



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská



Strojová klasifikace zdrojů akustické emise a stupeň poškození materiálu

Machine based acoustic emission classification and damage level of materials

Diplomová práce

Autor: **Bc. Jan Zavadil**
Vedoucí práce: **Ing. Václav Kůs, Ph.D.**
Akademický rok: 2023/2024

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zavadil** Jméno: **Jan** Osobní číslo: **486425**
Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**
Zadávající katedra/ústav: **Katedra matematiky**
Studijní program: **Aplikované matematicko-stochastické metody**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Strojová klasifikace zdrojů akustické emise a stupeň poškození materiálu

Název diplomové práce anglicky:

Machine based acoustic emission classification and damage level of materials

Pokyny pro vypracování:

- 1) Pokračujte ve vývoji modelů strojového učení pro klasifikaci signálů akustické emise (AE) z defektoskopicky poškozených materiálů.
- 2) Soustřeďte se na moderní metody klasifikace využívající celý signál AE, jako např. Inception time CNN sítě. Navrhněte vhodný model klasifikačního transformeru pro hodnocení zdroje AE.
- 3) Ve spolupráci s Ústavem termomechaniky AV ČR navrhněte a proveďte experiment, který proměřuje stupeň poškození/opotřebení vybraného materiálu.
- 4) Získaná data z měření očistěte, přezpracujte statistickými metodami a aplikujte dřívější i nové strojové metody klasifikace. Výsledky vzájemně porovnejte.
- 5) Testujte možnosti metod strojového učení pro predikci počtu a zastoupení komponent v distribuční směsi v Preisach-Mayergoyzově (PM) prostoru hysteretických materiálů. Výstup aplikujte na strojovou klasifikaci elasticity/poškození materiálu.

Seznam doporučené literatury:

- 1) S. Gholizadeh, Z. Lemana, B.T.H.T. Baharudin, A review of the application of acoustic emission technique in engineering, Structural Engineering and Mechanics, Vol. 54, No. 6, 2015, 1075-1095.
- 2) M. Z. Alom, M. Hasan, C. Yakopcic, et al., Inception recurrent convolutional neural network for object recognition. Machine Vision and Applications 32, Article no:28, 2021.
- 3) S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, Transformers in Vision: A Survey. Acm Computing Surveys 54, 41, 2022.
- 4) I. D. Mayergoyz, Mathematical Models of Hysteresis and Their Applications. AcademicPress-Elsevier, 2003.
- 5) G. Celeux, S. Frühwirth-Schnatter, Ch. Robert, Model Selection for Mixture Models--Perspectives and Strategies. Handbook of Mixture Analysis, CRC Press, hal-01961077, 2018.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Václav Kús, Ph.D. katedra matematiky FJFI


Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **31.10.2023**

Termín odevzdání diplomové práce: **10.05.2024**

Platnost zadání diplomové práce: **31.10.2025**


Ing. Václav Kús, Ph.D.
podpis vedoucí(ho) práce


prof. Ing. Zuzana Masáková, Ph.D.
podpis vedoucí(ho) ústavu/katedry



doc. Ing. Václav Čuba, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

14.11.2023

Datum převzetí zadání



Podpis studenta

Poděkování:

Chtěl bych poděkovat svému školiteli Ing. Václavu Kúsovi, Ph.D. za dlouhodobé, trpělivé a plodné vedení mé Diplomové práce. Dále děkuji pracovníkům oddělení Rázů a Vln v tělesech Ústavu termomechaniky Akademie věd České republiky, kteří umožnili provedení experimentů a přispěli cennými radami k vyhodnocení jejich výsledků.

