–297. DOI: 10.1016/j.neunet.2007.04.004. URL: https://doi.org/10.1016/j.neunet.2007.04.004 (cit. on p. 177).

[1406] J. J. Steil. “Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning.” In: *Neural Networks* 20.3 (Apr. 2007), pp. 353–364. DOI: 10.1016/j.neunet.2007.04.011. URL: https://doi.org/10.1016/j.neunet.2007.04.011 (cit. on p. 177).

[1407] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt. “Improving reservoirs using intrinsic plasticity.” In: *Neurocomputing* 71.7-9 (Mar. 2008), pp. 1159–1171. DOI: 10.1016/j.neucom.2007.12.020. URL: https://doi.org/10.1016/j.neucom.2007.12.020 (cit. on pp. 177, 178).

[1408] I. Farkaš, R. Bosák, and P. Gergeľ. “Computational analysis of memory capacity in echo state networks.” In: *Neural Networks* 83 (Nov. 2016), pp. 109–120. DOI: 10.1016/j.neunet.2016.07.012. URL: https://doi.org/10.1016/j.neunet.2016.07.012 (cit. on p. 177).

[1409] E. Wallace, H. R. Maei, and P. E. Latham. “Randomly Connected Networks Have Short Temporal Memory.” In: *Neural Computation* 25.6 (June 2013), pp. 1408–1439. DOI: 10.1162/neco_a_00449. URL: https://doi.org/10.1162/neco_a_00449 (cit. on p. 177).

[1410] N. Bertschinger and T. Natschläger. “Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks.” In: *Neural Computation* 16.7 (July 2004), pp. 1413–1436. DOI: 10.1162/089976604323057443. URL: https://doi.org/10.1162/089976604323057443 (cit. on p. 177).

[1411] P. Barančok and I. Farkaš. “Memory Capacity of Input-Driven Echo State Networks at the Edge of Chaos.” In: *Artificial Neural Networks and Machine Learning – ICANN 2014*. Springer International Publishing, 2014, pp. 41–48. DOI: 10.1007/978-3-319-11179-7_6. URL: https://doi.org/10.1007/978-3-319-11179-7_6 (cit. on p. 177).

[1412] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada. “Information processing in echo state networks at the edge of chaos.” In: *Theory in Biosciences* 131.3 (Dec. 2011), pp. 205–213. DOI: 10.1007/s12064-011-0146-8. URL: https://doi.org/10.1007/s12064-011-0146-8 (cit. on p. 177).

[1413] L. Büsing, B. Schrauwen, and R. Legenstein. “Connectivity, Dynamics, and Memory in Reservoir Computing with Binary and Analog Neurons.” In: *Neural Computation* 22.5 (May 2010), pp. 1272–1311. DOI: 10.1162/neco.2009.01-09-947. URL: https://doi.org/10.1162/neco.2009.01-09-947 (cit. on p. 177).

[1414] M. Inubushi and K. Yoshimura. “Reservoir Computing Beyond Memory-Nonlinearity Trade-off.” In: *Scientific Reports* 7.1 (Aug. 2017). DOI: 10.1038/s41598-017-10257-6. URL: https://doi.org/10.1038/s41598-017-10257-6 (cit. on pp. 177, 178).

[1415] S. Marzen. “Difference between memory and prediction in linear recurrent networks.” In: *Physical Review E* 96.3 (Sept. 2017). DOI: 10.1103/physreve.96.032308. URL: https://doi.org/10.1103/physreve.96.032308 (cit. on p. 177).

[1416] C. Gallicchio and A. Micheli. “Deep Reservoir Computing: A Critical Analysis.” In: *The European Symposium on Artificial Neural Networks*. Apr. 2016. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2016-175.pdf (cit. on p. 177).

[1417] C. Gallicchio, A. Micheli, and L. Pedrelli. “Deep reservoir computing: A critical experimental analysis.” In: *Neurocomputing* 268 (Dec. 2017), pp. 87–99. DOI: 10.1016/j.neucom.2016.12.089. URL: https://doi.org/10.1016/j.neucom.2016.12.089 (cit. on p. 177).

[1418] Q. Shen, H. Zhang, and Y. Mao. “Improving Deep Echo State Network with Neuronal Similarity-Based Iterative Pruning Merging Algorithm.” In: *Applied Sciences* 13.5 (Feb. 2023), p. 2918. DOI: 10.3390/app13052918. URL: https://doi.org/10.3390/app13052918 (cit. on p. 177).

[1419] C. Gallicchio, A. Micheli, and L. Pedrelli. “Design of deep echo state networks.” In: *Neural Networks* 108 (Dec. 2018), pp. 33–47. DOI: 10.1016/j.neunet.2018.08.002. URL: https://doi.org/10.1016/j.neunet.2018.08.002 (cit. on p. 177).

[1420] C. Gallicchio, A. Micheli, and L. Pedrelli. “Hierarchical Temporal Representation in Linear Reservoir Computing.” In: *Neural Advances in Processing Nonlinear Dynamic Signals*. Springer International Publishing, July 2018, pp. 119–129. DOI: 10.1007/978-3-319-95098-3_11. URL: https://doi.org/10.1007/978-3-319-95098-3_11 (cit. on p. 177).

[1421] C. Gallicchio and A. Micheli. “Echo State Property of Deep Reservoir Computing Networks.” In: *Cognitive Computation* 9.3 (May 2017), pp. 337–350. DOI: 10.1007/s12559-017-9461-9. URL: https://doi.org/10.1007/s12559-017-9461-9 (cit. on p. 177).

[1422] C. Gallicchio and A. Micheli. “Richness of Deep Echo State Network Dynamics.” In: *Advances in Computational Intelligence*. Springer International Publishing, 2019, pp. 480–491. DOI: 10.1007/978-3-030-20521-8_40. URL: https://doi.org/10.1007/978-3-030-20521-8_40 (cit. on p. 177).

[1423] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen. “Memory versus non-linearity in reservoirs.” In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2010. DOI: 10.1109/ijcnn.2010.5596492. URL: https://doi.org/10.1109/ijcnn.2010.5596492 (cit. on p. 177).

[1424] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. "An experimental unification of reservoir computing methods." In: *Neural Networks* 20.3 (Apr. 2007), pp. 391–403. DOI: 10.1016/j.neunet.2007.04.003. URL: https://doi.org/10.1016/j.neunet.2007.04.003 (cit. on p. 177).

[1425] D. Verstraeten and B. Schrauwen. "On the Quantification of Dynamics in Reservoir Computing." In: *Artificial Neural Networks – ICANN 2009*. Springer Berlin Heidelberg, 2009, pp. 985–994. DOI: 10.1007/978-3-642-04274-4_101. URL: https://doi.org/10.1007/978-3-642-04274-4_101 (cit. on p. 177).

[1426] A. Hart, J. Hook, and J. Dawes. "Embedding and approximation theorems for echo state networks." In: *Neural Networks* 128 (Aug. 2020), pp. 234–247. DOI: 10.1016/j.neunet.2020.05.013. URL: https://doi.org/10.1016/j.neunet.2020.05.013 (cit. on p. 177).

[1427] C. Gallicchio, A. Micheli, and L. Silvestri. "Local Lyapunov exponents of deep echo state networks." In: *Neurocomputing* 298 (July 2018), pp. 34–45. DOI: 10.1016/j.neucom.2017.11.073. URL: https://doi.org/10.1016/j.neucom.2017.11.073 (cit. on p. 177).

[1428] L. A. Thiede and U. Parlitz. "Gradient based hyperparameter optimization in Echo State Networks." In: *Neural Networks* 115 (July 2019), pp. 23–29. DOI: 10.1016/j.neunet.2019.02.001. URL: https://doi.org/10.1016/j.neunet.2019.02.001 (cit. on p. 177).

[1429] N. Trouvain, N. Rougier, and X. Hinaut. "Create Efficient and Complex Reservoir Computing Architectures with ReservoirPy." In: *From Animals to Animats 16*. Springer International Publishing, 2022, pp. 91–102. DOI: 10.1007/978-3-031-16770-6_8. URL: https://doi.org/10.1007/978-3-031-16770-6_8 (cit. on p. 177).

[1430] P. Vlachas, J. Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. "Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics." In: *Neural Networks* 126 (June 2020), pp. 191–217. DOI: 10.1016/j.neunet.2020.02.016. URL: https://doi.org/10.1016/j.neunet.2020.02.016 (cit. on p. 177).

[1431] C. Yang, J. Qiao, H. Han, and L. Wang. "Design of polynomial echo state networks for time series prediction." In: *Neurocomputing* 290 (May 2018), pp. 148–160. DOI: 10.1016/j.neucom.2018.02.036. URL: https://doi.org/10.1016/j.neucom.2018.02.036 (cit. on pp. 177, 178).

[1432] S. Zhong, X. Xie, L. Lin, and F. Wang. "Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction." In: *Neurocomputing* 238 (May 2017), pp. 191–204. DOI: 10.1016/j.neucom.2017.01.053. URL: https://doi.org/10.1016/j.neucom.2017.01.053 (cit. on pp. 177, 178).

[1433] G. Shi, D. Liu, and Q. Wei. "Energy consumption prediction of office buildings based on echo state networks." In: *Neurocomputing* 216 (Dec. 2016), pp. 478–488. DOI: 10.1016/j.neucom.2016.08.004. URL: https://doi.org/10.1016/j.neucom.2016.08.004 (cit. on p. 177).

[1434] X. Sun, T. Li, Q. Li, Y. Huang, and Y. Li. "Deep belief echo-state network and its application to time series prediction." In: *Knowledge-Based Systems* 130 (Aug. 2017), pp. 17–29. DOI: 10.1016/j.knosys.2017.05.022. URL: https://doi.org/10.1016/j.knosys.2017.05.022 (cit. on pp. 177, 178).

[1435] Y. Chen, Z. He, Z. Shang, C. Li, L. Li, and M. Xu. "A novel combined model based on echo state network for multi-step ahead wind speed forecasting: A case study of NREL." In: *Energy Conversion and Management* 179 (Jan. 2019), pp. 13–29. DOI: 10.1016/j.enconman.2018.10.068. URL: https://doi.org/10.1016/j.enconman.2018.10.068 (cit. on p. 177).

[1436] X. Yao, Z. Wang, and H. Zhang. "A novel photovoltaic power forecasting model based on echo state network." In: *Neurocomputing* 325 (Jan. 2019), pp. 182–189. DOI: 10.1016/j.neucom.2018.10.022. URL: https://doi.org/10.1016/j.neucom.2018.10.022 (cit. on pp. 177, 178).

[1437] L. Wang, H. Hu, X.-Y. Ai, and H. Liu. "Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm." In: *Energy* 153 (June 2018), pp. 801–815. DOI: 10.1016/j.energy.2018.04.078. URL: https://doi.org/10.1016/j.energy.2018.04.078 (cit. on pp. 177, 178).

[1438] M. A. Chitsazan, M. S. Fadali, and A. M. Trzynadlowski. "Wind speed and wind direction forecasting using echo state network with nonlinear functions." In: *Renewable Energy* 131 (Feb. 2019), pp. 879–889. DOI: 10.1016/j.renene.2018.07.060. URL: https://doi.org/10.1016/j.renene.2018.07.060 (cit. on p. 177).

[1439] F. M. Bianchi, S. Scardapane, A. Uncini, A. Rizzi, and A. Sadeghian. "Prediction of telephone calls load using Echo State Network with exogenous variables." In: *Neural Networks* 71 (Nov. 2015), pp. 204–213. DOI: 10.1016/j.neunet.2015.08.010. URL: https://doi.org/10.1016/j.neunet.2015.08.010 (cit. on p. 177).

[1440] M. Han and M. Xu. "Predicting Multivariate Time Series Using Subspace Echo State Network." In: *Neural Processing Letters* 41.2 (Sept. 2013), pp. 201–209. DOI: 10.1007/s11063-013-9324-7. URL: https://doi.org/10.1007/s11063-013-9324-7 (cit. on pp. 177, 178).

[1441] D. Li, M. Han, and J. Wang. "Chaotic Time Series Prediction Based on a Novel Robust Echo State Network." In: *IEEE Transactions on Neural Networks and Learning Systems* 23.5 (May 2012), pp. 787–799. DOI: 10.1109/tnnls.2012.2188414. URL: https://doi.org/10.1109/tnnls.2012.2188414 (cit. on pp. 177, 178).

[1442] E. A. Antonelo, E. Camponogara, and B. Foss. "Echo State Networks for data-driven downhole pressure estimation in gas-lift oil wells." In: *Neural Networks* 85 (Jan. 2017), pp. 106–117. DOI: 10.1016/j.neunet.2016.09.009. URL: https://doi.org/10.1016/j.neunet.2016.09.009 (cit. on p. 177).

[1443] Z. Shi and M. Han. "Support Vector Echo-State Machine for Chaotic Time-Series Prediction." In: *IEEE Transactions on Neural Networks* 18.2 (Mar. 2007), pp. 359–372. DOI: 10.1109/tnn.2006.885113. URL: https://doi.org/10.1109/tnn.2006.885113 (cit. on pp. 177, 178).

[1444] J. Xi, Z. Shi, and M. Han. "Analyzing the state space property of echo state networks for chaotic system prediction." In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* IEEE, 2005. DOI: 10.1109/ijcnn.2005.1556081. URL: https://doi.org/10.1109/ijcnn.2005.1556081 (cit. on p. 177).

[1445] K. Takano, C. Sugano, M. Inubushi, K. Yoshimura, S. Sunada, K. Kanno, and A. Uchida. "Compact reservoir computing with a photonic integrated circuit." In: *Optics Express* 26.22 (Oct. 2018), p. 29424. DOI: 10.1364/oe.26.029424. URL: https://doi.org/10.1364/oe.26.029424 (cit. on pp. 177, 178).

[1446] K. Bai and Y. Yi. "DFR: An Energy-efficient Analog Delay Feedback Reservoir Computing System for Brain-inspired Computing." In: *ACM Journal on Emerging Technologies in Computing Systems* 14.4 (Oct. 2018), pp. 1–22. DOI: 10.1145/3264659. URL: https://doi.org/10.1145/3264659 (cit. on pp. 177, 178).

[1447] M. L. Alomar, V. Canals, N. Perez-Mora, V. Martínez-Moll, and J. L. Rosselló. "FPGA-Based Stochastic Echo State Networks for Time-Series Forecasting." In: *Computational Intelligence and Neuroscience* 2016 (2016), pp. 1–14. DOI: 10.1155/2016/3917892. URL: https://doi.org/10.1155/2016/3917892 (cit. on pp. 177, 178).

[1448] X. Shi, J. Gao, L. L. Minku, J. J. Yu, and X. Yao. "Second-order Time Delay Reservoir Computing for Nonlinear Time Series Problems." In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI).* IEEE, Dec. 2021. DOI: 10.1109/ssci50451.2021.9659913. URL: https://doi.org/10.1109/ssci50451.2021.9659913 (cit. on p. 177).

[1449] R. Budhiraja, M. Kumar, M. K. Das, A. S. Bafila, and S. Singh. "A reservoir computing approach for forecasting and regenerating both dynamical and time-delay controlled financial system behavior." In: *PLOS ONE* 16.2 (Feb. 2021). Ed. by P. K. Arora, e0246737. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0246737. URL: http://dx.doi.org/10.1371/journal.pone.0246737 (cit. on p. 177).

[1450] A. Elsonbaty, A. A. Elsadany, and W. Adel. "On Reservoir Computing Approach for Digital Image Encryption and Forecasting of Hyperchaotic Finance Model." In: *Fractal and Fractional* 7.4 (Mar. 2023), p. 282. ISSN: 2504-3110. DOI: 10.3390/fractalfract7040282. URL: http://dx.doi.org/10.3390/fractalfract7040282 (cit. on p. 177).

[1451] W.-J. Wang, Y. Tang, J. Xiong, and Y.-C. Zhang. "Stock market index prediction based on reservoir computing models." In: *Expert Systems with Applications* 178 (Sept. 2021), p. 115022. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.115022. URL: http://dx.doi.org/10.1016/j.eswa.2021.115022 (cit. on p. 177).

[1452] B. Liu, Y. Xie, X. Jiang, Y. Ye, T. Song, J. Chai, Q. Tang, and M. Feng. "Forecasting stock market with nanophotonic reservoir computing system based on silicon optomechanical oscillators." In: *Optics Express* 30.13 (June 2022), p. 23359. ISSN: 1094-4087. DOI: 10.1364/oe.454973. URL: http://dx.doi.org/10.1364/oe.454973 (cit. on pp. 177, 178).

[1453] L. Wang, Z. Wang, and S. Liu. "An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm." In: *Expert Systems with Applications* 43 (Jan. 2016), pp. 237–249. DOI: 10.1016/j.eswa.2015.08.055. URL: https://doi.org/10.1016/j.eswa.2015.08.055 (cit. on p. 177).

[1454] M.-H. Yusoff, J. Chrol-Cannon, and Y. Jin. "Modeling neural plasticity in echo state networks for classification and regression." In: *Information Sciences* 364-365 (Oct. 2016), pp. 184–196. DOI: 10.1016/j.ins.2015.11.017. URL: https://doi.org/10.1016/j.ins.2015.11.017 (cit. on p. 177).

[1455] E. Trentin, S. Scherer, and F. Schwenker. "Emotion recognition from speech signals via a probabilistic echo-state network." In: *Pattern Recognition Letters* 66 (Nov. 2015), pp. 4–12. DOI: 10.1016/j.patrec.2014.10.015. URL: https://doi.org/10.1016/j.patrec.2014.10.015 (cit. on pp. 177, 178).

[1456] G. Shi, B. Zhao, C. Li, Q. Wei, and D. Liu. "An echo state network based approach to room classification of office buildings." In: *Neurocomputing* 333 (Mar. 2019), pp. 319–328. DOI: 10.1016/j.neucom.2018.12.033. URL: https://doi.org/10.1016/j.neucom.2018.12.033 (cit. on p. 177).

[1457] Q. Ma, L. Shen, W. Chen, J. Wang, J. Wei, and Z. Yu. "Functional echo state network for time series classification." In: *Information Sciences* 373 (Dec. 2016), pp. 1–20. DOI: 10.1016/j.ins.2016.08.081. URL: https://doi.org/10.1016/j.ins.2016.08.081 (cit. on pp. 177, 178).

[1458] S. E. Lacy, S. L. Smith, and M. A. Lones. "Using echo state networks for classification: A case study in Parkinson's disease diagnosis." In: *Artificial Intelligence in Medicine* 86 (Mar. 2018), pp. 53–59. DOI: 10.1016/j.artmed.2018.02.002. URL: https://doi.org/10.1016/j.artmed.2018.02.002 (cit. on p. 177).

[1459] M. Skowronski and J. Harris. "Minimum mean squared error time series classification using an echo state network prediction model." In: *2006 IEEE International Symposium on Circuits and Systems*. IEEE, 2006. DOI: 10.1109/iscas.2006.1693294. URL: https://doi.org/10.1109/iscas.2006.1693294 (cit. on p. 177).

[1460] M. D. Skowronski and J. G. Harris. "Automatic speech recognition using a predictive echo state network classifier." In: *Neural Networks* 20.3 (Apr. 2007), pp. 414–423. DOI: 10.1016/j.neunet.2007.04.006. URL: https://doi.org/10.1016/j.neunet.2007.04.006 (cit. on p. 177).

[1461] M. H. Tong, A. D. Bickett, E. M. Christiansen, and G. W. Cottrell. "Learning grammatical structure with Echo State Networks." In: *Neural Networks* 20.3 (Apr. 2007), pp. 424–432. DOI: 10.1016/j.neunet.2007.04.013. URL: https://doi.org/10.1016/j.neunet.2007.04.013 (cit. on p. 177).

[1462] S. Wang et al. "Echo state graph neural networks with analogue random resistive memory arrays." In: *Nature Machine Intelligence* 5.2 (Feb. 2023), pp. 104–113. DOI: 10.1038/s42256-023-00609-5. URL: https://doi.org/10.1038/s42256-023-00609-5 (cit. on pp. 177, 178).

[1463] K. Yeo. "Data-driven reconstruction of nonlinear dynamics from sparse observation." In: *Journal of Computational Physics* 395 (Oct. 2019), pp. 671–689. DOI: 10.1016/j.jcp.2019.06.039. URL: https://doi.org/10.1016/j.jcp.2019.06.039 (cit. on p. 177).

[1464] M. Magerl, V. Ceperic, and A. Baric. "Echo State Networks for Black-Box Modeling of Integrated Circuits." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.8 (Aug. 2016), pp. 1309–1317. DOI: 10.1109/tcad.2015.2501312. URL: https://doi.org/10.1109/tcad.2015.2501312 (cit. on p. 177).

[1465] Y. Yang, R. G. Harley, D. Divan, and T. G. Habetler. "Overhead conductor thermal dynamics identification by using Echo State Networks." In: *2009 International Joint Conference on Neural Networks*. IEEE, June 2009. DOI: 10.1109/ijcnn.2009.5179006. URL: https://doi.org/10.1109/ijcnn.2009.5179006 (cit. on p. 177).

[1466] A. Jalalvand, K. Demuynck, W. D. Neve, and J.-P. Martens. "On the application of reservoir computing networks for noisy image recognition." In: *Neurocomputing* 277 (Feb. 2018), pp. 237–248. DOI: 10.1016/j.neucom.2016.11.100. URL: https://doi.org/10.1016/j.neucom.2016.11.100 (cit. on p. 178).

[1467] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott. "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.4 (Apr. 2017), p. 041102. DOI: 10.1063/1.4979665. URL: https://doi.org/10.1063/1.4979665 (cit. on p. 178).

[1468] P. Antonik, M. Gulina, J. Pauwels, and S. Massar. "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography." In: *Physical Review E* 98.1 (July 2018). DOI: 10.1103/physreve.98.012215. URL: https://doi.org/10.1103/physreve.98.012215 (cit. on p. 178).

[1469] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-p. Martens. "Phoneme Recognition with Large Hierarchical Reservoirs." In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/2ca65f58e35d9ad45bf7f3ae5cfd08f1-Paper.pdf (cit. on p. 178).

[1470] M. R. Salehi, E. Abiri, and L. Dehyadegari. "An analytical approach to photonic reservoir computing – a network of SOA's – for noisy speech recognition." In: *Optics Communications* 306 (Oct. 2013), pp. 135–139. DOI: 10.1016/j.optcom.2013.05.036. URL: https://doi.org/10.1016/j.optcom.2013.05.036 (cit. on p. 178).

[1471] Q. An, K. Bai, L. Liu, F. Shen, and Y. Yi. "A unified information perceptron using deep reservoir computing." In: *Computers & Electrical Engineering* 85 (July 2020), p. 106705. DOI: 10.1016/j.compeleceng.2020.106705. URL: https://doi.org/10.1016/j.compeleceng.2020.106705 (cit. on p. 178).

[1472] S. Tan, Z. Wu, D. Yue, W. Wu, and G. Xia. "Spoken digit recognition utilizing a reservoir computing system based on mutually coupled VCSELs under optical injection." In: *Optics Continuum* 1.7 (July 2022), p. 1593. DOI: 10.1364/optcon.453196. URL: https://doi.org/10.1364/optcon.453196 (cit. on p. 178).

[1473] P. Buteneers, B. Schrauwen, D. Verstraeten, and D. Stroobandt. "Real-Time Epileptic Seizure Detection on Intra-cranial Rat Data Using Reservoir Computing." In: *Advances in Neuro-Information Processing*. Springer Berlin Heidelberg, 2009, pp. 56–63. DOI: 10.1007/978-3-642-02490-0_7. URL: https://doi.org/10.1007/978-3-642-02490-0_7 (cit. on p. 178).

[1474] P. Buteneers, D. Verstraeten, P. van Mierlo, T. Wyckhuys, D. Stroobandt, R. Raedt, H. Hallez, and B. Schrauwen. "Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing." In: *Artificial Intelligence in Medicine* 53.3 (Nov. 2011), pp. 215–223. DOI: 10.1016/j.artmed.2011.08.006. URL: https://doi.org/10.1016/j.artmed.2011.08.006 (cit. on p. 178).

[1475] A. Jalalvand, F. Triefenbach, K. Demuynck, and J.-P. Martens. "Robust continuous digit recognition using Reservoir Computing." In: *Computer Speech & Language* 30.1 (Mar. 2015), pp. 135–158. DOI: 10.1016/j.csl.2014.09.006. URL: https://doi.org/10.1016/j.csl.2014.09.006 (cit. on p. 178).

[1476] E. Antonelo, B. Schrauwen, and D. Stroobandt. "Event detection and localization for small mobile robots using reservoir computing." In: *Neural Networks* 21.6 (Aug. 2008), pp. 862–871. DOI: 10.1016/j.neunet.2008.06.010. URL: https://doi.org/10.1016/j.neunet.2008.06.010 (cit. on p. 178).

[1477] K. Lüdge and A. Röhm. "Computing with a camera." In: *Nature Machine Intelligence* 1.12 (Dec. 2019), pp. 551–552. DOI: 10.1038/s42256-019-0124-2. URL: https://doi.org/10.1038/s42256-019-0124-2 (cit. on p. 178).

[1478] S. Apostel, N. D. Haynes, E. Schöll, O. D'Huys, and D. J. Gauthier. "Reservoir Computing Using Autonomous Boolean Networks Realized on Field-Programmable Gate Arrays." In: *Natural Computing Series*. Springer Singapore, 2021, pp. 239–271. DOI: 10.1007/978-981-13-1687-6_11. URL: https://doi.org/10.1007/978-981-13-1687-6_11 (cit. on p. 178).

[1479] E. Iranmehr, S. B. Shouraki, and M. M. Faraji. "ILS-based Reservoir Computing for Handwritten Digits Recognition." In: *2020 8th Iranian Joint Congress on Fuzzy and intelligent Systems (CFIS)*. IEEE, Sept. 2020. DOI: 10.1109/cfis49607.2020.9238722. URL: https://doi.org/10.1109/cfis49607.2020.9238722 (cit. on p. 178).

[1480] P. Antonik, N. Marsal, D. Brunner, and D. Rontani. "Human action recognition with a large-scale brain-inspired photonic computer." In: *Nature Machine Intelligence* 1.11 (Nov. 2019), pp. 530–537. DOI: 10.1038/s42256-019-0110-8. URL: https://doi.org/10.1038/s42256-019-0110-8 (cit. on p. 178).

[1481] K. Hamedani, L. Liu, R. Atat, J. Wu, and Y. Yi. "Reservoir Computing Meets Smart Grids: Attack Detection Using Delayed Feedback Networks." In: *IEEE Transactions on Industrial Informatics* 14.2 (Feb. 2018), pp. 734–743. DOI: 10.1109/tii.2017.2769106. URL: https://doi.org/10.1109/tii.2017.2769106 (cit. on p. 178).

[1482] S.-x. Lun, X.-s. Yao, and H.-f. Hu. "A new echo state network with variable memory length." In: *Information Sciences* 370-371 (Nov. 2016), pp. 103–119. DOI: 10.1016/j.ins.2016.07.065. URL: https://doi.org/10.1016/j.ins.2016.07.065 (cit. on p. 178).

[1483] H. Wang, C. Ni, and X. Yan. "Optimizing the echo state network based on mutual information for modeling fed-batch bioprocesses." In: *Neurocomputing* 225 (Feb. 2017), pp. 111–118. DOI: 10.1016/j.neucom.2016.11.007. URL: https://doi.org/10.1016/j.neucom.2016.11.007 (cit. on p. 178).

[1484] H. Wang and X. Yan. "Optimizing the echo state network with a binary particle swarm optimization algorithm." In: *Knowledge-Based Systems* 86 (Sept. 2015), pp. 182–193. DOI: 10.1016/j.knosys.2015.06.003. URL: https://doi.org/10.1016/j.knosys.2015.06.003 (cit. on p. 178).

[1485] G. Wainrib and M. N. Galtier. "A local Echo State Property through the largest Lyapunov exponent." In: *Neural Networks* 76 (Apr. 2016), pp. 39–45. DOI: 10.1016/j.neunet.2015.12.013. URL: https://doi.org/10.1016/j.neunet.2015.12.013 (cit. on p. 178).

[1486] H. Cui, C. Feng, Y. Chai, R. P. Liu, and Y. Liu. "Effect of hybrid circle reservoir injected with wavelet-neurons on performance of echo state network." In: *Neural Networks* 57 (Sept. 2014), pp. 141–151. DOI: 10.1016/j.neunet.2014.05.013. URL: https://doi.org/10.1016/j.neunet.2014.05.013 (cit. on p. 178).

[1487] Y. Guo, F. Wang, B. Chen, and J. Xin. "Robust echo state networks based on correntropy induced loss function." In: *Neurocomputing* 267 (Dec. 2017), pp. 295–303. DOI: 10.1016/j.neucom.2017.05.087. URL: https://doi.org/10.1016/j.neucom.2017.05.087 (cit. on p. 178).

[1488] X. Yao, Z. Wang, and H. Zhang. "Identification method for a class of periodic discrete-time dynamic nonlinear systems based on Sinusoidal ESN." In: *Neurocomputing* 275 (Jan. 2018), pp. 1511–1521. DOI: 10.1016/j.neucom.2017.09.092. URL: https://doi.org/10.1016/j.neucom.2017.09.092 (cit. on p. 178).

[1489] S.-x. Lun, S. Wang, T.-t. Guo, and C.-j. Du. "An I–V model based on time warp invariant echo state network for photovoltaic array with shaded solar cells." In: *Solar Energy* 105 (July 2014), pp. 529–541. DOI: 10.1016/j.solener.2014.04.023. URL: https://doi.org/10.1016/j.solener.2014.04.023 (cit. on p. 178).

[1490] C. Gallicchio and A. Micheli. "Fast and Deep Graph Neural Networks." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 3898–3905. DOI: 10.1609/aaai.v34i04.5803. URL: https://doi.org/10.1609/aaai.v34i04.5803 (cit. on p. 178).

[1491] P. Mujal, R. Martínez-Peña, G. L. Giorgi, M. C. Soriano, and R. Zambrini. "Time-series quantum reservoir computing with weak and projective measurements." In: *npj Quantum Information* 9.1 (Feb. 2023). DOI: 10.1038/s41534-023-00682-z. URL: https://doi.org/10.1038/s41534-023-00682-z (cit. on p. 178).

[1492] R. Martínez-Peña and J.-P. Ortega. "Quantum reservoir computing in finite dimensions." In: *Physical Review E* 107.3 (Mar. 2023). DOI: 10.1103/physreve.107.035306. URL: https://doi.org/10.1103/physreve.107.035306 (cit. on p. 178).

[1493] G. Llodrà, C. Charalambous, G. L. Giorgi, and R. Zambrini. "Benchmarking the Role of Particle Statistics in Quantum Reservoir Computing." In: *Advanced Quantum Technologies* 6.1 (Nov. 2022), p. 2200100. DOI: 10.1002/qute.202200100. URL: https://doi.org/10.1002/qute.202200100 (cit. on p. 178).

[1494] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar. "Optoelectronic Reservoir Computing." In: *Scientific Reports* 2.1 (Feb. 2012). DOI: 10.1038/srep00287. URL: https://doi.org/10.1038/srep00287 (cit. on p. 178).

[1495] R. Martinenghi, S. Rybalko, M. Jacquot, Y. K. Chembo, and L. Larger. "Photonic Nonlinear Transient Computing with Multiple-Delay Wavelength Dynamics." In: *Physical Review Letters* 108.24 (June 2012). DOI: 10.1103/physrevlett.108.244101. URL: https://doi.org/10.1103/physrevlett.108.244101 (cit. on p. 178).

[1496] K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman, and J. V. Campenhout. "Toward optical signal processing using Photonic Reservoir Computing." In: *Optics Express* 16.15 (July 2008), p. 11182. DOI: 10.1364/oe.16.011182. URL: https://doi.org/10.1364/oe.16.011182 (cit. on p. 178).

[1497] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer. "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing." In: *Optics Express* 20.3 (Jan. 2012), p. 3241. DOI: 10.1364/oe.20.003241. URL: https://doi.org/10.1364/oe.20.003241 (cit. on p. 178).

[1498] Y. Chen, L. Yi, J. Ke, Z. Yang, Y. Yang, L. Huang, Q. Zhuge, and W. Hu. "Reservoir computing system with double optoelectronic feedback loops." In: *Optics Express* 27.20 (Sept. 2019), p. 27431. DOI: 10.1364/oe.27.027431. URL: https://doi.org/10.1364/oe.27.027431 (cit. on p. 178).

[1499] K. Vandoorne, M. Fiers, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman. "Photonic reservoir computing: A new approach to optical information processing." In: *2010 12th International Conference on Transparent Optical Networks*. IEEE, June 2010. DOI: 10.1109/icton.2010.5548990. URL: https://doi.org/10.1109/icton.2010.5548990 (cit. on p. 178).

[1500] K. Vandoorne, M. Fiers, T. V. Vaerenbergh, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman. "Advances in photonic reservoir computing on an integrated platform." In: *2011 13th International Conference on Transparent Optical Networks*. IEEE, June 2011. DOI: 10.1109/icton.2011.5970791. URL: https://doi.org/10.1109/icton.2011.5970791 (cit. on p. 178).

[1501] Vandoorne, Kristof. "Photonic reservoir computing with a network of coupled semiconductor optical amplifiers." eng. PhD thesis. Ghent University, 2011, XXXV, 185. ISBN: 9789085784517 (cit. on p. 178).

[1502] X. Li, N. Jiang, Q. Zhang, C. Tang, Y. Zhang, G. Hu, Y. Cao, and K. Qiu. "Performance-enhanced time-delayed photonic reservoir computing system using a reflective semiconductor optical amplifier." In: *Optics Express* 31.18 (Aug. 2023), p. 28764. ISSN: 1094-4087. DOI: 10.1364/oe.495697. URL: http://dx.doi.org/10.1364/oe.495697 (cit. on p. 178).

[1503] Q. Vinckier, F. Duport, A. Smerieri, K. Vandoorne, P. Bienstman, M. Haelterman, and S. Massar. "High-performance photonic reservoir computer based on a coherently driven passive cavity." In: *Optica* 2.5 (Apr. 2015), p. 438. DOI: 10.1364/optica.2.000438. URL: https://doi.org/10.1364/optica.2.000438 (cit. on p. 178).

[1504] A. Dejonckheere, F. Duport, A. Smerieri, L. Fang, J.-L. Oudar, M. Haelterman, and S. Massar. "All-optical reservoir computer based on saturation of absorption." In: *Optics Express* 22.9 (Apr. 2014), p. 10868. DOI: 10.1364/oe.22.010868. URL: https://doi.org/10.1364/oe.22.010868 (cit. on p. 178).

[1505] M. Abdalla, C. Zrounba, R. Cardoso, P. Jimenez, G. Ren, A. Boes, A. Mitchell, A. Bosio, I. O'Connor, and F. Pavanello. "Minimum complexity integrated photonic architecture for delay-based reservoir computing." In: *Optics Express* 31.7 (Mar. 2023), p. 11610. DOI: 10.1364/oe.484052. URL: https://doi.org/10.1364/oe.484052 (cit. on p. 178).

[1506] I. Bauwens, K. Harkhoe, P. Bienstman, G. Verschaffelt, and G. V. der Sande. "Transfer learning for photonic delay-based reservoir computing to compensate parameter drift." In: *Nanophotonics* 12.5 (Oct. 2022), pp. 949–961. DOI: 10.1515/nanoph-2022-0399. URL: https://doi.org/10.1515/nanoph-2022-0399 (cit. on p. 178).

[1507] G. V. der Sande, D. Brunner, and M. C. Soriano. "Advances in photonic reservoir computing." In: *Nanophotonics* 6.3 (May 2017), pp. 561–576. DOI: 10.1515/nanoph-2016-0132. URL: https://doi.org/10.1515/nanoph-2016-0132 (cit. on p. 178).

[1508] S. Masaad, E. Gooskens, S. Sackesyn, J. Dambre, and P. Bienstman. "Photonic reservoir computing for nonlinear equalization of 64-QAM signals with a Kramers–Kronig receiver." In: *Nanophotonics* 12.5 (Oct. 2022), pp. 925–935. DOI: 10.1515/nanoph-2022-0426. URL: https://doi.org/10.1515/nanoph-2022-0426 (cit. on p. 178).

[1509] H. Hasegawa, K. Kanno, and A. Uchida. "Parallel and deep reservoir computing using semiconductor lasers with optical feedback." In: *Nanophotonics* 12.5 (Oct. 2022), pp. 869–881. DOI: 10.1515/nanoph-2022-0440. URL: https://doi.org/10.1515/nanoph-2022-0440 (cit. on p. 178).

[1510] J. Nakayama, K. Kanno, and A. Uchida. "Laser dynamical reservoir computing with consistency: an approach of a chaos mask signal." In: *Optics Express* 24.8 (Apr. 2016), p. 8679. DOI: 10.1364/oe.24.008679. URL: https://doi.org/10.1364/oe.24.008679 (cit. on p. 178).

[1511] M. C. Soriano, S. Ortín, D. Brunner, L. Larger, C. R. Mirasso, I. Fischer, and L. Pesquera. "Optoelectronic reservoir computing: tackling noise-induced performance degradation." In: *Optics Express* 21.1 (Jan. 2013), p. 12. DOI: 10.1364/oe.21.000012. URL: https://doi.org/10.1364/oe.21.000012 (cit. on p. 178).

[1512] R. M. Nguimdo, E. Lacot, O. Jacquin, O. Hugon, G. V. der Sande, and H. G. de Chatellus. "Prediction performance of reservoir computing systems based on a diode-pumped erbium-doped microchip laser subject to optical feedback." In: *Optics Letters* 42.3 (Jan. 2017), p. 375. DOI: 10.1364/ol.42.000375. URL: https://doi.org/10.1364/ol.42.000375 (cit. on p. 178).

[1513] J. Bueno, D. Brunner, M. C. Soriano, and I. Fischer. "Conditions for reservoir computing performance using semiconductor lasers with delayed optical feedback." In: *Optics Express* 25.3 (Jan. 2017), p. 2401. DOI: 10.1364/oe.25.002401. URL: https://doi.org/10.1364/oe.25.002401 (cit. on p. 178).

[1514] Y. Hou, G. Xia, W. Yang, D. Wang, E. Jayaprasath, Z. Jiang, C. Hu, and Z. Wu. "Prediction performance of reservoir computing system based on a semiconductor laser subject to double optical feedback and optical injection." In: *Optics Express* 26.8 (Apr. 2018), p. 10211. DOI: 10.1364/oe.26.010211. URL: https://doi.org/10.1364/oe.26.010211 (cit. on p. 178).

[1515] J. Vatin, D. Rontani, and M. Sciamanna. "Enhanced performance of a reservoir computer using polarization dynamics in VCSELs." In: *Optics Letters* 43.18 (Sept. 2018), p. 4497. DOI: 10.1364/ol.43.004497. URL: https://doi.org/10.1364/ol.43.004497 (cit. on p. 178).

[1516] Y.-S. Hou, G.-Q. Xia, E. Jayaprasath, D.-Z. Yue, W.-Y. Yang, and Z.-M. Wu. "Prediction and classification performance of reservoir computing system using mutually delay-coupled semiconductor lasers." In: *Optics Communications* 433 (Feb. 2019), pp. 215–220. DOI: 10.1016/j.optcom.2018.10.014. URL: https://doi.org/10.1016/j.optcom.2018.10.014 (cit. on p. 178).

[1517] S. S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi. "Brain-Inspired Wireless Communications: Where Reservoir Computing Meets MIMO-OFDM." In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4694–4708. DOI: 10.1109/tnnls.2017.2766162. URL: https://doi.org/10.1109/tnnls.2017.2766162 (cit. on p. 178).

[1518] H. Numata, J. B. Heroux, T. Yamane, and D. Nakano. "FPGA-driven High Density Photonic Reservoir Computing." In: *2022 International Conference on Electronics Packaging (ICEP)*. IEEE, May 2022. DOI: 10.23919/icep55381.2022.9795514. URL: https://doi.org/10.23919/icep55381.2022.9795514 (cit. on p. 178).

[1519] S. Ortín, M. C. Soriano, L. Pesquera, D. Brunner, D. San-Martín, I. Fischer, C. R. Mirasso, and J. M. Gutiérrez. "A Unified Framework for Reservoir Computing and Extreme Learning Machines based on a Single Time-delayed Neuron." In: *Scientific Reports* 5.1 (Oct. 2015). DOI: 10.1038/srep14945. URL: https://doi.org/10.1038/srep14945 (cit. on p. 178).

[1520] A. Katumba, J. Heyvaert, B. Schneider, S. Uvin, J. Dambre, and P. Bienstman. "Low-Loss Photonic Reservoir Computing with Multimode Photonic Integrated Circuits." In: *Scientific Reports* 8.1 (Feb. 2018). DOI: 10.1038/s41598-018-21011-x. URL: https://doi.org/10.1038/s41598-018-21011-x (cit. on p. 178).

[1521] Y. Zhang, P. Li, Y. Jin, and Y. Choe. "A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition." In: *IEEE Transactions on Neural Networks and Learning Systems* 26.11 (Nov. 2015), pp. 2635–2649. DOI: 10.1109/tnnls.2015.2388544. URL: https://doi.org/10.1109/tnnls.2015.2388544 (cit. on p. 178).

[1522] A. N. Elbedwehy, A. M. El-Mohandes, A. Elnakib, and M. E. Abou-Elsoud. "FPGA-based reservoir computing system for ECG denoising." In: *Microprocessors and Microsystems* 91 (June 2022), p. 104549. DOI: 10.1016/j.micpro.2022.104549. URL: https://doi.org/10.1016/j.micpro.2022.104549 (cit. on p. 178).

[1523] P. Kumar, M. Jin, T. Bu, S. Kumar, and Y.-P. Huang. "Efficient reservoir computing using field programmable gate array and electro-optic modulation." In: *OSA Continuum* 4.3 (Mar. 2021), p. 1086. DOI: 10.1364/osac.417996. URL: https://doi.org/10.1364/osac.417996 (cit. on p. 178).

[1524] Y. Liao, H. Li, Y. Shen, and W. Li. "An FPGA Based Real Time Reservoir Computing System for Neuromorphic Processors." In: *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, July 2018. DOI: 10.1109/acirs.2018.8467252. URL: https://doi.org/10.1109/acirs.2018.8467252 (cit. on p. 178).

[1525] V. M. Gan, Y. Liang, L. Li, L. Liu, and Y. Yi. "A Cost-Efficient Digital ESN Architecture on FPGA for OFDM Symbol Detection." In: *ACM Journal on Emerging Technologies in Computing Systems* 17.4 (June 2021), pp. 1–15. DOI: 10.1145/3440017. URL: https://doi.org/10.1145/3440017 (cit. on p. 178).

[1526] E. S. Skibinsky-Gitlin, M. L. Alomar, C. F. Frasser, V. Canals, E. Isern, M. Roca, and J. L. Rosselló. "Cyclic Reservoir Computing with FPGA Devices for Efficient Channel Equalization." In: *Artificial Intelligence and Soft Computing*. Springer International Publishing, 2018, pp. 226–234. DOI: 10.1007/978-3-319-91253-0_22. URL: https://doi.org/10.1007/978-3-319-91253-0_22 (cit. on p. 178).

[1527] H. O. Sillin, R. Aguilera, H.-H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Gimzewski. "A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing." In: *Nanotechnology* 24.38 (Sept. 2013), p. 384004. DOI: 10.1088/0957-4484/24/38/384004. URL: https://doi.org/10.1088/0957-4484/24/38/384004 (cit. on p. 178).

[1528] L. Chua. "Memristor-The missing circuit element." In: *IEEE Transactions on Circuit Theory* 18.5 (1971), pp. 507–519. DOI: 10.1109/tct.1971.1083337. URL: https://doi.org/10.1109/tct.1971.1083337 (cit. on p. 178).

[1529] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. "The missing memristor found." In: *Nature* 453.7191 (May 2008), pp. 80–83. DOI: 10.1038/nature06932. URL: https://doi.org/10.1038/nature06932 (cit. on p. 178).

[1530] J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, and W. D. Lu. "Temporal data classification and forecasting using a memristor-based reservoir computing system." In: *Nature Electronics* 2.10 (Oct. 2019), pp. 480–487. DOI: 10.1038/s41928-019-0313-3. URL: https://doi.org/10.1038/s41928-019-0313-3 (cit. on p. 178).

[1531] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu. "Reservoir computing using dynamic memristors for temporal information processing." In: *Nature Communications* 8.1 (Dec. 2017). DOI: 10.1038/s41467-017-02337-y. URL: https://doi.org/10.1038/s41467-017-02337-y (cit. on p. 178).

[1532] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu. "Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing." In: *Nature Communications* 12.1 (Jan. 2021). DOI: 10.1038/s41467-020-20692-1. URL: https://doi.org/10.1038/s41467-020-20692-1 (cit. on p. 178).

[1533] G. Tanaka and R. Nakane. "Simulation platform for pattern recognition based on reservoir computing with memristor networks." In: *Scientific Reports* 12.1 (June 2022). DOI: 10.1038/s41598-022-13687-z. URL: https://doi.org/10.1038/s41598-022-13687-z (cit. on p. 178).

[1534] J. Ren, M. Ji'e, S. Xu, D. Yan, S. Duan, and L. Wang. "RC-MHM: reservoir computing with a 2D memristive hyperchaotic map." In: *The European Physical Journal Special Topics* (Mar. 2023). DOI: 10.1140/epjs/s11734-023-00773-0. URL: https://doi.org/10.1140/epjs/s11734-023-00773-0 (cit. on p. 178).

[1535] E. Wlaźlak, P. Zawal, and K. Szaciłowski. "Neuromorphic Applications of a Multivalued [SnI_4(C6H5)2SO_2] Memristor Incorporated in the Echo State Machine." In: *ACS Applied Electronic Materials* 2.2 (Jan. 2020), pp. 329–338. DOI: 10.1021/acsaelm.9b00750. URL: https://doi.org/10.1021/acsaelm.9b00750 (cit. on p. 178).

[1536] E. Wlaźlak, M. Marzec, P. Zawal, and K. Szaciłowski. "Memristor in a Reservoir System — Experimental Evidence for High-Level Computing and Neuromorphic Behavior of PbI_2." In: *ACS Applied Materials & Interfaces* 11.18 (Apr. 2019), pp. 17009–17018. DOI: 10.1021/acsami.9b01841. URL: https://doi.org/10.1021/acsami.9b01841 (cit. on p. 178).

[1537] D. Przyczyna, G. Hess, and K. Szaciłowski. "KNOWM memristors in a bridge synapse delay-based reservoir computing system for detection of epileptic seizures." In: *International Journal of Parallel, Emergent and Distributed Systems* 37.5 (June 2022), pp. 512–527. DOI: 10.1080/17445760.2022.2088751. URL: https://doi.org/10.1080/17445760.2022.2088751 (cit. on p. 178).

[1538] F. Hadaeghi. "Neuromorphic Electronic Systems for Reservoir Computing." In: *Natural Computing Series*. Springer Singapore, 2021, pp. 221–237. DOI: 10.1007/978-981-13-1687-6_10. URL: https://doi.org/10.1007/978-981-13-1687-6_10 (cit. on p. 178).

[1539] Y. Yang, H. Cui, S. Ke, M. Pei, K. Shi, C. Wan, and Q. Wan. "Reservoir computing based on electric-double-layer coupled InGaZnO artificial synapse." In: *Applied Physics Letters* 122.4 (Jan. 2023), p. 043508. DOI: 10.1063/5.0137647. URL: https://doi.org/10.1063/5.0137647 (cit. on p. 178).

[1540] O. Obst et al. "Nano-scale reservoir computing." In: *Nano Communication Networks* 4.4 (Dec. 2013), pp. 189–196. DOI: 10.1016/j.nancom.2013.08.005. URL: https://doi.org/10.1016/j.nancom.2013.08.005 (cit. on p. 178).

[1541] D. Nishioka, T. Tsuchiya, W. Namiki, M. Takayanagi, M. Imura, Y. Koide, T. Higuchi, and K. Terabe. "Edge-of-chaos learning achieved by ion-electron–coupled dynamics in an ion-gating reservoir." In: *Science Advances* 8.50 (Dec. 2022). DOI: 10.1126/sciadv.ade1156. URL: https://doi.org/10.1126/sciadv.ade1156 (cit. on p. 178).

[1542] S. Kan, K. Nakajima, T. Asai, and M. Akai-Kasaya. "Physical Implementation of Reservoir Computing through Electrochemical Reaction." In: *Advanced Science* 9.6 (Dec. 2021), p. 2104076. DOI: 10.1002/advs.202104076. URL: https://doi.org/10.1002/advs.202104076 (cit. on p. 178).

[1543] S.-G. Koh, H. Shima, Y. Naitoh, H. Akinaga, and K. Kinoshita. "Reservoir computing with dielectric relaxation at an electrode–ionic liquid interface." In: *Scientific Reports* 12.1 (Apr. 2022). DOI: 10.1038/s41598-022-10152-9. URL: https://doi.org/10.1038/s41598-022-10152-9 (cit. on p. 178).

[1544] T. Matsuo, D. Sato, S.-G. Koh, H. Shima, Y. Naitoh, H. Akinaga, T. Itoh, T. Nokami, M. Kobayashi, and K. Kinoshita. "Dynamic Nonlinear Behavior of Ionic Liquid-Based Reservoir Computing Devices." In: *ACS Applied Materials & Interfaces* 14.32 (July 2022), pp. 36890–36901. DOI: 10.1021/acsami.2c04167. URL: https://doi.org/10.1021/acsami.2c04167 (cit. on p. 178).

[1545] M. Nakajima, K. Minegishi, Y. Shimizu, Y. Usami, H. Tanaka, and T. Hasegawa. "In-materio reservoir working at low frequencies in a Ag_2S-island network." In: *Nanoscale* 14.20 (2022), pp. 7634–7640. DOI: 10.1039/d2nr01439d. URL: https://doi.org/10.1039/d2nr01439d (cit. on p. 178).

[1546] H. Tanaka et al. "In-materio computing in random networks of carbon nanotubes complexed with chemically dynamic molecules: a review." In: *Neuromorphic Computing and Engineering* 2.2 (May 2022), p. 022002. DOI: 10.1088/2634-4386/ac676a. URL: https://doi.org/10.1088/2634-4386/ac676a (cit. on p. 178).

[1547] D. Banerjee, T. Kotooka, S. Azhari, Y. Usami, T. Ogawa, J. K. Gimzewski, H. Tamukoh, and H. Tanaka. "Emergence of In-Materio Intelligence from an Incidental Structure of a Single-Walled Carbon Nanotube–Porphyrin Polyoxometalate Random Network." In: *Advanced Intelligent Systems* 4.4 (Jan. 2022), p. 2100145. DOI: 10.1002/aisy.202100145. URL: https://doi.org/10.1002/aisy.202100145 (cit. on p. 178).

[1548] Y. Suzuki, Q. Gao, K. C. Pradel, K. Yasuoka, and N. Yamamoto. "Natural quantum reservoir computing for temporal information processing." In: *Scientific Reports* 12.1 (Jan. 2022). DOI: 10.1038/s41598-022-05061-w. URL: https://doi.org/10.1038/s41598-022-05061-w (cit. on p. 178).

[1549] S. Ghosh, K. Nakajima, T. Krisnanda, K. Fujii, and T. C. H. Liew. "Quantum Neuromorphic Computing with Reservoir Computing Networks." In: *Advanced Quantum Technologies* 4.9 (July 2021), p. 2100053. DOI: 10.1002/qute.202100053. URL: https://doi.org/10.1002/qute.202100053 (cit. on p. 178).

[1550] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini. "Opportunities in Quantum Reservoir Computing and Extreme Learning Machines." In: *Advanced Quantum Technologies* 4.8 (June 2021), p. 2100027. DOI: 10.1002/qute.202100027. URL: https://doi.org/10.1002/qute.202100027 (cit. on p. 178).

[1551] E. Wlaźlak, W. Macyk, W. Nitek, and K. Szaciłowski. "Influence of π-Iodide Intermolecular Interactions on Electronic Properties of Tin(IV) Iodide Semiconducting Complexes." In: *Inorganic Chemistry* 55.12 (June 2016), pp. 5935–5945. DOI: 10.1021/acs.inorgchem.6b00336. URL: https://doi.org/10.1021/acs.inorgchem.6b00336 (cit. on p. 178).

[1552] K. Liu, B. Dang, T. Zhang, Z. Yang, L. Bao, L. Xu, C. Cheng, R. Huang, and Y. Yang. "Multilayer Reservoir Computing Based on Ferroelectric α-In_2Se_3 for Hierarchical Information Processing." In: *Advanced Materials* 34.48 (Feb. 2022), p. 2108826. DOI: 10.1002/adma.202108826. URL: https://doi.org/10.1002/adma.202108826 (cit. on p. 178).

[1553] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice. "The rise of intelligent matter." In: *Nature* 594.7863 (June 2021), pp. 345–355. DOI: 10.1038/s41586-021-03453-y. URL: https://doi.org/10.1038/s41586-021-03453-y (cit. on p. 178).

[1554] S. Ganguli, D. Huh, and H. Sompolinsky. "Memory traces in dynamical systems." In: *Proceedings of the National Academy of Sciences* 105.48 (Dec. 2008), pp. 18970–18975. DOI: 10.1073/pnas.0804451105. URL: https://doi.org/10.1073/pnas.0804451105 (cit. on p. 178).

[1555] S. Ganguli and H. Sompolinsky. "Short-term memory in neuronal networks through dynamical compressed sensing." In: vol. 23. Jan. 2010, pp. 667–675. URL: https://ganguli-gang.stanford.edu/pdf/DynCompSense.pdf (cit. on p. 178).

[1556] L. Grigoryeva and J.-P. Ortega. "Echo state networks are universal." In: *Neural Networks* 108 (Dec. 2018), pp. 495–508. DOI: 10.1016/j.neunet.2018.08.025. URL: https://doi.org/10.1016/j.neunet.2018.08.025 (cit. on p. 178).

[1557] L. Gonon and J.-P. Ortega. "Reservoir Computing Universality With Stochastic Inputs." In: *IEEE Transactions on Neural Networks and Learning Systems* 31.1 (Jan. 2020), pp. 100–112. DOI: 10.1109/tnnls.2019.2899649. URL: https://doi.org/10.1109/tnnls.2019.2899649 (cit. on p. 178).

[1558] E. Bollt. "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (Jan. 2021), p. 013108. DOI: 10.1063/5.0024890. URL: https://doi.org/10.1063/5.0024890 (cit. on p. 178).

[1559] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa. "Next generation reservoir computing." In: *Nature Communications* 12.1 (Sept. 2021). DOI: 10.1038/s41467-021-25801-2. URL: https://doi.org/10.1038/s41467-021-25801-2 (cit. on p. 179).

[1560] H. Zhang and D. V. Vargas. "A Survey on Reservoir Computing and its Interdisciplinary Applications Beyond Traditional Machine Learning." In: *IEEE Access* 11 (2023), pp. 81033–81070. ISSN: 2169-3536. DOI: 10.1109/access.2023.3299296. URL: http://dx.doi.org/10.1109/access.2023.3299296 (cit. on p. 179).

[1561] M. Lukoševičius, H. Jaeger, and B. Schrauwen. "Reservoir Computing Trends." In: *KI - Künstliche Intelligenz* 26.4 (May 2012), pp. 365–371. DOI: 10.1007/s13218-012-0204-5. URL: https://doi.org/10.1007/s13218-012-0204-5 (cit. on p. 179).

[1562] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. "Recent advances in physical reservoir computing: A review." In: *Neural Networks* 115 (July 2019), pp. 100–123. DOI: 10.1016/j.neunet.2019.03.005. URL: https://doi.org/10.1016/j.neunet.2019.03.005 (cit. on p. 179).

[1563] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu. "Are Graph Convolutional Networks With Random Weights Feasible?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–18. DOI: 10.1109/tpami.2022.3183143. URL: https://doi.org/10.1109/tpami.2022.3183143 (cit. on p. 179).

[1564] Z. Liao. "A random matrix framework for large dimensional machine learning and neural networks." PhD thesis. Université Paris-Saclay, 2019. URL: https://theses.hal.science/tel-02397287/document (cit. on pp. 179, 186).

[1565] T. R. Davidson, L. Falorsi, N. D. Cao, T. Kipf, and J. M. Tomczak. "Hyperspherical Variational Auto-Encoders." In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*. Ed. by A. Globerson and R. Silva. AUAI Press, 2018, pp. 856–865. URL: http://auai.org/uai2018/proceedings/papers/309.pdf (cit. on pp. 179, 181).

[1566] Y. Chen, X.-H. Yang, Z. Wei, A. A. Heidari, N. Zheng, Z. Li, H. Chen, H. Hu, Q. Zhou, and Q. Guan. "Generative Adversarial Networks in Medical Image augmentation: A review." In: *Computers in Biology and Medicine* 144 (May 2022), p. 105382. DOI: 10.1016/j.compbiomed.2022.105382. URL: https://doi.org/10.1016/j.compbiomed.2022.105382 (cit. on p. 179).

[1567] M. Brundage et al. *The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation*. Tech. rep. 2018. DOI: 10.17863/CAM.22520. URL: https://www.repository.cam.ac.uk/handle/1810/275332 (cit. on p. 179).

[1568] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation." In: *CoRR* abs/1710.10196 (2017). arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196 (cit. on p. 179).

[1569] J. Wu, W. Gan, Z. Chen, S. Wan, and H. Lin. *AI-Generated Content (AIGC): A Survey*. 2023. DOI: 10.48550/ARXIV.2304.06632. URL: https://arxiv.org/abs/2304.06632 (cit. on pp. 179, 182, 183).

[1570] Y. Pang, J. Lin, T. Qin, and Z. Chen. "Image-to-Image Translation: Methods and Applications." In: *IEEE Transactions on Multimedia* 24 (2022), pp. 3859–3881. DOI: 10.1109/tmm.2021.3109419. URL: https://doi.org/10.1109/tmm.2021.3109419 (cit. on p. 179).

[1571] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.632. URL: https://doi.org/10.1109/cvpr.2017.632 (cit. on pp. 179, 182).

[1572] Y. Chen, X. Yu, S. Liu, W. Gao, and G. Li. "Zero-shot unsupervised image-to-image translation via exploiting semantic attributes." In: *Image and Vision Computing* 124 (Aug. 2022), p. 104489. DOI: 10.1016/j.imavis.2022.104489. URL: https://doi.org/10.1016/j.imavis.2022.104489 (cit. on p. 179).

[1573] V. Talasila, N. M. R, and M. M. V. "Optimized GAN for Text-to-Image Synthesis: Hybrid Whale Optimization Algorithm and Dragonfly Algorithm." In: *Advances in Engineering Software* 173 (Nov. 2022), p. 103222. DOI: 10.1016/j.advengsoft.2022.103222. URL: https://doi.org/10.1016/j.advengsoft.2022.103222 (cit. on pp. 179, 182).

[1574] T. Qiao, J. Zhang, D. Xu, and D. Tao. "Learn, Imagine and Create: Text-to-Image Generation from Prior Knowledge." In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/d18f655c3fce66ca401d5f38b48c89af-Paper.pdf (cit. on pp. 179, 182).

[1575] U. Ullah, J.-S. Lee, C.-H. An, H. Lee, S.-Y. Park, R.-H. Baek, and H.-C. Choi. "A Review of Multi-Modal Learning from the Text-Guided Visual Processing Viewpoint." In: *Sensors* 22.18 (Sept. 2022), p. 6816. DOI: 10.3390/s22186816. URL: https://doi.org/10.3390/s22186816 (cit. on pp. 179, 183).

[1576] R. Mechrez, I. Talmi, and L. Zelnik-Manor. "The Contextual Loss for Image Transformation with Non-aligned Data." In: *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 800–815. DOI: 10.1007/978-3-030-01264-9_47. URL: https://doi.org/10.1007/978-3-030-01264-9_47 (cit. on p. 179).

[1577] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution." In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 694–711. DOI: 10.1007/978-3-319-46475-6_43. URL: https://doi.org/10.1007/978-3-319-46475-6_43 (cit. on p. 180).

[1578] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.244. URL: https://doi.org/10.1109/iccv.2017.244 (cit. on pp. 180, 182).

[1579] M. Liu, Y. Ding, M. Xia, X. Liu, E. Ding, W. Zuo, and S. Wen. "STGAN: A Unified Selective Transfer Network for Arbitrary Image Attribute Editing." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.00379. URL: https://doi.org/10.1109/cvpr.2019.00379 (cit. on pp. 180, 182).

[1580] C. Louizos, U. Shalit, J. Mooij, D. Sontag, R. Zemel, and M. Welling. "Causal Effect Inference with Deep Latent-Variable Models." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6449–6459. ISBN: 9781510860964 (cit. on pp. 180, 181).

[1581] Y. Lu, H. Wang, and W. Wei. *Machine Learning for Synthetic Data Generation: A Review*. 2023. DOI: 10.48550/ARXIV.2302.04062. URL: https://arxiv.org/abs/2302.04062 (cit. on p. 180).

[1582] J.-F. Rajotte, R. Bergen, D. L. Buckeridge, K. E. Emam, R. Ng, and E. Strome. "Synthetic data as an enabler for machine learning applications in medicine." In: *iScience* 25.11 (Nov. 2022), p. 105331. DOI: 10.1016/j.isci.2022.105331. URL: https://doi.org/10.1016/j.isci.2022.105331 (cit. on p. 180).

[1583] J. Tremblay, T. To, and S. Birchfield. "Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2018. DOI: 10.1109/cvprw.2018.00275. URL: https://doi.org/10.1109/cvprw.2018.00275 (cit. on p. 180).

[1584] K. Maharana, S. Mondal, and B. Nemade. "A review: Data pre-processing and data augmentation techniques." In: *Global Transitions Proceedings* 3.1 (June 2022), pp. 91–99. DOI: 10.1016/j.gltp.2022.04.020. URL: https://doi.org/10.1016/j.gltp.2022.04.020 (cit. on p. 180).

[1585] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. *Building Generalizable Agents with a Realistic and Rich 3D Environment*. 2018. arXiv: 1801.02209 [cs.LG] (cit. on p. 180).

[1586] M. Fabbri, G. Braso, G. Maugeri, O. Cetintas, R. Gasparini, A. Osep, S. Calderara, L. Leal-Taixe, and R. Cucchiara. *MOTSynth: How Can Synthetic Data Help Pedestrian Detection and Tracking?* 2021. arXiv: 2108.09518 [cs.CV] (cit. on p. 180).

[1587] C. Shorten and T. M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning." In: *Journal of Big Data* 6.1 (July 2019). DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/Fs40537-019-0197-0 (cit. on p. 180).

[1588] F. Zhou, S. Yang, H. Fujita, D. Chen, and C. Wen. "Deep learning fault diagnosis method based on global optimization GAN for unbalanced data." In: *Knowledge-Based Systems* 187 (Jan. 2020), p. 104837. DOI: 10.1016/j.knosys.2019.07.008. URL: https://doi.org/10.1016/j.knosys.2019.07.008 (cit. on pp. 180, 183).

[1589] J. Luo, L. Zhu, Q. Li, D. Liu, and M. Chen. "Imbalanced Fault Diagnosis of Rotating Machinery Based on Deep Generative Adversarial Networks with Gradient Penalty." In: *Processes* 9.10 (Sept. 2021), p. 1751. DOI: 10.3390/pr9101751. URL: https://doi.org/10.3390/pr9101751 (cit. on pp. 180, 182, 183).

[1590] M. Abufadda and K. Mansour. "A Survey of Synthetic Data Generation for Machine Learning." In: *2021 22nd International Arab Conference on Information Technology (ACIT)*. IEEE, Dec. 2021. DOI: 10.1109/acit53391.2021.9677302. URL: https://doi.org/10.1109/acit53391.2021.9677302 (cit. on p. 180).

[1591] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. "Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing." In: SEC'14. San Diego, CA: USENIX Association, 2014, pp. 17–32. ISBN: 9781931971157 (cit. on p. 180).

[1592] M. Fredrikson, S. Jha, and T. Ristenpart. "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures." In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2015. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677 (cit. on p. 180).

[1593] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang. "Membership Inference Attacks on Machine Learning: A Survey." In: *ACM Computing Surveys* 54.11s (Jan. 2022), pp. 1–37. DOI: 10.1145/3523273. URL: https://doi.org/10.1145/3523273 (cit. on p. 180).

[1594] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. "Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting." In: *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, July 2018. DOI: 10.1109/csf.2018.00027. URL: https://doi.org/10.1109/csf.2018.00027 (cit. on p. 180).

[1595] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. "A Methodology for Formalizing Model-Inversion Attacks." In: *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, June 2016. DOI: 10.1109/csf.2016.32. URL: https://doi.org/10.1109/csf.2016.32 (cit. on p. 180).

[1596] S. Yeom, I. Giacomelli, A. Menaged, M. Fredrikson, and S. Jha. "Overfitting, robustness, and malicious algorithms: A study of potential causes of privacy risk in machine learning." In: *Journal of Computer Security* 28.1 (Feb. 2020), pp. 35–70. DOI: 10.3233/jcs-191362. URL: https://doi.org/10.3233/jcs-191362 (cit. on p. 180).

[1597] M. Xu and X. Li. "Subject Property Inference Attack in Collaborative Learning." In: *2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. IEEE, Aug. 2020. DOI: 10.1109/ihmsc49165.2020.00057. URL: https://doi.org/10.1109/ihmsc49165.2020.00057 (cit. on p. 180).

[1598] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri. "Enhanced Membership Inference Attacks against Machine Learning Models." In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Nov. 2022. DOI: 10.1145/3548606.3560675. URL: https://doi.org/10.1145/3548606.3560675 (cit. on p. 180).

[1599] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. "Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays." In: *PLoS Genetics* 4.8 (Aug. 2008). Ed. by P. M. Visscher, e1000167. DOI: 10.1371/journal.pgen.1000167. URL: https://doi.org/10.1371/journal.pgen.1000167 (cit. on p. 180).

[1600] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. "Membership Inference Attacks Against Machine Learning Models." In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2017. DOI: 10.1109/sp.2017.41. URL: https://doi.org/10.1109/sp.2017.41 (cit. on p. 180).

[1601] X. Gong, Y. Chen, Q. Wang, M. Wang, and S. Li. "Private Data Inference Attacks against Cloud: Model, Technologies, and Research Directions." In: *IEEE Communications Magazine* 60.9 (Sept. 2022), pp. 46–52. DOI: 10.1109/mcom.004.2100867. URL: https://doi.org/10.1109/mcom.004.2100867 (cit. on p. 180).

[1602] M. Backes, P. Berrang, M. Humbert, and P. Manoharan. "Membership Privacy in MicroRNA-based Studies." In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016. DOI: 10.1145/2976749.2978355. URL: https://doi.org/10.1145/2976749.2978355 (cit. on p. 180).

[1603] Y. Long, L. Wang, D. Bu, V. Bindschaedler, X. Wang, H. Tang, C. A. Gunter, and K. Chen. "A Pragmatic Approach to Membership Inferences on Machine Learning Models." In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, Sept. 2020. DOI: 10.1109/eurosp48549.2020.00040. URL: https://doi.org/10.1109/eurosp48549.2020.00040 (cit. on p. 180).

[1604] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models." In: *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019. DOI: 10.14722/ndss.2019.23119. URL: https://doi.org/10.14722/ndss.2019.23119 (cit. on p. 180).

[1605] C. Dwork and A. Roth. "The Algorithmic Foundations of Differential Privacy." In: *Foundations and Trends® in Theoretical Computer Science* 9.3-4 (2013), pp. 211–407. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042 (cit. on p. 180).

[1606] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. "Deep Learning with Differential Privacy." In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145/2976749.2978318 (cit. on p. 180).

[1607] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. "Differentially Private Empirical Risk Minimization." In: *J. Mach. Learn. Res.* 12 (July 2011), pp. 1069–1109. ISSN: 1532-4435 (cit. on p. 180).

[1608] P. Wang, Z. Yang, Y. Lei, Y. Ying, and H. Zhang. "Differentially private empirical risk minimization for AUC maximization." In: *Neurocomputing* 461 (Oct. 2021), pp. 419–437. DOI: 10.1016/j.neucom.2021.07.001. URL: https://doi.org/10.1016/j.neucom.2021.07.001 (cit. on p. 180).

[1609] X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, and A. V. Vasilakos. "Privacy and Security Issues in Deep Learning: A Survey." In: *IEEE Access* 9 (2021), pp. 4566–4593. DOI: 10.1109/access.2020.3045078. URL: https://doi.org/10.1109/access.2020.3045078 (cit. on p. 180).

[1610] A. Honkela, M. Das, A. Nieminen, O. Dikmen, and S. Kaski. "Efficient differentially private learning improves drug sensitivity prediction." In: *Biology Direct* 13.1 (Feb. 2018). DOI: 10.1186/s13062-017-0203-4. URL: https://doi.org/10.1186/s13062-017-0203-4 (cit. on p. 180).

[1611] S. Tople, A. Sharma, and A. V. Nori. "Alleviating Privacy Attacks via Causal Learning." In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020 (cit. on p. 180).

[1612] M. Gong, Y. Xie, K. Pan, K. Feng, and A. Qin. "A Survey on Differentially Private Machine Learning [Review Article]." In: *IEEE Computational Intelligence Magazine* 15.2 (May 2020), pp. 49–64. DOI: 10.1109/mci.2020.2976185. URL: https://doi.org/10.1109/mci.2020.2976185 (cit. on p. 180).

[1613] P. Jain and A. Thakurta. "Differentially Private Learning with Kernels." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 118–126. URL: https://proceedings.mlr.press/v28/jain13.html (cit. on p. 180).

[1614] P. Jain, P. Kothari, and A. Thakurta. "Differentially Private Online Learning." In: *Proceedings of the 25th Annual Conference on Learning Theory*. Ed. by S. Mannor, N. Srebro, and R. C. Williamson. Vol. 23. Proceedings of Machine Learning Research. Edinburgh, Scotland: PMLR, 25–27 Jun 2012, pp. 24.1–24.34. URL: https://proceedings.mlr.press/v23/jain12.html (cit. on p. 180).

[1615] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex. "Differentially Private Model Publishing for Deep Learning." In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2019. DOI: 10.1109/sp.2019.00019. URL: https://doi.org/10.1109/sp.2019.00019 (cit. on p. 180).

[1616] M. Jagielski, J. Ullman, and A. Oprea. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546 (cit. on p. 180).

[1617] F. Tramer and D. Boneh. "Differentially Private Learning Needs Better Features (or Much More Data)." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YTWGvpFOQD- (cit. on p. 180).

[1618] J. Wang and Z.-H. Zhou. "Differentially Private Learning with Small Public Data." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 6219–6226. DOI: 10.1609/aaai.v34i04.6088. URL: https://doi.org/10.1609/aaai.v34i04.6088 (cit. on p. 180).

[1619] N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. Thakurta. *How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy*. 2023. DOI: 10.48550/ARXIV.2303.00654. URL: https://arxiv.org/abs/2303.00654 (cit. on p. 180).

[1620] Z. Zhang, C. Yan, and B. A. Malin. "Membership inference attacks against synthetic health data." In: *Journal of Biomedical Informatics* 125 (Jan. 2022), p. 103977. DOI: 10.1016/j.jbi.2021.103977. URL: https://doi.org/10.1016/j.jbi.2021.103977 (cit. on p. 180).

[1621] K. S. Liu, C. Xiao, B. Li, and J. Gao. "Performing Co-membership Attacks Against Deep Generative Models." In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2019. DOI: 10.1109/icdm.2019.00056. URL: https://doi.org/10.1109/icdm.2019.00056 (cit. on p. 180).

[1622] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro. "LOGAN: Membership Inference Attacks Against Generative Models." In: *Proceedings on Privacy Enhancing Technologies* 2019.1 (Dec. 2018), pp. 133–152. DOI: 10.2478/popets-2019-0008. URL: https://doi.org/10.2478/popets-2019-0008 (cit. on p. 180).

[1623] D. Chen, N. Yu, Y. Zhang, and M. Fritz. "GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models." In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2020. DOI: 10.1145/3372297.3417238. URL: https://doi.org/10.1145/3372297.3417238 (cit. on p. 180).

[1624] B. Hilprecht, M. Härterich, and D. Bernau. "Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models." In: *Proceedings on Privacy Enhancing Technologies* 2019.4 (July 2019), pp. 232–249. DOI: 10.2478/popets-2019-0067. URL: https://doi.org/10.2478/popets-2019-0067 (cit. on p. 180).

[1625] S. Mukherjee, Y. Xu, A. Trivedi, N. Patowary, and J. L. Ferres. "privGAN: Protecting GANs from membership inference attacks at low cost to utility." In: *Proceedings on Privacy Enhancing Technologies* 2021.3 (Apr. 2021), pp. 142–163. DOI: 10.2478/popets-2021-0041. URL: https://doi.org/10.2478/popets-2021-0041 (cit. on p. 180).

[1626] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks." In: *Proceedings of the 28th USENIX Security Symposium*. USENIX Associatio, Aug. 2019. URL: https://www.usenix.org/system/files/sec19-carlini.pdf (cit. on p. 180).

[1627] G. E. Hinton. "Training Products of Experts by Minimizing Contrastive Divergence." In: *Neural Computation* 14.8 (Aug. 2002), pp. 1771–1800. DOI: 10.1162/089976602760128018. URL: https://doi.org/10.1162/089976602760128018 (cit. on p. 180).

[1628] V. Upadhya and P. S. Sastry. "An Overview of Restricted Boltzmann Machines." In: *Journal of the Indian Institute of Science* 99.2 (Feb. 2019), pp. 225–236. DOI: 10.1007/s41745-019-0102-z. URL: https://doi.org/10.1007/s41745-019-0102-z (cit. on p. 180).

[1629] M. Ranzato, A. Krizhevsky, and G. Hinton. "Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Y. W. Teh and M. Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 621–628. URL: https://proceedings.mlr.press/v9/ranzato10a.html (cit. on p. 180).

[1630] G. W. Taylor, G. E. Hinton, and S. Roweis. "Modeling Human Motion Using Binary Latent Variables." In: *Advances in Neural Information Processing Systems*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Vol. 19. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper_files/paper/2006/file/1091660f3dff84fd648efe31391c5524-Paper.pdf (cit. on p. 180).

[1631] R. Souriau, J. Lerbet, H. Chen, and V. Vigneron. "A review on generative Boltzmann networks applied to dynamic systems." In: *Mechanical Systems and Signal Processing* 147 (Jan. 2021), p. 107072. DOI: 10.1016/j.ymssp.2020.107072. URL: https://doi.org/10.1016/j.ymssp.2020.107072 (cit. on p. 180).

[1632] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." In: *Science* 313.5786 (July 2006), pp. 504–507. DOI: 10.1126/science.1127647. URL: https://doi.org/10.1126/science.1127647 (cit. on p. 180).

[1633] G. E. Hinton, S. Osindero, and Y.-W. Teh. "A Fast Learning Algorithm for Deep Belief Nets." In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527. URL: https://doi.org/10.1162/neco.2006.18.7.1527 (cit. on p. 180).

[1634] S. Osindero and G. E. Hinton. “Modeling image patches with a directed hierarchy of Markov random fields.” In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Ed. by J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis. Curran Associates, Inc., 2007, pp. 1121–1128. URL: https://proceedings.neurips.cc/paper/2007/hash/9232fe81225bcaef853ae32870a2b0fe-Abstract.html (cit. on p. 180).

[1635] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.” In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, June 2009. DOI: 10.1145/1553374.1553453. URL: https://doi.org/10.1145/1553374.1553453 (cit. on pp. 180, 181).

[1636] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. “Stacked Generative Adversarial Networks.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.202. URL: https://doi.org/10.1109/cvpr.2017.202 (cit. on pp. 180, 182).

[1637] R. Salakhutdinov and G. Hinton. “Deep Boltzmann Machines.” In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by D. van Dyk and M. Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 448–455. URL: https://proceedings.mlr.press/v5/salakhutdinov09a.html (cit. on p. 180).

[1638] Y. Li, L. Zou, L. Jiang, and X. Zhou. “Fault Diagnosis of Rotating Machinery Based on Combination of Deep Belief Network and One-dimensional Convolutional Neural Network.” In: *IEEE Access* 7 (2019), pp. 165710–165723. DOI: 10.1109/access.2019.2953490. URL: https://doi.org/10.1109/access.2019.2953490 (cit. on p. 181).

[1639] T. Pan, J. Chen, and Z. Zhou. “Intelligent Fault Diagnosis of Rolling Bearing via Deep-Layerwise Feature Extraction Using Deep Belief Network.” In: *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*. IEEE, Aug. 2018. DOI: 10.1109/sdpc.2018.8664995. URL: https://doi.org/10.1109/sdpc.2018.8664995 (cit. on p. 181).

[1640] H. Lee, P. Pham, Y. Largman, and A. Ng. “Unsupervised feature learning for audio classification using convolutional deep belief networks.” In: *Advances in Neural Information Processing Systems*. Ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta. Vol. 22. Curran Associates, Inc., 2009. URL: https://proceedings.neurips.cc/paper_files/paper/2009/file/a113c1ecd3cace2237256f4c712f61b5-Paper.pdf (cit. on p. 181).

[1641] Y. Zhang and J. Ji. “Intelligent Fault Diagnosis of a Reciprocating Compressor Using Mode Isolation Convolutional Deep Belief Networks.” In: *IEEE/ASME Transactions on Mechatronics* 26.3 (June 2021), pp. 1668–1677. DOI: 10.1109/tmech.2020.3027912. URL: https://doi.org/10.1109/tmech.2020.3027912 (cit. on p. 181).

[1642] T. Osogami. *Boltzmann machines and energy-based models*. 2017. DOI: 10.48550/ARXIV.1708.06008. URL: https://arxiv.org/abs/1708.06008 (cit. on p. 181).

[1643] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf (cit. on p. 181).

[1644] M. Arbel, L. Zhou, and A. Gretton. “Generalized Energy Based Models.” In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=0PtUPB9z6qK (cit. on p. 181).

[1645] M. Roder, G. H. de Rosa, V. H. C. de Albuquerque, A. L. D. Rossi, and J. P. Papa. “Energy-Based Dropout in Restricted Boltzmann Machines: Why Not Go Random.” In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 6.2 (Apr. 2022), pp. 276–286. DOI: 10.1109/tetci.2020.3043764. URL: https://doi.org/10.1109/tetci.2020.3043764 (cit. on p. 181).

[1646] P. Duda, L. Rutkowski, P. Woldan, and P. Najgebauer. “The Streaming Approach to Training Restricted Boltzmann Machines.” In: *Artificial Intelligence and Soft Computing*. Springer International Publishing, 2021, pp. 308–317. DOI: 10.1007/978-3-030-87986-0_27. URL: https://doi.org/10.1007/978-3-030-87986-0_27 (cit. on p. 181).

[1647] S. Elfwing, E. Uchibe, and K. Doya. “Expected energy-based restricted Boltzmann machine for classification.” In: *Neural Networks* 64 (Apr. 2015), pp. 29–38. DOI: 10.1016/j.neunet.2014.09.006. URL: https://doi.org/10.1016/j.neunet.2014.09.006 (cit. on p. 181).

[1648] J. Zhang, S. Ding, T. Sun, and L. Guo. “A Gaussian RBM with binary auxiliary units.” In: *International Journal of Machine Learning and Cybernetics* 13.9 (Mar. 2022), pp. 2425–2433. DOI: 10.1007/s13042-022-01534-6. URL: https://doi.org/10.1007/s13042-022-01534-6 (cit. on p. 181).

[1649] J. Zhai, X. Zhou, S. Zhang, and T. Wang. "Ensemble RBM-based classifier using fuzzy integral for big data classification." In: *International Journal of Machine Learning and Cybernetics* 10.11 (May 2019), pp. 3327–3337. DOI: 10.1007/s13042-019-00960-3. URL: https://doi.org/10.1007/s13042-019-00960-3 (cit. on p. 181).

[1650] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. "Neural Autoregressive Distribution Estimation." In: *J. Mach. Learn. Res.* 17.1 (Jan. 2016), pp. 7184–7220. ISSN: 1532-4435 (cit. on p. 181).

[1651] H. Larochelle and I. Murray. "The Neural Autoregressive Distribution Estimator." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 29–37. URL: https://proceedings.mlr.press/v15/larochelle11a.html (cit. on p. 181).

[1652] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. "Deep AutoRegressive Networks." In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by E. P. Xing and T. Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1242–1250. URL: https://proceedings.mlr.press/v32/gregor14.html (cit. on p. 181).

[1653] K. Gregor and Y. LeCun. *Learning Representations by Maximizing Compression*. 2011. DOI: 10.48550/ARXIV.1108.1169. URL: https://arxiv.org/abs/1108.1169 (cit. on p. 181).

[1654] M. Germain, K. Gregor, I. Murray, and H. Larochelle. "MADE: Masked Autoencoder for Distribution Estimation." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 881–889. URL: https://proceedings.mlr.press/v37/germain15.html (cit. on p. 181).

[1655] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Pixel Recurrent Neural Networks." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1747–1756. URL: https://proceedings.mlr.press/v48/oord16.html (cit. on p. 181).

[1656] L. Theis and M. Bethge. "Generative Image Modeling Using Spatial LSTMs." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1927–1935 (cit. on p. 181).

[1657] R. Wei and A. Mahmood. "Recent Advances in Variational Autoencoders With Representation Learning for Biomedical Informatics: A Survey." In: *IEEE Access* 9 (2021), pp. 4939–4956. DOI: 10.1109/access.2020.3048309. URL: https://doi.org/10.1109/access.2020.3048309 (cit. on p. 181).

[1658] C. Doersch. *Tutorial on Variational Autoencoders*. 2016. DOI: 10.48550/ARXIV.1606.05908. URL: https://arxiv.org/abs/1606.05908 (cit. on p. 181).

[1659] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. "Semi-Supervised Learning with Deep Generative Models." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 3581–3589 (cit. on p. 181).

[1660] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by E. P. Xing and T. Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1278–1286. URL: https://proceedings.mlr.press/v32/rezende14.html (cit. on p. 181).

[1661] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. DOI: 10.1109/tpami.2013.50. URL: https://doi.org/10.1109/tpami.2013.50 (cit. on p. 181).

[1662] A. T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli. "The Autoencoding Variational Autoencoder." In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546 (cit. on p. 181).

[1663] G. Alain and Y. Bengio. "What Regularized Auto-Encoders Learn from the Data-Generating Distribution." In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 3563–3593. ISSN: 1532-4435 (cit. on p. 181).

[1664] Y. Zhu, M. R. Min, A. Kadav, and H. P. Graf. "S3VAE: Self-Supervised Sequential VAE for Representation Disentanglement and Data Generation." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.00657. URL: https://doi.org/10.1109/cvpr42600.2020.00657 (cit. on p. 181).

[1665] J. Bai, W. Wang, and C. P. Gomes. "Contrastively Disentangled Sequential Variational Autoencoder." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021. URL: https://openreview.net/forum?id=rWPxhfz2_S (cit. on p. 181).

[1666] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. "Neural Discrete Representation Learning." In: NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6309–6318. ISBN: 9781510860964 (cit. on p. 181).

[1667] A. Razavi, A. van den Oord, and O. Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2." In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019. URL: https://dl.acm.org/doi/10.5555/3454287.3455618 (cit. on p. 181).

[1668] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. "Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Australia: AAAI Press, 2017, pp. 1965–1972. ISBN: 9780999241103 (cit. on p. 181).

[1669] Z. Li, Y. Zhao, H. Xu, W. Chen, S. Xu, Y. Li, and D. Pei. "Unsupervised Clustering through Gaussian Mixture Variational AutoEncoder with Non-Reparameterized Variational Inference and Std Annealing." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2020. DOI: 10.1109/ijcnn48605.2020.9207493. URL: https://doi.org/10.1109/ijcnn48605.2020.9207493 (cit. on p. 181).

[1670] S. Zhao, J. Song, and S. Ermon. "InfoVAE: Balancing Learning and Inference in Variational Autoencoders." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 5885–5892. DOI: 10.1609/aaai.v33i01.33015885. URL: https://doi.org/10.1609/aaai.v33i01.33015885 (cit. on p. 181).

[1671] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl (cit. on p. 181).

[1672] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. "DRAW: A Recurrent Neural Network For Image Generation." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1462–1471. URL: https://proceedings.mlr.press/v37/gregor15.html (cit. on p. 181).

[1673] A. Vahdat and J. Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder." In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546. URL: https://dl.acm.org/doi/abs/10.5555/3495724.3497374 (cit. on p. 181).

[1674] U. Hwang, J. Park, H. Jang, S. Yoon, and N. I. Cho. "PuVAE: A Variational Autoencoder to Purify Adversarial Examples." In: *IEEE Access* 7 (2019), pp. 126582–126593. DOI: 10.1109/access.2019.2939352. URL: https://doi.org/10.1109/access.2019.2939352 (cit. on p. 181).

[1675] H. Xu, S. Lu, Z. Sun, C. Ma, and C. Guo. "VAE based Text Style Transfer with Pivot Words Enhancement Learning." In: *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*. National Institute of Technology Silchar, Silchar, India: NLP Association of India (NLPAI), Dec. 2021, pp. 162–172. URL: https://aclanthology.org/2021.icon-main.20 (cit. on p. 181).

[1676] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. "Autoencoding beyond pixels using a learned similarity metric." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1558–1566. URL: https://proceedings.mlr.press/v48/larsen16.html (cit. on pp. 181, 182).

[1677] J. Li, D. Kang, W. Pei, X. Zhe, Y. Zhang, Z. He, and L. Bao. "Audio2Gestures: Generating Diverse Gestures from Speech Audio with Conditional Variational Autoencoders." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01110. URL: https://doi.org/10.1109/iccv48922.2021.01110 (cit. on p. 181).

[1678] L. Ternes, M. Dane, S. Gross, M. Labrie, G. Mills, J. Gray, L. Heiser, and Y. H. Chang. "A multi-encoder variational autoencoder controls multiple transformational features in single-cell image analysis." In: *Communications Biology* 5.1 (Mar. 2022). DOI: 10.1038/s42003-022-03218-x. URL: https://doi.org/10.1038/s42003-022-03218-x (cit. on p. 181).

[1679] H. Hadipour, C. Liu, R. Davis, S. T. Cardona, and P. Hu. "Deep clustering of small molecules at large-scale via variational autoencoder embedding and K-means." In: *BMC Bioinformatics* 23.S4 (Apr. 2022). ISSN: 1471-2105. DOI: 10.1186/s12859-022-04667-1. URL: http://dx.doi.org/10.1186/s12859-022-04667-1 (cit. on p. 181).

[1680] N. Russkikh, D. Antonets, D. Shtokalo, A. Makarov, Y. Vyatkin, A. Zakharov, and E. Terentyev. "Style transfer with variational autoencoders is a promising approach to RNA-Seq data harmonization and analysis." In: *Bioinformatics* 36.20 (July 2020). Ed. by Z. Lu, pp. 5076–5085. DOI: 10.1093/bioinformatics/btaa624. URL: https://doi.org/10.1093/bioinformatics/btaa624 (cit. on p. 181).

[1681] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2672–2680 (cit. on pp. 181, 182).

[1682] H. Alqahtani, M. Kavakli-Thorne, and G. Kumar. "Applications of Generative Adversarial Networks (GANs): An Updated Review." In: *Archives of Computational Methods in Engineering* 28.2 (Dec. 2019), pp. 525–552. DOI: 10.1007/s11831-019-09388-y. URL: https://doi.org/10.1007/s11831-019-09388-y (cit. on pp. 182, 184).

[1683] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang. "GAN Inversion: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–17. DOI: 10.1109/tpami.2022.3181070. URL: https://doi.org/10.1109/tpami.2022.3181070 (cit. on p. 182).

[1684] X. Wu, K. Xu, and P. Hall. "A survey of image synthesis and editing with generative adversarial networks." In: *Tsinghua Science and Technology* 22.6 (Dec. 2017), pp. 660–674. DOI: 10.23919/tst.2017.8195348. URL: https://doi.org/10.23919/tst.2017.8195348 (cit. on p. 182).

[1685] Z. Zhang, Z. Li, K. Wei, S. Pan, and C. Deng. "A survey on multimodal-guided visual content synthesis." In: *Neurocomputing* 497 (Aug. 2022), pp. 110–128. DOI: 10.1016/j.neucom.2022.04.126. URL: https://doi.org/10.1016/j.neucom.2022.04.126 (cit. on p. 182).