Čestné prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 5. května 2024

Jan Zavadil

Název práce:

Strojová klasifikace zdrojů akustické emise a stupeň poškození materiálu

Autor: Bc. Jan Zavadil

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: Ing. Václav Kůs, Ph.D., České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská, Katedra matematiky

Abstrakt: Hlavním cílem této práce je identifikace, implementace a porovnání metod hlubokého učení za účelem rozpoznávání signálů akustické emise. Pro získání relevantních dat sloužících k porovnání jednotlivých metod byly provedeny dva experimenty. V první části práce jsou shrnuty poznatky z teorie neuronových sítí a je představeno pět vybraných architektur pracujících přímo s 1D signály jako vstupními daty. Tyto modely jsou porovnány na klasifikační úloze nad daty z obou experimentů. Adaptovaná podoba nejúspěšnější sítě Pooled Inception Time je použita v regresní úloze při spojitě predikci závislé proměnné. V druhé části práce je představena problematika odhadu tvaru rozdělení pravděpodobnosti na Preisach-Mayergoyzově prostoru podle výsledných hysterezních křivek. Neuronové sítě představené v první části práce jsou využity pro predikci tvaru distribuční směsi.

Klíčová slova: Akustická emise, konvoluční neuronové sítě, Inception Time, PM prostor, distribuční směsi

Title:

Machine based acoustic emission classification and damage level of materials

Author: Bc. Jan Zavadil

Abstract: The main objective of this thesis is to identify, implement, and compare deep learning methods for the recognition of acoustic emission signals. Two experiments were conducted to obtain relevant data for comparing these methods. In the first part of the thesis, we summarize the findings from neural network theory and present five selected architectures designed to work directly with 1D signals as input data. These models are compared based on their performance in a classification task using the data from the experiments. Additionally, an adapted version of the best-performing Pooled Inception Time network is utilized in a regression task to predict continuous dependent variable. The second part of the thesis addresses the problem of estimating the shape of the probability distribution on the Preisach-Mayergoyz space from resulting hysteresis curves. We employ the neural networks introduced in the first part to predict the shape of the distribution mixture.

Key words: Acoustic emission, convolutional neural networks, Inception time, PM space, distribution mixtures

Obsah

Úvod	3
1 Akustická emise	4
2 Experimenty - původ dat	6
2.1 Dataset 1 - Rotační vrták	6
2.2 Dataset 2 - Ložiska	8
3 Architektura neuronových sítí	11
3.1 Vrstvy neuronové sítě	11
3.1.1 Hustá vrstva	11
3.1.2 Aktivační vrstva	11
Multilayer Perceptron	13
3.1.3 Konvoluční vrstva	13
3.1.4 Poolingové vrstvy	14
CNN.	14
3.1.5 Regularizační vrstvy	15
3.2 Učení neuronové sítě	18
3.2.1 Ztrátová funkce	18
3.2.2 Inicializace vah	19
3.2.3 Backpropagation	19
3.2.4 Algoritmy učení	20
Adam.	20
3.3 Transformer	21
3.3.1 Attention mechanismus	21
3.3.2 Poziční kódování	23
3.3.3 Encoder - Decoder	23
4 Klasifikační metody	26
4.1 Metriky úspěšnosti klasifikace	26
4.2 Conv2Net	27
4.3 ResNet-16	28
4.4 Inception Time	29
4.5 Pooled Inception Time	31
4.6 TSiT	33

5	Výsledky analýzy AE	35
5.1	Úspěšnost klasifikace podle délky filtrů	35
5.1.1	Conv2Net	35
5.1.2	Inception Time	37
5.2	Experiment 1	37
5.3	Experiment 2	40
5.4	Úspěšnost klasifikace podle délky vybíraných signálů	42
5.4.1	Experiment 1	42
5.4.2	Experiment 2	45
5.5	Experiment 2 - citlivost snímačů	46
5.6	Experiment 2 - regresní úloha	46
6	Hystereze v materiálech	49
6.1	Preisach-Mayergoyzův (PM) model	50
6.2	Identifikace PM prostoru	51
6.3	Hysterezní data	53
6.4	Výsledky pro hysterezní data	56
6.4.1	Klasifikace dominantního rozdělení ve směsi	56
6.4.2	Odhad koeficientů v distribuční směsi	57
	Závěr	59

Úvod

V poslední dekádě zaznamenal obor strojového učení, zejména hlubokých neuronových sítí, zásadní posun. Tento rychlý vývoj společně s růstem výpočetního výkonu dostupného hardwaru umožnil využití hlubokého učení v řadě nových odvětví a v posledních letech začal zásadním způsobem ovlivňovat každodenní život. V této práci se zabýváme aplikacemi metod hlubokého učení v oboru nedestruktivní defektoskopie s motivací zvýšit efektivitu metod využívaných např. k bezpečnostní kontrole tlakových potrubí, aktivnímu řízení obráběcích procesů či k odhadu poškození zemětřesných tlumičů.

V práci se věnujeme především metodě testování materiálů využívající signály akustické emise. Naším cílem je identifikovat vhodné metody hlubokého učení, které budou pracovat přímo s naměřenými signály akustické emise. Navazujeme tak na práci [1], která řešila stejný problém přístupem extrakce příznaků ze signálů. Zaměřujeme se tedy zejména na konvoluční neuronové sítě, které se osvědčily v oboru rozpoznávání obrazu, což je příbuzná úloha lišící se zejména dimenzí vstupních dat. Dále ověřujeme také využitelnost metod na bázi transformerů, které dosáhly výborných výsledků jak v oboru zpracování obrazu, tak ve zpracování přirozeného jazyka. Vybrané typy neuronových sítí implementujeme a porovnááme jejich výsledky na datech pocházejících ze dvou experimentů provedených ve spolupráci s Ústavem termomechaniky AV ČR (vrtací a ložiskové experimenty).

Pro ověření využitelnosti představených metod hlubokého učení v jiných materiálových aplikacích se v závěru práce věnujeme mechanické elastické hysterezi. Navazujeme na výsledky práce [32], přičemž výsledky z našich strojových metod využívající přímo neupravené 'signály' hysterezi při odhadu podoby rozdělení pravděpodobnosti v Preisach-Mayergoyzově modelu porovnááme s dřívějšími přístupy.

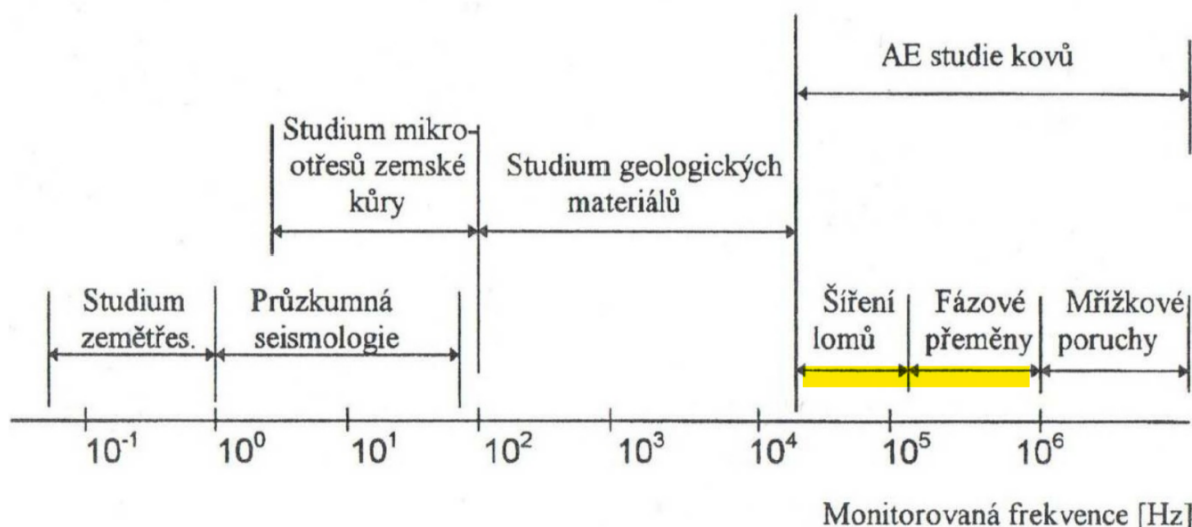
Implementace jednotlivých neuronových sítí proběhla v jazyce *python* s využitím zavedených knihoven pro strojové učení *pytorch*, *numpy*, *scikit-learn*, *tsai*. K vizualizaci a archivaci záznamů o průběhu učení jsme využili platformu *Weights & Biases*. Záznam dat během experimentů a základní předzpracování signálů probíhalo v prostředí *Matlab*. Vzhledem k velkému objemu dat a výpočetní náročnosti metod hlubokého učení byl k trénování sítí využit výpočetní klastr při FJFI *Helios*.