[1686] L. Lan, L. You, Z. Zhang, Z. Fan, W. Zhao, N. Zeng, Y. Chen, and X. Zhou. "Generative Adversarial Networks and Its Applications in Biomedical Informatics." In: *Frontiers in Public Health* 8 (May 2020). DOI: 10.3389/fpubh.2020.00164. URL: https://doi.org/10.3389/fpubh.2020.00164 (cit. on p. 182).

[1687] R. Gao, X. Hou, J. Qin, J. Chen, L. Liu, F. Zhu, Z. Zhang, and L. Shao. "Zero-VAE-GAN: Generating Unseen Features for Generalized and Transductive Zero-Shot Learning." In: *IEEE Transactions on Image Processing* 29 (2020), pp. 3665–3680. DOI: 10.1109/tip.2020.2964429. URL: https://doi.org/10.1109/tip.2020.2964429 (cit. on p. 182).

[1688] Y. Xian, S. Sharma, B. Schiele, and Z. Akata. "F-VAEGAN-D2: A Feature Generating Framework for Any-Shot Learning." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.01052. URL: https://doi.org/10.1109/cvpr.2019.01052 (cit. on p. 182).

[1689] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata. "Feature Generating Networks for Zero-Shot Learning." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00581. URL: https://doi.org/10.1109/cvpr.2018.00581 (cit. on p. 182).

[1690] B. Zhao, W. Li, and W. Gong. "Real-aware motion deblurring using multi-attention CycleGAN with contrastive guidance." In: *Digital Signal Processing* 135 (Apr. 2023), p. 103953. DOI: 10.1016/j.dsp.2023.103953. URL: https://doi.org/10.1016/j.dsp.2023.103953 (cit. on p. 182).

[1691] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. "Toward Multimodal Image-to-Image Translation." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 465–476. ISBN: 9781510860964. URL: https://dl.acm.org/doi/10.5555/3294771.3294816 (cit. on p. 182).

[1692] Q. Qi, J. Guo, and W. Jin. "EGAN: Non-uniform image deblurring based on edge adversarial mechanism and partial weight sharing network." In: *Signal Processing: Image Communication* 88 (Oct. 2020), p. 115952. DOI: 10.1016/j.image.2020.115952. URL: https://doi.org/10.1016/j.image.2020.115952 (cit. on p. 182).

[1693] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. "StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.629. URL: https://doi.org/10.1109/iccv.2017.629 (cit. on p. 182).

[1694] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. "StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.8 (Aug. 2019), pp. 1947–1962. DOI: 10.1109/tpami.2018.2856256. URL: https://doi.org/10.1109/tpami.2018.2856256 (cit. on p. 182).

[1695] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. "Learning What and Where to Draw." In: NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 217–225. ISBN: 9781510838819 (cit. on p. 182).

[1696] T. R. Shaham, T. Dekel, and T. Michaeli. "SinGAN: Learning a Generative Model From a Single Natural Image." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00467. URL: https://doi.org/10.1109/iccv.2019.00467 (cit. on p. 182).

[1697] X. He and Z. Fu. "Recurrent SinGAN." In: *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*. ACM, Oct. 2021. DOI: 10.1145/3501409.3501476. URL: https://doi.org/10.1145/3501409.3501476 (cit. on p. 182).

[1698] X. Wang and A. Gupta. "Generative Image Modeling Using Style and Structure Adversarial Networks." In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 318–335. DOI: 10.1007/978-3-319-46493-0_20. URL: https://doi.org/10.1007/978-3-319-46493-0_20 (cit. on p. 182).

[1699] M. Mehralian and B. Karasfi. "RDCGAN: Unsupervised Representation Learning With Regularized Deep Convolutional Generative Adversarial Networks." In: *2018 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium*. IEEE, Dec. 2018. DOI: 10.1109/aiar.2018.8769811. URL: https://doi.org/10.1109/aiar.2018.8769811 (cit. on p. 182).

[1700] A. Odena, C. Olah, and J. Shlens. "Conditional Image Synthesis with Auxiliary Classifier GANs." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2642–2651. URL: https://proceedings.mlr.press/v70/odena17a.html (cit. on p. 182).

[1701] Y. Jiang, S. Chang, and Z. Wang. "TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 14745–14758. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/7c220a2091c26a7f5e9f1cfb099511e3-Paper.pdf (cit. on p. 182).

[1702] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. "Self-Attention Generative Adversarial Networks." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 7354–7363. URL: https://proceedings.mlr.press/v97/zhang19d.html (cit. on p. 182).

[1703] M.-Y. Liu and O. Tuzel. "Coupled Generative Adversarial Networks." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/502e4a16930e414107ee22b6198c578f-Paper.pdf (cit. on p. 182).

[1704] J. Wang, E. Zhang, S. Cui, J. Wang, Q. Zhang, J. Fan, and J. Peng. "GGD-GAN: Gradient-Guided dual-Branch adversarial networks for relic sketch generation." In: *Pattern Recognition* 141 (Sept. 2023), p. 109586. DOI: 10.1016/j.patcog.2023.109586. URL: https://doi.org/10.1016/j.patcog.2023.109586 (cit. on p. 182).

[1705] A. Dudhane, H. S. Aulakh, and S. Murala. "RI-GAN: An End-To-End Network for Single Image Haze Removal." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2019. DOI: 10.1109/cvprw.2019.00253. URL: https://doi.org/10.1109/cvprw.2019.00253 (cit. on p. 182).

[1706] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00143. URL: https://doi.org/10.1109/cvpr.2018.00143 (cit. on p. 182).

[1707] X. Gong, S. Chang, Y. Jiang, and Z. Wang. "AutoGAN: Neural Architecture Search for Generative Adversarial Networks." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00332. URL: https://doi.org/10.1109/iccv.2019.00332 (cit. on p. 182).

[1708] B. Wang, T. Wu, M. Zhu, and P. Du. "Interactive Image Synthesis With Panoptic Layout Generation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 7783–7792. URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Wang_Interactive_Image_Synthesis_With_Panoptic_Layout_Generation_CVPR_2022_paper.pdf (cit. on p. 182).

[1709] M. Tao, H. Tang, F. Wu, X.-Y. Jing, B.-K. Bao, and C. Xu. "DF-GAN: A Simple and Effective Baseline for Text-to-Image Synthesis." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 16515–16525. URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Tao_DF-GAN_A_Simple_and_Effective_Baseline_for_Text-to-Image_Synthesis_CVPR_2022_paper.pdf (cit. on p. 182).

[1710] P. Sumi, S. Sindhuja, and S. Sureshkumar. "A Comparison between AttnGAN and DF GAN: Text to Image Synthesis." In: *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. IEEE, May 2021. DOI: 10.1109/icspc51351.2021.9451789. URL: https://doi.org/10.1109/icspc51351.2021.9451789 (cit. on p. 182).

[1711] M. Bahani, A. E. Ouaazizi, and K. Maalmi. "AraBERT and DF-GAN fusion for Arabic text-to-image generation." In: *Array* 16 (Dec. 2022), p. 100260. DOI: 10.1016/j.array.2022.100260. URL: https://doi.org/10.1016/j.array.2022.100260 (cit. on p. 182).

[1712] L. Feng, G. Geng, Q. Li, Y. Jiang, Z. Li, and K. Li. "CRPGAN: Learning image-to-image translation of two unpaired images by cross-attention mechanism and parallelization strategy." In: *PLOS ONE* 18.1 (Jan. 2023). Ed. by X. Kong, e0280073. DOI: 10.1371/journal.pone.0280073. URL: https://doi.org/10.1371/journal.pone.0280073 (cit. on p. 182).

[1713] X. Luo, X. Chen, X. He, L. Qing, and X. Tan. "CMAFGAN: A Cross-Modal Attention Fusion based Generative Adversarial Network for attribute word-to-face synthesis." In: *Knowledge-Based Systems* 255 (Nov. 2022), p. 109750. DOI: 10.1016/j.knosys.2022.109750. URL: https://doi.org/10.1016/j.knosys.2022.109750 (cit. on p. 182).

[1714] K. C. Dharma, C. T. Morrison, and B. Walls. "Texture Generation Using a Graph Generative Adversarial Network and Differentiable Rendering." In: *Image and Vision Computing*. Springer Nature Switzerland, 2023, pp. 388–401. DOI: 10.1007/978-3-031-25825-1_28. URL: https://doi.org/10.1007/978-3-031-25825-1_28 (cit. on pp. 182, 183).

[1715] J. Yin, Z. Zhou, S. Xu, R. Yang, and K. Liu. "A Generative Adversarial Network Fused with Dual-Attention Mechanism and Its Application in Multitarget Image Fine Segmentation." In: *Computational Intelligence and Neuroscience* 2021 (Dec. 2021). Ed. by Y. Yi, pp. 1–16. DOI: 10.1155/2021/2464648. URL: https://doi.org/10.1155/2021/2464648 (cit. on p. 182).

[1716] H. Zhang, H. Zhu, S. Yang, and W. Li. "DGattGAN: Cooperative Up-Sampling Based Dual Generator Attentional GAN on Text-to-Image Synthesis." In: *IEEE Access* 9 (2021), pp. 29584–29598. DOI: 10.1109/access.2021.3058674. URL: https://doi.org/10.1109/access.2021.3058674 (cit. on p. 182).

[1717] Y. Ma, G. Zhong, W. Liu, Y. Wang, P. Jiang, and R. Zhang. "ML-CGAN: Conditional Generative Adversarial Network with a Meta-learner Structure for High-Quality Image Generation with Few Training Data." In: *Cognitive Computation* 13.2 (Jan. 2021), pp. 418–430. DOI: 10.1007/s12559-020-09796-4. URL: https://doi.org/10.1007/s12559-020-09796-4 (cit. on p. 182).

[1718] A. Phaphuangwittayakul, F. Ying, Y. Guo, L. Zhou, and N. Chakpitak. "Few-shot image generation based on contrastive meta-learning generative adversarial network." In: *The Visual Computer* (July 2022). DOI: 10.1007/s00371-022-02566-3. URL: https://doi.org/10.1007/s00371-022-02566-3 (cit. on p. 182).

[1719] T. Nguyen, T. Le, H. Vu, and D. Phung. "Dual Discriminator Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/e60e81c4cbe5171cd654662d9887aec2-Paper.pdf (cit. on p. 182).

[1720] B. Liu, L. Wang, J. Wang, and J. Zhang. "Dual Discriminator Weighted Mixture Generative Adversarial Network for image generation." In: *Journal of Ambient Intelligence and Humanized Computing* (Feb. 2022). DOI: 10.1007/s12652-021-03667-y. URL: https://doi.org/10.1007/s12652-021-03667-y (cit. on p. 182).

[1721] G. G. Chrysos, J. Kossaifi, and S. Zafeiriou. "RoCGAN: Robust Conditional GAN." In: *International Journal of Computer Vision* 128.10-11 (July 2020), pp. 2665–2683. DOI: 10.1007/s11263-020-01348-5. URL: https://doi.org/10.1007/s11263-020-01348-5 (cit. on p. 182).

[1722] S. Mohammadjafari, M. Cevik, and A. Basar. "VARGAN: variance enforcing network enhanced GAN." In: *Applied Intelligence* 53.1 (Apr. 2022), pp. 69–95. DOI: 10.1007/s10489-022-03199-8. URL: https://doi.org/10.1007/s10489-022-03199-8 (cit. on p. 182).

[1723] X. Liang, Y. Li, X. Li, Y. Zhang, and Y. Ding. "A Dual Stream Generative Adversarial Network with Phase Awareness for Speech Enhancement." In: *Information* 14.4 (Apr. 2023), p. 221. DOI: 10.3390/info14040221. URL: https://doi.org/10.3390/info14040221 (cit. on p. 182).

[1724] J. Kossaifi, L. Tran, Y. Panagakis, and M. Pantic. "GAGAN: Geometry-Aware Generative Adversarial Networks." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00098. URL: https://doi.org/10.1109/cvpr.2018.00098 (cit. on p. 182).

[1725] L. Sun, B. Jiang, C. Yang, J. Dai, and W. Zeng. "RePGAN: image inpainting via residual partial connection and mask discriminator." In: *International Journal of Machine Learning and Cybernetics* (Apr. 2023). DOI: 10.1007/s13042-023-01827-4. URL: https://doi.org/10.1007/s13042-023-01827-4 (cit. on p. 182).

[1726] F. Zhang, X. Wang, T. Sun, and X. Xu. “SE-DCGAN: a New Method of Semantic Image Restoration.” In: *Cognitive Computation* 13.4 (May 2021), pp. 981–991. DOI: 10.1007/s12559-021-09877-y. URL: https://doi.org/10.1007/s12559-021-09877-y (cit. on p. 182).

[1727] Z. Wenjun, S. Benpeng, F. Ruiqi, P. Xihua, and C. Shanxiong. “EA-GAN: restoration of text in ancient Chinese books based on an example attention generative adversarial network.” In: *Heritage Science* 11.1 (Mar. 2023). DOI: 10.1186/s40494-023-00882-y. URL: https://doi.org/10.1186/s40494-023-00882-y (cit. on p. 182).

[1728] D. Lazcano, N. F. Franco, and W. Creixell. “HGAN: Hyperbolic Generative Adversarial Network.” In: *IEEE Access* 9 (2021), pp. 96309–96320. DOI: 10.1109/access.2021.3094723. URL: https://doi.org/10.1109/access.2021.3094723 (cit. on p. 182).

[1729] M. Armandpour, A. Sadeghian, C. Li, and M. Zhou. “Partition-Guided GANs.” In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00506. URL: https://doi.org/10.1109/cvpr46437.2021.00506 (cit. on p. 182).

[1730] J. Chen, Y. Zhang, X. Hu, and C. Y.-C. Chen. “Cascading residual–residual attention generative adversarial network for image super resolution.” In: *Soft Computing* 25.14 (Mar. 2021), pp. 9651–9662. DOI: 10.1007/s00500-021-05730-4. URL: https://doi.org/10.1007/s00500-021-05730-4 (cit. on p. 182).

[1731] M. Arjovsky, S. Chintala, and L. Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML] (cit. on p. 182).

[1732] A. Radford, L. Metz, and S. Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG] (cit. on p. 182).

[1733] A. Brock, J. Donahue, and K. Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis.” In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=B1xsqj09Fm (cit. on p. 182).

[1734] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. “StarGAN v2: Diverse Image Synthesis for Multiple Domains.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Choi_StarGAN_v2_Diverse_Image_Synthesis_for_Multiple_Domains_CVPR_2020_paper.pdf (cit. on p. 182).

[1735] J. Kim, H. Yang, and K. Min. “24-GAN: Portrait Generation with Composite Attributes.” In: *Mathematics* 10.20 (Oct. 2022), p. 3887. DOI: 10.3390/math10203887. URL: https://doi.org/10.3390/math10203887 (cit. on p. 182).

[1736] S. Yang, J.-T. Zhang, C.-W. Lin, and C.-C. Hsu. “GAN-Based Criminal Suspect Face Generator.” In: *Communications in Computer and Information Science*. Springer Nature Singapore, 2022, pp. 329–340. DOI: 10.1007/978-981-19-9582-8_29. URL: https://doi.org/10.1007/978-981-19-9582-8_29 (cit. on p. 182).

[1737] X. Huang, A. Mallya, T.-C. Wang, and M.-Y. Liu. “Multimodal Conditional Image Synthesis with Product-of-Experts GANs.” In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 91–109. DOI: 10.1007/978-3-031-19787-1_6. URL: https://doi.org/10.1007/978-3-031-19787-1_6 (cit. on p. 182).

[1738] J. Fu, S. Li, Y. Jiang, K.-Y. Lin, C. Qian, C. C. Loy, W. Wu, and Z. Liu. “StyleGAN-Human: A Data-Centric Odyssey of Human Generation.” In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 1–19. DOI: 10.1007/978-3-031-19787-1_1. URL: https://doi.org/10.1007/978-3-031-19787-1_1 (cit. on p. 182).

[1739] P. Celard, E. L. Iglesias, J. M. Sorribes-Fdez, R. Romero, A. S. Vieira, and L. Borrajo. “A survey on deep learning applied to medical images: from simple artificial neural networks to generative models.” In: *Neural Computing and Applications* 35.3 (Nov. 2022), pp. 2291–2323. DOI: 10.1007/s00521-022-07953-4. URL: https://doi.org/10.1007/s00521-022-07953-4 (cit. on p. 182).

[1740] J. Zhang, E. Sangineto, H. Tang, A. Siarohin, Z. Zhong, N. Sebe, and W. Wang. “3D-Aware Semantic-Guided Generative Model for Human Synthesis.” In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 339–356. DOI: 10.1007/978-3-031-19784-0_20. URL: https://doi.org/10.1007/978-3-031-19784-0_20 (cit. on p. 182).

[1741] C. Yang, Y. Shen, and B. Zhou. “Semantic Hierarchy Emerges in Deep Generative Representations for Scene Synthesis.” In: *International Journal of Computer Vision* 129.5 (Feb. 2021), pp. 1451–1466. DOI: 10.1007/s11263-020-01429-5. URL: https://doi.org/10.1007/s11263-020-01429-5 (cit. on p. 182).

[1742] Q. Zhou, J. Zhang, G. Han, Z. Ruan, and Y. Wei. “Enhanced self-supervised GANs with blend ratio classification.” In: *Multimedia Tools and Applications* 81.6 (Jan. 2022), pp. 7651–7667. DOI: 10.1007/s11042-022-12056-2. URL: https://doi.org/10.1007/s11042-022-12056-2 (cit. on p. 182).

[1743] D. Bang, S. Kang, and H. Shim. “Discriminator Feature-Based Inference by Recycling the Discriminator of GANs.” In: *International Journal of Computer Vision* 128.10-11 (Mar. 2020), pp. 2436–2458. DOI: 10.1007/s11263-020-01311-4. URL: https://doi.org/10.1007/s11263-020-01311-4 (cit. on p. 182).

[1744] Q. Zhou, J. Zhang, and G. Han. “Improved Generative Adversarial Network Learning via Structural Pattern Classification.” In: *Neural Processing Letters* (Mar. 2023). DOI: 10.1007/s11063-023-11221-4. URL: https://doi.org/10.1007/s11063-023-11221-4 (cit. on p. 182).

[1745] Y. Skandarani, P.-M. Jodoin, and A. Lalande. “GANs for Medical Image Synthesis: An Empirical Study.” In: *Journal of Imaging* 9.3 (Mar. 2023), p. 69. DOI: 10.3390/jimaging9030069. URL: https://doi.org/10.3390/jimaging9030069 (cit. on p. 182).

[1746] J. Chen, S. Luo, M. Xiong, T. Peng, P. Zhu, M. Jiang, and X. Qin. “HybridGAN: hybrid generative adversarial networks for MR image synthesis.” In: *Multimedia Tools and Applications* 79.37-38 (July 2020), pp. 27615–27631. DOI: 10.1007/s11042-020-09387-3. URL: https://doi.org/10.1007/s11042-020-09387-3 (cit. on p. 182).

[1747] Z. Liu, K. Niu, and Z. He. “ML-CookGAN: Multi-Label Generative Adversarial Network for Food Image Generation.” In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 19.2s (Feb. 2023), pp. 1–21. DOI: 10.1145/3554738. URL: https://doi.org/10.1145/3554738 (cit. on p. 182).

[1748] Y. Fang, X. Zhang, H. Cao, J. Nie, Z. Chen, and Z. He. “Insulator Image Dataset Generation based on Generative Adversarial Network.” In: *2023 4th International Conference on Computer Vision, Image and Deep Learning (CVIDL)*. IEEE, May 2023. DOI: 10.1109/cvidl58838.2023.10166407. URL: https://doi.org/10.1109/cvidl58838.2023.10166407 (cit. on pp. 182, 183).

[1749] Z. Zhang, Y. Xie, and L. Yang. “Photographic Text-to-Image Synthesis with a Hierarchically-Nested Adversarial Network.” In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00649. URL: https://doi.org/10.1109/cvpr.2018.00649 (cit. on p. 182).

[1750] Y. X. Tan, C. P. Lee, M. Neo, K. M. Lim, and J. Y. Lim. “Enhanced Text-to-Image Synthesis With Self-Supervision.” In: *IEEE Access* 11 (2023), pp. 39508–39519. DOI: 10.1109/access.2023.3268869. URL: https://doi.org/10.1109/access.2023.3268869 (cit. on p. 182).

[1751] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. “Generative Adversarial Text to Image Synthesis.” In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1060–1069. URL: https://proceedings.mlr.press/v48/reed16.html (cit. on p. 182).

[1752] A. Sauer, T. Karras, S. Laine, A. Geiger, and T. Aila. *StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis*. 2023. DOI: 10.48550/ARXIV.2301.09515. URL: https://arxiv.org/abs/2301.09515 (cit. on p. 182).

[1753] A. Kushwaha, C. P, and K. P. Singh. “Text to Face generation using Wasserstein stackGAN.” In: *2022 IEEE 9th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. IEEE, Dec. 2022. DOI: 10.1109/upcon56432.2022.9986391. URL: https://doi.org/10.1109/upcon56432.2022.9986391 (cit. on p. 182).

[1754] E. Raparla, V. Raavipaati, S. N. G, S. S. T. Md, and K. K. S. “Different Techniques of Facial Image Generation from Textual Input : A Survey.” In: *2022 2nd International Conference on Intelligent Technologies (CONIT)*. IEEE, June 2022. DOI: 10.1109/conit55038.2022.9848228. URL: https://doi.org/10.1109/conit55038.2022.9848228 (cit. on p. 182).

[1755] Y. Zhou, R. Zhang, C. Chen, C. Li, C. Tensmeyer, T. Yu, J. Gu, J. Xu, and T. Sun. “Towards Language-Free Training for Text-to-Image Generation.” In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01738. URL: https://doi.org/10.1109/cvpr52688.2022.01738 (cit. on p. 182).

[1756] J. Sun, Q. Deng, Q. Li, M. Sun, M. Ren, and Z. Sun. “AnyFace: Free-style Text-to-Face Synthesis and Manipulation.” In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01813. URL: https://doi.org/10.1109/cvpr52688.2022.01813 (cit. on p. 182).

[1757] M. Berrahal and M. Azizi. “Optimal text-to-image synthesis model for generating portrait images using generative adversarial network techniques.” In: *Indonesian Journal of Electrical Engineering and Computer Science* 25.2 (Feb. 2022), p. 972. DOI: 10.11591/ijeecs.v25.i2.pp972-979. URL: https://doi.org/10.11591/ijeecs.v25.i2.pp972-979 (cit. on p. 182).

[1758] Y. Zhou. “Generative Adversarial Network for Text-to-Face Synthesis and Manipulation.” In: *Proceedings of the 29th ACM International Conference on Multimedia*. ACM, Oct. 2021. DOI: 10.1145/3474085.3481026. URL: https://doi.org/10.1145/3474085.3481026 (cit. on p. 182).

[1759] H. Ku and M. Lee. “TextControlGAN: Text-to-Image Synthesis with Controllable Generative Adversarial Networks.” In: *Applied Sciences* 13.8 (Apr. 2023), p. 5098. DOI: 10.3390/app13085098. URL: https://doi.org/10.3390/app13085098 (cit. on p. 182).

[1760] S. Luo. “A Survey on Multimodal Deep Learning for Image Synthesis.” In: *2021 the 5th International Conference on Innovation in Artificial Intelligence*. ACM, Mar. 2021. DOI: 10.1145/3461353.3461388. URL: https://doi.org/10.1145/3461353.3461388 (cit. on pp. 182, 184).

[1761] Z. Deng, X. He, and Y. Peng. “LFR-GAN: Local Feature Refinement based Generative Adversarial Network for Text-to-Image Generation.” In: *ACM Transactions on Multimedia Computing, Communications, and Applications* (Mar. 2023). DOI: 10.1145/3589002. URL: https://doi.org/10.1145/3589002 (cit. on p. 182).

[1762] S. Pande, S. Chouhan, R. Sonavane, R. Walambe, G. Ghinea, and K. Kotecha. “Development and deployment of a generative model-based framework for text to photorealistic image generation.” In: *Neurocomputing* 463 (Nov. 2021), pp. 1–16. DOI: 10.1016/j.neucom.2021.08.055. URL: https://doi.org/10.1016/j.neucom.2021.08.055 (cit. on p. 182).

[1763] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon. *Text-to-image Diffusion Models in Generative AI: A Survey*. 2023. DOI: 10.48550/ARXIV.2303.07909. URL: https://arxiv.org/abs/2303.07909 (cit. on pp. 182, 183).

[1764] L. Gao, D. Chen, J. Song, X. Xu, D. Zhang, and H. T. Shen. “Perceptual Pyramid Adversarial Networks for Text-to-Image Synthesis.” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 8312–8319. DOI: 10.1609/aaai.v33i01.33018312. URL: https://doi.org/10.1609/aaai.v33i01.33018312 (cit. on p. 182).

[1765] Z. Zhang, C. Fu, W. Weng, and J. Zhou. “Text-Guided Customizable Image Synthesis and Manipulation.” In: *Applied Sciences* 12.20 (Oct. 2022), p. 10645. DOI: 10.3390/app122010645. URL: https://doi.org/10.3390/app122010645 (cit. on p. 182).

[1766] V. Sushko, E. Schönfeld, D. Zhang, J. Gall, B. Schiele, and A. Khoreva. “OASIS: Only Adversarial Supervision for Semantic Image Synthesis.” In: *International Journal of Computer Vision* 130.12 (Sept. 2022), pp. 2903–2923. DOI: 10.1007/s11263-022-01673-x. URL: https://doi.org/10.1007/s11263-022-01673-x (cit. on p. 182).

[1767] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski. “StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery.” In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.00209. URL: https://doi.org/10.1109/iccv48922.2021.00209 (cit. on p. 182).

[1768] W. Xia, Y. Yang, J.-H. Xue, and B. Wu. “TediGAN: Text-Guided Diverse Face Image Generation and Manipulation.” In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00229. URL: https://doi.org/10.1109/cvpr46437.2021.00229 (cit. on p. 182).

[1769] C.-K. T. Chao and Y. Gingold. *Text-guided Image-and-Shape Editing and Generation: A Short Survey*. 2023. DOI: 10.48550/ARXIV.2304.09244. URL: https://arxiv.org/abs/2304.09244 (cit. on pp. 182–184).

[1770] J. Ko, K. Cho, D. Choi, K. Ryoo, and S. Kim. “3D GAN Inversion with Pose Optimization.” In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00298. URL: https://doi.org/10.1109/wacv56688.2023.00298 (cit. on p. 182).

[1771] S. H. Lee, G. Oh, W. Byeon, C. Kim, W. J. Ryoo, S. H. Yoon, H. Cho, J. Bae, J. Kim, and S. Kim. *Sound-Guided Semantic Video Generation*. 2022. DOI: 10.48550/ARXIV.2204.09273. URL: https://arxiv.org/abs/2204.09273 (cit. on p. 182).

[1772] Y. Alaluf, O. Tov, R. Mokady, R. Gal, and A. Bermano. “HyperStyle: StyleGAN Inversion with HyperNetworks for Real Image Editing.” In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01796. URL: https://doi.org/10.1109/cvpr52688.2022.01796 (cit. on p. 182).

[1773] Y. Liu, Q. Li, Q. Deng, and Z. Sun. “Towards Spatially Disentangled Manipulation of Face Images With Pre-Trained StyleGANs.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 33.4 (Apr. 2023), pp. 1725–1739. DOI: 10.1109/tcsvt.2022.3213662. URL: https://doi.org/10.1109/tcsvt.2022.3213662 (cit. on p. 182).

[1774] Q. Bai, W. Xia, F. Yin, and Y. Yang. “Identity-Guided Face Generation with Multi-Modal Contour Conditions.” In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, Oct. 2022. DOI: 10.1109/icip46576.2022.9897459. URL: https://doi.org/10.1109/icip46576.2022.9897459 (cit. on p. 182).

[1775] Z. Guo, M. Shao, and S. Li. “Image-to-image translation using an offset-based multi-scale codes GAN encoder.” In: *The Visual Computer* (Mar. 2023). DOI: 10.1007/s00371-023-02810-4. URL: https://doi.org/10.1007/s00371-023-02810-4 (cit. on p. 182).

[1776] J. Chen, S. Chen, L. Wee, A. Dekker, and I. Bermejo. “Deep learning based unpaired image-to-image translation applications for medical physics: a systematic review.” In: *Physics in Medicine & Biology* 68.5 (Feb. 2023), 05TR01. DOI: 10.1088/1361-6560/acba74. URL: https://doi.org/10.1088/1361-6560/acba74 (cit. on p. 182).

[1777] M. Liu, Y. Wei, X. Wu, W. Zuo, and L. Zhang. “Survey on leveraging pre-trained generative adversarial networks for image editing and restoration.” In: *Science China Information Sciences* 66.5 (Apr. 2023). DOI: 10.1007/s11432-022-3679-0. URL: https://doi.org/10.1007/s11432-022-3679-0 (cit. on p. 182).

[1778] Y. Alaluf, O. Patashnik, Z. Wu, A. Zamir, E. Shechtman, D. Lischinski, and D. Cohen-Or. “Third Time’s the Charm? Image and Video Editing with StyleGAN3.” In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2023, pp. 204–220. DOI: 10.1007/978-3-031-25063-7_13. URL: https://doi.org/10.1007/978-3-031-25063-7_13 (cit. on p. 182).

[1779] Y. Liu, R. Gal, A. H. Bermano, B. Chen, and D. Cohen-Or. “Self-Conditioned GANs for Image Editing.” In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. ACM, Aug. 2022. DOI: 10.1145/3528233.3530698. URL: https://doi.org/10.1145/3528233.3530698 (cit. on p. 182).

[1780] Y. Zhang, Q. Wang, and B. Hu. “MinimalGAN: diverse medical image synthesis for data augmentation using minimal training data.” In: *Applied Intelligence* 53.4 (June 2022), pp. 3899–3916. DOI: 10.1007/s10489-022-03609-x. URL: https://doi.org/10.1007/s10489-022-03609-x (cit. on p. 182).

[1781] C.-Y. Chung and S.-H. Huang. “Interactively transforming chinese ink paintings into realistic images using a border enhance generative adversarial network.” In: *Multimedia Tools and Applications* 82.8 (Aug. 2022), pp. 11663–11696. DOI: 10.1007/s11042-022-13684-4. URL: https://doi.org/10.1007/s11042-022-13684-4 (cit. on p. 182).

[1782] A. Mehta, H. Sinha, P. Narang, and M. Mandal. “HIDeGan: A Hyperspectral-guided Image Dehazing GAN.” In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2020. DOI: 10.1109/cvprw50498.2020.00114. URL: https://doi.org/10.1109/cvprw50498.2020.00114 (cit. on p. 182).

[1783] K. Regmi and A. Borji. “Cross-view image synthesis using geometry-guided conditional GANs.” In: *Computer Vision and Image Understanding* 187 (Oct. 2019), p. 102788. DOI: 10.1016/j.cviu.2019.07.008. URL: https://doi.org/10.1016/j.cviu.2019.07.008 (cit. on p. 182).

[1784] Y. Mao, T. Zhang, B. Fu, and D. N. H. Thanh. “A Self-Attention Based Wasserstein Generative Adversarial Networks for Single Image Inpainting.” In: *Pattern Recognition and Image Analysis* 32.3 (Sept. 2022), pp. 591–599. DOI: 10.1134/s1054661822030245. URL: https://doi.org/10.1134/s1054661822030245 (cit. on p. 182).

[1785] G. Chen, P. Kang, X. Wu, Z. Yang, and W. Liu. “Adaptive Visual Field Multi-scale Generative Adversarial Networks Image Inpainting Base on Coordinate-Attention.” In: *Neural Processing Letters* (Mar. 2023). DOI: 10.1007/s11063-023-11233-0. URL: https://doi.org/10.1007/s11063-023-11233-0 (cit. on p. 182).

[1786] Y. Chen and H. Hu. “An Improved Method for Semantic Image Inpainting with GANs: Progressive Inpainting.” In: *Neural Processing Letters* 49.3 (June 2018), pp. 1355–1367. DOI: 10.1007/s11063-018-9877-6. URL: https://doi.org/10.1007/s11063-018-9877-6 (cit. on p. 182).

[1787] G. Chen, G. Zhang, Z. Yang, and W. Liu. “Multi-scale patch-GAN with edge detection for image inpainting.” In: *Applied Intelligence* 53.4 (June 2022), pp. 3917–3932. DOI: 10.1007/s10489-022-03577-2. URL: https://doi.org/10.1007/s10489-022-03577-2 (cit. on p. 182).

[1788] Y. Chen, H. Zhang, L. Liu, X. Chen, Q. Zhang, K. Yang, R. Xia, and J. Xie. “Research on image Inpainting algorithm of improved GAN based on two-discriminations networks.” In: *Applied Intelligence* 51.6 (Nov. 2020), pp. 3460–3474. DOI: 10.1007/s10489-020-01971-2. URL: https://doi.org/10.1007/s10489-020-01971-2 (cit. on p. 182).

[1789] Y. Chen, L. Liu, J. Tao, R. Xia, Q. Zhang, K. Yang, J. Xiong, and X. Chen. “The improved image inpainting algorithm via encoder and similarity constraint.” In: *The Visual Computer* 37.7 (July 2020), pp. 1691–1705. DOI: 10.1007/s00371-020-01932-3. URL: https://doi.org/10.1007/s00371-020-01932-3 (cit. on p. 182).

[1790] C. Lv, Z. Li, Y. Shen, J. Li, and J. Zheng. “SeparaFill: Two generators connected mural image restoration based on generative adversarial network with skip connect.” In: *Heritage Science* 10.1 (Aug. 2022). DOI: 10.1186/s40494-022-00771-w. URL: https://doi.org/10.1186/s40494-022-00771-w (cit. on p. 182).

[1791] S. Ma, J. Cao, Z. Li, Z. Chen, and X. Hu. “An improved algorithm for superresolution reconstruction of ancient murals with a generative adversarial network based on asymmetric pyramid modules.” In: *Heritage Science* 10.1 (May 2022). DOI: 10.1186/s40494-022-00700-x. URL: https://doi.org/10.1186/s40494-022-00700-x (cit. on p. 182).

[1792] J. Cao, Y. Jia, M. Yan, and X. Tian. "Superresolution reconstruction method for ancient murals based on the stable enhanced generative adversarial network." In: *EURASIP Journal on Image and Video Processing* 2021.1 (July 2021). DOI: 10.1186/s13640-021-00569-z. URL: https://doi.org/10.1186/s13640-021-00569-z (cit. on p. 182).

[1793] J. Cao, Z. Zhang, A. Zhao, H. Cui, and Q. Zhang. "Ancient mural restoration based on a modified generative adversarial network." In: *Heritage Science* 8.1 (Jan. 2020). DOI: 10.1186/s40494-020-0355-x. URL: https://doi.org/10.1186/s40494-020-0355-x (cit. on p. 182).

[1794] J. Li, H. Wang, Z. Deng, M. Pan, and H. Chen. "Restoration of non-structural damaged murals in Shenzhen Bao'an based on a generator–discriminator network." In: *Heritage Science* 9.1 (Jan. 2021). DOI: 10.1186/s40494-020-00478-w. URL: https://doi.org/10.1186/s40494-020-00478-w (cit. on p. 182).

[1795] R. Liu, R. Yang, S. Li, Y. Shi, and X. Jin. "Painting completion with generative translation models." In: *Multimedia Tools and Applications* 79.21-22 (Oct. 2018), pp. 14375–14388. DOI: 10.1007/s11042-018-6761-3. URL: https://doi.org/10.1007/s11042-018-6761-3 (cit. on p. 182).

[1796] P. Kumar and V. Gupta. "Restoration of damaged artworks based on a generative adversarial network." In: *Multimedia Tools and Applications* (Apr. 2023). DOI: 10.1007/s11042-023-15222-2. URL: https://doi.org/10.1007/s11042-023-15222-2 (cit. on p. 182).

[1797] K. Falahkheirkhah, T. Guo, M. Hwang, P. Tamboli, C. G. Wood, J. A. Karam, K. Sircar, and R. Bhargava. "A generative adversarial approach to facilitate archival-quality histopathologic diagnoses from frozen tissue sections." In: *Laboratory Investigation* 102.5 (May 2022), pp. 554–559. DOI: 10.1038/s41374-021-00718-y. URL: https://doi.org/10.1038/s41374-021-00718-y (cit. on p. 182).

[1798] D. Lee, W.-J. Moon, and J. C. Ye. "Assessing the importance of magnetic resonance contrasts using collaborative generative adversarial networks." In: *Nature Machine Intelligence* 2.1 (Jan. 2020), pp. 34–42. DOI: 10.1038/s42256-019-0137-x. URL: https://doi.org/10.1038/s42256-019-0137-x (cit. on p. 182).

[1799] J. Denck, J. Guehring, A. Maier, and E. Rothgang. "MR-contrast-aware image-to-image translations with generative adversarial networks." In: *International Journal of Computer Assisted Radiology and Surgery* 16.12 (June 2021), pp. 2069–2078. DOI: 10.1007/s11548-021-02433-x. URL: https://doi.org/10.1007/s11548-021-02433-x (cit. on p. 182).

[1800] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers. "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks." In: *Scientific Reports* 9.1 (Nov. 2019). DOI: 10.1038/s41598-019-52737-x. URL: https://doi.org/10.1038/s41598-019-52737-x (cit. on p. 182).

[1801] B. Sun, S. Jia, X. Jiang, and F. Jia. "Double U-Net CycleGAN for 3D MR to CT image synthesis." In: *International Journal of Computer Assisted Radiology and Surgery* 18.1 (Aug. 2022), pp. 149–156. DOI: 10.1007/s11548-022-02732-x. URL: https://doi.org/10.1007/s11548-022-02732-x (cit. on p. 182).

[1802] P. Qian, K. Xu, T. Wang, Q. Zheng, H. Yang, A. Baydoun, J. Zhu, B. Traughber, and R. F. Muzic. "Estimating CT from MR Abdominal Images Using Novel Generative Adversarial Networks." In: *Journal of Grid Computing* 18.2 (Mar. 2020), pp. 211–226. DOI: 10.1007/s10723-020-09513-3. URL: https://doi.org/10.1007/s10723-020-09513-3 (cit. on p. 182).

[1803] S. Yao, J. Tan, Y. Chen, and Y. Gu. "A weighted feature transfer gan for medical image synthesis." In: *Machine Vision and Applications* 32.1 (Nov. 2020). DOI: 10.1007/s00138-020-01152-8. URL: https://doi.org/10.1007/s00138-020-01152-8 (cit. on p. 182).

[1804] S. Pang, A. Du, M. A. Orgun, Z. Yu, Y. Wang, Y. Wang, and G. Liu. "CTumorGAN: a unified framework for automatic computed tomography tumor segmentation." In: *European Journal of Nuclear Medicine and Molecular Imaging* 47.10 (Mar. 2020), pp. 2248–2268. DOI: 10.1007/s00259-020-04781-3. URL: https://doi.org/10.1007/s00259-020-04781-3 (cit. on p. 182).

[1805] L. Chen, H. Song, C. Wang, Y. Cui, J. Yang, X. Hu, and L. Zhang. "Liver tumor segmentation in CT volumes using an adversarial densely connected network." In: *BMC Bioinformatics* 20.S16 (Dec. 2019). DOI: 10.1186/s12859-019-3069-x. URL: https://doi.org/10.1186/s12859-019-3069-x (cit. on p. 182).

[1806] H. M. Azni, M. Afsharchi, and A. Allahverdi. "Improving brain tumor segmentation performance using CycleGAN based feature extraction." In: *Multimedia Tools and Applications* 82.12 (Nov. 2022), pp. 18039–18058. DOI: 10.1007/s11042-022-14174-3. URL: https://doi.org/10.1007/s11042-022-14174-3 (cit. on p. 182).

[1807] P. Yang, X. Peng, J. Xiao, X. Wu, J. Zhou, and Y. Wang. "Automatic Head-and-Neck Tumor Segmentation in MRI via an End-to-End Adversarial Network." In: *Neural Processing Letters* (Apr. 2023). DOI: 10.1007/s11063-023-11232-1. URL: https://doi.org/10.1007/s11063-023-11232-1 (cit. on p. 182).

[1808] S. G. Domadia, F. N. Thakkar, and M. A. Ardeshana. “Recent advancement in learning methodology for segmenting brain tumor from magnetic resonance imaging -a review.” In: *Multimedia Tools and Applications* (Feb. 2023). DOI: 10.1007/s11042-023-14857-5. URL: https://doi.org/10.1007/s11042-023-14857-5 (cit. on p. 182).

[1809] Q. Yang, N. Li, Z. Zhao, X. Fan, E. I.-C. Chang, and Y. Xu. “MRI Cross-Modality Image-to-Image Translation.” In: *Scientific Reports* 10.1 (Feb. 2020). DOI: 10.1038/s41598-020-60520-6. URL: https://doi.org/10.1038/s41598-020-60520-6 (cit. on p. 182).

[1810] H. A. Amirkolaee and H. A. Amirkolaee. “Medical image translation using an edge-guided generative adversarial network with global-to-local feature fusion.” In: *The Journal of Biomedical Research* 36.6 (2022), p. 409. DOI: 10.7555/jbr.36.20220037. URL: https://doi.org/10.7555/jbr.36.20220037 (cit. on p. 182).

[1811] X. Zhang, C. Feng, A. Wang, L. Yang, and Y. Hao. “CT super-resolution using multiple dense residual block based GAN.” In: *Signal, Image and Video Processing* 15.4 (Oct. 2020), pp. 725–733. DOI: 10.1007/s11760-020-01790-5. URL: https://doi.org/10.1007/s11760-020-01790-5 (cit. on p. 182).

[1812] Z. Wang, J. Li, and M. Enoh. “Removing ring artifacts in CBCT images via generative adversarial networks with unidirectional relative total variation loss.” In: *Neural Computing and Applications* 31.9 (Jan. 2019), pp. 5147–5158. DOI: 10.1007/s00521-018-04007-6. URL: https://doi.org/10.1007/s00521-018-04007-6 (cit. on p. 182).

[1813] P. Li, Z. Li, X. Pang, H. Wang, W. Lin, and W. Wu. “Multi-scale residual denoising GAN model for producing super-resolution CTA images.” In: *Journal of Ambient Intelligence and Humanized Computing* 13.3 (Mar. 2021), pp. 1515–1524. DOI: 10.1007/s12652-021-03009-y. URL: https://doi.org/10.1007/s12652-021-03009-y (cit. on p. 182).

[1814] M.-Y. Liu, T. Breuel, and J. Kautz. “Unsupervised Image-to-Image Translation Networks.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/dc6a6489640ca02b0d42dabeb8e46bb7-Paper.pdf (cit. on p. 182).

[1815] H. Hoyez, C. Schockaert, J. Rambach, B. Mirbach, and D. Stricker. “Unsupervised Image-to-Image Translation: A Review.” In: *Sensors* 22.21 (Nov. 2022), p. 8540. DOI: 10.3390/s22218540. URL: https://doi.org/10.3390/s22218540 (cit. on p. 182).

[1816] J. Sun, B. Bhattarai, Z. Chen, and T.-K. Kim. *SeCGAN: Parallel Conditional Generative Adversarial Networks for Face Editing via Semantic Consistency*. 2021. DOI: 10.48550/ARXIV.2111.09298. URL: https://arxiv.org/abs/2111.09298 (cit. on p. 182).

[1817] C. Ledig et al. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.19. URL: https://doi.org/10.1109/cvpr.2017.19 (cit. on p. 182).

[1818] M. Zareapoor, H. Zhou, and J. Yang. “Perceptual image quality using dual generative adversarial network.” In: *Neural Computing and Applications* 32.18 (May 2019), pp. 14521–14531. DOI: 10.1007/s00521-019-04239-0. URL: https://doi.org/10.1007/s00521-019-04239-0 (cit. on p. 182).

[1819] H. Li, Z. Xuan, J. Zhou, X. Hu, and B. Yang. “Fast and accurate super-resolution of MR images based on lightweight generative adversarial network.” In: *Multimedia Tools and Applications* 82.2 (June 2022), pp. 2465–2487. DOI: 10.1007/s11042-022-13326-9. URL: https://doi.org/10.1007/s11042-022-13326-9 (cit. on p. 182).

[1820] Z. Chen, J. Wang, C. Jia, and X. Ye. “Pathological image super-resolution using mix-attention generative adversarial network.” In: *International Journal of Machine Learning and Cybernetics* (Mar. 2023). DOI: 10.1007/s13042-023-01806-9. URL: https://doi.org/10.1007/s13042-023-01806-9 (cit. on p. 182).

[1821] X. Chen and H. Zhao. “A Novel Fast Reconstruction Method for Single Image Super Resolution Task.” In: *Neural Processing Letters* (Mar. 2023). DOI: 10.1007/s11063-023-11235-y. URL: https://doi.org/10.1007/s11063-023-11235-y (cit. on p. 182).

[1822] F. Nan, Q. Zeng, Y. Xing, and Y. Qian. “Single Image Super-Resolution Reconstruction based on the ResNeXt Network.” In: *Multimedia Tools and Applications* 79.45-46 (June 2020), pp. 34459–34470. DOI: 10.1007/s11042-020-09053-8. URL: https://doi.org/10.1007/s11042-020-09053-8 (cit. on p. 182).

[1823] V. Chudasama and K. Upla. “RSRGAN: computationally efficient real-world single image super-resolution using generative adversarial network.” In: *Machine Vision and Applications* 32.1 (Oct. 2020). DOI: 10.1007/s00138-020-01135-9. URL: https://doi.org/10.1007/s00138-020-01135-9 (cit. on p. 182).

[1824] J. Daihong, Z. Sai, D. Lei, and D. Yueming. "Multi-scale generative adversarial network for image super-resolution." In: *Soft Computing* 26.8 (Feb. 2022), pp. 3631–3641. DOI: 10.1007/s00500-022-06822-5. URL: https://doi.org/10.1007/s00500-022-06822-5 (cit. on p. 182).

[1825] B. Das and S. D. Roy. "Edge-Aware Image Super-Resolution Using a Generative Adversarial Network." In: *SN Computer Science* 4.2 (Jan. 2023). DOI: 10.1007/s42979-022-01561-8. URL: https://doi.org/10.1007/s42979-022-01561-8 (cit. on p. 182).

[1826] Y. Choi and H. Park. "Improving ESRGAN with an additional image quality loss." In: *Multimedia Tools and Applications* 82.2 (July 2022), pp. 3123–3137. DOI: 10.1007/s11042-022-13452-4. URL: https://doi.org/10.1007/s11042-022-13452-4 (cit. on p. 182).

[1827] J. Huang. "Image super-resolution reconstruction based on generative adversarial network model with double discriminators." In: *Multimedia Tools and Applications* 79.39-40 (Aug. 2020), pp. 29639–29662. DOI: 10.1007/s11042-020-09524-y. URL: https://doi.org/10.1007/s11042-020-09524-y (cit. on p. 182).

[1828] J. Qiao, H. Song, K. Zhang, and X. Zhang. "Conditional generative adversarial network with densely-connected residual learning for single image super-resolution." In: *Multimedia Tools and Applications* 80.3 (Sept. 2020), pp. 4383–4397. DOI: 10.1007/s11042-020-09817-2. URL: https://doi.org/10.1007/s11042-020-09817-2 (cit. on p. 182).

[1829] T. Ma and W. Tian. "Back-projection-based progressive growing generative adversarial network for single image super-resolution." In: *The Visual Computer* 37.5 (Apr. 2020), pp. 925–938. DOI: 10.1007/s00371-020-01843-3. URL: https://doi.org/10.1007/s00371-020-01843-3 (cit. on p. 182).

[1830] M. Wang, Z. Chen, Q. M. J. Wu, and M. Jian. "Improved face super-resolution generative adversarial networks." In: *Machine Vision and Applications* 31.4 (Apr. 2020). DOI: 10.1007/s00138-020-01073-6. URL: https://doi.org/10.1007/s00138-020-01073-6 (cit. on p. 182).

[1831] L. Zhang, W. Zhang, G. Lu, P. Yang, and Z. Rao. "Feature-level interpolation-based GAN for image super-resolution." In: *Personal and Ubiquitous Computing* 26.4 (Jan. 2021), pp. 995–1010. DOI: 10.1007/s00779-020-01488-y. URL: https://doi.org/10.1007/s00779-020-01488-y (cit. on p. 182).

[1832] S. Viriyavisuthisakul, N. Kaothanthong, P. Sanguansat, M. L. Nguyen, and C. Haruechaiyasak. "Parametric regularization loss in super-resolution reconstruction." In: *Machine Vision and Applications* 33.5 (July 2022). DOI: 10.1007/s00138-022-01315-9. URL: https://doi.org/10.1007/s00138-022-01315-9 (cit. on p. 182).

[1833] Y. Gu et al. "MedSRGAN: medical images super-resolution using generative adversarial networks." In: *Multimedia Tools and Applications* 79.29-30 (May 2020), pp. 21815–21840. DOI: 10.1007/s11042-020-08980-w. URL: https://doi.org/10.1007/s11042-020-08980-w (cit. on p. 182).

[1834] Z. Yin, K. Xia, S. Wang, Z. He, J. Zhang, and B. Zu. "Unpaired low-dose CT denoising via an improved cycle-consistent adversarial network with attention ensemble." In: *The Visual Computer* (Aug. 2022). DOI: 10.1007/s00371-022-02599-8. URL: https://doi.org/10.1007/s00371-022-02599-8 (cit. on p. 182).

[1835] W. Du, H. Chen, H. Yang, and Y. Zhang. "Disentangled generative adversarial network for low-dose CT." In: *EURASIP Journal on Advances in Signal Processing* 2021.1 (July 2021). DOI: 10.1186/s13634-021-00749-z. URL: https://doi.org/10.1186/s13634-021-00749-z (cit. on p. 182).

[1836] R. Li, C. Wang, J. Wang, G. Liu, H.-Y. Zhang, B. Zeng, and S. Liu. "UPHDR-GAN: Generative Adversarial Network for High Dynamic Range Imaging With Unpaired Data." In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.11 (Nov. 2022), pp. 7532–7546. DOI: 10.1109/tcsvt.2022.3190057. URL: https://doi.org/10.1109/tcsvt.2022.3190057 (cit. on p. 182).

[1837] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas. "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00854. URL: https://doi.org/10.1109/cvpr.2018.00854 (cit. on p. 182).

[1838] W.-Z. Shao, Y.-Y. Liu, L.-Y. Ye, L.-Q. Wang, Q. Ge, B.-K. Bao, and H.-B. Li. "DeblurGAN+: Revisiting blind motion deblurring using conditional adversarial networks." In: *Signal Processing* 168 (Mar. 2020), p. 107338. DOI: 10.1016/j.sigpro.2019.107338. URL: https://doi.org/10.1016/j.sigpro.2019.107338 (cit. on p. 182).

[1839] S. Sharif, R. A. Naqvi, F. Ali, and M. Biswas. "DarkDeblur: Learning single-shot image deblurring in low-light condition." In: *Expert Systems with Applications* 222 (July 2023), p. 119739. DOI: 10.1016/j.eswa.2023.119739. URL: https://doi.org/10.1016/j.eswa.2023.119739 (cit. on p. 182).

[1840] W. Xiao, Z. Tang, J. Luo, and J. Liu. "FS-DeblurGAN: a spatiotemporal deblurring method for zinc froth flotation." In: *The European Physical Journal Special Topics* 231.10 (Feb. 2022), pp. 1983–1993. DOI: 10.1140/epjs/s11734-022-00459-z. URL: https://doi.org/10.1140/epjs/s11734-022-00459-z (cit. on p. 182).

[1841] L. Song, Q. Wang, H. Li, J. Fan, and B. Hu. "Spatio-Temporal Learning for Video Deblurring based on Two-Stream Generative Adversarial Network." In: *Neural Processing Letters* 53.4 (Apr. 2021), pp. 2701–2714. DOI: 10.1007/s11063-021-10520-y. URL: https://doi.org/10.1007/s11063-021-10520-y (cit. on p. 182).

[1842] L. Zhou, W. Min, D. Lin, Q. Han, and R. Liu. "Detecting Motion Blurred Vehicle Logo in IoV Using Filter-DeblurGAN and VL-YOLO." In: *IEEE Transactions on Vehicular Technology* 69.4 (Apr. 2020), pp. 3604–3614. DOI: 10.1109/tvt.2020.2969427. URL: https://doi.org/10.1109/tvt.2020.2969427 (cit. on p. 182).

[1843] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz. "PassGAN: A Deep Learning Approach for Password Guessing." In: *Applied Cryptography and Network Security*. Springer International Publishing, 2019, pp. 217–237. DOI: 10.1007/978-3-030-21568-2_11. URL: https://doi.org/10.1007/978-3-030-21568-2_11 (cit. on p. 182).

[1844] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz. *PassGAN: A Deep Learning Approach for Password Guessing*. 2017. DOI: 10.48550/ARXIV.1709.00440. URL: https://arxiv.org/abs/1709.00440 (cit. on p. 183).

[1845] H. Liu, H. Zhao, J. Wang, S. Yuan, and W. Feng. "LSTM-GAN-AE: A Promising Approach for Fault Diagnosis in Machine Health Monitoring." In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–13. DOI: 10.1109/tim.2021.3135328. URL: https://doi.org/10.1109/tim.2021.3135328 (cit. on p. 183).

[1846] H. Zhang, R. Wang, R. Pan, and H. Pan. "Imbalanced Fault Diagnosis of Rolling Bearing Using Enhanced Generative Adversarial Networks." In: *IEEE Access* 8 (2020), pp. 185950–185963. DOI: 10.1109/access.2020.3030058. URL: https://doi.org/10.1109/access.2020.3030058 (cit. on p. 183).

[1847] P. Samangouei, M. Kabkab, and R. Chellappa. "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models." In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=BkJ3ibb0- (cit. on p. 183).

[1848] K. Nakashima, Y. Iwashita, and R. Kurazume. "Generative Range Imaging for Learning Scene Priors of 3D LiDAR Data." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00131. URL: https://doi.org/10.1109/wacv56688.2023.00131 (cit. on p. 183).

[1849] Z. Canfes, M. F. Atasoy, A. Dirik, and P. Yanardag. "Text and Image Guided 3D Avatar Generation and Manipulation." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00440. URL: https://doi.org/10.1109/wacv56688.2023.00440 (cit. on p. 183).

[1850] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler. "GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=GAUwreODU5L (cit. on p. 183).

[1851] D. Pavllo, J. Kohler, T. Hofmann, and A. Lucchi. "Learning Generative Models of Textured 3D Meshes from Real-World Images." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01362. URL: https://doi.org/10.1109/iccv48922.2021.01362 (cit. on p. 183).

[1852] D. Pavllo, G. Spinks, T. Hofmann, M.-F. Moens, and A. Lucchi. "Convolutional Generation of Textured 3D Meshes." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 870–882. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/098d86c982354a96556bd861823ebfbd-Paper.pdf (cit. on p. 183).

[1853] R. Wu and C. Zheng. "Learning to Generate 3D Shapes from a Single Example." In: *ACM Transactions on Graphics* 41.6 (Nov. 2022), pp. 1–19. DOI: 10.1145/3550454.3555480. URL: https://doi.org/10.1145/3550454.3555480 (cit. on p. 183).

[1854] Z. Shi, S. Peng, Y. Xu, Y. Liao, and Y. Shen. *Deep Generative Models on 3D Representations: A Survey*. 2022. DOI: 10.48550/ARXIV.2210.15663. URL: https://arxiv.org/abs/2210.15663 (cit. on p. 183).

[1855] W. Kang, L. Lin, S. Sun, and S. Wu. "Three-round learning strategy based on 3D deep convolutional GANs for Alzheimer's disease staging." In: *Scientific Reports* 13.1 (Apr. 2023). DOI: 10.1038/s41598-023-33055-9. URL: https://doi.org/10.1038/s41598-023-33055-9 (cit. on p. 183).

[1856] C. Ren and Y. Xu. “A Fully Data-Driven Method Based on Generative Adversarial Networks for Power System Dynamic Security Assessment With Missing Data.” In: *IEEE Transactions on Power Systems* 34.6 (Nov. 2019), pp. 5044–5052. DOI: 10.1109/tpwrs.2019.2922671. URL: https://doi.org/10.1109/tpwrs.2019.2922671 (cit. on p. 183).

[1857] Y. Raghuvamsi and K. Teeparthi. “Distribution System State Estimation with Convolutional Generative Adversarial Imputation Networks for Missing Measurement Data.” In: *Arabian Journal for Science and Engineering* (Nov. 2023). DOI: 10.1007/s13369-023-08393-5. URL: https://doi.org/10.1007/s13369-023-08393-5 (cit. on p. 183).

[1858] R. Li, S. Liu, G. Wang, G. Liu, and B. Zeng. “JigsawGAN: Auxiliary Learning for Solving Jigsaw Puzzles With Generative Adversarial Networks.” In: *IEEE Transactions on Image Processing* 31 (2022), pp. 513–524. DOI: 10.1109/tip.2021.3120052. URL: https://doi.org/10.1109/tip.2021.3120052 (cit. on p. 183).

[1859] J. Fei, Z. Xia, B. Tondi, and M. Barni. *Supervised GAN Watermarking for Intellectual Property Protection*. 2022. DOI: 10.48550/ARXIV.2209.03466. URL: https://arxiv.org/abs/2209.03466 (cit. on p. 183).

[1860] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2017, pp. 146–157. DOI: 10.1007/978-3-319-59050-9_12. URL: https://doi.org/10.1007/978-3-319-59050-9_12 (cit. on p. 183).

[1861] N. Bhatt, D. R. Prados, N. Hodzic, C. Karanassios, and H. Tizhoosh. “Unsupervised Detection of Lung Nodules in Chest Radiography Using Generative Adversarial Networks.” In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, Nov. 2021. DOI: 10.1109/embc46164.2021.9630340. URL: https://doi.org/10.1109/embc46164.2021.9630340 (cit. on p. 183).

[1862] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth. “f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks.” In: *Medical Image Analysis* 54 (May 2019), pp. 30–44. DOI: 10.1016/j.media.2019.01.010. URL: https://doi.org/10.1016/j.media.2019.01.010 (cit. on p. 183).

[1863] S. Park, K. H. Lee, B. Ko, and N. Kim. “Unsupervised anomaly detection with generative adversarial networks in mammography.” In: *Scientific Reports* 13.1 (Feb. 2023). DOI: 10.1038/s41598-023-29521-z. URL: https://doi.org/10.1038/s41598-023-29521-z (cit. on p. 183).

[1864] C. Zhao, T. Wang, and B. Lei. “Medical image fusion method based on dense block and deep convolutional generative adversarial network.” In: *Neural Computing and Applications* 33.12 (Oct. 2020), pp. 6595–6610. DOI: 10.1007/s00521-020-05421-5. URL: https://doi.org/10.1007/s00521-020-05421-5 (cit. on p. 183).

[1865] M. Esmaeili, A. Toosi, A. Roshanpoor, V. Changizi, M. Ghazisaeedi, A. Rahmim, and M. Sabokrou. “Generative Adversarial Networks for Anomaly Detection in Biomedical Imaging: A Study on Seven Medical Image Datasets.” In: *IEEE Access* 11 (2023), pp. 17906–17921. DOI: 10.1109/access.2023.3244741. URL: https://doi.org/10.1109/access.2023.3244741 (cit. on p. 183).

[1866] B. Yao, J. Li, S. Xue, J. Wu, H. Guan, J. Chang, and Z. Ding. “GARAT: Generative Adversarial Learning for Robust and Accurate Tracking.” In: *Neural Networks* 148 (Apr. 2022), pp. 206–218. DOI: 10.1016/j.neunet.2022.01.010. URL: https://doi.org/10.1016/j.neunet.2022.01.010 (cit. on p. 183).

[1867] X. Guo and L. Zhao. “A Systematic Survey on Deep Generative Models for Graph Generation.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–20. DOI: 10.1109/tpami.2022.3214832. URL: https://doi.org/10.1109/tpami.2022.3214832 (cit. on p. 183).

[1868] Z. Liu, N. Vouitsis, S. K. Gorti, J. Ba, and G. Loaiza-Ganem. *TRoN: Translator Networks for 0-Shot Plug-and-Play Conditional Generation*. 2023. DOI: 10.48550/ARXIV.2304.13742. URL: https://arxiv.org/abs/2304.13742 (cit. on p. 183).

[1869] Y. Gao, X. Liu, and J. Xiang. “Fault Detection in Gears Using Fault Samples Enlarged by a Combination of Numerical Simulation and a Generative Adversarial Network.” In: *IEEE/ASME Transactions on Mechatronics* 27.5 (Oct. 2022), pp. 3798–3805. DOI: 10.1109/tmech.2021.3132459. URL: https://doi.org/10.1109/tmech.2021.3132459 (cit. on p. 183).

[1870] S. Shao, P. Wang, and R. Yan. “Generative adversarial networks for data augmentation in machine fault diagnosis.” In: *Computers in Industry* 106 (Apr. 2019), pp. 85–93. DOI: 10.1016/j.compind.2019.01.001. URL: https://doi.org/10.1016/j.compind.2019.01.001 (cit. on p. 183).

[1871] L. Huo, H. Qi, S. Fei, C. Guan, and J. Li. “A Generative Adversarial Network Based a Rolling Bearing Data Generation Method Towards Fault Diagnosis.” In: *Computational Intelligence and Neuroscience* 2022 (July 2022). Ed. by S. Mumtaz, pp. 1–21. DOI: 10.1155/2022/7592258. URL: https://doi.org/10.1155/2022/7592258 (cit. on p. 183).

[1872] J. Liu, G. Yang, X. Li, Q. Wang, Y. He, and X. Yang. "Wind turbine anomaly detection based on SCADA: A deep autoencoder enhanced by fault instances." In: *ISA Transactions* (Apr. 2023). DOI: 10.1016/j.isatra.2023.03.045. URL: https://doi.org/10.1016/j.isatra.2023.03.045 (cit. on p. 183).

[1873] W. Fabian, K. Timo, B. Moritz, K. Markus, and L. Marcus. "Generation of synthetic data with low-dimensional features for condition monitoring utilizing Generative Adversarial Networks." In: *Procedia Computer Science* 207 (2022), pp. 634–643. DOI: 10.1016/j.procs.2022.09.118. URL: https://doi.org/10.1016/j.procs.2022.09.118 (cit. on p. 183).

[1874] A. Kebaili, J. Lapuyade-Lahorgue, and S. Ruan. "Deep Learning Approaches for Data Augmentation in Medical Imaging: A Review." In: *Journal of Imaging* 9.4 (Apr. 2023), p. 81. DOI: 10.3390/jimaging9040081. URL: https://doi.org/10.3390/jimaging9040081 (cit. on pp. 183, 184).

[1875] P. Chen, Y. Deng, Q. Zou, L. Lu, and H. Li. "EAAE: A Generative Adversarial Mechanism Based Classfication Method for Small-scale Datasets." In: *Neural Processing Letters* 55.2 (June 2022), pp. 969–987. ISSN: 1573-773X. DOI: 10.1007/s11063-022-10921-7. URL: http://dx.doi.org/10.1007/s11063-022-10921-7 (cit. on p. 183).

[1876] W. Mao, Y. Liu, L. Ding, and Y. Li. "Imbalanced Fault Diagnosis of Rolling Bearing Based on Generative Adversarial Network: A Comparative Study." In: *IEEE Access* 7 (2019), pp. 9515–9530. DOI: 10.1109/access.2018.2890693. URL: https://doi.org/10.1109/access.2018.2890693 (cit. on p. 183).

[1877] B. Zhao and Q. Yuan. "Improved generative adversarial network for vibration-based fault diagnosis with imbalanced data." In: *Measurement* 169 (Feb. 2021), p. 108522. DOI: 10.1016/j.measurement.2020.108522. URL: https://doi.org/10.1016/j.measurement.2020.108522 (cit. on p. 183).

[1878] W. S. Wong, M. Amer, T. Maul, I. Y. Liao, and A. Ahmed. "Conditional Generative Adversarial Networks for Data Augmentation in Breast Cancer Classification." In: *Recent Advances on Soft Computing and Data Mining*. Springer International Publishing, Dec. 2019, pp. 392–402. ISBN: 9783030360566. DOI: 10.1007/978-3-030-36056-6_37. URL: http://dx.doi.org/10.1007/978-3-030-36056-6_37 (cit. on p. 183).

[1879] O. N. Oyelade, A. E. Ezugwu, M. S. Almutairi, A. K. Saha, L. Abualigah, and H. Chiroma. "A generative adversarial network for synthetization of regions of interest based on digital mammograms." In: *Scientific Reports* 12.1 (Apr. 2022). DOI: 10.1038/s41598-022-09929-9. URL: https://doi.org/10.1038/s41598-022-09929-9 (cit. on p. 183).

[1880] D. Mukherkjee, P. Saha, D. Kaplun, A. Sinitca, and R. Sarkar. "Brain tumor image generation using an aggregation of GAN models with style transfer." In: *Scientific Reports* 12.1 (June 2022). DOI: 10.1038/s41598-022-12646-y. URL: https://doi.org/10.1038/s41598-022-12646-y (cit. on p. 183).

[1881] R. Touati and S. Kadoury. "A least square generative network based on invariant contrastive feature pair learning for multimodal MR image synthesis." In: *International Journal of Computer Assisted Radiology and Surgery* (Apr. 2023). DOI: 10.1007/s11548-023-02916-z. URL: https://doi.org/10.1007/s11548-023-02916-z (cit. on p. 183).

[1882] R. A. Werner, T. Higuchi, N. Nose, F. Toriumi, Y. Matsusaka, I. Kuji, and K. Kazuhiro. "Generative adversarial network-created brain SPECTs of cerebral ischemia are indistinguishable to scans from real patients." In: *Scientific Reports* 12.1 (Nov. 2022). DOI: 10.1038/s41598-022-23325-3. URL: https://doi.org/10.1038/s41598-022-23325-3 (cit. on p. 183).

[1883] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. "Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.374. URL: https://doi.org/10.1109/cvpr.2017.374 (cit. on p. 183).

[1884] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html (cit. on p. 183).

[1885] Y. Song and S. Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019 (cit. on p. 183).

[1886] P. Dhariwal and A. Nichol. "Diffusion Models Beat GANs on Image Synthesis." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 8780–8794. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf (cit. on p. 183).

[1887] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. *Diffusion Models: A Comprehensive Survey of Methods and Applications*. 2022. DOI: 10.48550/ARXIV.2209.00796. URL: https://arxiv.org/abs/2209.00796 (cit. on pp. 183, 184).

[1888] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann. *Conditional Image Generation with Score-Based Diffusion Models*. 2021. DOI: 10.48550/ARXIV.2111.13606. URL: https://arxiv.org/abs/2111.13606 (cit. on p. 183).

[1889] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. "Cascaded Diffusion Models for High Fidelity Image Generation." In: *Journal of Machine Learning Research* 23.47 (2022), pp. 1–33. URL: http://jmlr.org/papers/v23/21-0635.html (cit. on p. 183).

[1890] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. "Structured Denoising Diffusion Models in Discrete State-Spaces." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021. URL: https://openreview.net/forum?id=h7-XixPCAL (cit. on p. 183).

[1891] E. Luhman and T. Luhman. "Denoising Synthesis: A module for fast image synthesis using denoising-based models." In: *Software Impacts* 9 (Aug. 2021), p. 100076. DOI: 10.1016/j.simpa.2021.100076. URL: https://doi.org/10.1016/j.simpa.2021.100076 (cit. on pp. 183, 184).

[1892] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. "SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=aBsCjcPu_tE (cit. on p. 183).

[1893] A. Graikos, N. Malkin, N. Jojic, and D. Samaras. "Diffusion Models as Plug-and-Play Priors." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=yhlMZ3iR7Pu (cit. on pp. 183, 184).

[1894] V. Kulikov, S. Yadin, M. Kleiner, and T. Michaeli. *SinDDM: A Single Image Denoising Diffusion Model*. 2022. DOI: 10.48550/ARXIV.2211.16582. URL: https://arxiv.org/abs/2211.16582 (cit. on p. 183).

[1895] B. Kawar, R. Ganz, and M. Elad. "Enhancing Diffusion-Based Image Synthesis with Robust Classifier Guidance." In: *Transactions on Machine Learning Research* (2023). ISSN: 2835-8856. URL: https://openreview.net/forum?id=tEVpz2xJWX (cit. on p. 183).

[1896] A. Farshad, Y. Yeganeh, Y. Chi, C. Shen, B. Ommer, and N. Navab. *SceneGenie: Scene Graph Guided Diffusion Models for Image Synthesis*. 2023. DOI: 10.48550/ARXIV.2304.14573. URL: https://arxiv.org/abs/2304.14573 (cit. on p. 183).

[1897] G. C. Tarrés, D. Ruta, T. Bui, and J. Collomosse. *PARASOL: Parametric Style Control for Diffusion Image Synthesis*. 2023. DOI: 10.48550/ARXIV.2303.06464. URL: https://arxiv.org/abs/2303.06464 (cit. on p. 183).

[1898] X. Liu, D. H. Park, S. Azadi, G. Zhang, A. Chopikyan, Y. Hu, H. Shi, A. Rohrbach, and T. Darrell. "More Control for Free! Image Synthesis with Semantic Diffusion Guidance." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00037. URL: https://doi.org/10.1109/wacv56688.2023.00037 (cit. on p. 183).

[1899] F. Khader et al. "Denoising diffusion probabilistic models for 3D medical image generation." In: *Scientific Reports* 13.1 (May 2023). DOI: 10.1038/s41598-023-34341-2. URL: https://doi.org/10.1038/s41598-023-34341-2 (cit. on p. 183).

[1900] I. Han, S. Yang, T. Kwon, and J. C. Ye. *Highly Personalized Text Embedding for Image Manipulation by Stable Diffusion*. 2023. DOI: 10.48550/ARXIV.2303.08767. URL: https://arxiv.org/abs/2303.08767 (cit. on p. 183).

[1901] H. Go, Y. Lee, J.-Y. Kim, S. Lee, M. Jeong, H. S. Lee, and S. Choi. *Towards Practical Plug-and-Play Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2212.05973. URL: https://arxiv.org/abs/2212.05973 (cit. on p. 183).

[1902] V. T. Hu, D. W. Zhang, Y. M. Asano, G. J. Burghouts, and C. G. M. Snoek. "Self-Guided Diffusion Model." In: *NeurIPS 2022 Workshop on Score-Based Methods*. 2022. URL: https://openreview.net/forum?id=Mf6NLebyqdq (cit. on p. 183).

[1903] X. Liu, L. Wu, M. Ye, and qiang liu. "Let us Build Bridges: Understanding and Extending Diffusion Generative Models." In: *NeurIPS 2022 Workshop on Score-Based Methods*. 2022. URL: https://openreview.net/forum?id=0ef0CRKC9uZ (cit. on p. 183).

[1904] X. Liu, L. Wu, M. Ye, and Q. Liu. *Let us Build Bridges: Understanding and Extending Diffusion Generative Models*. 2022. DOI: 10.48550/ARXIV.2208.14699. URL: https://arxiv.org/abs/2208.14699 (cit. on pp. 183, 184).

[1905] T. Chen, R. ZHANG, and G. Hinton. "Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=3itjR9QxFw (cit. on p. 183).

[1906] V. Fernandez, W. H. L. Pinaya, P. Borges, P.-D. Tudosiu, M. S. Graham, T. Vercauteren, and M. J. Cardoso. "Can Segmentation Models Be Trained with Fully Synthetically Generated Data?" In: *Simulation and Synthesis in Medical Imaging*. Springer International Publishing, 2022, pp. 79–90. DOI: 10.1007/978-3-031-16980-9_8. URL: https://doi.org/10.1007/978-3-031-16980-9_8 (cit. on p. 183).

[1907] F. Bao, S. Nie, K. Xue, C. Li, S. Pu, Y. Wang, G. Yue, Y. Cao, H. Su, and J. Zhu. *One Transformer Fits All Distributions in Multi-Modal Diffusion at Scale*. 2023. DOI: 10.48550/ARXIV.2303.06555. URL: https://arxiv.org/abs/2303.06555 (cit. on pp. 183, 184).

[1908] X. Xu, Z. Wang, E. Zhang, K. Wang, and H. Shi. *Versatile Diffusion: Text, Images and Variations All in One Diffusion Model*. 2022. DOI: 10.48550/ARXIV.2211.08332. URL: https://arxiv.org/abs/2211.08332 (cit. on p. 183).

[1909] A. Pokle, Z. Geng, and J. Z. Kolter. "Deep Equilibrium Approaches to Diffusion Models." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=zGPeowwxWb (cit. on p. 183).

[1910] L. Ruan, Y. Ma, H. Yang, H. He, B. Liu, J. Fu, N. J. Yuan, Q. Jin, and B. Guo. *MM-Diffusion: Learning Multi-Modal Diffusion Models for Joint Audio and Video Generation*. 2022. DOI: 10.48550/ARXIV.2212.09478. URL: https://arxiv.org/abs/2212.09478 (cit. on pp. 183, 184).

[1911] Y. Nikankin, N. Haim, and M. Irani. *SinFusion: Training Diffusion Models on a Single Image or Video*. 2022. DOI: 10.48550/ARXIV.2211.11743. URL: https://arxiv.org/abs/2211.11743 (cit. on p. 183).

[1912] D. Zhou, W. Wang, H. Yan, W. Lv, Y. Zhu, and J. Feng. *MagicVideo: Efficient Video Generation With Latent Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2211.11018. URL: https://arxiv.org/abs/2211.11018 (cit. on p. 183).

[1913] W. Harvey, S. Naderiparizi, V. Masrani, C. D. Weilbach, and F. Wood. "Flexible Diffusion Modeling of Long Videos." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=0RTJcuvHtIu (cit. on p. 183).

[1914] J. Ho, T. Salimans, A. A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. "Video Diffusion Models." In: *ICLR Workshop on Deep Generative Models for Highly Structured Data*. 2022. URL: https://openreview.net/forum?id=BBelR2NdDZ5 (cit. on p. 183).

[1915] Anonymous. "Diffusion Probabilistic Modeling for Video Generation." In: *Submitted to Transactions on Machine Learning Research* (2023). Rejected. URL: https://openreview.net/forum?id=Sw4aYWX21a (cit. on p. 183).

[1916] R. Yang, P. Srivastava, and S. Mandt. *Diffusion Probabilistic Modeling for Video Generation*. 2022. DOI: 10.48550/ARXIV.2203.09481. URL: https://arxiv.org/abs/2203.09481 (cit. on p. 183).

[1917] S. Yu, K. Sohn, S. Kim, and J. Shin. *Video Probabilistic Diffusion Models in Projected Latent Space*. 2023. DOI: 10.48550/ARXIV.2302.07685. URL: https://arxiv.org/abs/2302.07685 (cit. on p. 183).

[1918] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu. *MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model*. 2022. DOI: 10.48550/ARXIV.2208.15001. URL: https://arxiv.org/abs/2208.15001 (cit. on p. 183).

[1919] J. Z. Wu, Y. Ge, X. Wang, W. Lei, Y. Gu, Y. Shi, W. Hsu, Y. Shan, X. Qie, and M. Z. Shou. *Tune-A-Video: One-Shot Tuning of Image Diffusion Models for Text-to-Video Generation*. 2022. DOI: 10.48550/ARXIV.2212.11565. URL: https://arxiv.org/abs/2212.11565 (cit. on p. 183).

[1920] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. *Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2304.08818. URL: https://arxiv.org/abs/2304.08818 (cit. on p. 183).

[1921] Y. He, T. Yang, Y. Zhang, Y. Shan, and Q. Chen. *Latent Video Diffusion Models for High-Fidelity Long Video Generation*. 2022. DOI: 10.48550/ARXIV.2211.13221. URL: https://arxiv.org/abs/2211.13221 (cit. on p. 183).

[1922] J. Ho et al. *Imagen Video: High Definition Video Generation with Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2210.02303. URL: https://arxiv.org/abs/2210.02303 (cit. on p. 183).

[1923] A. Ulhaq, N. Akhtar, and G. Pogrebna. *Efficient Diffusion Models for Vision: A Survey*. 2022. DOI: 10.48550/ARXIV.2210.09292. URL: https://arxiv.org/abs/2210.09292 (cit. on pp. 183, 184).

[1924] C. Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=08Yk-n5l2Al (cit. on p. 183).

[1925] Z. Pan, X. Zhou, and H. Tian. "Arbitrary Style Guidance for Enhanced Diffusion-Based Text-to-Image Generation." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00444. URL: https://doi.org/10.1109/wacv56688.2023.00444 (cit. on p. 183).

[1926] A. Voynov, K. Aberman, and D. Cohen-Or. *Sketch-Guided Text-to-Image Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2211.13752. URL: https://arxiv.org/abs/2211.13752 (cit. on p. 183).

[1927] Y. Balaji et al. *eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers*. 2022. DOI: 10.48550/ARXIV.2211.01324. URL: https://arxiv.org/abs/2211.01324 (cit. on p. 183).

[1928] N. Huang, Y. Zhang, F. Tang, C. Ma, H. Huang, Y. Zhang, W. Dong, and C. Xu. *DiffStyler: Controllable Dual Diffusion for Text-Driven Image Stylization*. 2022. DOI: 10.48550/ARXIV.2211.10682. URL: https://arxiv.org/abs/2211.10682 (cit. on p. 183).

[1929] G. Kim and S. Y. Chun. *DATID-3D: Diversity-Preserved Domain Adaptation Using Text-to-Image Diffusion for 3D Generative Model*. 2022. DOI: 10.48550/ARXIV.2211.16374. URL: https://arxiv.org/abs/2211.16374 (cit. on p. 183).

[1930] L. Struppek, D. Hintersdorf, and K. Kersting. *Rickrolling the Artist: Injecting Backdoors into Text Encoders for Text-to-Image Synthesis*. 2022. DOI: 10.48550/ARXIV.2211.02408. URL: https://arxiv.org/abs/2211.02408 (cit. on pp. 183, 184).

[1931] L. Struppek, D. Hintersdorf, F. Friedrich, M. Brack, P. Schramowski, and K. Kersting. *Exploiting Cultural Biases via Homoglyphs in Text-to-Image Synthesis*. 2022. DOI: 10.48550/ARXIV.2209.08891. URL: https://arxiv.org/abs/2209.08891 (cit. on p. 183).

[1932] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-or. "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=NAQvF08TcyG (cit. on p. 183).

[1933] Z. Dong, P. Wei, and L. Lin. *DreamArtist: Towards Controllable One-Shot Text-to-Image Generation via Positive-Negative Prompt-Tuning*. 2022. DOI: 10.48550/ARXIV.2211.11337. URL: https://arxiv.org/abs/2211.11337 (cit. on p. 183).

[1934] W. Chen, H. Hu, Y. Li, N. Ruiz, X. Jia, M.-W. Chang, and W. W. Cohen. *Subject-driven Text-to-Image Generation via Apprenticeship Learning*. 2023. DOI: 10.48550/ARXIV.2304.00186. URL: https://arxiv.org/abs/2304.00186 (cit. on p. 183).

[1935] Z. Liu, Y. Shin, B.-C. Okogwu, Y. Yun, L. Coleman, P. Schaldenbrand, J. Kim, and J. Oh. *Towards Equitable Representation in Text-to-Image Synthesis Models with the Cross-Cultural Understanding Benchmark (CCUB) Dataset*. 2023. DOI: 10.48550/ARXIV.2301.12073. URL: https://arxiv.org/abs/2301.12073 (cit. on p. 183).

[1936] X. Jia, Y. Zhao, K. C. K. Chan, Y. Li, H. Zhang, B. Gong, T. Hou, H. Wang, and Y.-C. Su. *Taming Encoder for Zero Fine-tuning Image Customization with Text-to-Image Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2304.02642. URL: https://arxiv.org/abs/2304.02642 (cit. on p. 183).

[1937] R. Zbinden. *Implementing and Experimenting with Diffusion Models for Text-to-Image Generation*. 2022. DOI: 10.48550/ARXIV.2209.10948. URL: https://arxiv.org/abs/2209.10948 (cit. on p. 183).

[1938] S. Sheynin, O. Ashual, A. Polyak, U. Singer, O. Gafni, E. Nachmani, and Y. Taigman. "kNN-Diffusion: Image Generation via Large-Scale Retrieval." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=x5mtJD2ovc (cit. on p. 183).

[1939] Y. Zhou, B. Liu, Y. Zhu, X. Yang, C. Chen, and J. Xu. *Shifted Diffusion for Text-to-image Generation*. 2022. DOI: 10.48550/ARXIV.2211.15388. URL: https://arxiv.org/abs/2211.15388 (cit. on p. 183).

[1940] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel. *Plug-and-Play Diffusion Features for Text-Driven Image-to-Image Translation*. 2022. DOI: 10.48550/ARXIV.2211.12572. URL: https://arxiv.org/abs/2211.12572 (cit. on p. 183).

[1941] C. Meng, R. Gao, D. P. Kingma, S. Ermon, J. Ho, and T. Salimans. "On Distillation of Guided Diffusion Models." In: *NeurIPS 2022 Workshop on Score-Based Methods*. 2022. URL: https://openreview.net/forum?id=6QHpSQt6VR- (cit. on p. 183).

[1942] Y. Zhu and Y. Zhao. *Diffusion Models in NLP: A Survey*. 2023. DOI: 10.48550/ARXIV.2303.07576. URL: https://arxiv.org/abs/2303.07576 (cit. on pp. 183, 184).

[1943] O. Avrahami, D. Lischinski, and O. Fried. "Blended Diffusion for Text-driven Editing of Natural Images." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01767. URL: https://doi.org/10.1109/cvpr52688.2022.01767 (cit. on p. 183).

[1944] A. Q. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. Mcgrew, I. Sutskever, and M. Chen. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 16784–16804. URL: https://proceedings.mlr.press/v162/nichol22a.html (cit. on p. 183).

[1945] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo. "Vector Quantized Diffusion Model for Text-to-Image Synthesis." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01043. URL: https://doi.org/10.1109/cvpr52688.2022.01043 (cit. on p. 183).

[1946] W.-C. Fan, Y.-C. Chen, D. Chen, Y. Cheng, L. Yuan, and Y.-C. F. Wang. *Frido: Feature Pyramid Diffusion for Complex Scene Image Synthesis*. 2022. DOI: 10.48550/ARXIV.2208.13753. URL: https://arxiv.org/abs/2208.13753 (cit. on p. 183).

[1947] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. *DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation*. 2022. DOI: 10.48550/ARXIV.2208.12242. URL: https://arxiv.org/abs/2208.12242 (cit. on p. 183).

[1948] M. F. Sutedy and N. N. Qomariyah. "Text to Image Latent Diffusion Model with Dreambooth Fine Tuning for Automobile Image Generation." In: *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, Dec. 2022. DOI: 10.1109/isriti56927.2022.10052908. URL: https://doi.org/10.1109/isriti56927.2022.10052908 (cit. on p. 183).

[1949] B. Kawar, S. Zada, O. Lang, O. Tov, H. Chang, T. Dekel, I. Mosseri, and M. Irani. *Imagic: Text-Based Real Image Editing with Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2210.09276. URL: https://arxiv.org/abs/2210.09276 (cit. on p. 183).

[1950] A. Voynov, Q. Chu, D. Cohen-Or, and K. Aberman. *P+: Extended Textual Conditioning in Text-to-Image Generation*. 2023. DOI: 10.48550/ARXIV.2303.09522. URL: https://arxiv.org/abs/2303.09522 (cit. on p. 183).

[1951] D. Valevski, M. Kalman, Y. Matias, and Y. Leviathan. *UniTune: Text-Driven Image Editing by Fine Tuning an Image Generation Model on a Single Image*. 2022. DOI: 10.48550/ARXIV.2210.09477. URL: https://arxiv.org/abs/2210.09477 (cit. on p. 183).

[1952] L. Zhang and M. Agrawala. *Adding Conditional Control to Text-to-Image Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2302.05543. URL: https://arxiv.org/abs/2302.05543 (cit. on p. 183).

[1953] L. Yang, Z. Huang, Y. Song, S. Hong, G. Li, W. Zhang, B. Cui, B. Ghanem, and M.-H. Yang. *Diffusion-Based Scene Graph to Image Generation with Masked Contrastive Pre-Training*. 2022. DOI: 10.48550/ARXIV.2211.11138. URL: https://arxiv.org/abs/2211.11138 (cit. on p. 183).

[1954] J. An, S. Zhang, H. Yang, S. Gupta, J.-B. Huang, J. Luo, and X. Yin. *Latent-Shift: Latent Diffusion with Temporal Shift for Efficient Text-to-Video Generation*. 2023. DOI: 10.48550/ARXIV.2304.08477. URL: https://arxiv.org/abs/2304.08477 (cit. on p. 183).

[1955] U. Singer et al. "Make-A-Video: Text-to-Video Generation without Text-Video Data." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=nJfylDvgzlq (cit. on p. 183).

[1956] C. Qi, X. Cun, Y. Zhang, C. Lei, X. Wang, Y. Shan, and Q. Chen. *FateZero: Fusing Attentions for Zero-shot Text-based Video Editing*. 2023. DOI: 10.48550/ARXIV.2303.09535. URL: https://arxiv.org/abs/2303.09535 (cit. on p. 183).

[1957] E. J. C. Findlay, H. Zhang, Z. Chang, and H. P. H. Shum. *Denoising Diffusion Probabilistic Models for Styled Walking Synthesis*. 2022. DOI: 10.48550/ARXIV.2209.14828. URL: https://arxiv.org/abs/2209.14828 (cit. on p. 183).

[1958] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano. "Human Motion Diffusion Model." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=SJ1kSyO2jwu (cit. on p. 183).

[1959] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz. *PhysDiff: Physics-Guided Human Motion Diffusion Model*. 2022. DOI: 10.48550/ARXIV.2212.02500. URL: https://arxiv.org/abs/2212.02500 (cit. on p. 183).

[1960] M. Zhang, X. Guo, L. Pan, Z. Cai, F. Hong, H. Li, L. Yang, and Z. Liu. *ReMoDiffuse: Retrieval-Augmented Motion Diffusion Model*. 2023. DOI: 10.48550/ARXIV.2304.01116. URL: https://arxiv.org/abs/2304.01116 (cit. on p. 183).

[1961] J. Kim, J. Kim, and S. Choi. *FLAME: Free-form Language-based Motion Synthesis & Editing*. 2022. DOI: 10.48550/ARXIV.2209.00349. URL: https://arxiv.org/abs/2209.00349 (cit. on p. 183).

[1962] J. Tseng, R. Castellon, and C. K. Liu. *EDGE: Editable Dance Generation From Music*. 2022. DOI: 10.48550/ARXIV.2211.10658. URL: https://arxiv.org/abs/2211.10658 (cit. on p. 183).

[1963] H. Ni, C. Shi, K. Li, S. X. Huang, and M. R. Min. *Conditional Image-to-Video Generation with Latent Flow Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2303.13744. URL: https://arxiv.org/abs/2303.13744 (cit. on p. 183).

[1964] T. Höppe, A. Mehrjou, S. Bauer, D. Nielsen, and A. Dittadi. "Diffusion Models for Video Prediction and Infilling." In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. URL: https://openreview.net/forum?id=lf0lr4AYM6 (cit. on p. 183).

[1965] M. Özbey, O. Dalmaz, S. U. Dar, H. A. Bedel, Ş. Özturk, A. Güngör, and T. undefinedukur. *Unsupervised Medical Image Translation with Adversarial Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2207.08208. URL: https://arxiv.org/abs/2207.08208 (cit. on pp. 183, 184).

[1966] H. Chung and J. C. Ye. "Score-based diffusion models for accelerated MRI." In: *Medical Image Analysis* 80 (Aug. 2022), p. 102479. DOI: 10.1016/j.media.2022.102479. URL: https://doi.org/10.1016/j.media.2022.102479 (cit. on p. 183).

[1967] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. "Palette: Image-to-Image Diffusion Models." In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. ACM, Aug. 2022. DOI: 10.1145/3528233.3530757. URL: https://doi.org/10.1145/3528233.3530757 (cit. on pp. 183, 184).

[1968] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn. "Diffusion Autoencoders: Toward a Meaningful and Decodable Representation." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01036. URL: https://doi.org/10.1109/cvpr52688.2022.01036 (cit. on p. 183).