Kapitola 1

Akustická emise

Fenomén a původ akustické emise (AE) byl podrobněji popsán v [1] a [2], zde z těchto zdrojů čerpáme a přinášíme pouze stručné shrnutí pro uvedení do problematiky.

V této práci se zabýváme pouze využitím vln AE v rámci jedné z metod nedestruktivního testování v oboru *Structural health monitoring*. Aplikace akustické emise v nedestruktivním testování materiálů typicky pracují s vlnami o frekvencích mezi 20 kHz a 1 MHz [3]. Na obrázku 1.1 je toto zvýrazněné pracovní rozpětí frekvencí zaneseno mezi další obory pracující s jevem akustické emise v jiných odvětvích.

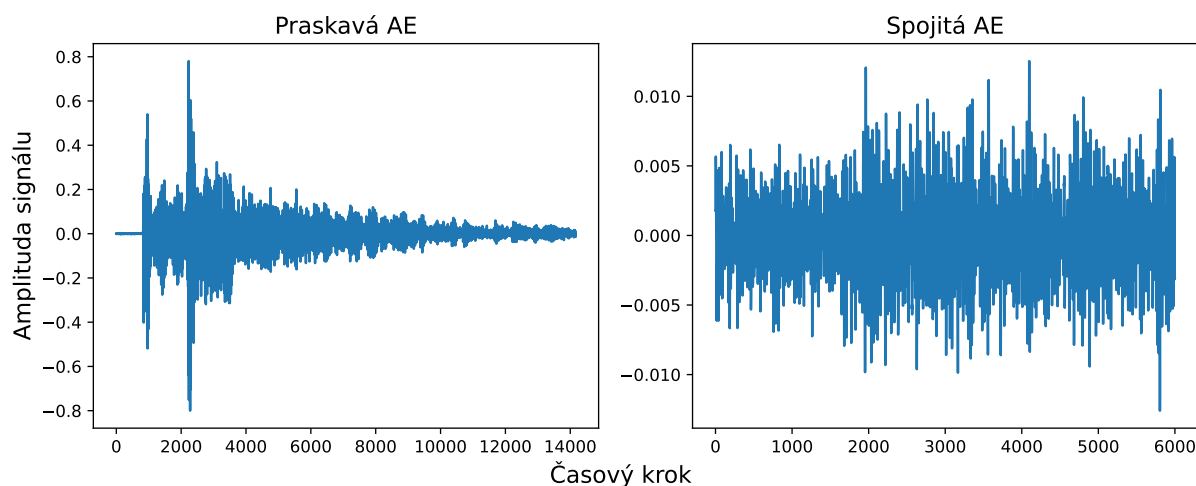


Obrázek 1.1: Využití AE dle frekvence vln [4].