[1969] Y. Song, L. Shen, L. Xing, and S. Ermon. "Solving Inverse Problems in Medical Imaging with Score-Based Generative Models." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=vaRCHVj0uGI (cit. on p. 183).

[1970] B. Kawar, M. Elad, S. Ermon, and J. Song. "Denoising Diffusion Restoration Models." In: *ICLR Workshop on Deep Generative Models for Highly Structured Data*. 2022. URL: https://openreview.net/forum?id=BExXihVOvWq (cit. on pp. 183, 184).

[1971] Y. Wang, J. Yu, and J. Zhang. "Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=mRieQgMtNTQ (cit. on pp. 183, 184).

[1972] G. Kwon and J. C. Ye. "Diffusion-based Image Translation using disentangled style and content representation." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=Nayau9fwXU (cit. on p. 183).

[1973] G. Kim, T. Kwon, and J. C. Ye. "DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.00246. URL: https://doi.org/10.1109/cvpr52688.2022.00246 (cit. on p. 183).

[1974] C. Kong, D. Jeon, O. Kwon, and N. Kwak. "Leveraging Off-the-shelf Diffusion Model for Multi-attribute Fashion Image Manipulation." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00091. URL: https://doi.org/10.1109/wacv56688.2023.00091 (cit. on p. 183).

[1975] H. Ravi, S. Kelkar, M. Harikumar, and A. Kale. *PRedItOR: Text Guided Image Editing with Diffusion Prior*. 2023. DOI: 10.48550/ARXIV.2302.07979. URL: https://arxiv.org/abs/2302.07979 (cit. on p. 183).

[1976] N. Starodubcev, D. Baranchuk, V. Khrulkov, and A. Babenko. *Towards Real-time Text-driven Image Manipulation with Unconditional Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2304.04344. URL: https://arxiv.org/abs/2304.04344 (cit. on p. 183).

[1977] N. G. Nair, K. Mei, and V. M. Patel. "AT-DDPM: Restoring Faces Degraded by Atmospheric Turbulence Using Denoising Diffusion Probabilistic Models." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00343. URL: https://doi.org/10.1109/wacv56688.2023.00343 (cit. on p. 183).

[1978] K. Karchev, N. A. Montel, A. Coogan, and C. Weniger. *Strong-Lensing Source Reconstruction with Denoising Diffusion Restoration Models*. 2022. DOI: 10.48550/ARXIV.2211.04365. URL: https://arxiv.org/abs/2211.04365 (cit. on p. 183).

[1979] Q. Lyu and G. Wang. *Conversion Between CT and MRI Images Using Diffusion and Score-Matching Models*. 2022. DOI: 10.48550/ARXIV.2209.12104. URL: https://arxiv.org/abs/2209.12104 (cit. on p. 183).

[1980] H. Chung and J. C. Ye. *Score-based diffusion models for accelerated MRI*. 2021. DOI: 10.48550/ARXIV.2110.05243. URL: https://arxiv.org/abs/2110.05243 (cit. on p. 183).

[1981] A. Güngör, S. U. Dar, Ş. Öztürk, Y. Korkmaz, G. Elmas, M. Özbey, and T. undefinedukur. *Adaptive Diffusion Priors for Accelerated MRI Reconstruction*. 2022. DOI: 10.48550/ARXIV.2207.05876. URL: https://arxiv.org/abs/2207.05876 (cit. on p. 183).

[1982] B. Kim, Y. Oh, and J. C. Ye. "Diffusion Adversarial Representation Learning for Self-supervised Vessel Segmentation." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=H0gdPxSwkPb (cit. on pp. 183, 184).

[1983] P. Rouzrokh, B. Khosravi, S. Faghani, M. Moassefi, S. Vahdati, and B. J. Erickson. *Multitask Brain Tumor Inpainting with Diffusion Models: A Methodological Report*. 2022. DOI: 10.48550/ARXIV.2210.12113. URL: https://arxiv.org/abs/2210.12113 (cit. on p. 183).

[1984] A. Kazerouni, E. K. Aghdam, M. Heidari, R. Azad, M. Fayyaz, I. Hacihaliloglu, and D. Merhof. *Diffusion Models for Medical Image Analysis: A Comprehensive Survey*. 2022. DOI: 10.48550/ARXIV.2211.07804. URL: https://arxiv.org/abs/2211.07804 (cit. on pp. 183, 184).

[1985] T. Chen, C. Wang, and H. Shan. *BerDiff: Conditional Bernoulli Diffusion Model for Medical Image Segmentation*. 2023. DOI: 10.48550/ARXIV.2304.04429. URL: https://arxiv.org/abs/2304.04429 (cit. on p. 183).

[1986] A. Rahman, J. M. J. Valanarasu, I. Hacihaliloglu, and V. M. Patel. *Ambiguous Medical Image Segmentation using Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2304.04745. URL: https://arxiv.org/abs/2304.04745 (cit. on p. 183).

[1987] H. Chung, E. S. Lee, and J. C. Ye. "MR Image Denoising and Super-Resolution Using Regularized Reverse Diffusion." In: *IEEE Transactions on Medical Imaging* 42.4 (Apr. 2023), pp. 922–934. DOI: 10.1109/tmi.2022.3220681. URL: https://doi.org/10.1109/tmi.2022.3220681 (cit. on pp. 183, 184).

[1988] H. Chung, B. Sim, and J. C. Ye. "Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01209. URL: https://doi.org/10.1109/cvpr52688.2022.01209 (cit. on pp. 183, 184).

[1989] C. Peng, P. Guo, S. K. Zhou, V. M. Patel, and R. Chellappa. "Towards Performant and Reliable Undersampled MR Reconstruction via Diffusion Model Sampling." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 623–633. DOI: 10.1007/978-3-031-16446-0_59. URL: https://doi.org/10.1007/978-3-031-16446-0_59 (cit. on p. 183).

[1990] Y. Xie and Q. Li. "Measurement-Conditioned Denoising Diffusion Probabilistic Model for Under-Sampled Medical Image Reconstruction." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 655–664. DOI: 10.1007/978-3-031-16446-0_62. URL: https://doi.org/10.1007/978-3-031-16446-0_62 (cit. on p. 183).

[1991] C. Shin, H. Kim, C. H. Lee, S.-g. Lee, and S. Yoon. *Edit-A-Video: Single Video Editing with Object-Aware Consistency*. 2023. DOI: 10.48550/ARXIV.2303.07945. URL: https://arxiv.org/abs/2303.07945 (cit. on p. 183).

[1992] B. Kawar, J. Song, S. Ermon, and M. Elad. "JPEG Artifact Correction using Denoising Diffusion Restoration Models." In: *NeurIPS 2022 Workshop on Score-Based Methods*. 2022. URL: https://openreview.net/forum?id=O3WJOt79289 (cit. on p. 183).

[1993] D. Baranchuk, A. Voynov, I. Rubachev, V. Khrulkov, and A. Babenko. "Label-Efficient Semantic Segmentation with Diffusion Models." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=SlxSY2UZQT (cit. on p. 183).

[1994] L. Zbinden, L. Doorenbos, T. Pissas, A. T. Huber, R. Sznitman, and P. Márquez-Neila. *Stochastic Segmentation with Conditional Categorical Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2303.08888. URL: https://arxiv.org/abs/2303.08888 (cit. on p. 183).

[1995] E. A. Brempong, S. Kornblith, T. Chen, N. Parmar, M. Minderer, and M. Norouzi. "Denoising Pretraining for Semantic Segmentation." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2022. DOI: 10.1109/cvprw56347.2022.00462. URL: https://doi.org/10.1109/cvprw56347.2022.00462 (cit. on p. 183).

[1996] J. Xu, S. Liu, A. Vahdat, W. Byeon, X. Wang, and S. De Mello. *Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2303.04803. URL: https://arxiv.org/abs/2303.04803 (cit. on p. 183).

[1997] T. Amit, T. Shaharbany, E. Nachmani, and L. Wolf. *SegDiff: Image Segmentation with Diffusion Probabilistic Models*. 2021. DOI: 10.48550/ARXIV.2112.00390. URL: https://arxiv.org/abs/2112.00390 (cit. on p. 183).

[1998] J. Wolleb, R. Sandkühler, F. Bieder, P. Valmaggia, and P. C. Cattin. *Diffusion Models for Implicit Image Segmentation Ensembles*. 2021. DOI: 10.48550/ARXIV.2112.03145. URL: https://arxiv.org/abs/2112.03145 (cit. on p. 183).

[1999] B. Kolbeinsson and K. Mikolajczyk. *Multi-Class Segmentation from Aerial Views using Recursive Noise Diffusion*. 2022. DOI: 10.48550/ARXIV.2212.00787. URL: https://arxiv.org/abs/2212.00787 (cit. on p. 183).

[2000] B. Kim, Y. Oh, and J. C. Ye. *Diffusion Adversarial Representation Learning for Self-supervised Vessel Segmentation*. 2022. DOI: 10.48550/ARXIV.2209.14566. URL: https://arxiv.org/abs/2209.14566 (cit. on pp. 183, 184).

[2001] X. L. Li, J. Thickstun, I. Gulrajani, P. Liang, and T. Hashimoto. "Diffusion-LM Improves Controllable Text Generation." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=3s9IrEsjLyk (cit. on p. 183).

[2002] T. Rahman, H.-Y. Lee, J. Ren, S. Tulyakov, S. Mahajan, and L. Sigal. *Make-A-Story: Visual Memory Conditioned Consistent Story Generation*. 2022. DOI: 10.48550/ARXIV.2211.13319. URL: https://arxiv.org/abs/2211.13319 (cit. on p. 183).

[2003] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong. "DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=jQj-_rLVXsj (cit. on p. 183).

[2004] X. Han, S. Kumar, and Y. Tsvetkov. *SSD-LM: Semi-autoregressive Simplex-based Diffusion Language Model for Text Generation and Modular Control*. 2022. DOI: 10.48550/ARXIV.2210.17432. URL: https://arxiv.org/abs/2210.17432 (cit. on p. 183).

[2005] T. Wang et al. *Rodin: A Generative Model for Sculpting 3D Digital Avatars Using Diffusion*. 2022. DOI: 10.48550/ARXIV.2212.06135. URL: https://arxiv.org/abs/2212.06135 (cit. on p. 183).

[2006] A. Raj et al. *DreamBooth3D: Subject-Driven Text-to-3D Generation*. 2023. DOI: 10.48550/ARXIV.2303.13508. URL: https://arxiv.org/abs/2303.13508 (cit. on p. 183).

[2007] J. Xu, X. Wang, W. Cheng, Y.-P. Cao, Y. Shan, X. Qie, and S. Gao. *Dream3D: Zero-Shot Text-to-3D Synthesis Using 3D Shape Prior and Text-to-Image Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2212.14704. URL: https://arxiv.org/abs/2212.14704 (cit. on p. 183).

[2008] Z. Liu, P. Dai, R. Li, X. Qi, and C.-W. Fu. *ISS++: Image as Stepping Stone for Text-Guided 3D Shape Generation*. 2023. DOI: 10.48550/ARXIV.2303.15181. URL: https://arxiv.org/abs/2303.15181 (cit. on p. 183).

[2009] A. Karnewar, A. Vedaldi, D. Novotny, and N. Mitra. *HOLODIFFUSION: Training a 3D Diffusion Model using 2D Images*. 2023. DOI: 10.48550/ARXIV.2303.16509. URL: https://arxiv.org/abs/2303.16509 (cit. on p. 183).

[2010] N. Müller, Y. Siddiqui, L. Porzi, S. R. Bulò, P. Kontschieder, and M. Nießner. *DiffRF: Rendering-Guided 3D Radiance Field Diffusion*. 2022. DOI: 10.48550/ARXIV.2212.01206. URL: https://arxiv.org/abs/2212.01206 (cit. on p. 183).

[2011] Z. Zhou and S. Tulsiani. *SparseFusion: Distilling View-conditioned Diffusion for 3D Reconstruction*. 2022. DOI: 10.48550/ARXIV.2212.00792. URL: https://arxiv.org/abs/2212.00792 (cit. on pp. 183, 184).

[2012] G. Chou, Y. Bahat, and F. Heide. *Diffusion-SDF: Conditional Generative Modeling of Signed Distance Functions*. 2022. DOI: 10.48550/ARXIV.2211.13757. URL: https://arxiv.org/abs/2211.13757 (cit. on p. 183).

[2013] C. Sbrolli, P. Cudrano, M. Frosi, and M. Matteucci. *IC3D: Image-Conditioned 3D Diffusion for Shape Generation*. 2022. DOI: 10.48550/ARXIV.2211.10865. URL: https://arxiv.org/abs/2211.10865 (cit. on p. 183).

[2014] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. "DreamFusion: Text-to-3D using 2D Diffusion." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=FjNys5c7VyY (cit. on p. 183).

[2015] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. *Magic3D: High-Resolution Text-to-3D Content Creation*. 2022. DOI: 10.48550/ARXIV.2211.10440. URL: https://arxiv.org/abs/2211.10440 (cit. on p. 183).

[2016] S. Hong, D. Ahn, and S. Kim. *Debiasing Scores and Prompts of 2D Diffusion for Robust Text-to-3D Generation*. 2023. DOI: 10.48550/ARXIV.2303.15413. URL: https://arxiv.org/abs/2303.15413 (cit. on p. 183).

[2017] L. Zhou, Y. Du, and J. Wu. "3D Shape Generation and Completion through Point-Voxel Diffusion." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.00577. URL: https://doi.org/10.1109/iccv48922.2021.00577 (cit. on p. 184).

[2018] S. Luo and W. Hu. "Diffusion Probabilistic Models for 3D Point Cloud Generation." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00286. URL: https://doi.org/10.1109/cvpr46437.2021.00286 (cit. on p. 184).

[2019] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis. "LION: Latent Point Diffusion Models for 3D Shape Generation." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=tHK5ntjp-5K (cit. on p. 184).

[2020] D. Watson, W. Chan, R. M. Brualla, J. Ho, A. Tagliasacchi, and M. Norouzi. "Novel View Synthesis with Diffusion Models." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=HtoA0oT30jC (cit. on p. 184).

[2021] H. Chen, J. Gu, A. Chen, W. Tian, Z. Tu, L. Liu, and H. Su. *Single-Stage Diffusion NeRF: A Unified Approach to 3D Generation and Reconstruction*. 2023. DOI: 10.48550/ARXIV.2304.06714. URL: https://arxiv.org/abs/2304.06714 (cit. on p. 184).

[2022] S. W. Kim, B. Brown, K. Yin, K. Kreis, K. Schwarz, D. Li, R. Rombach, A. Torralba, and S. Fidler. *NeuralField-LDM: Scene Generation with Hierarchical Latent Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2304.09787. URL: https://arxiv.org/abs/2304.09787 (cit. on p. 184).

[2023] Q. Wang, H. Deng, Y. Qi, D. Li, and Y.-Z. Song. "SketchKnitter: Vectorized Sketch Generation with Diffusion Models." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=4eJ43EN2g6l (cit. on p. 184).

[2024] P. Yu, S. Xie, X. Ma, B. Jia, B. Pang, R. Gao, Y. Zhu, S.-C. Zhu, and Y. N. Wu. "Latent Diffusion Energy-Based Model for Interpretable Text Modelling." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 25702–25720. URL: https://proceedings.mlr.press/v162/yu22h.html (cit. on p. 184).

[2025] Y. Li, K. Zhou, W. X. Zhao, and J.-R. Wen. *Diffusion Models for Non-autoregressive Text Generation: A Survey*. 2023. DOI: 10.48550/ARXIV.2303.06574. URL: https://arxiv.org/abs/2303.06574 (cit. on p. 184).

[2026] S. Dieleman et al. *Continuous diffusion for categorical data*. 2022. DOI: 10.48550/ARXIV.2211.15089. URL: https://arxiv.org/abs/2211.15089 (cit. on p. 184).

[2027] H. Li, Y. Yang, M. Chang, S. Chen, H. Feng, Z. Xu, Q. Li, and Y. Chen. "SRDiff: Single image super-resolution with diffusion probabilistic models." In: *Neurocomputing* 479 (Mar. 2022), pp. 47–59. DOI: 10.1016/j.neucom.2022.01.029. URL: https://doi.org/10.1016/j.neucom.2022.01.029 (cit. on p. 184).

[2028] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. "Image Super-Resolution Via Iterative Refinement." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–14. DOI: 10.1109/tpami.2022.3204461. URL: https://doi.org/10.1109/tpami.2022.3204461 (cit. on p. 184).

[2029] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01042. URL: https://doi.org/10.1109/cvpr52688.2022.01042 (cit. on p. 184).

[2030] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool. "RePaint: Inpainting using Denoising Diffusion Probabilistic Models." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.01117. URL: https://doi.org/10.1109/cvpr52688.2022.01117 (cit. on p. 184).

[2031] S. H. Lee, S. Kim, I. Yoo, F. Yang, D. Cho, Y. Kim, H. Chang, J. Kim, and S. Kim. *Soundini: Sound-Guided Diffusion for Natural Video Editing*. 2023. DOI: 10.48550/ARXIV.2304.06818. URL: https://arxiv.org/abs/2304.06818 (cit. on p. 184).

[2032] F. Zhang, N. Ji, F. Gao, and Y. Li. "DiffMotion: Speech-Driven Gesture Synthesis Using Denoising Diffusion Model." In: *MultiMedia Modeling*. Springer International Publishing, 2023, pp. 231–242. DOI: 10.1007/978-3-031-27077-2_18. URL: https://doi.org/10.1007/978-3-031-27077-2_18 (cit. on p. 184).

[2033] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan. "WaveGrad: Estimating Gradients for Waveform Generation." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=NsMLjcFaO8O (cit. on p. 184).

[2034] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. "DiffWave: A Versatile Diffusion Model for Audio Synthesis." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=a-xFK8Ymz5J (cit. on p. 184).

[2035] C. Zhang, C. Zhang, S. Zheng, M. Zhang, M. Qamar, S.-H. Bae, and I. S. Kweon. *A Survey on Audio Diffusion Models: Text To Speech Synthesis and Enhancement in Generative AI*. 2023. DOI: 10.48550/ARXIV.2303.13336. URL: https://arxiv.org/abs/2303.13336 (cit. on p. 184).

[2036] A. Levkovitch, E. Nachmani, and L. Wolf. "Zero-Shot Voice Conditioning for Denoising Diffusion TTS Models." In: *Interspeech 2022*. ISCA, Sept. 2022. DOI: 10.21437/interspeech.2022-10045. URL: https://doi.org/10.21437/interspeech.2022-10045 (cit. on p. 184).

[2037] J. Tae, H. Kim, and T. Kim. *EdiTTS: Score-based Editing for Controllable Text-to-Speech*. 2021. DOI: 10.48550/ARXIV.2110.02584. URL: https://arxiv.org/abs/2110.02584 (cit. on p. 184).

[2038] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov. "Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech." In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8599–8608. URL: https://proceedings.mlr.press/v139/popov21a.html (cit. on p. 184).

[2039] S. Wu and Z. Shi. "ItôWave: Itô Stochastic Differential Equation is all You Need for Wave Generation." In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022. DOI: 10.1109/icassp43922.2022.9746153. URL: https://doi.org/10.1109/icassp43922.2022.9746153 (cit. on p. 184).

[2040] R. Huang, Z. Zhao, H. Liu, J. Liu, C. Cui, and Y. Ren. "ProDiff: Progressive Fast Diffusion Model for High-Quality Text-to-Speech." In: *Proceedings of the 30th ACM International Conference on Multimedia*. ACM, Oct. 2022. DOI: 10.1145/3503161.3547855. URL: https://doi.org/10.1145/3503161.3547855 (cit. on p. 184).

[2041] H. Kim, S. Kim, and S. Yoon. "Guided-TTS: A Diffusion Model for Text-to-Speech via Classifier Guidance." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 11119–11133. URL: https://proceedings.mlr.press/v162/kim22d.html (cit. on p. 184).

[2042] S. Kim, H. Kim, and S. Yoon. *Guided-TTS 2: A Diffusion Model for High-quality Adaptive Text-to-Speech with Untranscribed Data*. 2022. DOI: 10.48550/ARXIV.2205.15370. URL: https://arxiv.org/abs/2205.15370 (cit. on p. 184).

[2043] D. Yang, J. Yu, H. Wang, W. Wang, C. Weng, Y. Zou, and D. Yu. "Diffsound: Discrete Diffusion Model for Text-to-Sound Generation." In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 31 (2023), pp. 1720–1733. DOI: 10.1109/taslp.2023.3268730. URL: https://doi.org/10.1109/taslp.2023.3268730 (cit. on p. 184).

[2044] J. Zhang, S. Jayasuriya, and V. Berisha. "Restoring Degraded Speech via a Modified Diffusion Model." In: *Interspeech 2021*. ISCA, Aug. 2021. DOI: 10.21437/interspeech.2021-1889. URL: https://doi.org/10.21437/interspeech.2021-1889 (cit. on p. 184).

[2045] K. Saito, N. Murata, T. Uesaka, C.-H. Lai, Y. Takida, T. Fukui, and Y. Mitsufuji. "Unsupervised Vocal Dereverberation with Diffusion-Based Generative Models." In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2023. DOI: 10.1109/icassp49357.2023.10095761. URL: https://doi.org/10.1109/icassp49357.2023.10095761 (cit. on p. 184).

[2046] L. Lin, Z. Li, R. Li, X. Li, and J. Gao. *Diffusion Models for Time Series Applications: A Survey*. 2023. DOI: 10.48550/ARXIV.2305.00624. URL: https://arxiv.org/abs/2305.00624 (cit. on p. 184).

[2047] Y. Li, X. Lu, Y. Wang, and D. Dou. "Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=rG0jm74xtx (cit. on p. 184).

[2048] J. L. Alcaraz and N. Strodthoff. "Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models." In: *Transactions on Machine Learning Research* (2023). ISSN: 2835-8856. URL: https://openreview.net/forum?id=hHiIbk7ApW (cit. on p. 184).

[2049] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf. "Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting." In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8857–8868. URL: https://proceedings.mlr.press/v139/rasul21a.html (cit. on p. 184).

[2050] Y. Tashiro, J. Song, Y. Song, and S. Ermon. "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021. URL: https://openreview.net/forum?id=VzuIzbRDrum (cit. on p. 184).

[2051] S. W. Park, K. Lee, and J. Kwon. "Neural Markov Controlled SDE: Stochastic Optimization for Continuous-Time Data." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=7DI6op61AY (cit. on p. 184).

[2052] M. S. Graham, W. H. L. Pinaya, P.-D. Tudosiu, P. Nachev, S. Ourselin, and M. J. Cardoso. *Denoising diffusion models for out-of-distribution detection*. 2022. DOI: 10.48550/ARXIV.2211.07740. URL: https://arxiv.org/abs/2211.07740 (cit. on p. 184).

[2053] Y. Wang, D. Guo, S. Li, and Y. Fu. *Towards Explainable Visual Anomaly Detection*. 2023. DOI: 10.48550/ARXIV.2302.06670. URL: https://arxiv.org/abs/2302.06670 (cit. on p. 184).

[2054] J. Wyatt, A. Leach, S. M. Schmon, and C. G. Willcocks. "AnoDDPM: Anomaly Detection with Denoising Diffusion Probabilistic Models using Simplex Noise." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2022. DOI: 10.1109/cvprw56347.2022.00080. URL: https://doi.org/10.1109/cvprw56347.2022.00080 (cit. on p. 184).

[2055] J. Wolleb, F. Bieder, R. Sandkühler, and P. C. Cattin. "Diffusion Models for Medical Anomaly Detection." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 35–45. DOI: 10.1007/978-3-031-16452-1_4. URL: https://doi.org/10.1007/978-3-031-16452-1_4 (cit. on p. 184).

[2056] W. G. C. Bandara, N. G. Nair, and V. M. Patel. *DDPM-CD: Remote Sensing Change Detection using Denoising Diffusion Probabilistic Models*. 2022. DOI: 10.48550/ARXIV.2206.11892. URL: https://arxiv.org/abs/2206.11892 (cit. on p. 184).

[2057] Z. Sun and Y. Yang. *DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization*. 2023. DOI: 10.48550/ARXIV.2302.08224. URL: https://arxiv.org/abs/2302.08224 (cit. on p. 184).

[2058] I. Corley and P. Najafirad. *Single-View Height Estimation with Conditional Diffusion Probabilistic Models*. 2023. DOI: 10.48550/ARXIV.2304.13214. URL: https://arxiv.org/abs/2304.13214 (cit. on p. 184).

[2059] P. Fernandez, G. Couairon, H. Jégou, M. Douze, and T. Furon. *The Stable Signature: Rooting Watermarks in Latent Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2303.15435. URL: https://arxiv.org/abs/2303.15435 (cit. on p. 184).

[2060] M. Zhang, M. Qamar, T. Kang, Y. Jung, C. Zhang, S.-H. Bae, and C. Zhang. "A Survey on Graph Diffusion Models: Generative AI in Science for Molecule, Protein and Material." In: (2023). DOI: 10.48550/ARXIV.2304.01565. URL: https://arxiv.org/abs/2304.01565 (cit. on p. 184).

[2061] Z. Guo, J. Liu, Y. Wang, M. Chen, D. Wang, D. Xu, and J. Cheng. *Diffusion Models in Bioinformatics: A New Wave of Deep Learning Revolution in Action*. 2023. DOI: 10.48550/ARXIV.2302.10907. URL: https://arxiv.org/abs/2302.10907 (cit. on p. 184).

[2062] C. Liu, W. Fan, Y. Liu, J. Li, H. Li, H. Liu, J. Tang, and Q. Li. *Generative Diffusion Models on Graphs: Methods and Applications*. 2023. DOI: 10.48550/ARXIV.2302.02591. URL: https://arxiv.org/abs/2302.02591 (cit. on p. 184).

[2063] L. Wu, C. Gong, X. Liu, M. Ye, and qiang liu. "Diffusion-based Molecule Generation with Informative Prior Bridges." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=TJUNtiZiTKE (cit. on p. 184).

[2064] A. Morehead and J. Cheng. *Geometry-Complete Diffusion for 3D Molecule Generation*. 2023. DOI: 10.48550/ARXIV.2302.04313. URL: https://arxiv.org/abs/2302.04313 (cit. on p. 184).

[2065] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling. "Equivariant Diffusion for Molecule Generation in 3D." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 8867–8887. URL: https://proceedings.mlr.press/v162/hoogeboom22a.html (cit. on p. 184).

[2066] B. Jing, G. Corso, J. Chang, R. Barzilay, and T. S. Jaakkola. "Torsional Diffusion for Molecular Conformer Generation." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=w6fj2r62r_H (cit. on p. 184).

[2067] N. Anand and T. Achim. *Protein Structure and Sequence Generation with Equivariant Denoising Diffusion Probabilistic Models*. 2022. DOI: 10.48550/ARXIV.2205.15019. URL: https://arxiv.org/abs/2205.15019 (cit. on p. 184).

[2068] K. E. Wu, K. K. Yang, R. v. d. Berg, J. Y. Zou, A. X. Lu, and A. P. Amini. *Protein structure generation via folding diffusion*. 2022. DOI: 10.48550/ARXIV.2209.15611. URL: https://arxiv.org/abs/2209.15611 (cit. on p. 184).

[2069] B. Jing, E. Erives, P. Pao-Huang, G. Corso, B. Berger, and T. S. Jaakkola. "EigenFold: Generative Protein Structure Prediction with Diffusion Models." In: *ICLR 2023 - Machine Learning for Drug Discovery workshop*. 2023. URL: https://openreview.net/forum?id=BgbRVzfQqFp (cit. on p. 184).

[2070] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. S. Jaakkola. "DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=kKF8_K-mBbS (cit. on p. 184).

[2071] M. A. Ketata, C. Laue, R. Mammadov, H. Stark, M. Wu, G. Corso, C. Marquet, R. Barzilay, and T. S. Jaakkola. "DiffDock-PP: Rigid Protein-Protein Docking with Diffusion Models." In: *ICLR 2023 - Machine Learning for Drug Discovery workshop*. 2023. URL: https://openreview.net/forum?id=AM7WbQxuRS (cit. on p. 184).

[2072] B. L. Trippe, J. Yim, D. Tischer, D. Baker, T. Broderick, R. Barzilay, and T. S. Jaakkola. "Diffusion Probabilistic Modeling of Protein Backbones in 3D for the motif-scaffolding problem." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=6TxBxqNME1Y (cit. on p. 184).

[2073] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang. "GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=PzcvxEMzvQC (cit. on p. 184).

[2074] C. Shi, S. Luo, M. Xu, and J. Tang. "Learning Gradient Fields for Molecular Conformation Generation." In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 9558–9568. URL: https://proceedings.mlr.press/v139/shi21b.html (cit. on p. 184).

[2075] S. Luo, C. Shi, M. Xu, and J. Tang. "Predicting Molecular Conformation via Dynamic Graph Score Matching." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 19784–19795. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/a45a1d12ee0fb7f1f872ab91da18f899-Paper.pdf (cit. on p. 184).

[2076] T. Xie, X. Fu, O.-E. Ganea, R. Barzilay, and T. S. Jaakkola. "Crystal Diffusion Variational Autoencoder for Periodic Material Generation." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=03RLpj-tc_ (cit. on p. 184).

[2077] S. Luo, Y. Su, X. Peng, S. Wang, J. Peng, and J. Ma. "Antigen-Specific Antibody Design and Optimization with Diffusion-Based Generative Models for Protein Structures." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=jSorGn2Tjg (cit. on p. 184).

[2078] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar. "Diffusion Models for Adversarial Purification." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 16805–16827. URL: https://proceedings.mlr.press/v162/nie22a.html (cit. on p. 184).

[2079] T. Blau, R. Ganz, B. Kawar, A. Bronstein, and M. Elad. *Threat Model-Agnostic Adversarial Defense using Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2207.08089. URL: https://arxiv.org/abs/2207.08089 (cit. on p. 184).

[2080] J. Yoon, S. J. Hwang, and J. Lee. "Adversarial Purification with Score-based Generative Models." In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 12062–12072. URL: https://proceedings.mlr.press/v139/yoon21a.html (cit. on p. 184).

[2081] Q. Wu, H. Ye, and Y. Gu. *Guided Diffusion Model for Adversarial Purification from Random Noise*. 2022. DOI: 10.48550/ARXIV.2206.10875. URL: https://arxiv.org/abs/2206.10875 (cit. on p. 184).

[2082] J. Wang, Z. Lyu, D. Lin, B. Dai, and H. Fu. *Guided Diffusion Model for Adversarial Purification*. 2022. DOI: 10.48550/ARXIV.2205.14969. URL: https://arxiv.org/abs/2205.14969 (cit. on p. 184).

[2083] J. Sun, W. Nie, Z. Yu, Z. M. Mao, and C. Xiao. *PointDP: Diffusion-driven Purification against Adversarial Attacks on 3D Point Cloud Recognition*. 2022. DOI: 10.48550/ARXIV.2208.09801. URL: https://arxiv.org/abs/2208.09801 (cit. on p. 184).

[2084] C. Xiao, Z. Chen, K. Jin, J. Wang, W. Nie, M. Liu, A. Anandkumar, B. Li, and D. Song. "DensePure: Understanding Diffusion Models for Adversarial Robustness." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=p7hvOJ6Gq0i (cit. on p. 184).

[2085] S. Wu, J. Wang, W. Ping, W. Nie, and C. Xiao. "Defending against Adversarial Audio via Diffusion Model." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=5-Df3tljit7 (cit. on p. 184).

[2086] C. Shi, C. Holtz, and G. Mishne. "Online Adversarial Purification based on Self-supervised Learning." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=_i3ASPp12WS (cit. on p. 184).

[2087] N. Carlini, F. Tramer, K. D. Dvijotham, L. Rice, M. Sun, and J. Z. Kolter. "(Certified!!) Adversarial Robustness for Free!" In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=JLg5aHHv7j (cit. on p. 184).

[2088] C. Meng, L. Yu, Y. Song, J. Song, and S. Ermon. "Autoregressive Score Matching." In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546. URL: https://dl.acm.org/doi/pdf/10.5555/3495724.3496284 (cit. on p. 184).

[2089] C. Meng, J. Song, Y. Song, S. Zhao, and S. Ermon. "Improved Autoregressive Modeling with Distribution Smoothing." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=rJA5Pz7lHKb (cit. on p. 184).

[2090] A. Vahdat, K. Kreis, and J. Kautz. "Score-based Generative Modeling in Latent Space." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021. URL: https://openreview.net/forum?id=P9TYG0j-wtG (cit. on p. 184).

[2091] C.-W. Huang, J. H. Lim, and A. C. Courville. "A Variational Perspective on Diffusion-Based Generative Models and Score Matching." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 22863–22876. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/c11abfd29e4d9b4d4b566b01114d8486-Paper.pdf (cit. on p. 184).

[2092] C. Luo. *Understanding Diffusion Models: A Unified Perspective*. 2022. DOI: 10.48550/ARXIV.2208.11970. URL: https://arxiv.org/abs/2208.11970 (cit. on p. 184).

[2093] Z. Wang, H. Zheng, P. He, W. Chen, and M. Zhou. "Diffusion-GAN: Training GANs with Diffusion." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=HZf7UbpWHuA (cit. on p. 184).

[2094] Z. Xiao, K. Kreis, and A. Vahdat. "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=JprM0p-q0Co (cit. on p. 184).

[2095] D. Klein and K. Yang. *FUDGE: Controlled Text Generation With Future Discriminators*. 2021. DOI: 10.48448/S9SW-6G59. URL: https://underline.io/lecture/19580-fudge-controlled-text-generation-with-future-discriminators (cit. on p. 184).

[2096] K. Yang and D. Klein. "FUDGE: Controlled Text Generation With Future Discriminators." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021. DOI: 10.18653/v1/2021.naacl-main.276. URL: https://doi.org/10.18653/v1/2021.naacl-main.276 (cit. on p. 184).

[2097] Q. Zhang and Y. Chen. "Diffusion Normalizing Flow." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021. URL: https://openreview.net/forum?id=x1Lp2bOlVIo (cit. on p. 184).

[2098] W. Gong and Y. Li. *Interpreting diffusion score matching using normalizing flow*. 2021. DOI: 10.48550/ARXIV.2107.10072. URL: https://arxiv.org/abs/2107.10072 (cit. on p. 184).

[2099] D. Kim, B. Na, S. J. Kwon, D. Lee, W. Kang, and I.-c. Moon. "Maximum Likelihood Training of Implicit Nonlinear Diffusion Model." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=TQn44YPuOR2 (cit. on p. 184).

[2100] R. Gao, Y. Song, B. Poole, Y. N. Wu, and D. P. Kingma. "Learning Energy-Based Models by Diffusion Recovery Likelihood." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=v_1Soh8QUNc (cit. on p. 184).

[2101] H. Cao, C. Tan, Z. Gao, G. Chen, P.-A. Heng, and S. Z. Li. *A Survey on Generative Diffusion Model*. 2022. DOI: 10.48550/ARXIV.2209.02646. URL: https://arxiv.org/abs/2209.02646 (cit. on p. 184).

[2102] W. Luo. *A Comprehensive Survey on Knowledge Distillation of Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2304.04262. URL: https://arxiv.org/abs/2304.04262 (cit. on p. 184).

[2103] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah. "Diffusion Models in Vision: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), pp. 1–20. DOI: 10.1109/tpami.2023.3261988. URL: https://doi.org/10.1109/tpami.2023.3261988 (cit. on p. 184).

[2104] M. Jovanovic and M. Campbell. "Generative Artificial Intelligence: Trends and Prospects." In: *Computer* 55.10 (Oct. 2022), pp. 107–112. DOI: 10.1109/mc.2022.3192720. URL: https://doi.org/10.1109/mc.2022.3192720 (cit. on p. 184).

[2105] Z. Zhao, J. C. Ye, and Y. Bresler. "Generative Models for Inverse Imaging Problems: From mathematical foundations to physics-driven applications." In: *IEEE Signal Processing Magazine* 40.1 (Jan. 2023), pp. 148–163. DOI: 10.1109/msp.2022.3215282. URL: https://doi.org/10.1109/msp.2022.3215282 (cit. on p. 184).

[2106] H. Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach*. GMD Report 159, German National Research Center for Information Technology. 2002. URL: https://www.ai.rug.nl/minds/uploads/ESNTutorialRev.pdf (cit. on p. 185).

[2107] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass. "The role of feedback in morphological computation with compliant bodies." In: *Biological Cybernetics* 106.10 (Sept. 2012), pp. 595–613. DOI: 10.1007/s00422-012-0516-4. URL: https://doi.org/10.1007/s00422-012-0516-4 (cit. on p. 185).

[2108] L. Righetti and A. J. Ijspeert. "Pattern generators with sensory feedback for the control of quadruped locomotion." In: *2008 IEEE International Conference on Robotics and Automation*. IEEE, May 2008. DOI: 10.1109/robot.2008.4543306. URL: https://doi.org/10.1109/robot.2008.4543306 (cit. on p. 185).

[2109] L. A. Gatys, A. S. Ecker, and M. Bethge. "Texture Synthesis Using Convolutional Neural Networks." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 262–270 (cit. on p. 186).

[2110] A. Mahendran and A. Vedaldi. "Understanding deep image representations by inverting them." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. DOI: 10.1109/cvpr.2015.7299155. URL: https://doi.org/10.1109/cvpr.2015.7299155 (cit. on p. 186).

[2111] L. Gatys, A. Ecker, and M. Bethge. "A Neural Algorithm of Artistic Style." In: *Journal of Vision* 16.12 (Sept. 2016), p. 326. DOI: 10.1167/16.12.326. URL: https://doi.org/10.1167/16.12.326 (cit. on p. 186).

[2112] M. Abbas and Y. EL-Manzalawy. "Machine learning based refined differential gene expression analysis of pediatric sepsis." In: *BMC Medical Genomics* 13.1 (Aug. 2020). DOI: 10.1186/s12920-020-00771-4. URL: https://doi.org/10.1186/s12920-020-00771-4 (cit. on pp. xlvi, 192).

[2113] G. K. Smyth et al. *limma*. 2017. DOI: 10.18129/B9.BIOC.LIMMA. URL: https://bioconductor.org/packages/limma (cit. on pp. 192, 193, 270).

[2114] M. E. Ritchie, B. Phipson, D. Wu, Y. Hu, C. W. Law, W. Shi, and G. K. Smyth. "limma powers differential expression analyses for RNA-sequencing and microarray studies." In: *Nucleic Acids Research* 43.7 (Jan. 2015), e47–e47. DOI: 10.1093/nar/gkv007. URL: https://doi.org/10.1093/nar/gkv007 (cit. on pp. 192, 193, 270).

[2115] B. Efron and C. Morris. "Stein's Estimation Rule and Its Competitors–An Empirical Bayes Approach." In: *Journal of the American Statistical Association* 68.341 (Mar. 1973), p. 117. ISSN: 0162-1459. DOI: 10.2307/2284155. URL: http://dx.doi.org/10.2307/2284155 (cit. on p. 192).

[2116] G. K. Smyth. "Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments." In: *Statistical Applications in Genetics and Molecular Biology* 3.1 (Jan. 2004), pp. 1–25. ISSN: 1544-6115. DOI: 10.2202/1544-6115.1027. URL: http://dx.doi.org/10.2202/1544-6115.1027 (cit. on p. 192).

[2117] M. Loni, A. Mohan, M. Asadi, and M. Lindauer. *Learning Activation Functions for Sparse Neural Networks*. 2023. DOI: 10.48550/ARXIV.2305.10964. URL: https://arxiv.org/abs/2305.10964 (cit. on p. 194).

[2118] A. Liu, H. Hu, T. Qiu, Q. Zhou, Q. Guan, and X. Li. "Exploring Optimal Adaptive Activation Functions for Various Tasks." In: *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, Dec. 2020. DOI: 10.1109/bibm49941.2020.9313386. URL: http://dx.doi.org/10.1109/BIBM49941.2020.9313386 (cit. on p. 194).

[2119] H. Bostrom et al. "On evidential combination rules for ensemble classifiers." In: *Information Fusion, 2008 11th International Conference on*. June 2008, pp. 1–8 (cit. on p. 221).

[2120] Y. Freund and R. E. Schapire. "Experiments with a New Boosting Algorithm." In: *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996, pp. 148–156. URL: http://cseweb.ucsd.edu/~yfreund/papers/boostingexperiments.pdf (cit. on p. 221).

[2121] L. Rokach. "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography." In: *Computational Statistics & Data Analysis* 53.12 (Oct. 2009), pp. 4046–4072. DOI: 10.1016/j.csda.2009.07.017. URL: http://dx.doi.org/10.1016/j.csda.2009.07.017 (cit. on p. 221).

[2122] L. Rokach. "Ensemble-based Classifiers." In: *Artif. Intell. Rev.* 33.1-2 (Feb. 2010), pp. 1–39. ISSN: 0269-2821. DOI: 10.1007/s10462-009-9124-7. URL: http://dx.doi.org/10.1007/s10462-009-9124-7 (cit. on p. 221).

[2123] O. Vinyals et al. "Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017), pp. 652–663. DOI: 10.1109/tpami.2016.2587640. URL: https://doi.org/10.1109/tpami.2016.2587640 (cit. on p. 221).

[2124] I. Nigam et al. "Ensemble Knowledge Transfer for Semantic Segmentation." In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2018. DOI: 10.1109/wacv.2018.00168. URL: https://doi.org/10.1109/wacv.2018.00168 (cit. on p. 221).

[2125] P. Yang et al. “A Review of Ensemble Methods in Bioinformatics.” In: *CBIO* 5.4 (Dec. 2010), pp. 296–308. DOI: 10.2174/157489310794072508. URL: http://dx.doi.org/10.2174/157489310794072508 (cit. on p. 221).

[2126] G. Valentini et al. “Bagged ensembles of Support Vector Machines for gene expression data analysis.” In: *Proceedings of the International Joint Conference on Neural Networks, 2003*. IEEE, 2003. DOI: 10.1109/ijcnn.2003.1223688. URL: http://dx.doi.org/10.1109/IJCNN.2003.1223688 (cit. on p. 221).

[2127] L. I. Kuncheva. *Combining Pattern Classifiers*. John Wiley & Sons, Inc., July 2004. DOI: 10.1002/0471660264. URL: http://dx.doi.org/10.1002/0471660264 (cit. on p. 221).

[2128] A. K. Tiwari and R. Srivastava. “A Survey of Computational Intelligence Techniques in Protein Function Prediction.” In: *International Journal of Proteomics* 2014 (2014), pp. 1–22. DOI: 10.1155/2014/845479. URL: http://dx.doi.org/10.1155/2014/845479 (cit. on p. 221).

[2129] E. Jones et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 2015-05-12]. 2001–. URL: http://www.scipy.org/ (cit. on p. 225).

[2130] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://arxiv.org/pdf/1201.0490v2.pdf (cit. on p. 225).

[2131] W. McKinney. “Data Structures for Statistical Computing in Python.” In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and J. Millman. 2010, pp. 51–56 (cit. on p. 225).

[2132] S. van der Walt et al. “The NumPy Array: A Structure for Efficient Numerical Computation.” In: *Computing in Science & Engineering* 13.2 (Mar. 2011), pp. 22–30. DOI: 10.1109/mcse.2011.37. URL: http://dx.doi.org/10.1109/MCSE.2011.37 (cit. on p. 225).

[2133] J. D. Hunter. “Matplotlib: A 2D Graphics Environment.” In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/mcse.2007.55. URL: http://dx.doi.org/10.1109/MCSE.2007.55 (cit. on p. 225).

[2134] M. Waskom et al. *seaborn: v0.7.1 (June 2016)*. June 2016. DOI: 10.5281/zenodo.54844. URL: https://doi.org/10.5281/zenodo.54844 (cit. on p. 225).

[2135] C. Yun, S. Sra, and A. Jadbabaie. “Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/dbea3d0e2a17c170c412c74273778159-Paper.pdf (cit. on p. 277).

[2136] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. “Understanding deep learning requires rethinking generalization.” In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Sy8gdB9xx (cit. on p. 277).

[2137] R. Vershynin. “Memory Capacity of Neural Networks with Threshold and Rectified Linear Unit Activations.” In: *SIAM Journal on Mathematics of Data Science* 2.4 (Jan. 2020), pp. 1004–1033. ISSN: 2577-0187. DOI: 10.1137/20m1314884. URL: http://dx.doi.org/10.1137/20M1314884 (cit. on p. 277).

[2138] G.-B. Huang. “Learning capability and storage capacity of two-hidden-layer feedforward networks.” In: *IEEE Transactions on Neural Networks* 14.2 (Mar. 2003), pp. 274–281. ISSN: 1045-9227. DOI: 10.1109/tnn.2003.809401. URL: http://dx.doi.org/10.1109/TNN.2003.809401 (cit. on p. 277).

[2139] D. A. Medler and M. R. W. Dawson. “Training redundant artificial neural networks: Imposing biology on technology.” In: *Psychological Research* 57.1 (Nov. 1994), pp. 54–62. DOI: 10.1007/bf00452996. URL: https://doi.org/10.1007/bf00452996 (cit. on p. 278).

[2140] D. A. Medler and M. R. W. Dawson. “Using Redundancy to Improve The Performance of Artificial Neural Networks.” In: *Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*. New Jersey: IEEE, 1994 (cit. on p. 278).

[2141] M. Johnson and S. Chartier. “Is There a Purpose to Network Redundancy?” In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2018. DOI: 10.1109/ijcnn.2018.8489203. URL: https://doi.org/10.1109/ijcnn.2018.8489203 (cit. on p. 278).

[2142] A. T. Nguyen, J. Xu, D. K. Luu, Q. Zhao, and Z. Yang. “Advancing System Performance with Redundancy: From Biological to Artificial Designs.” In: *Neural Computation* 31.3 (Mar. 2019), pp. 555–573. DOI: 10.1162/neco_a_01166. URL: https://doi.org/10.1162/neco_a_01166 (cit. on p. 278).

[2143] I. Safran and O. Shamir. “On the Quality of the Initial Basin in Overspecified Neural Networks.” In: ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 774–782. URL: http://proceedings.mlr.press/v48/safran16.html (cit. on p. 278).

[2144] Q. Nguyen and M. Hein. "The Loss Surface of Deep and Wide Neural Networks." In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 2603–2612 (cit. on p. 278).

[2145] D. C. Freeman and J. Bruna. "Topology and Geometry of Half-Rectified Network Optimization." In: 5th International Conference on Learning Representations. ICLR, 2017. arXiv: 1611.01540 (cit. on p. 278).

[2146] C. Lee, Y.-B. Kim, H. Ji, Y. Lee, Y. Hur, and H. Lim. "On the Redundancy in the Rank of Neural Network Parameters and Its Controllability." In: *Applied Sciences* 11.2 (Jan. 2021), p. 725. ISSN: 2076-3417. DOI: 10.3390/app11020725. URL: http://dx.doi.org/10.3390/app11020725 (cit. on p. 278).

[2147] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. "Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks." In: *CoRR* abs/1909.12228 (2019). arXiv: 1909.12228. URL: http://arxiv.org/abs/1909.12228 (cit. on p. 278).

[2148] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. "Modeling wine preferences by data mining from physicochemical properties." In: *Decision Support Systems* 47.4 (Nov. 2009), pp. 547–553. ISSN: 0167-9236. DOI: 10.1016/j.dss.2009.05.016. URL: http://dx.doi.org/10.1016/j.dss.2009.05.016 (cit. on p. 282).

[2149] W. Wolberg, W. Street, and O. Mangasarian. *Breast Cancer Wisconsin (Prognostic)*. 1995. DOI: 10.24432/C5GK50. URL: https://archive.ics.uci.edu/dataset/16 (cit. on p. 282).

[2150] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. "Visualizing the Loss Landscape of Neural Nets." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf (cit. on p. 288).

[2151] A. Anuarbekov. "Neural network learning visualization." Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, June 2021 (cit. on p. 288).

[2152] J. R. Beveridge et al. "The challenge of face recognition from digital point-and-shoot cameras." In: *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*. IEEE, Sept. 2013. DOI: 10.1109/btas.2013.6712704. URL: http://dx.doi.org/10.1109/BTAS.2013.6712704 (cit. on p. 289).

[2153] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments." In: *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*. Erik Learned-Miller and Andras Ferencz and Frédéric Jurie. Marseille, France, Oct. 2008. URL: https://inria.hal.science/inria-00321923 (cit. on p. 289).

[2154] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. "Attribute and simile classifiers for face verification." In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, Sept. 2009. DOI: 10.1109/iccv.2009.5459250. URL: http://dx.doi.org/10.1109/ICCV.2009.5459250 (cit. on p. 289).

[2155] P. Phillips, H. Moon, S. Rizvi, and P. Rauss. "The FERET evaluation methodology for face-recognition algorithms." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.10 (2000), pp. 1090–1104. ISSN: 0162-8828. DOI: 10.1109/34.879790. URL: http://dx.doi.org/10.1109/34.879790 (cit. on p. 289).

[2156] P. Phillips, H. Wechsler, J. Huang, and P. J. Rauss. "The FERET database and evaluation procedure for face-recognition algorithms." In: *Image and Vision Computing* 16.5 (Apr. 1998), pp. 295–306. ISSN: 0262-8856. DOI: 10.1016/s0262-8856(97)00070-x. URL: http://dx.doi.org/10.1016/S0262-8856(97)00070-X (cit. on p. 289).

[2157] A. M. Martínez and R. Benavente. *The AR Face Database*. CVC Technical Report #24. 1998. URL: https://www2.ece.ohio-state.edu/~aleix/ARdatabase.html (cit. on p. 289).

[2158] A. Martínez and A. Kak. "PCA versus LDA." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.2 (2001), pp. 228–233. ISSN: 0162-8828. DOI: 10.1109/34.908974. URL: http://dx.doi.org/10.1109/34.908974 (cit. on p. 289).

[2159] A. Georghiades, P. Belhumeur, and D. Kriegman. "From few to many: illumination cone models for face recognition under variable lighting and pose." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.6 (June 2001), pp. 643–660. ISSN: 0162-8828. DOI: 10.1109/34.927464. URL: http://dx.doi.org/10.1109/34.927464 (cit. on p. 289).

[2160] K.-C. Lee, J. Ho, and D. Kriegman. "Acquiring linear subspaces for face recognition under variable lighting." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.5 (May 2005), pp. 684–698. ISSN: 0162-8828. DOI: 10.1109/tpami.2005.92. URL: http://dx.doi.org/10.1109/TPAMI.2005.92 (cit. on p. 289).

[2161] B. Zhang, L. Zhang, D. Zhang, and L. Shen. “Directional binary code with application to PolyU near-infrared face database.” In: *Pattern Recognition Letters* 31.14 (Oct. 2010), pp. 2337–2344. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2010.07.006. URL: http://dx.doi.org/10.1016/j.patrec.2010.07.006 (cit. on p. 289).

[2162] S. Sabour, N. Frosst, and G. E. Hinton. “Dynamic Routing Between Capsules.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/2cad8fa47bbef282badbb8de5374b894-Paper.pdf (cit. on p. 289).

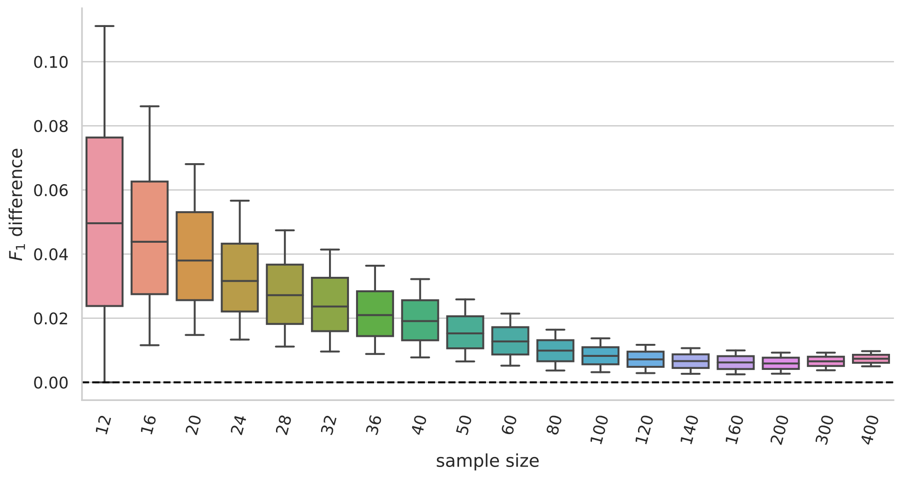
[2163] S. Bhuvaji, A. Kadam, P. Bhumkar, S. Dedge, and S. Kanchan. *Brain Tumor Classification (MRI)*. 2020. DOI: 10.34740/kaggle/dsv/1183165. URL: https://www.kaggle.com/dsv/1183165 (cit. on p. 289).

[2164] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius. *Accelerating Sparse Deep Neural Networks*. 2021. DOI: 10.48550/ARXIV.2104.08378. URL: https://arxiv.org/abs/2104.08378 (cit. on p. 289).

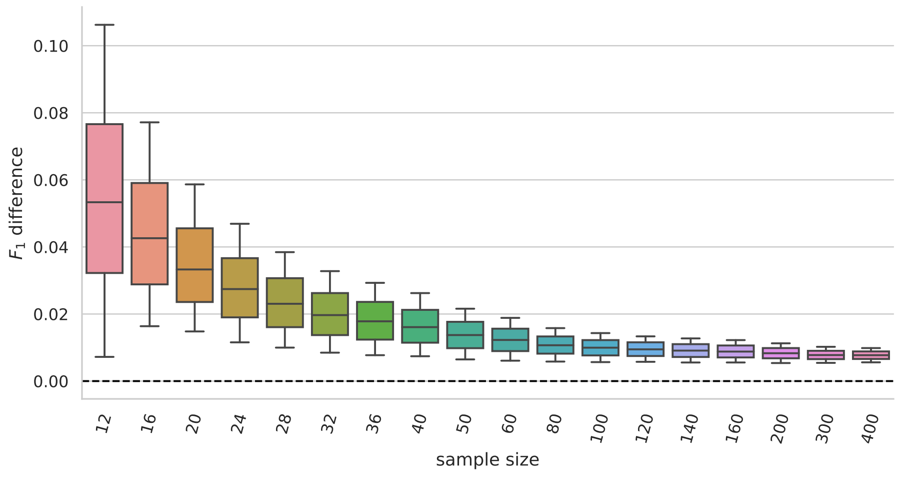
APPENDIX

A ADDITIONAL FIGURES

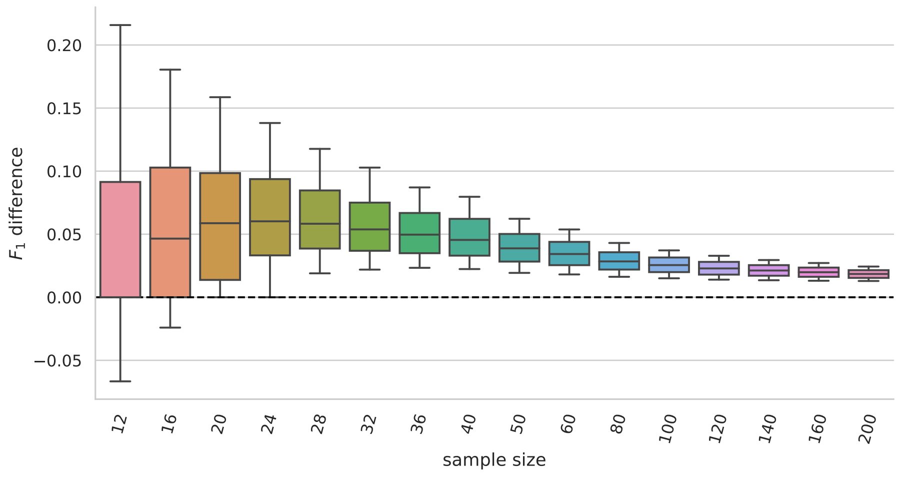
A.1 DISTRIBUTIONS OF DIFFERENCES OF VARIOUS METRICS USING REAL PHENOTYPES



(a) Breast × Colon

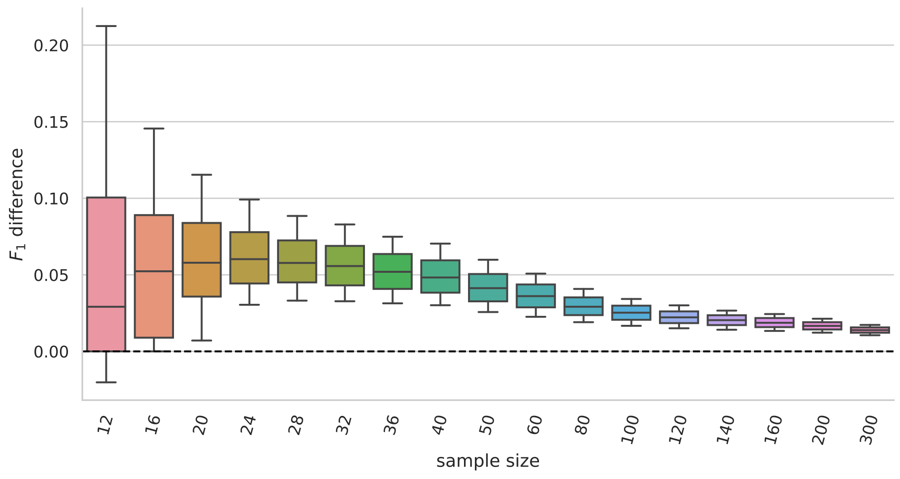


(b) Breast × Kidney

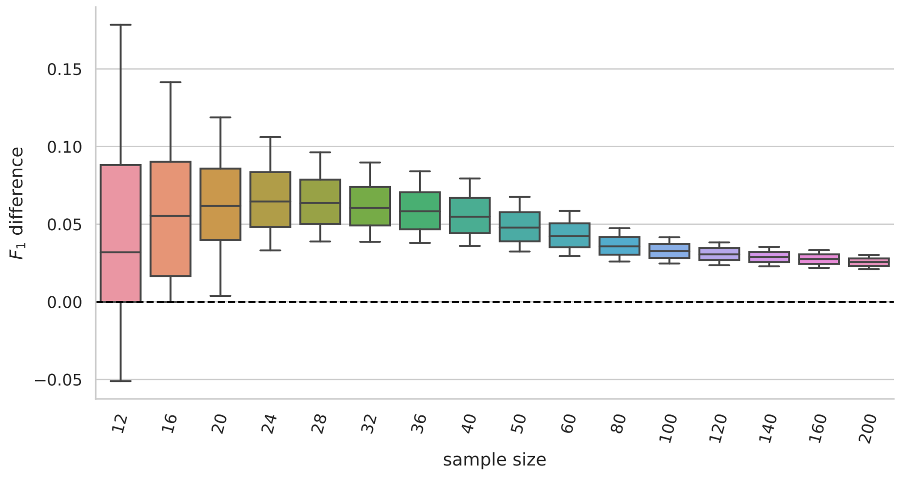


(c) Breast × Lung

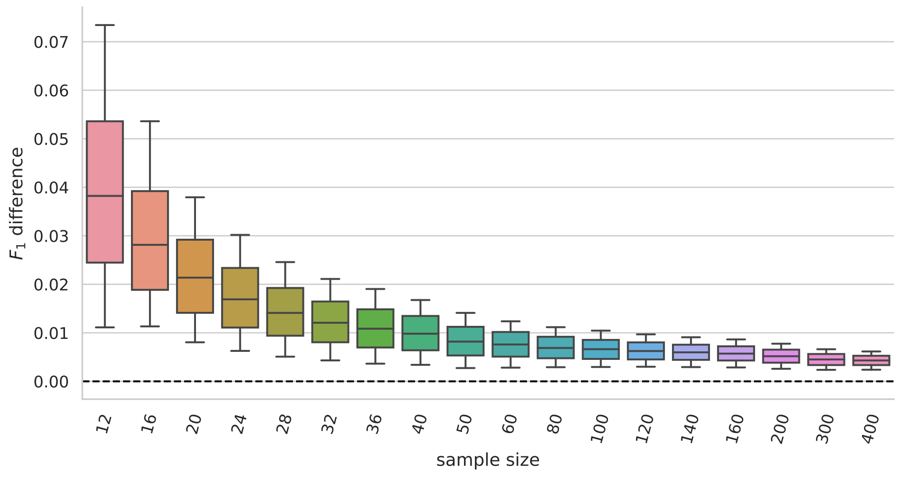
Figure A.1: **Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. The larger variant of Fig. 6.11.



(d) Breast × Ovary

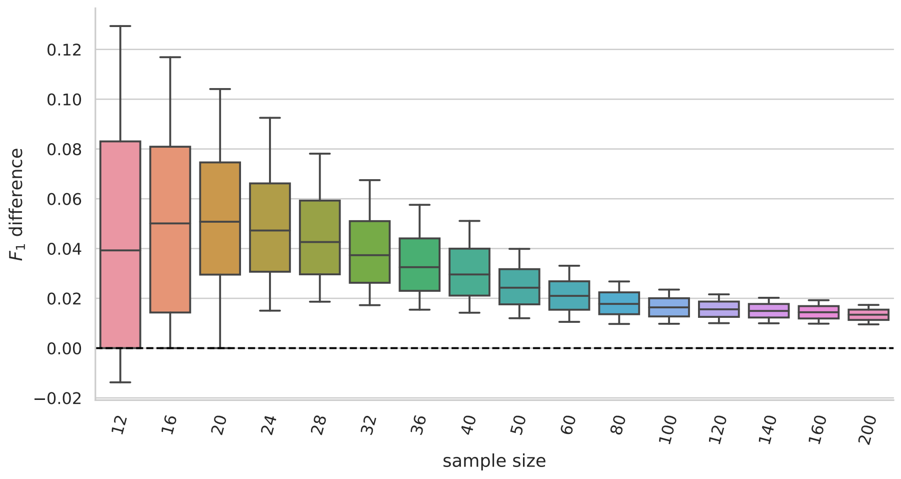


(e) Breast × Uterus

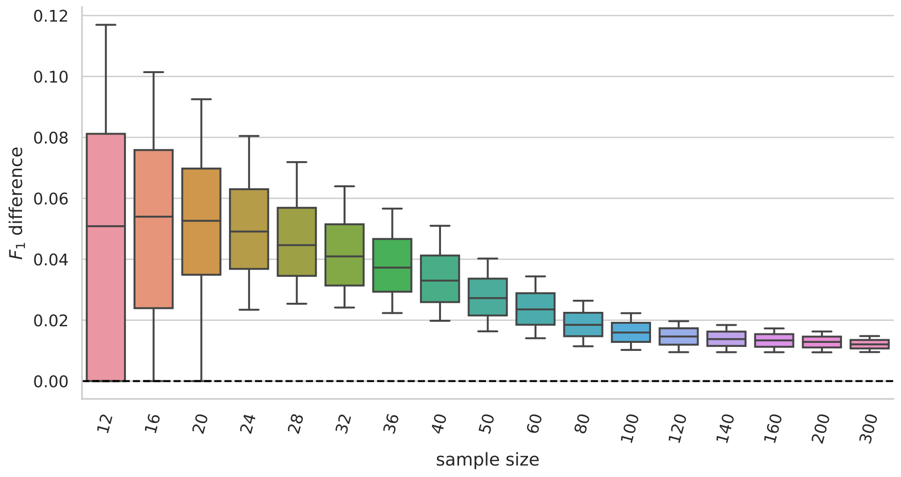


(f) Colon × Kidney

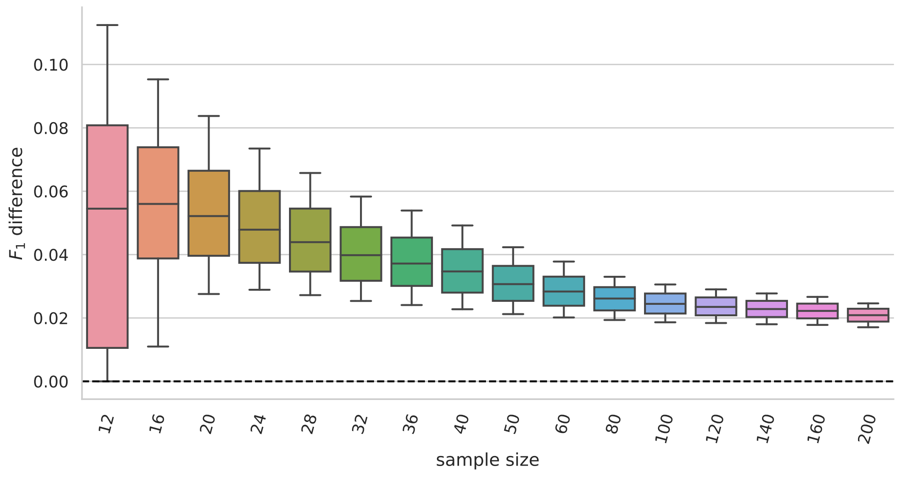
Figure A.1: **(cont.) Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. The larger variant of Fig. 6.11. Continuation of Fig. A.1.



(g) Colon × Lung

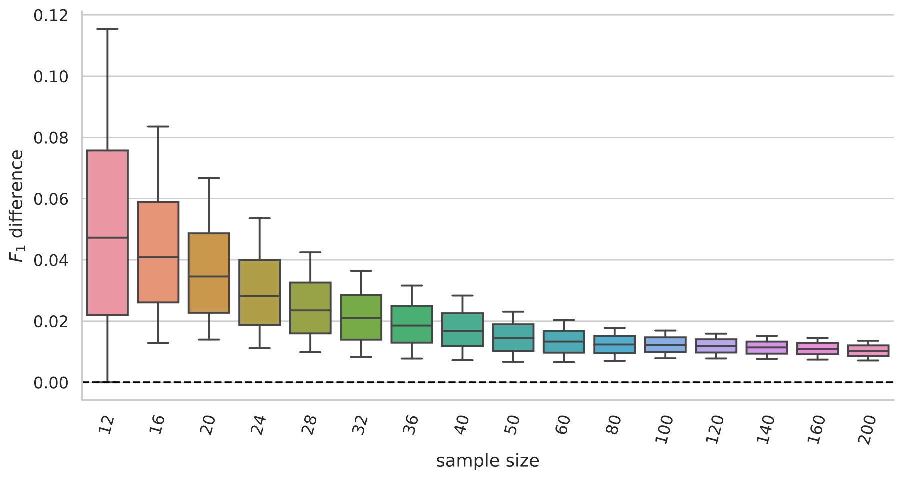


(h) Colon × Ovary

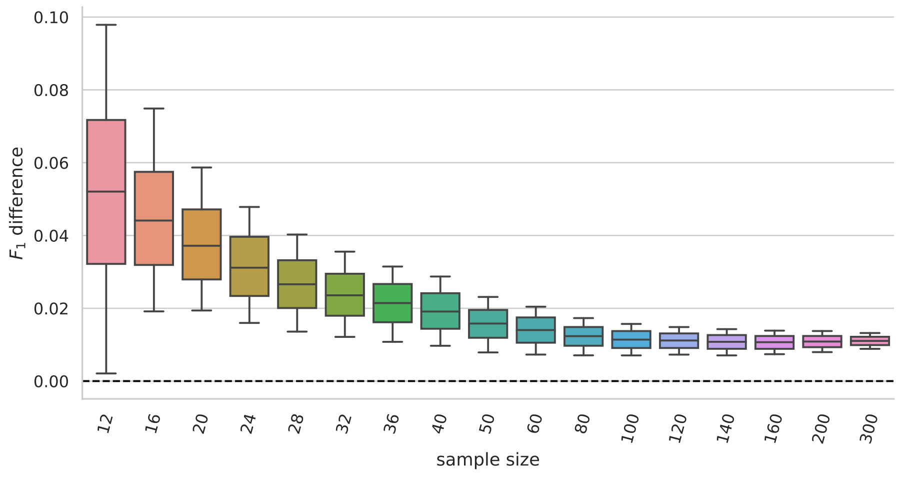


(i) Colon × Uterus

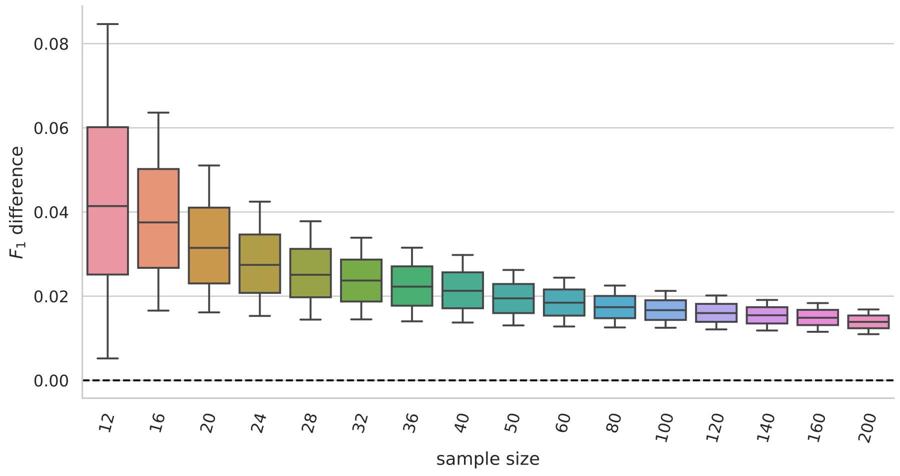
Figure A.1: **(cont.) Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. The larger variant of Fig. 6.11. Continuation of Fig. A.1.



(j) Kidney × Lung

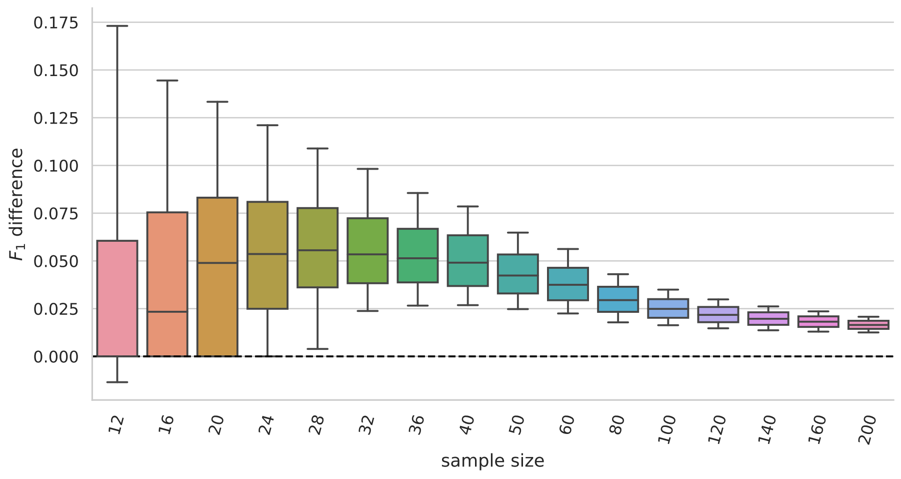


(k) Kidney × Ovary

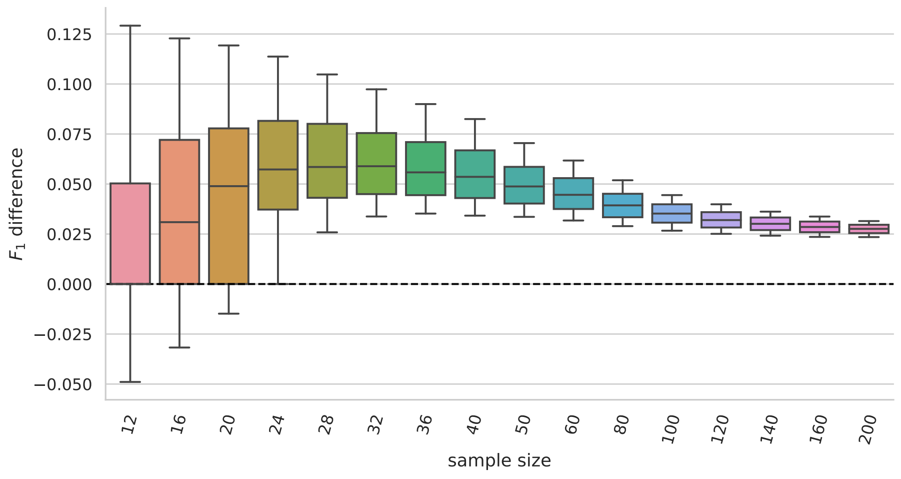


(l) Kidney × Uterus

Figure A.1: **(cont.) Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. The larger variant of Fig. 6.11. Continuation of Fig. A.1.

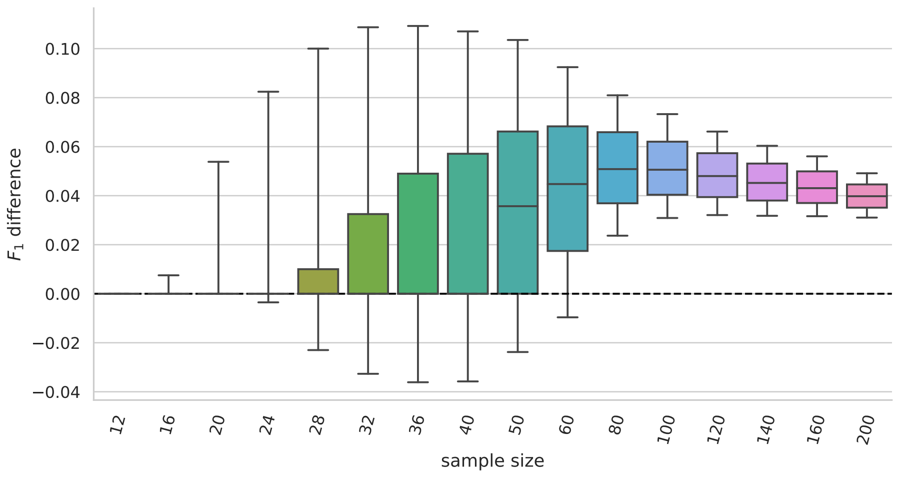


(m) Lung × Ovary



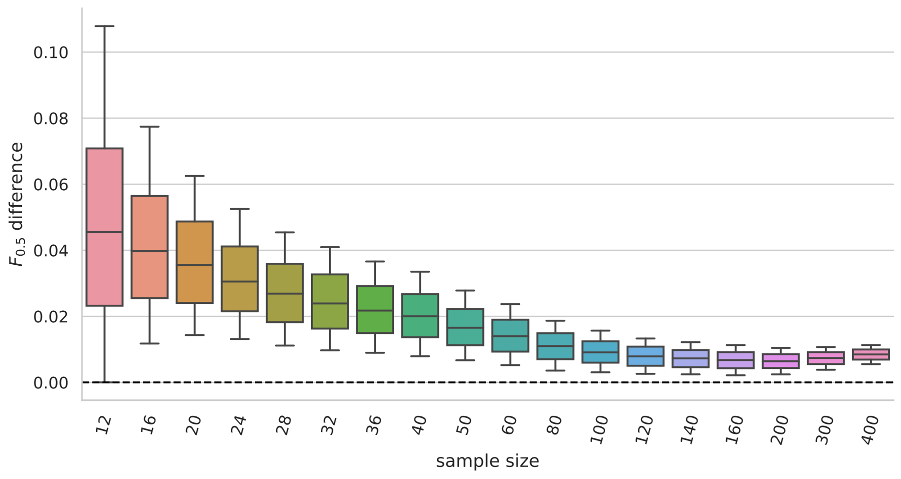
(n) Lung × Uterus

Figure A.1: **(cont.) Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. The larger variant of Fig. 6.11. Continuation of Fig. A.1.

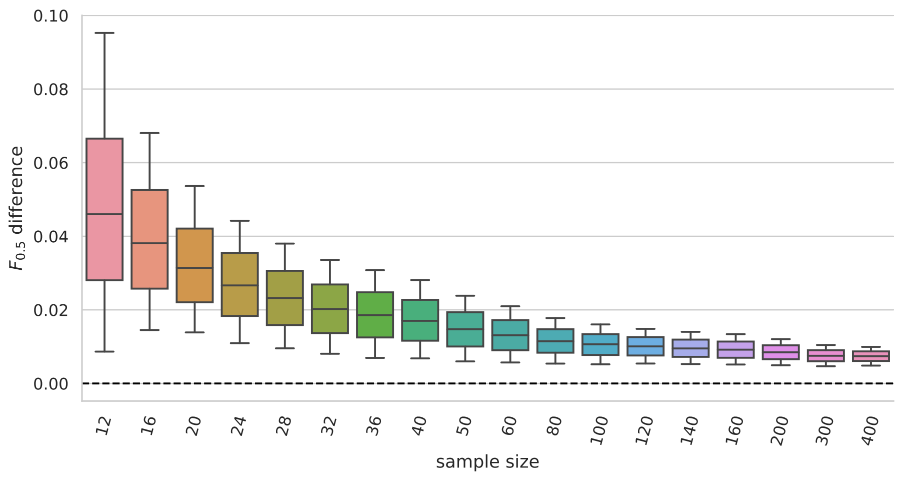


(o) Ovary × Uterus

Figure A.1: **(cont.) Distributions of F_1 score differences for the real phenotypes** Distributions of the F_1 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. The larger variant of Fig. 6.11. Continuation of Fig. A.1.

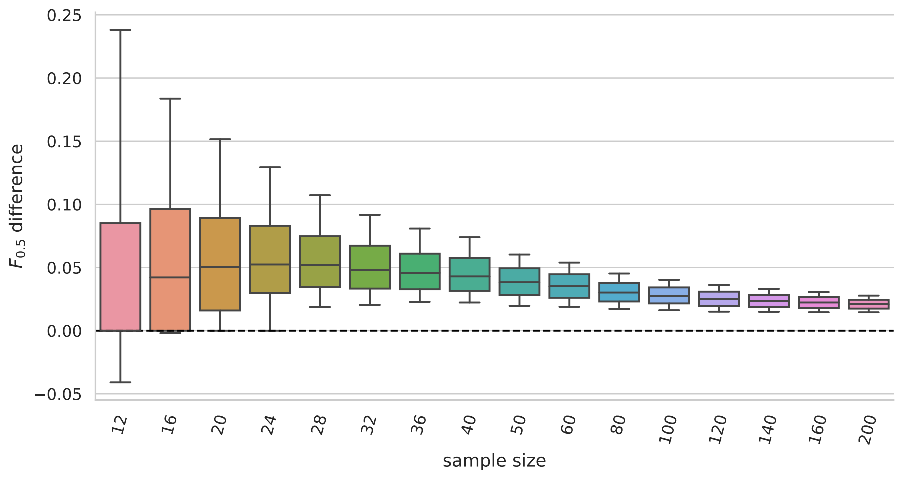


(a) Breast × Colon

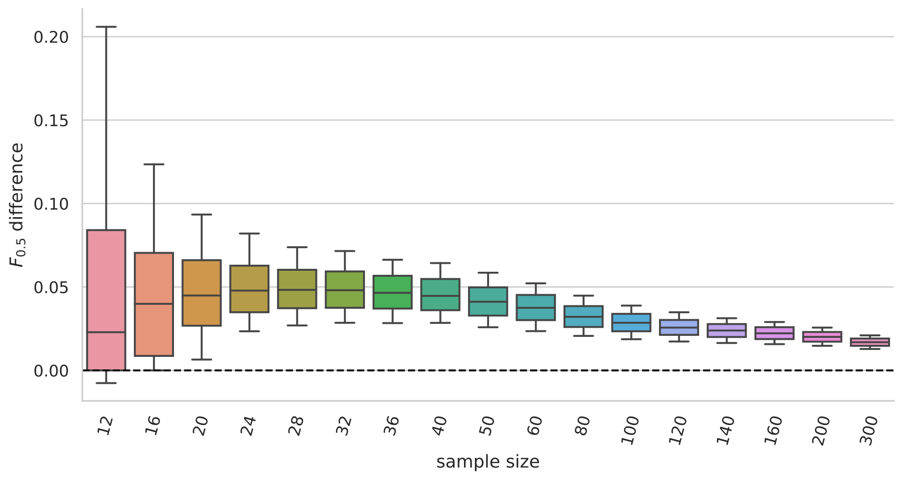


(b) Breast × Kidney

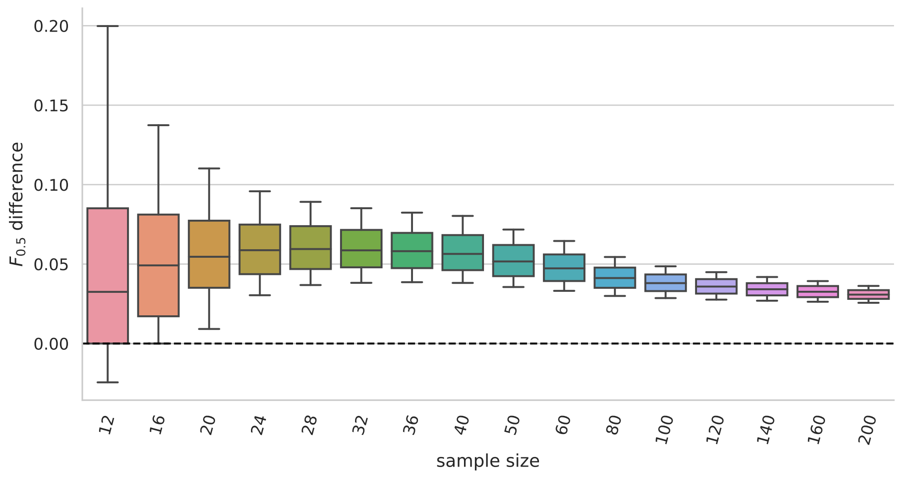
Figure A.2: **Distributions of $F_{0.5}$ score differences for the real phenotypes** Distributions of the $F_{0.5}$ score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles.



(c) Breast × Lung

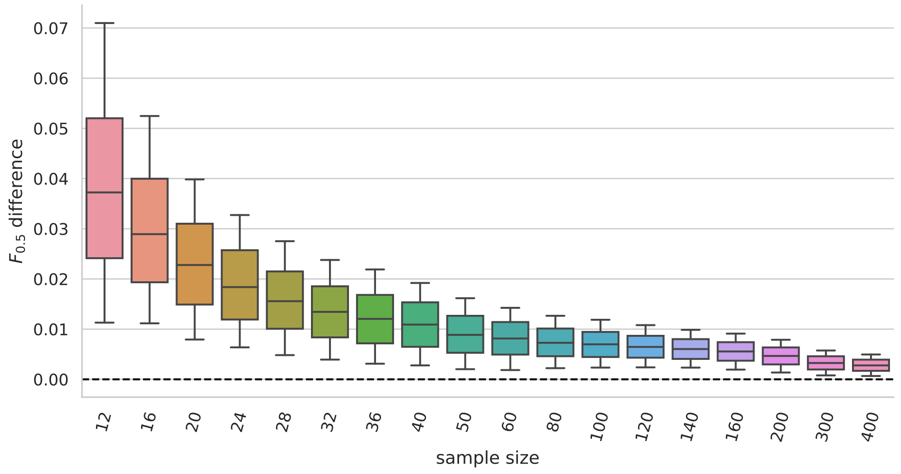


(d) Breast × Ovary

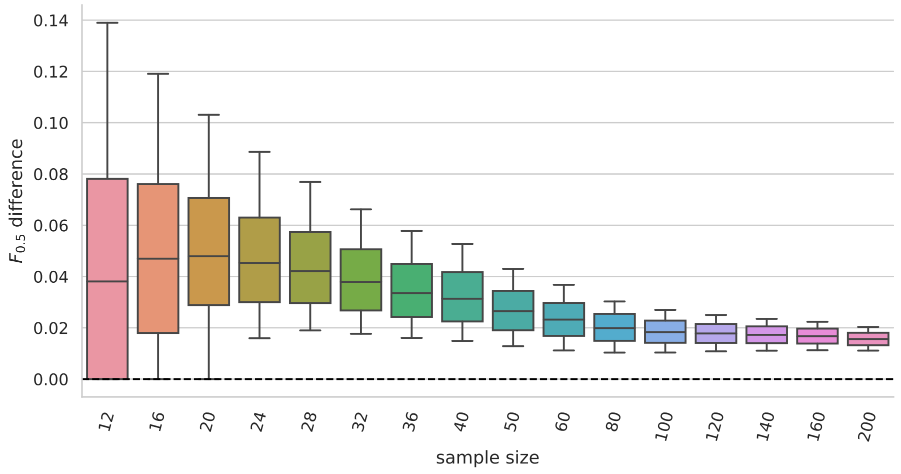


(e) Breast × Uterus

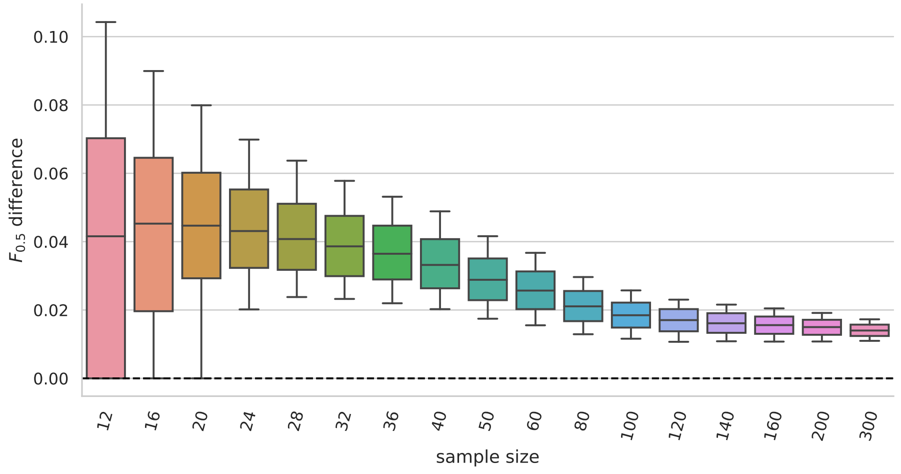
Figure A.2: **(cont.) Distributions of $F_{0.5}$ score differences for the real phenotypes** Distributions of the $F_{0.5}$ score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.2.



(f) Colon × Kidney

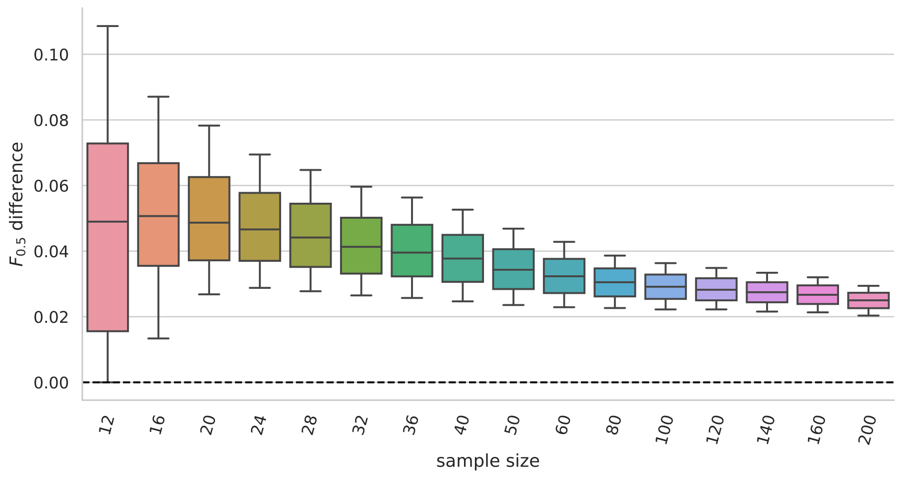


(g) Colon × Lung

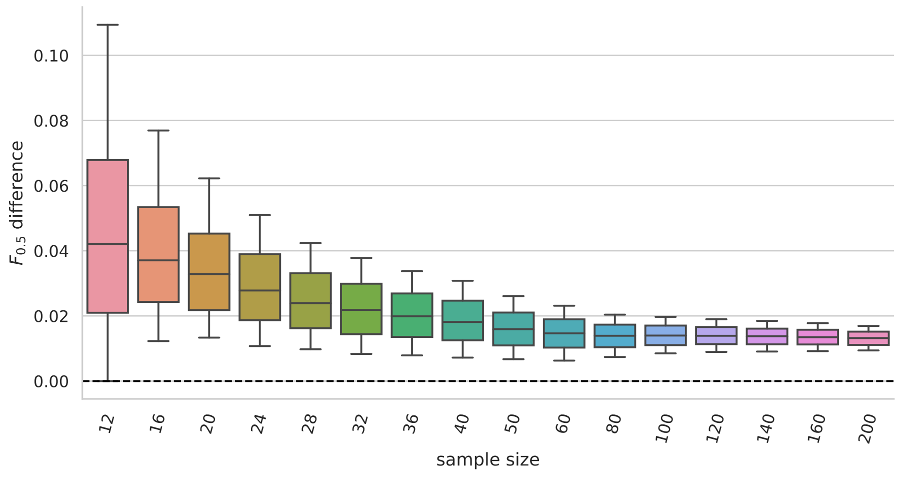


(h) Colon × Ovary

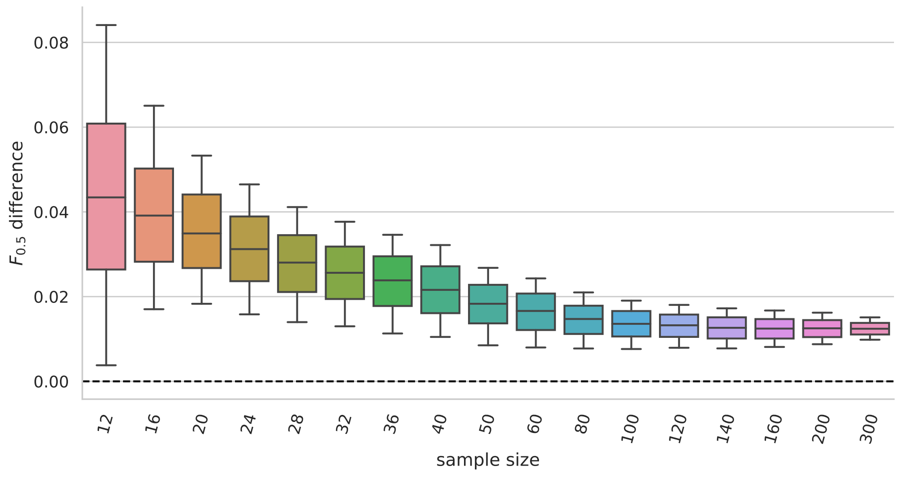
Figure A.2: **(cont.) Distributions of $F_{0.5}$ score differences for the real phenotypes** Distributions of the $F_{0.5}$ score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.2.