AE je jev, při kterém se v materiálu uvolňují elastické vlny. Dochází k ní například při tvorbě trhliny, plastické deformaci, teplotní deformaci, nebo vlivem externích mechanických sil. Akustická emise se tedy vyskytuje při valně většině nezvratných změn v interní struktuře materiálu, zejména se objevuje při mechanickém namáhání materiálů v podobě tzv. stresových vln [5], vznikajících při náhlém uvolnění nahromaděné energie. Tyto vlny se od místa svého vzniku šíří materiálem až po dosažení rozhraní povrchu. Část energie vlny je přeměněna na teplo, část se vrací do tělesa v podobě odražené vlny a zbylá část energie vyvolá na povrchu tělesa tzv. Rayleighovu vlnu, kterou je možné zachytit snímači AE [6].

Na rozdíl od běžných metod ultrazvukového testování, které zkoumají vliv materiálu na externě generované vlny, nástroje AE zkoumají vlny vyvolané při namáhání, případně selhání materiálu. Monitorování úrovně aktivity akustické emise během více zátěžových cyklů je základem bezpečnostně inspekčních metod využívajících fenoménu AE. Konkrétním příkladem aplikace je kontrola jaderného reaktoru za provozu či kontrola těsnosti tlakových nádob apod. Moderní aplikací akustické emise je aktivní kontrola ve výrobním procesu, která umožňuje uzpůsobovat například parametry obráběcího procesu, jako je rychlost řezání či tloušťka třísky, v závislosti na stavu obráběcího stroje.

Standardně se akustická emise rozlišuje na praskavou a spojitou emisi. Signál praskavé akustické emise je charakterizován jednotlivě oddělitelnými emisními událostmi, například většina lomových procesů v materiálech či přetržení vlákna v kompozitu. V signálu spojitě akustické emise nelze vysledovat jeho jednotlivé původce, slitý signál se může zdánlivě jevit jako běžný šum. Příkladem původce spojitě emise je únik média z tlakové nádoby, či obrábění materiálu. Na obrázku 1.2 je zobrazen příklad signálu praskavé a spojitě AE. Pro záznam akustické emise se používají piezoelektrické snímače umístěné na povrch tělesa. Přichycují se na tzv. vlnovod nebo přímo na povrch magnetem nebo různými lepicími hmotami.



Obrázek 1.2: Porovnání praskavé a spojitě AE.

Kapitola 2

Experimenty - původ dat

V této práci byla využita data pocházející ze tří různých experimentů. Všechny experimenty byly provedeny na půdě Ústavu termomechaniky Akademie věd za asistence jejich pracovníků. V obou případech byl signál akustické emise měřen USB osciloskopem *Handyscope HS6 DIFF* výrobce TiePie za pomoci tří typů snímačů:

1. Nerezový s korundovou keramickou styčnou plochou *Dakel IDK09*,
2. Nerezový magnetický s poniklovanou styčnou plochou *Dakel MDK13*,
3. Nerezový s ocelovou styčnou plochou *BOTEG UTS-400-BNC*.

Nemagnetické snímače byly lepeny kyanoakrylátovým lepidlem, magnetické snímače byly podmazány silikonovou vazelínou. Důvodem použití různých typů snímačů je jejich různá frekvenční charakteristika, neboli citlivost na různé frekvence ve snímaném signálu. Frekvenční charakteristika je dána materiálem, tvarem i konstrukčním postupem při výrobě daného snímače, liší se tedy drobně i mezi dvěma snímači stejného typu. Použití různých snímačů vnáší do experimentu další variabilitu a umožňuje potenciálně porovnání klasifikační výtěžnosti jednotlivých typů snímačů.