(i) Colon × Uterus

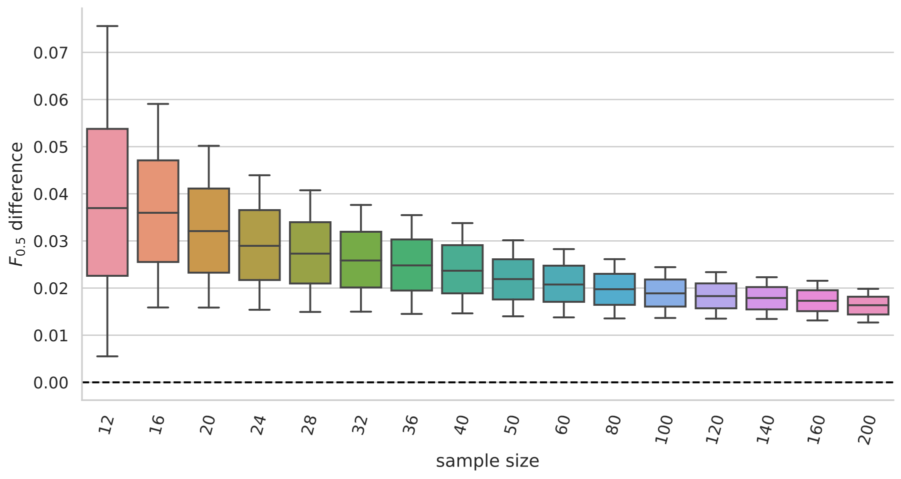


(j) Kidney × Lung

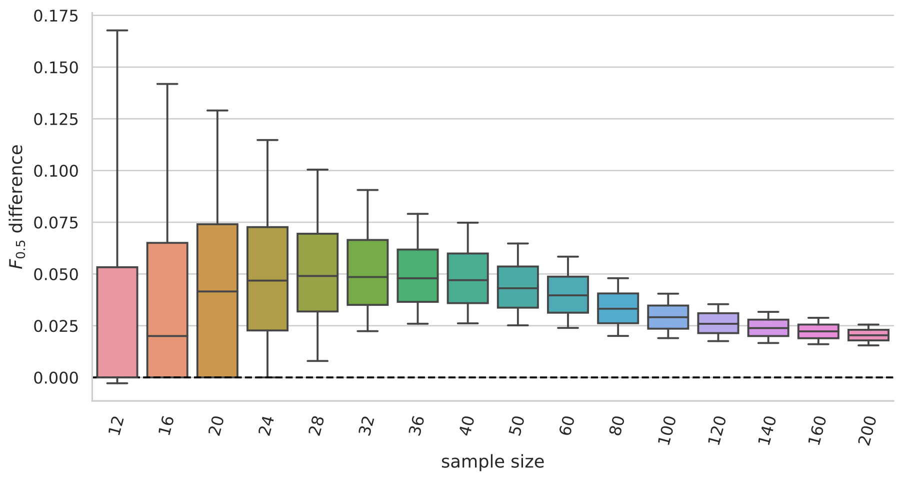


(k) Kidney × Ovary

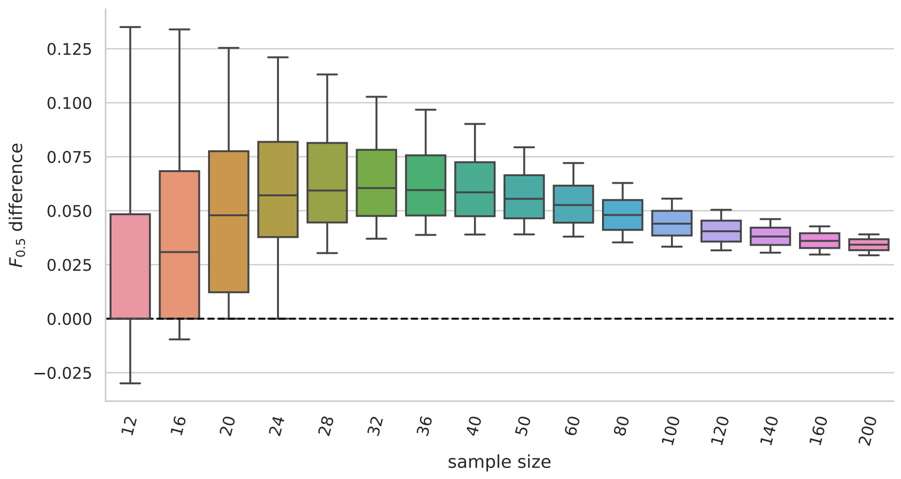
Figure A.2: **(cont.) Distributions of $F_{0.5}$ score differences for the real phenotypes** Distributions of the $F_{0.5}$ score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.2.



(l) Kidney × Uterus

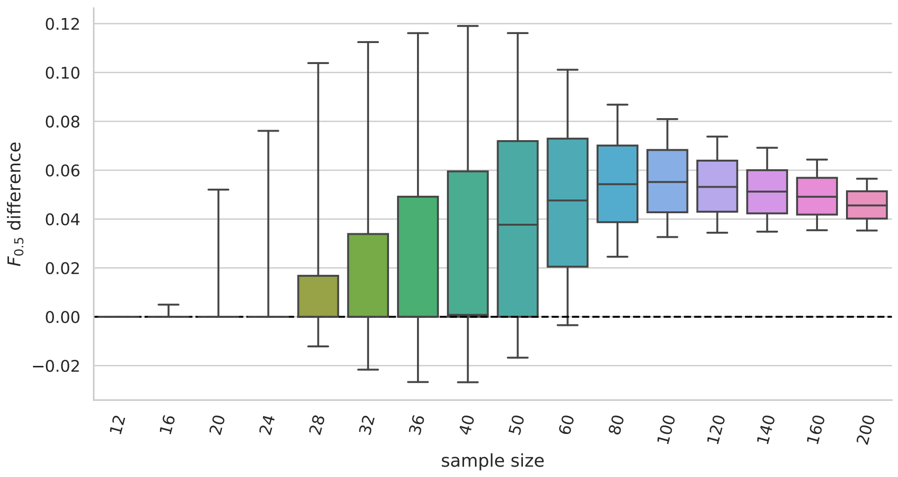


(m) Lung × Ovary



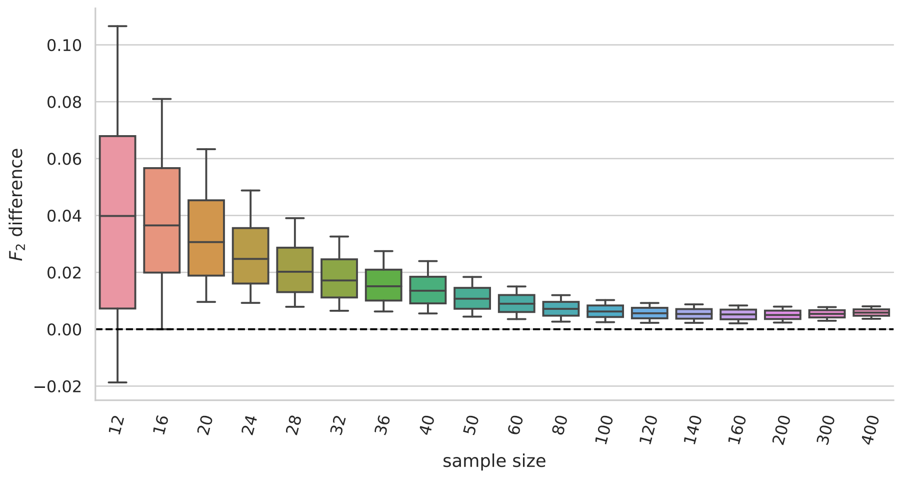
(n) Lung × Uterus

Figure A.2: **(cont.) Distributions of $F_{0.5}$ score differences for the real phenotypes** Distributions of the $F_{0.5}$ score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.2.

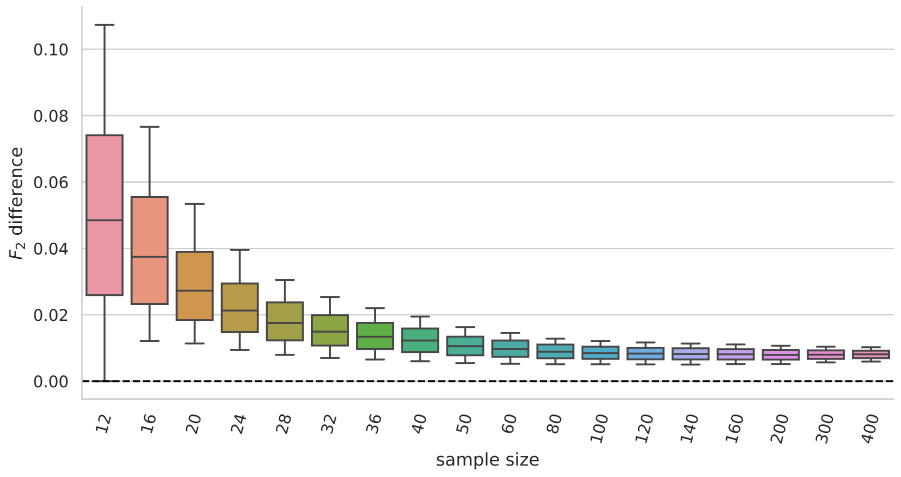


(o) Ovary × Uterus

Figure A.2: **(cont.) Distributions of $F_{0.5}$ score differences for the real phenotypes** Distributions of the $F_{0.5}$ score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.2.

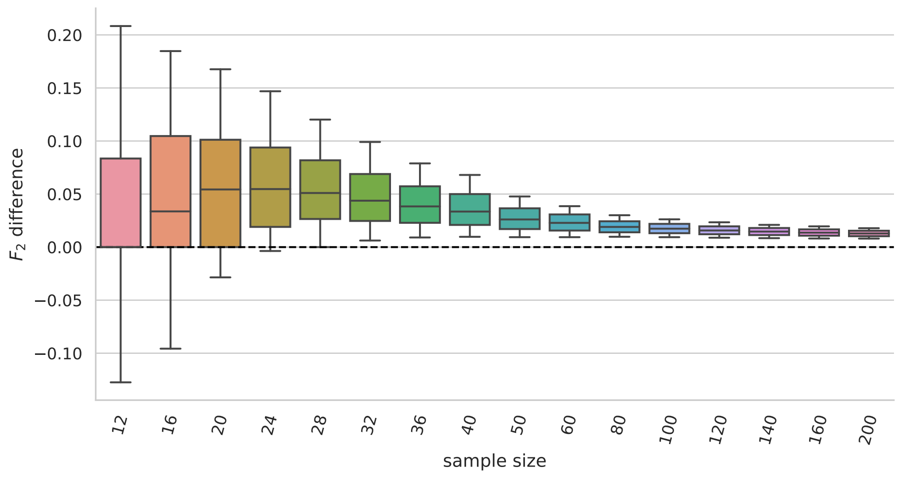


(a) Breast × Colon

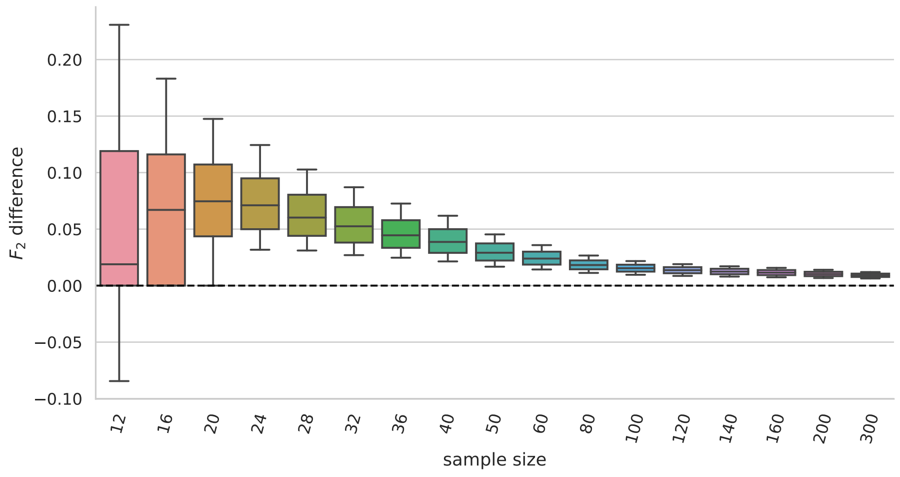


(b) Breast × Kidney

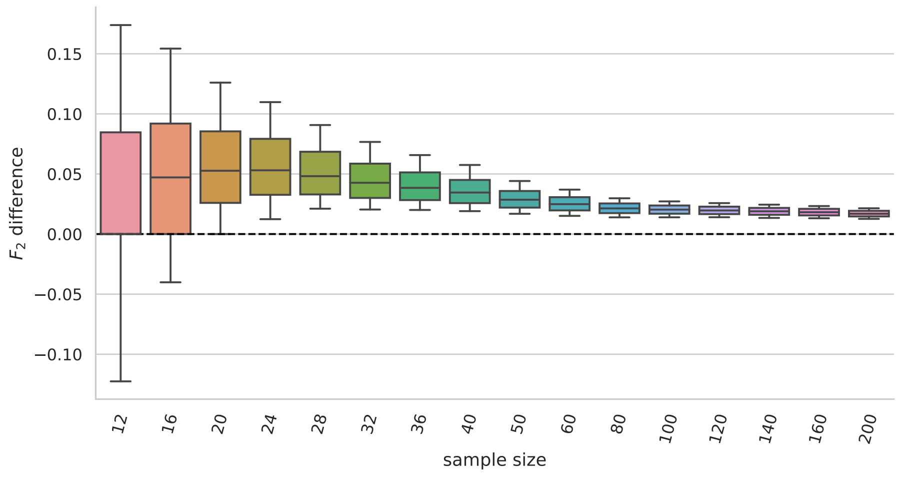
Figure A.3: **Distributions of F_2 score differences for the real phenotypes** Distributions of the F_2 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles.



(c) Breast × Lung

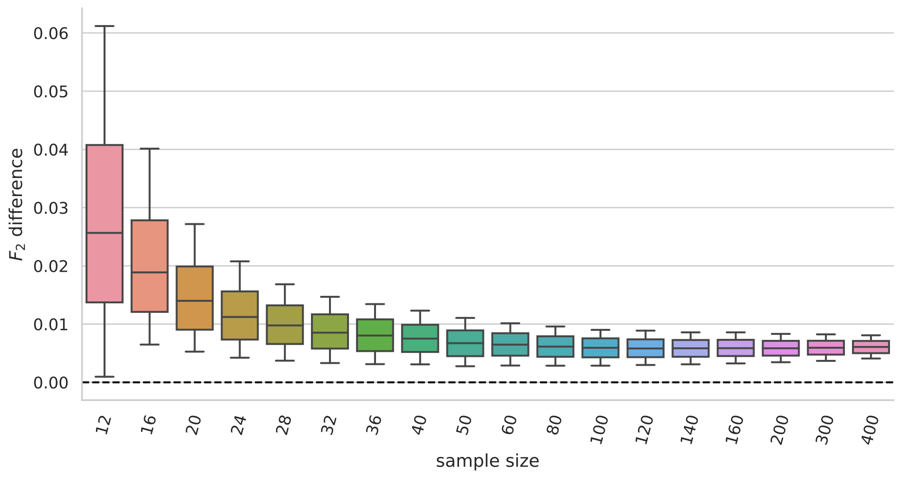


(d) Breast × Ovary

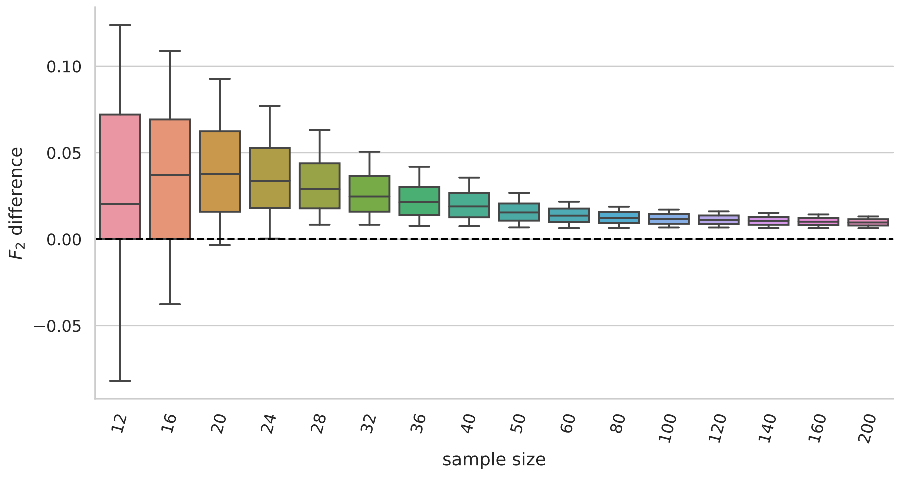


(e) Breast × Uterus

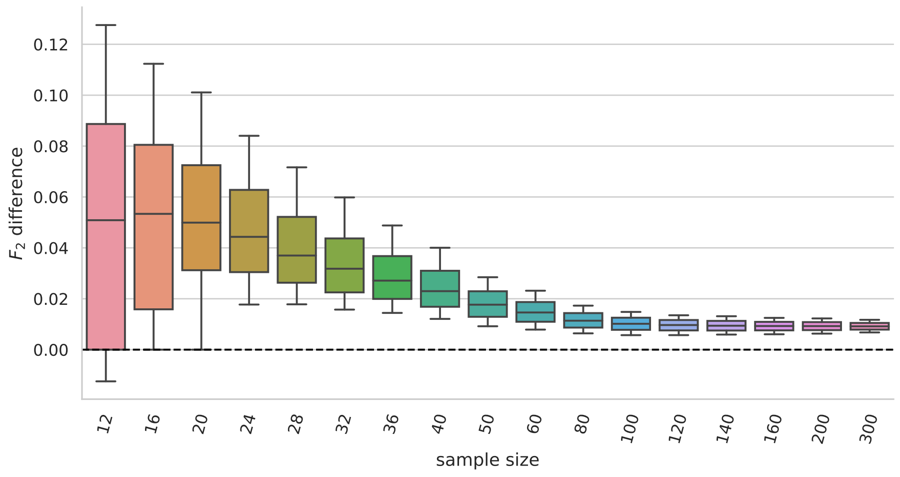
Figure A.3: **(cont.) Distributions of F_2 score differences for the real phenotypes** Distributions of the F_2 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.3.



(f) Colon × Kidney

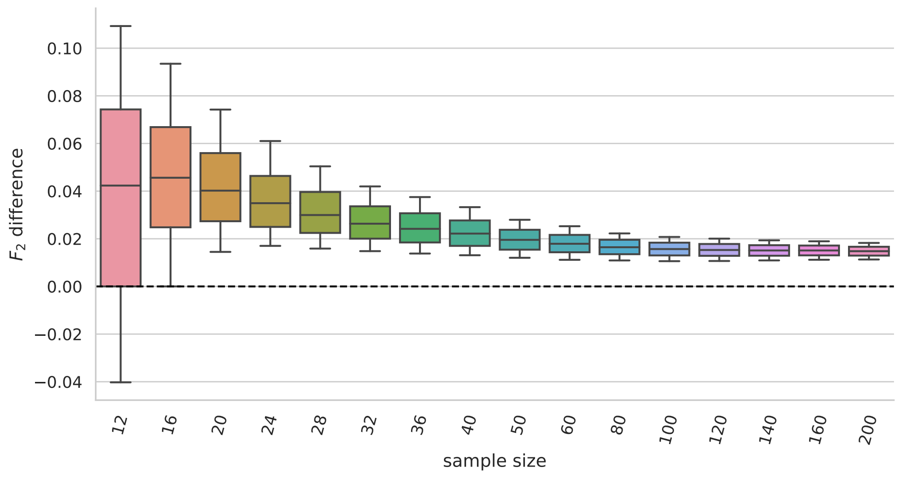


(g) Colon × Lung

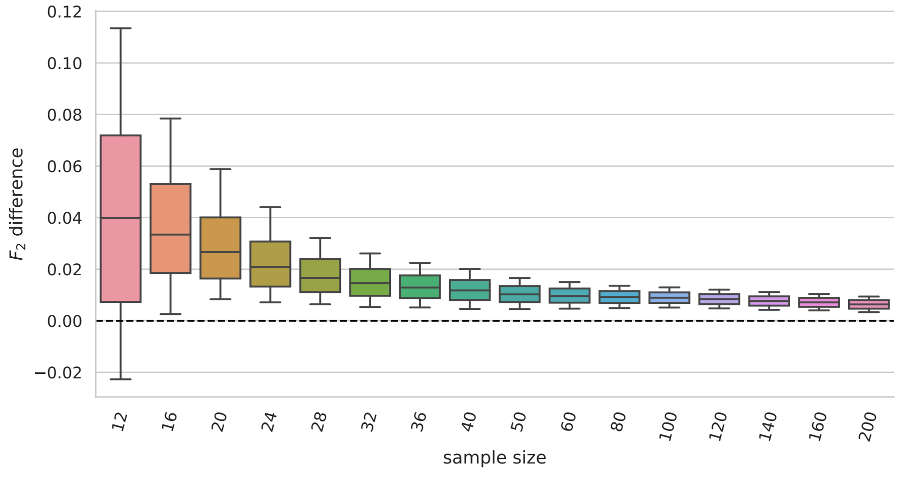


(h) Colon × Ovary

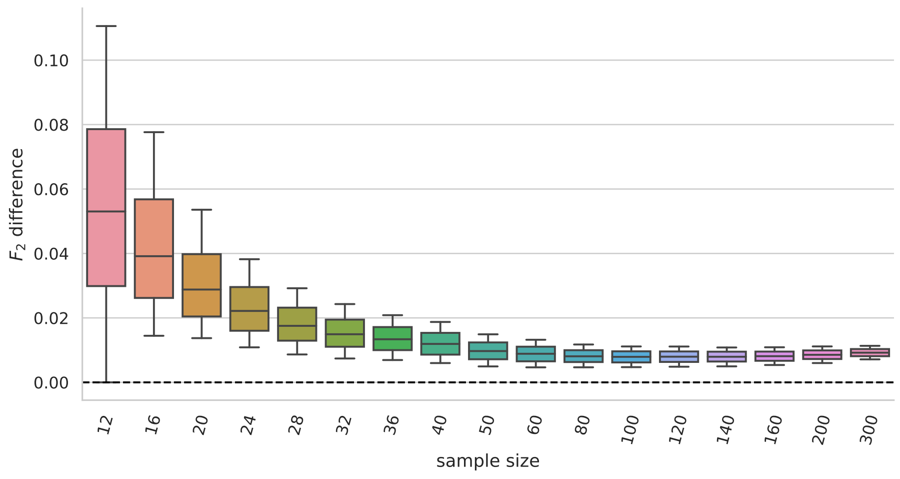
Figure A.3: **(cont.) Distributions of F_2 score differences for the real phenotypes** Distributions of the F_2 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.3.



(i) Colon × Uterus

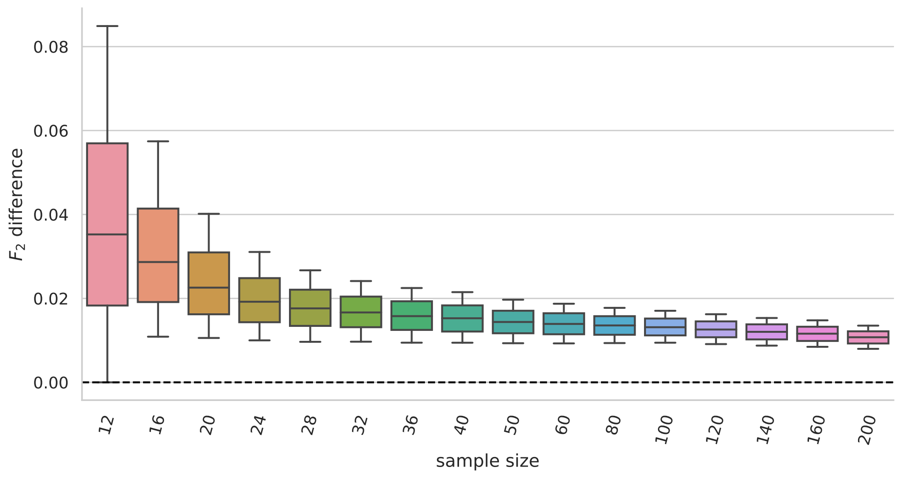


(j) Kidney × Lung

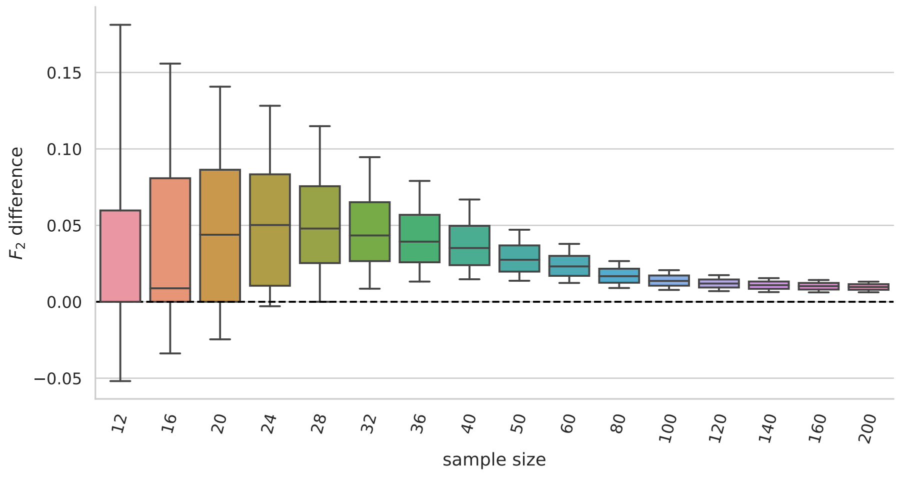


(k) Kidney × Ovary

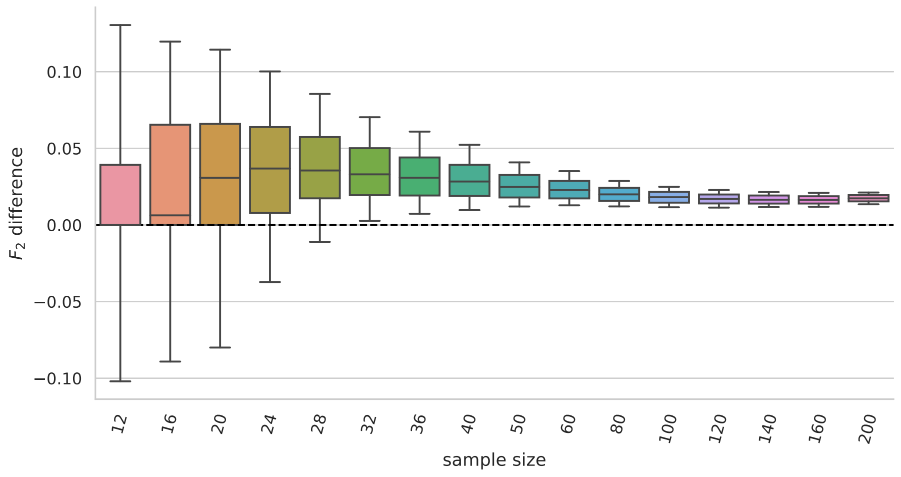
Figure A.3: **(cont.) Distributions of F_2 score differences for the real phenotypes** Distributions of the F_2 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.3.



(l) Kidney × Uterus

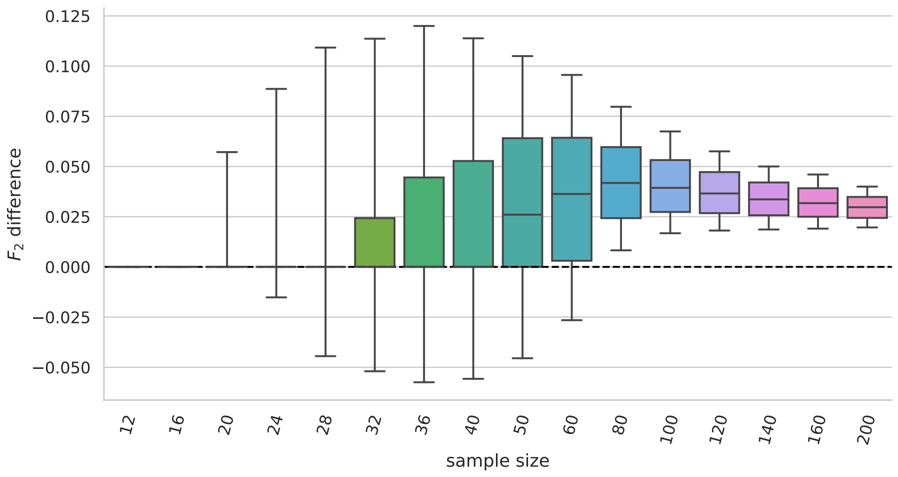


(m) Lung × Ovary



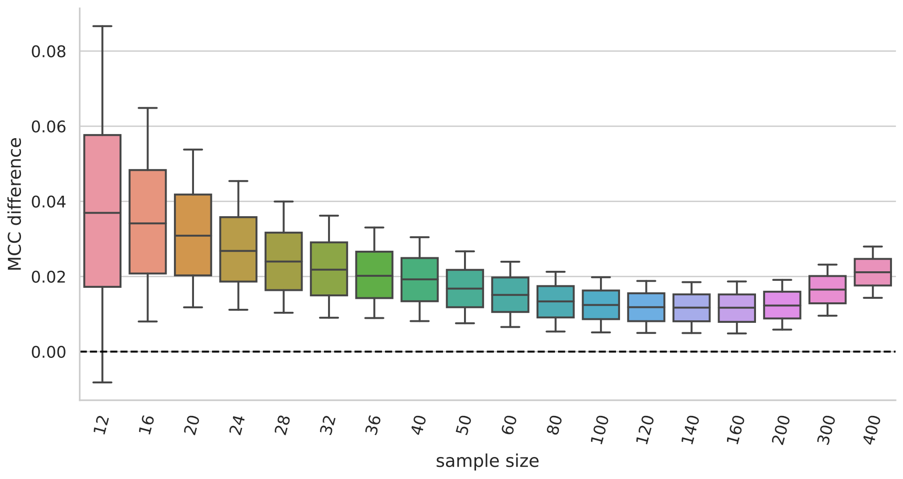
(n) Lung × Uterus

Figure A.3: **(cont.) Distributions of F_2 score differences for the real phenotypes** Distributions of the F_2 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.3.

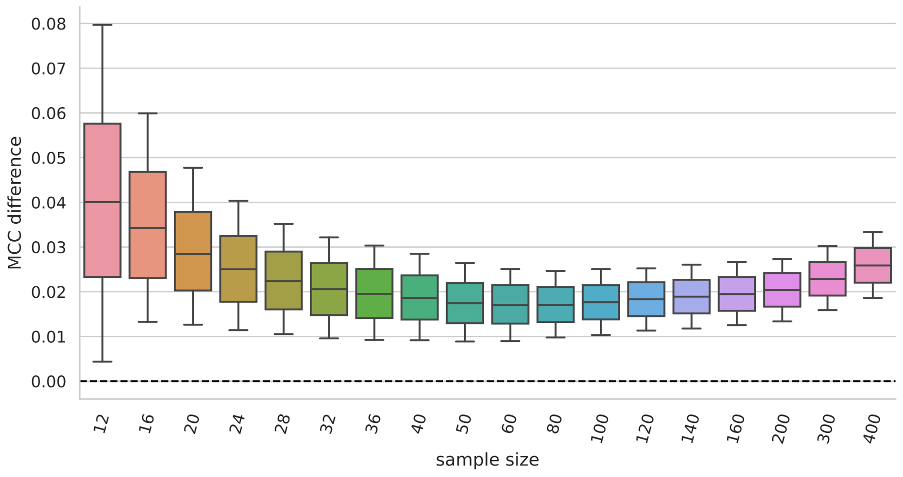


(o) Ovary × Uterus

Figure A.3: **(cont.) Distributions of F_2 score differences for the real phenotypes** Distributions of the F_2 score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.3.

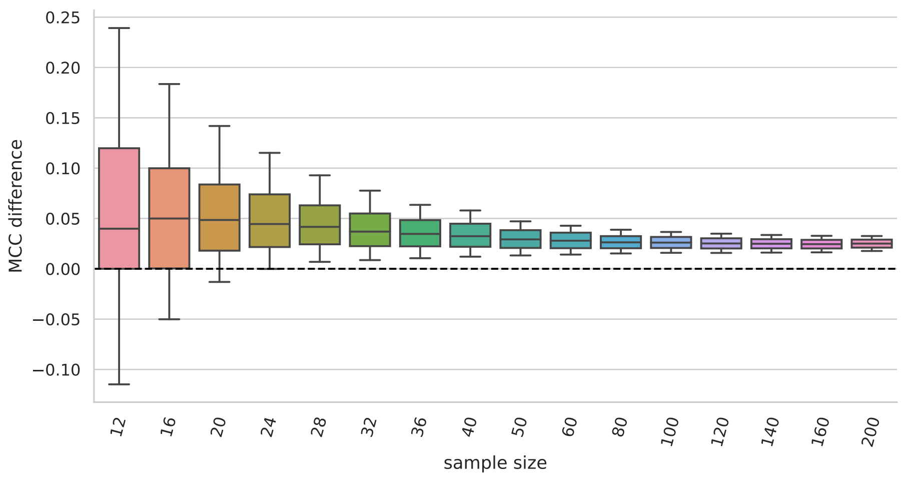


(a) Breast × Colon

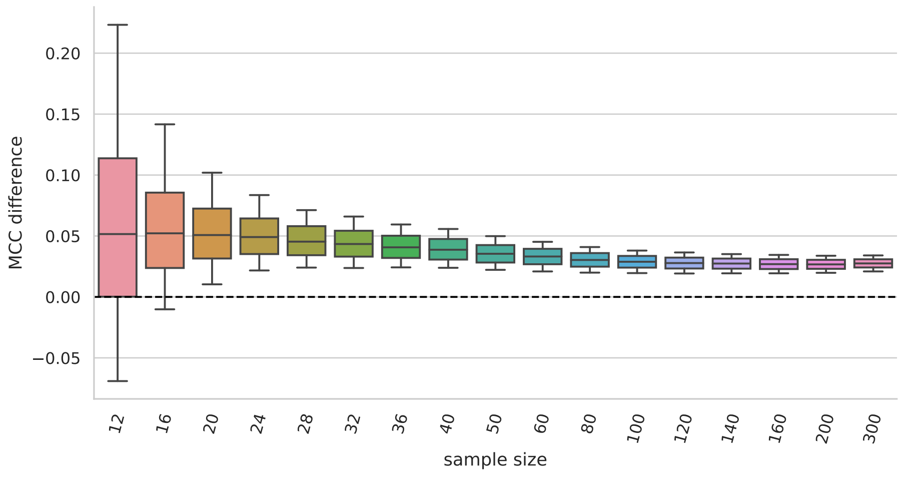


(b) Breast × Kidney

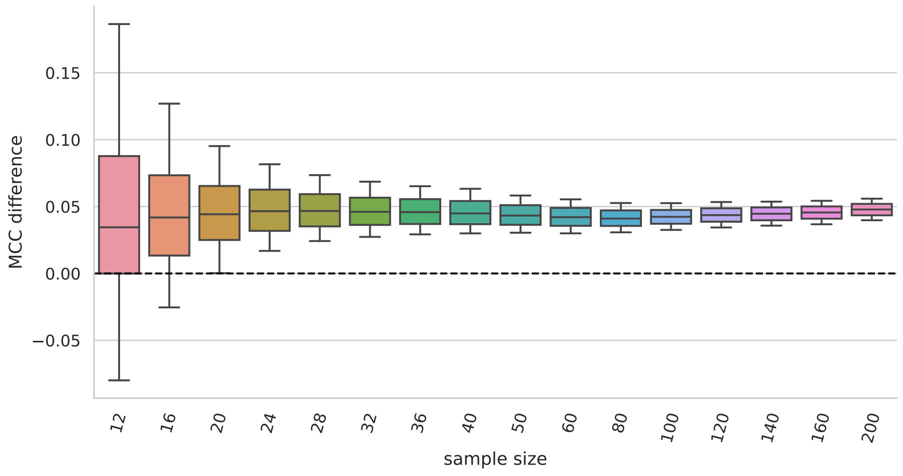
Figure A.4: **Distributions of MCC differences for the real phenotypes**
Distributions of the MCC pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles.



(c) Breast × Lung

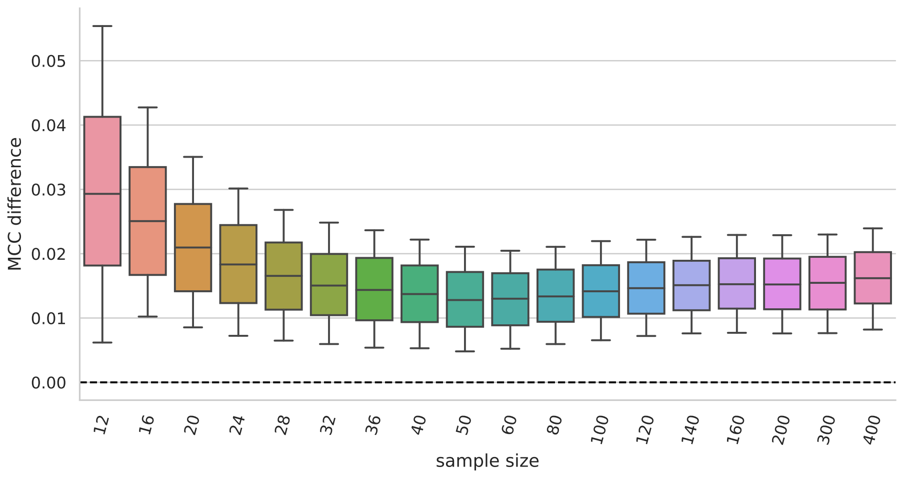


(d) Breast × Ovary

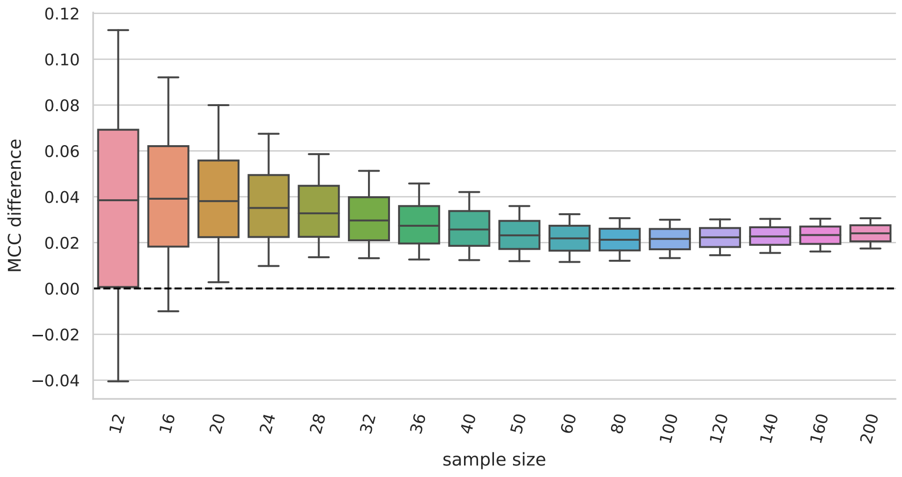


(e) Breast × Uterus

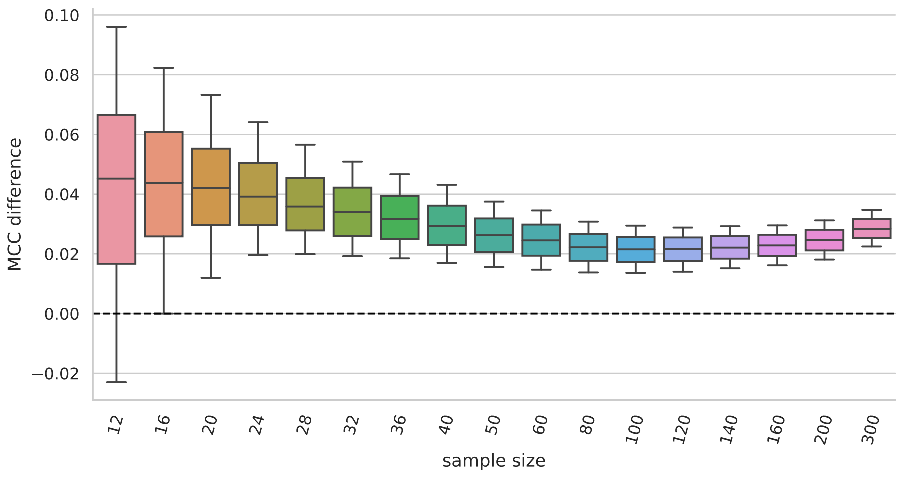
Figure A.4: **(cont.) Distributions of MCC differences for the real phenotypes** Distributions of the MCC score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.4.



(f) Colon × Kidney

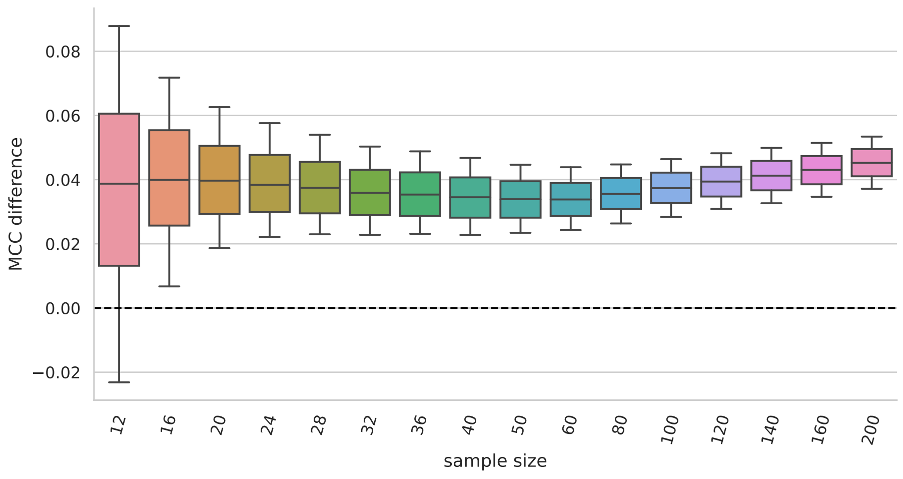


(g) Colon × Lung

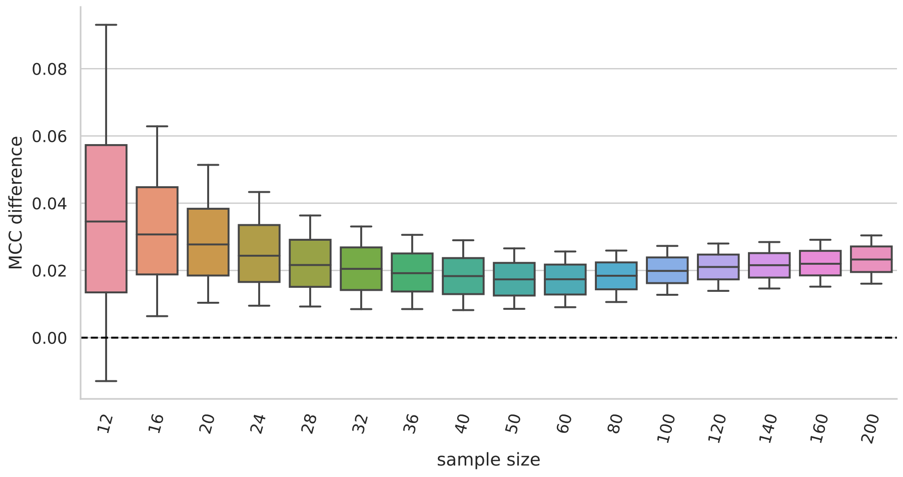


(h) Colon × Ovary

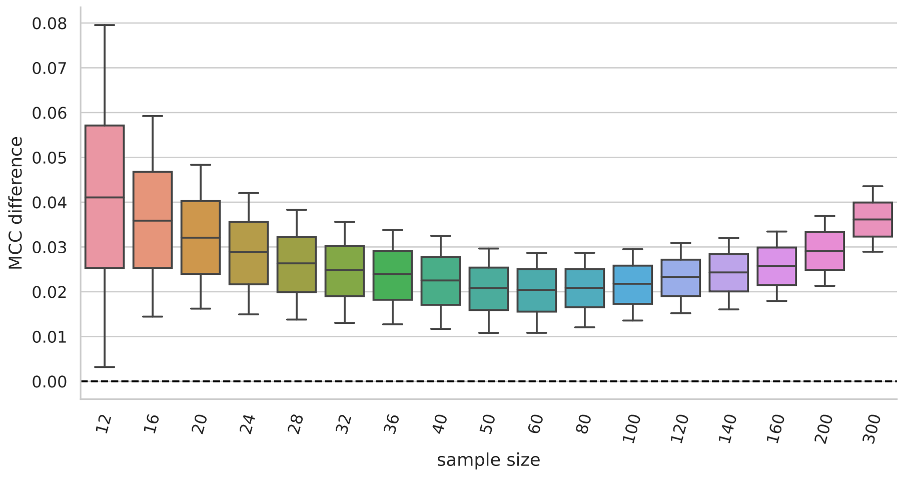
Figure A.4: **(cont.) Distributions of MCC differences for the real phenotypes** Distributions of the MCC score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.4.



(i) Colon × Uterus

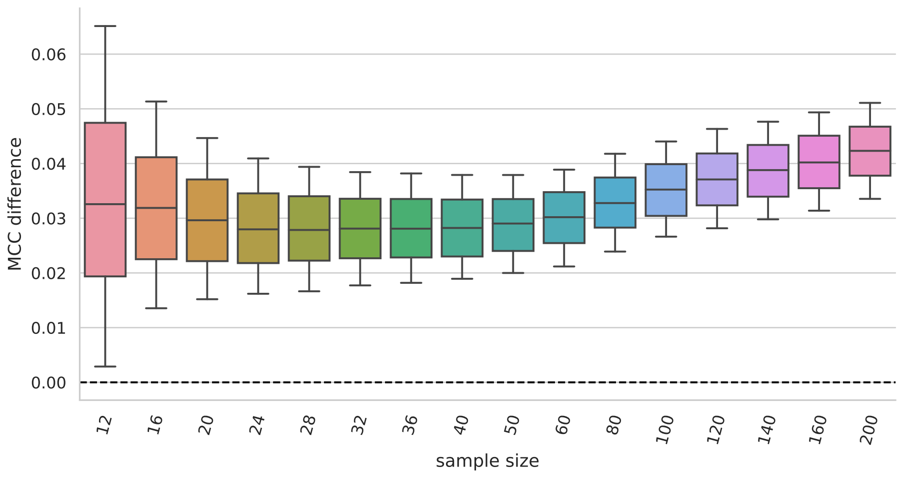


(j) Kidney × Lung

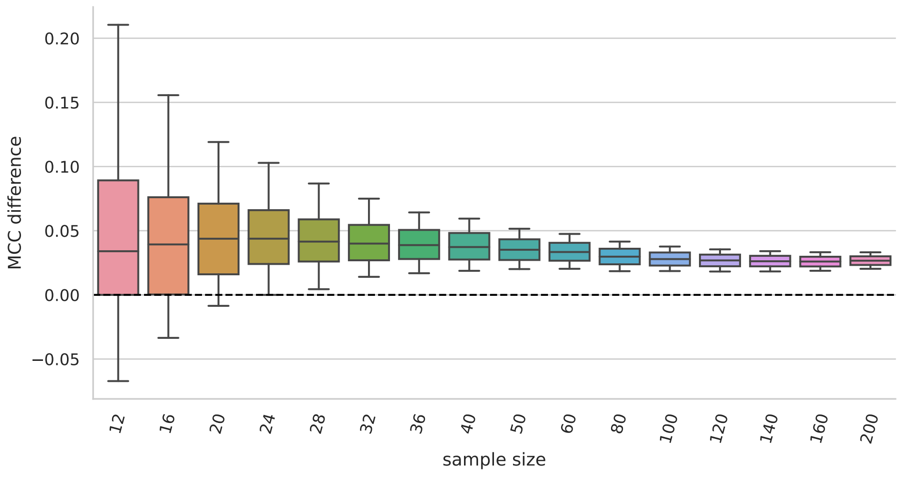


(k) Kidney × Ovary

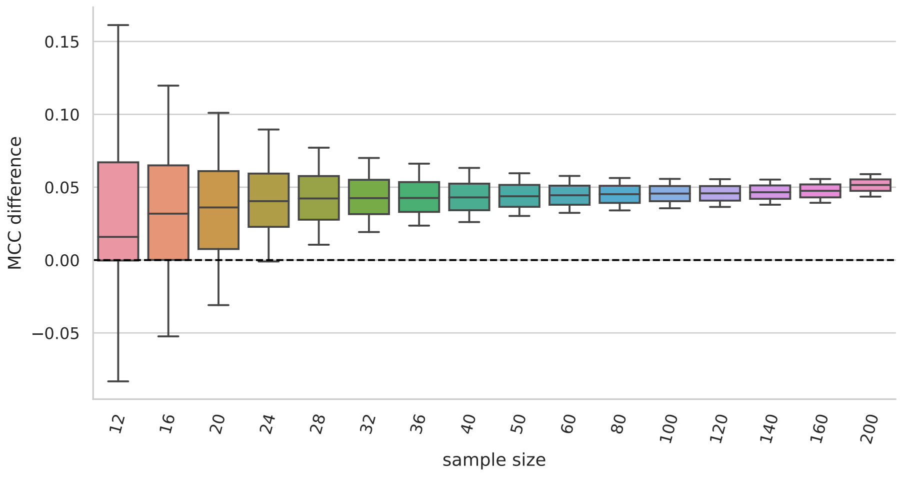
Figure A.4: **(cont.) Distributions of MCC differences for the real phenotypes** Distributions of the MCC score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.4.



(l) Kidney × Uterus

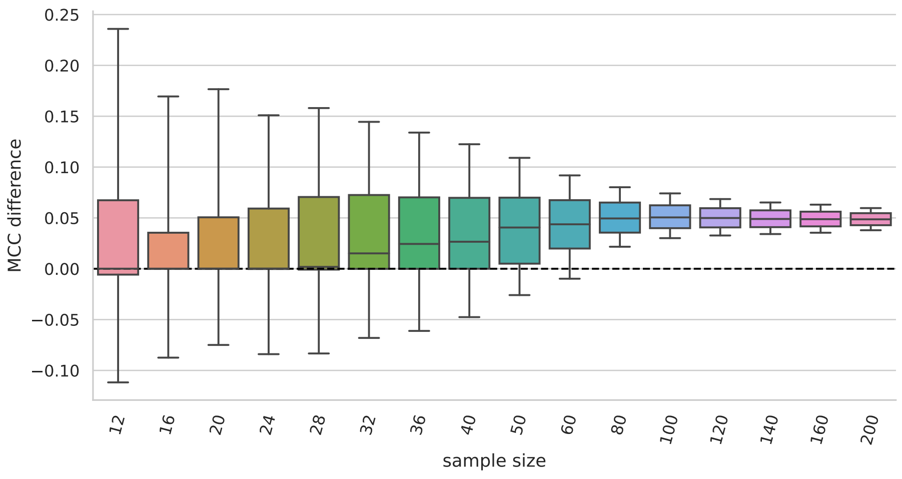


(m) Lung × Ovary



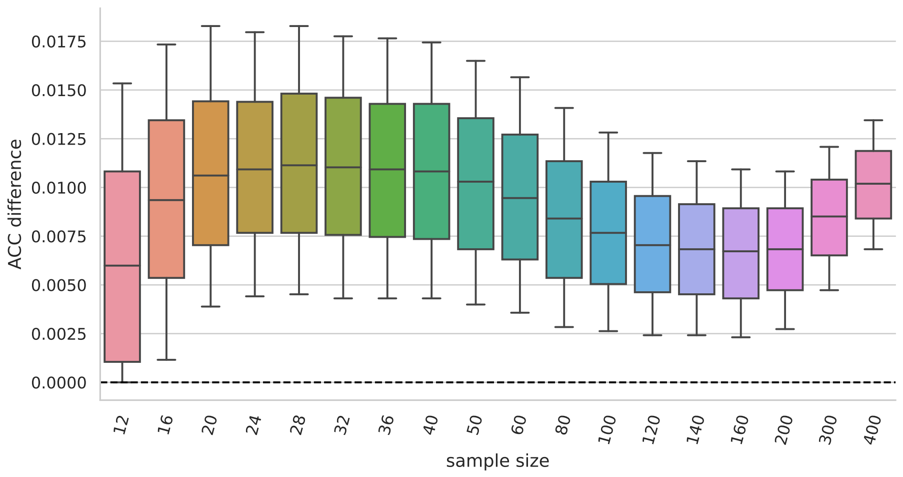
(n) Lung × Uterus

Figure A.4: **(cont.) Distributions of MCC differences for the real phenotypes** Distributions of the MCC score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.4.

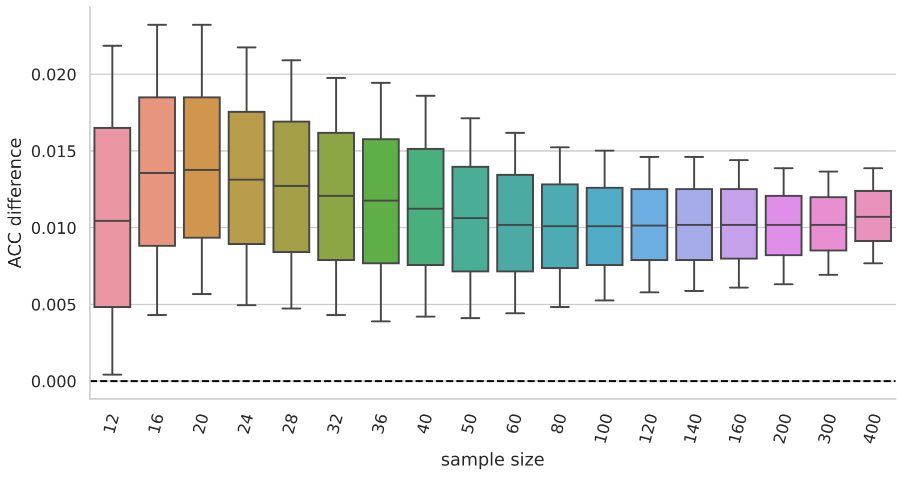


(o) Ovary × Uterus

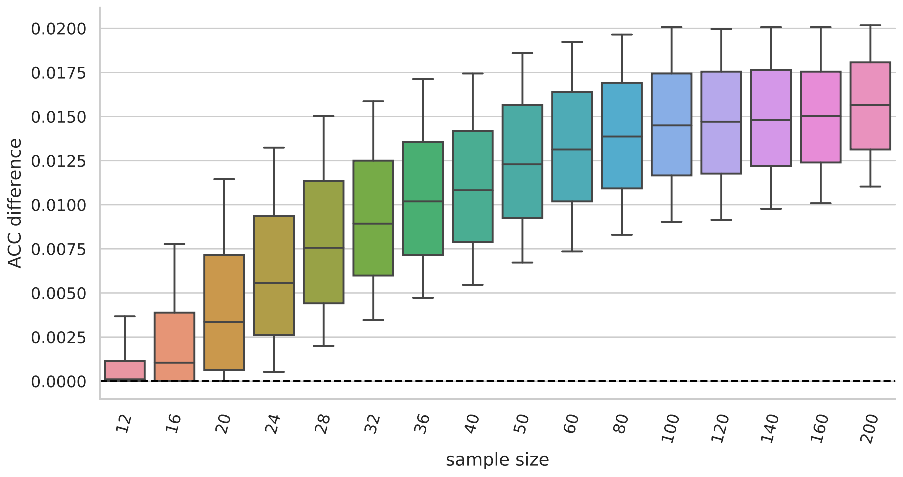
Figure A.4: **(cont.) Distributions of MCC differences for the real phenotypes** Distributions of the MCC score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Continuation of Fig. A.4.



(a) Breast × Colon

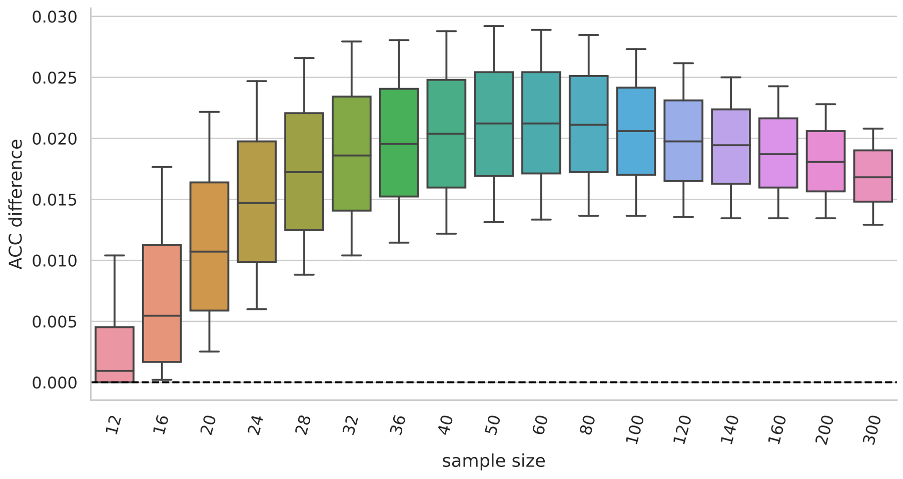


(b) Breast × Kidney

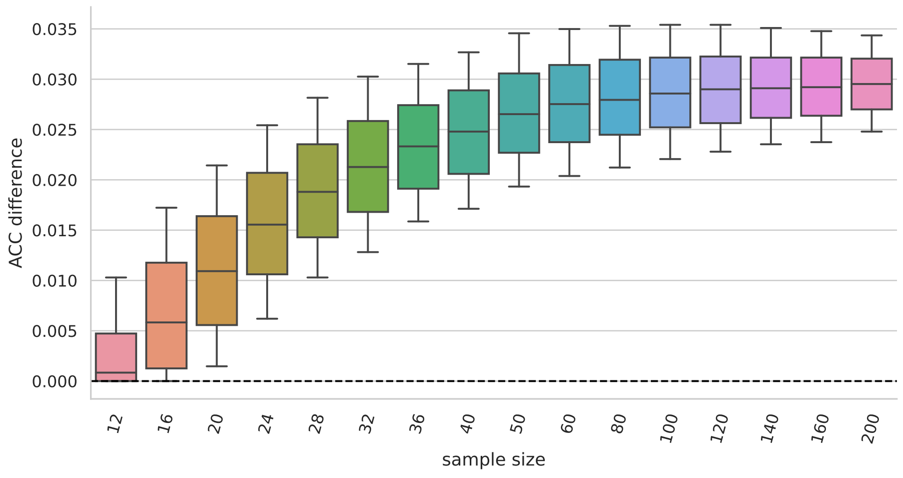


(c) Breast × Lung

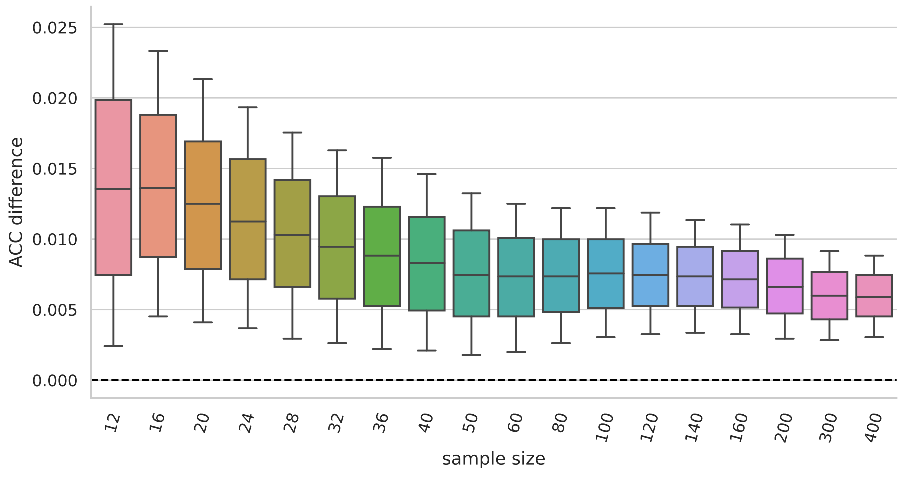
Figure A.5: **Distributions of accuracy differences for the real phenotypes** Distributions of the accuracy pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Note that unlike MCC shown in Fig. A.4, the accuracy does not take the class imbalance into account the accuracy, F_1, $F_{0.5}$, F_2, and MCC scores.



(d) Breast × Ovary

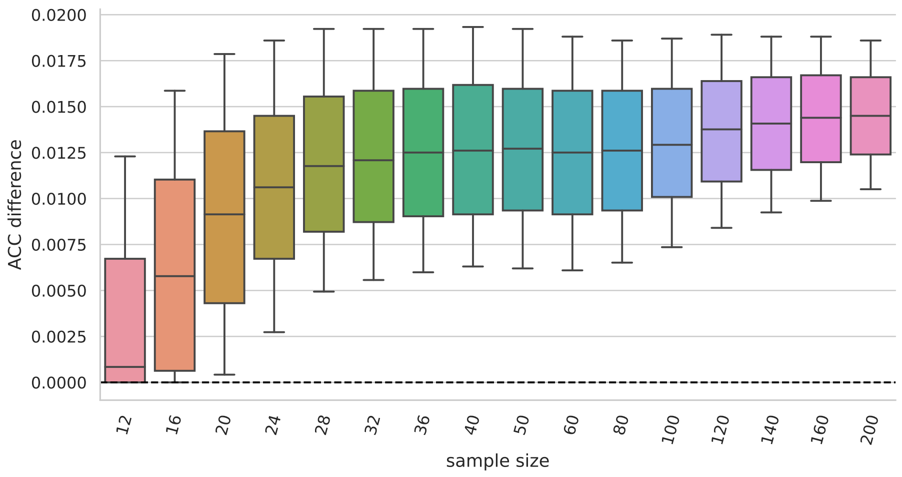


(e) Breast × Uterus

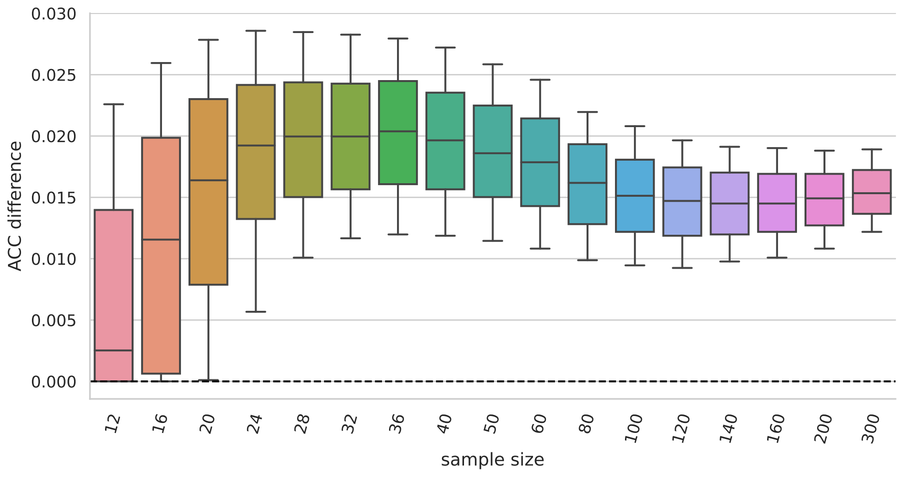


(f) Colon × Kidney

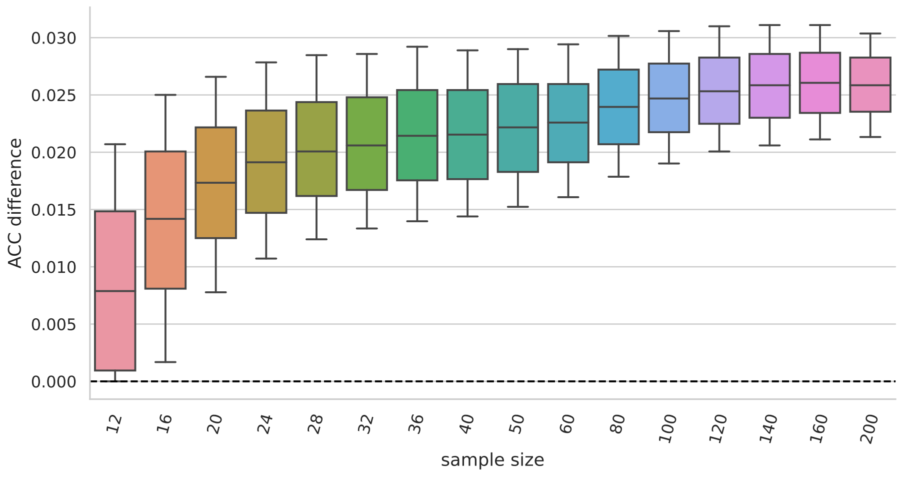
Figure A.5: **(cont.) Distributions of accuracy differences for the real phenotypes** Distributions of the accuracy score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Note that unlike MCC shown in Fig. A.4, the accuracy does not take the class imbalance into account. Continuation of Fig. A.5.



(g) Colon × Lung

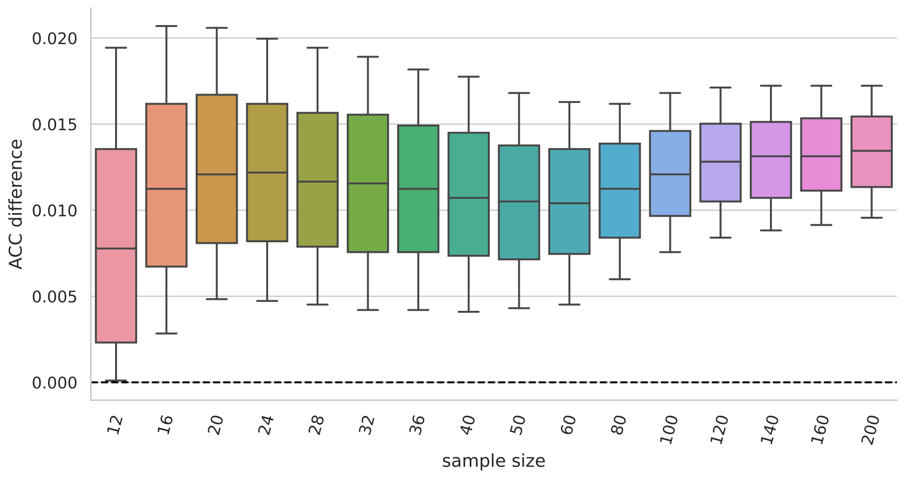


(h) Colon × Ovary

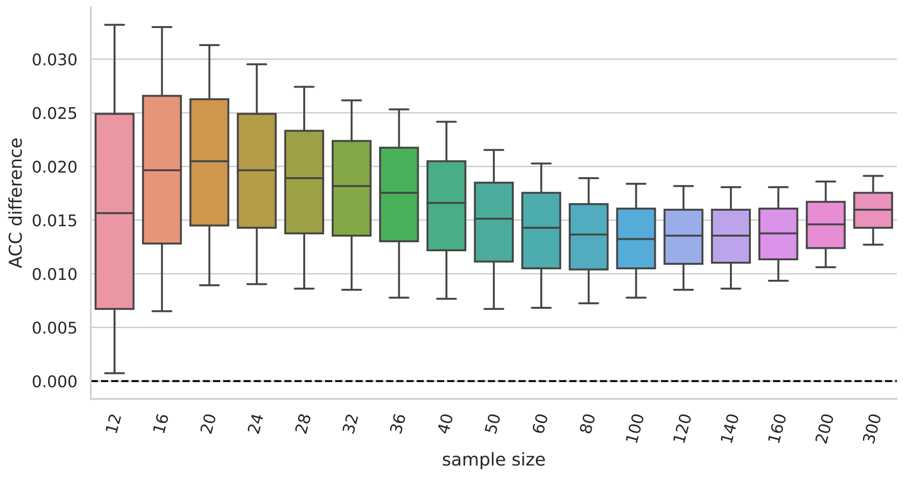


(i) Colon × Uterus

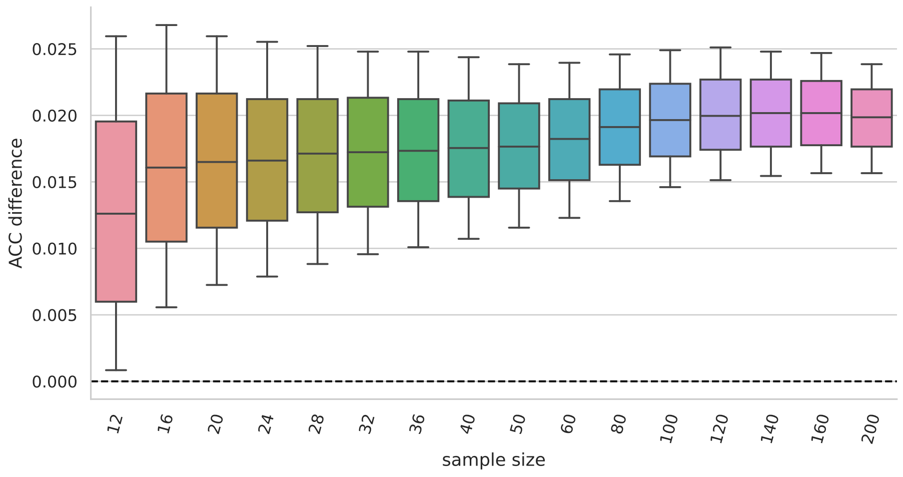
Figure A.5: **(cont.) Distributions of accuracy differences for the real phenotypes** Distributions of the accuracy score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Note that unlike MCC shown in Fig. A.4, the accuracy does not take the class imbalance into account. Continuation of Fig. A.5.



(j) Kidney × Lung

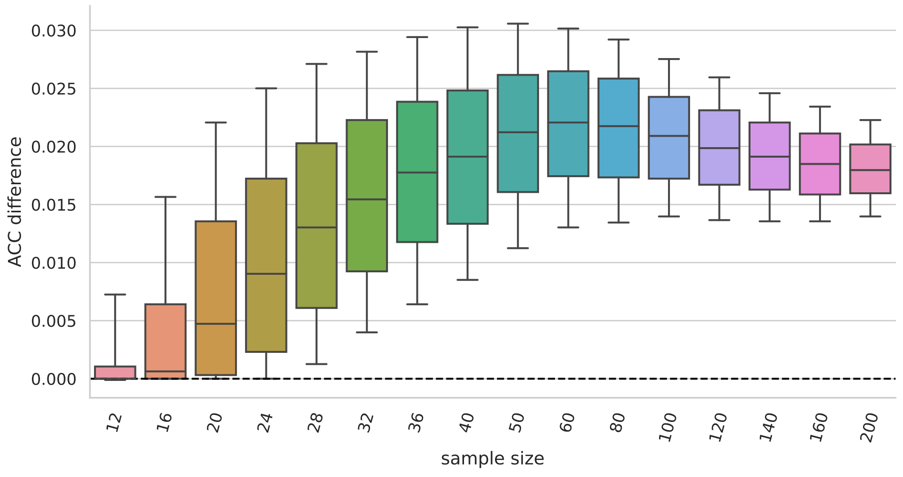


(k) Kidney × Ovary

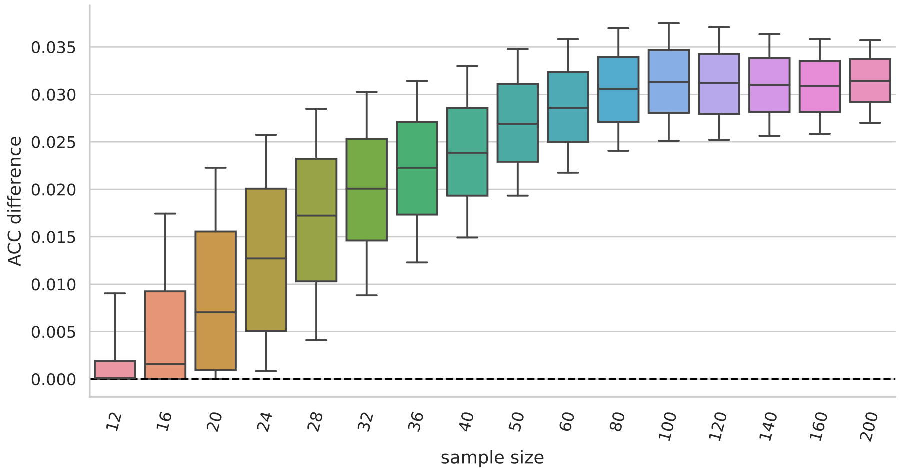


(l) Kidney × Uterus

Figure A.5: **(cont.) Distributions of accuracy differences for the real phenotypes** Distributions of the accuracy score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Note that unlike MCC shown in Fig. A.4, the accuracy does not take the class imbalance into account. Continuation of Fig. A.5.

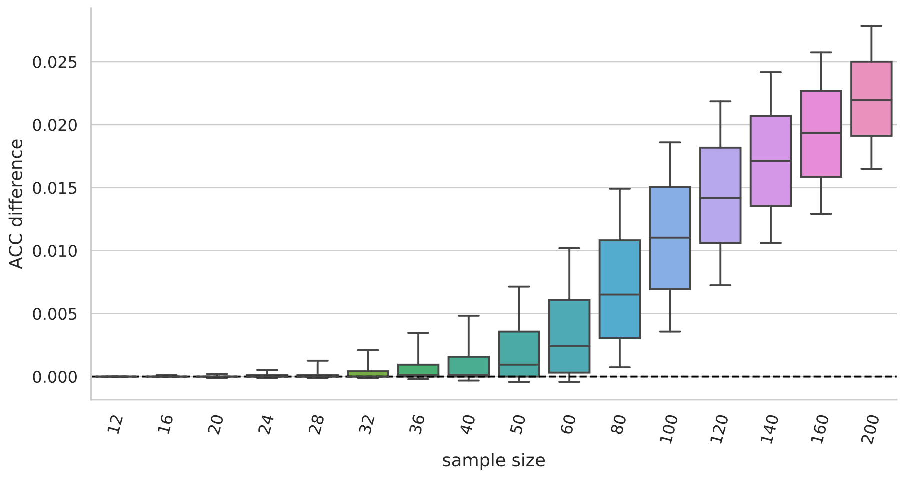


(m) Lung × Ovary



(n) Lung × Uterus

Figure A.5: **(cont.) Distributions of accuracy differences for the real phenotypes** Distributions of the accuracy score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Note that unlike MCC shown in Fig. A.4, the accuracy does not take the class imbalance into account. Continuation of Fig. A.5.



(o) Ovary × Uterus

Figure A.5: **(cont.) Distributions of accuracy differences for the real phenotypes** Distributions of the accuracy score pairwise differences of the D–GEX with TAAFs and the plain D–GEX of 5,000 repetitions for each sample size for different tissues. The whiskers show the 10th and 90th percentiles. Note that unlike MCC shown in Fig. A.4, the accuracy does not take the class imbalance into account. Continuation of Fig. A.5.

A.2 PERFORMANCE IMPACT OF LAYER CONFIGURATION OF THE INFERENCE NETWORK

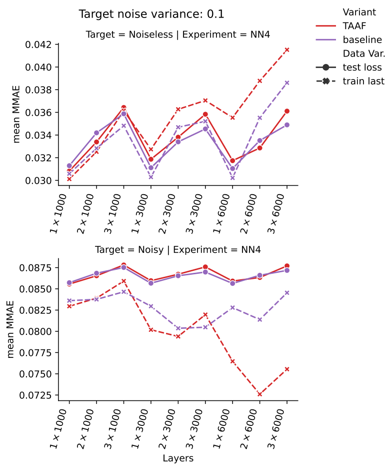


Figure A.6: **Absolute performance by layer configuration with target noise** The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on targets with noise with a standard deviation of 0.1.

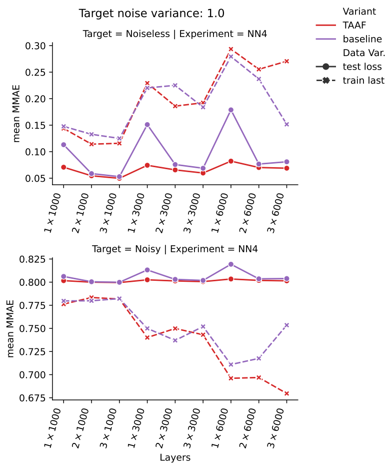


Figure A.7: **Absolute performance by layer configuration with target noise** The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on targets with noise with a standard deviation of 1.0.

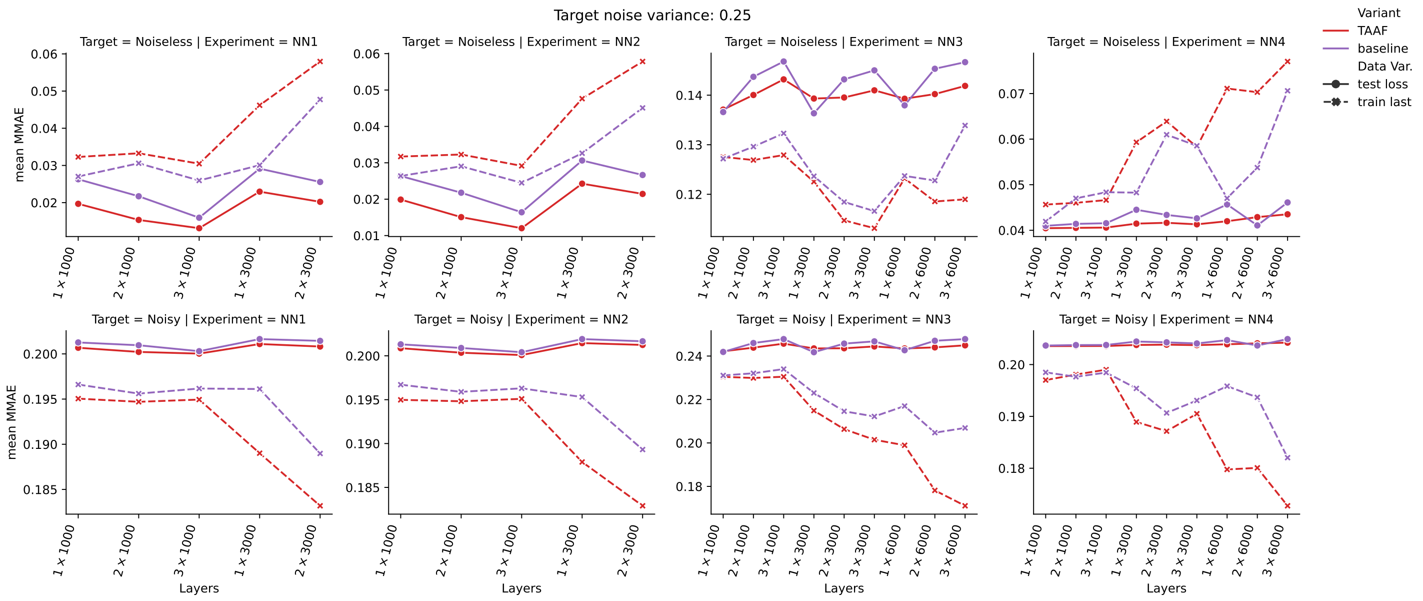


Figure A.8: **Absolute performance by layer configuration with target noise**
The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on targets with noise with a standard deviation of 0.25.

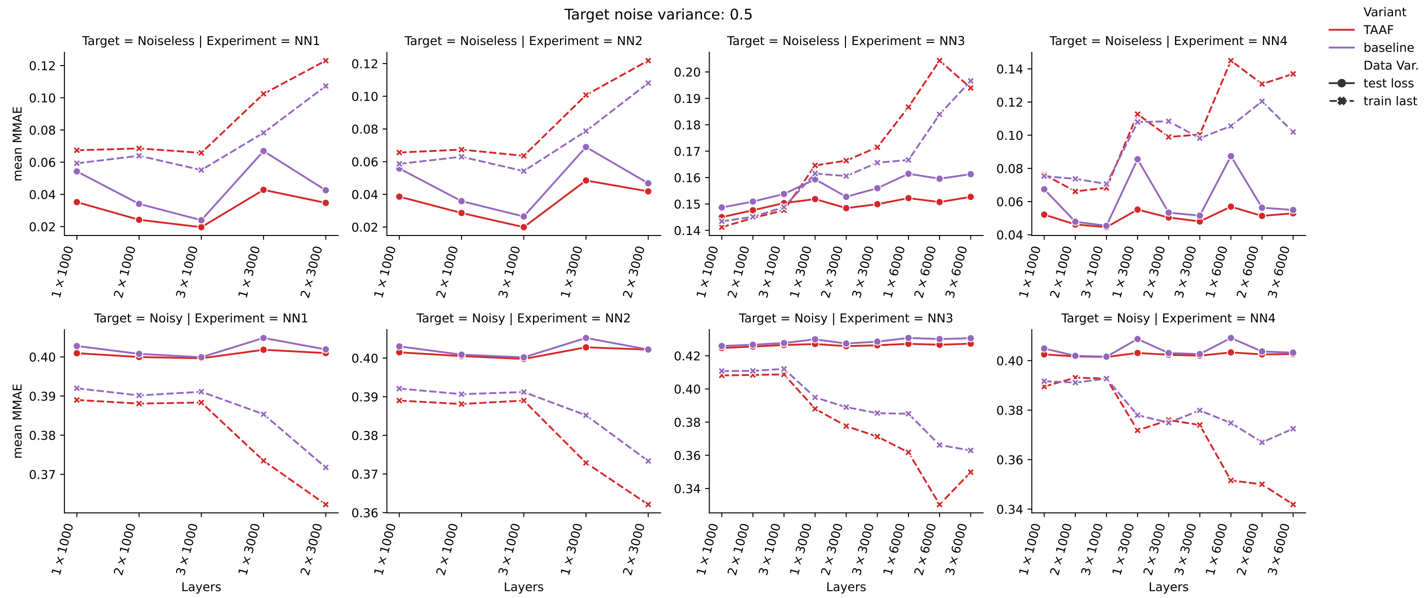


Figure A.9: **Absolute performance by layer configuration with target noise**
The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on targets with noise with a standard deviation of 0.5.

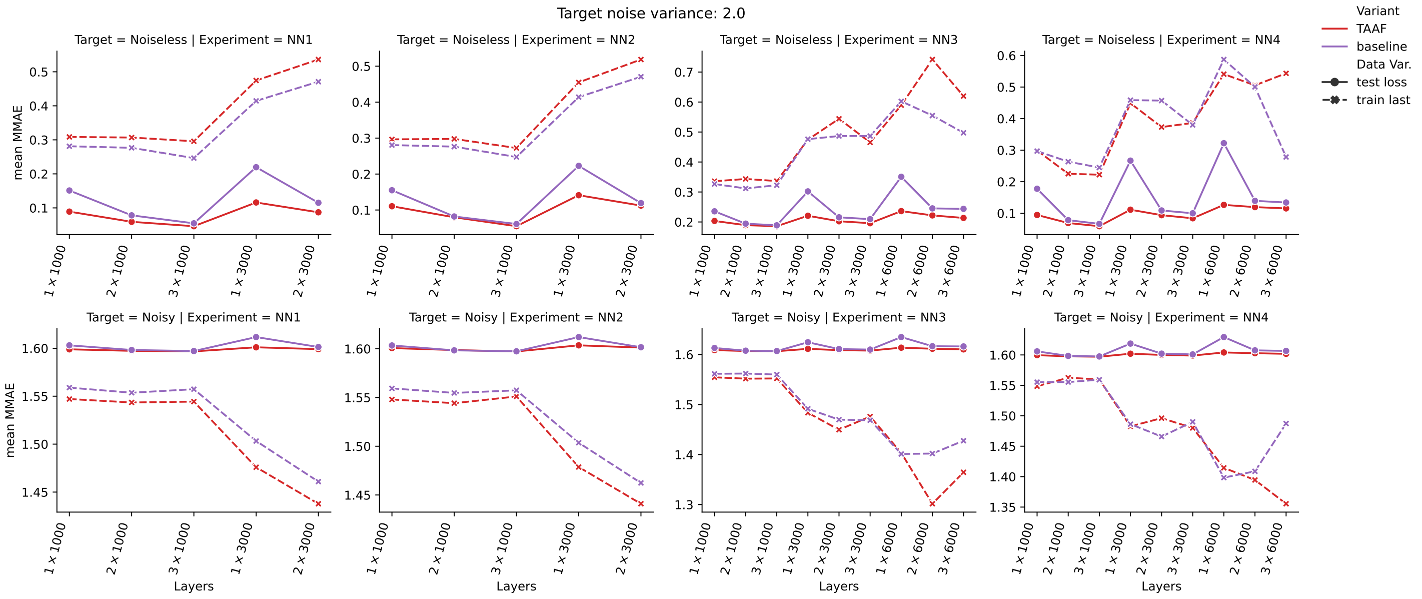


Figure A.10: **Absolute performance by layer configuration with target noise**
The absolute performance different configuration of hidden layers of the inference network. Only the OOS performance of the best model on the validation set and the in-sample performance of models trained on the training set till the last epoch is shown. It shows the mean MMAE over all relevant parameterizations for models trained on targets with noise with a standard deviation of 2.0.

DECLARATION

I declare that I elaborated this thesis on my own and that I mentioned all the information sources that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis[1].

Moreover, I state that this thesis has neither been submitted nor accepted for any other degree. The results presented in this thesis were achieved during my own research in cooperation with my thesis supervisor doc. Ing. Jiří Kléma, Ph.D. and are based on the works listed in this thesis (see Publications).

Prague, February 2024

Vladimír Kunc

1 EN: `https://www.cvut.cz/sites/default/files/content/d1dc93cd-5894-4521-b799-c7e715d3c59e/en/20230926-methodical-guideline-no-12009.pdf`
CZ: `https://intranet.fel.cvut.cz/cz/rozvoj/MP_2009_01.pdf`

This document was typeset in LaTeX using the typographical look-and-feel classicthesis developed by André Miede and Ivo Pletikosić which was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*.

Final Version as of February 16, 2024 (v1.0).