2.1 Dataset 1 - Rotační vrták

První experiment byl proveden jako součást práce [7] a následně zpracován v rámci [1], odkud přebíráme jeho popis. Předmětem pozorování je signál generovaný odkrajováním materiálu v průběhu vrtání díry do ocelové desky. Mechanismus opotřebení řezného břitu je blíže popsán v [6], kde je zároveň potvrzena využitelnost AE pro predikci tohoto opotřebení. Při vrtání dochází ke vzniku spojitě akustické emise v důsledku odkrajování tenkých vrstev materiálu důsledkem tření styčných ploch. Cílem je zaznamenat několik sad této spojitě emise s postupným otupením vrtáku. Očekáváme, že tupý vrták bude mít větší problém odkrajovat materiál rovnoměrně a bude ve vrtané díře způsobovat více nárazových poprasknutí. Díky tomu očekáváme rozdíly ve spojitě emisi, díky kterým budeme schopni odhalit stav tuposti vrtáku.

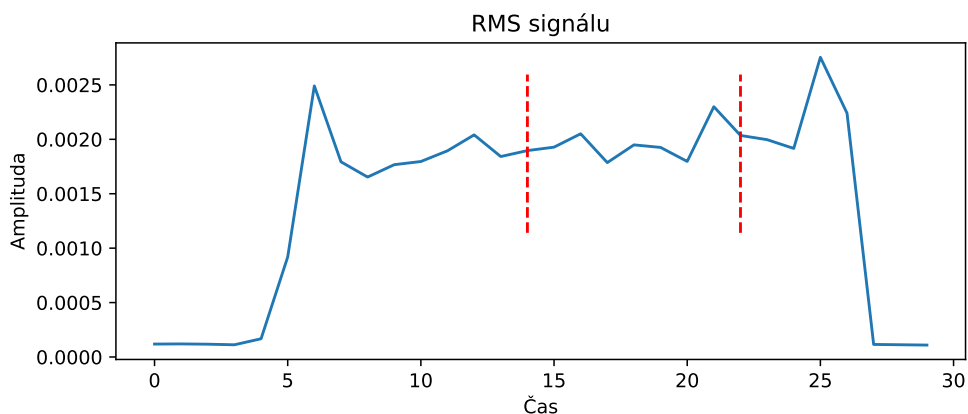
Na obrázku 2.4 je zachycena měřicí aparatura. Do stojanu umožňujícího vrtat pod konstantním zatížením byla uchycena vrtačka výrobce Bosch, s jejíž pomocí bylo vrtáno do ocelové desky o rozměrech 70 x 93 x 10 mm. K desce byly přichyceny dva snímače typu 1, dále po jednom snímači typů 2 a 3 jak je znázorněno na obrázku 2.3. Celkem bylo provedeno pět sad měření mezi kterými byl vrták uměle otupen za použití brusného kamene. V každé sadě měření bylo

vrtáno do děr 1 až 5 po dobu přibližně pěti sekund za konstantního zatížení vrtáku, celkem jsme tak dostali pět signálů spojité akustické emise pro každou úroveň otupení vrtáku, z nichž každý pocházel z jiného místa. Díky různým umístěním zdroje signálu se při určování tuposti vrtáku budeme moci zaměřit na charakteristiky signálu, které nezávisí na jeho poloze. Měřicí aparatura snímala se vzorkovací frekvencí 3,125 MHz na všech čtyřech kanálech.

Zaznamenané signály spojité AE zahrnovaly kromě doby vrtání pod konstantním zatížením i dobu před a po něm, nejprve tedy bylo nutné pro každý z 25 signálů manuálně vybrat pouze žádanou část záznamu. To bylo možné díky zvýšené amplitudě signálu při zasouvání vrtáku do díry a jeho následovném vyjmutí. Z každého signálu jsme vyřizli jeho část odpovídající čtyřem sekundám souvislého vrtání. Na obrázku 2.1 je zobrazen příklad průběhu RMS signálu. Pro snazší interpretaci byl signál rozdělen na půlsekundové intervaly, které byly zprůměrovány za pomoci root mean square (RMS) a vynesena byla pouze průměrná hodnota za daný úsek signálu. Nechť x_1, x_2, \dots, x_n jsou hodnoty signálu, potom jejich RMS průměr definujeme jako

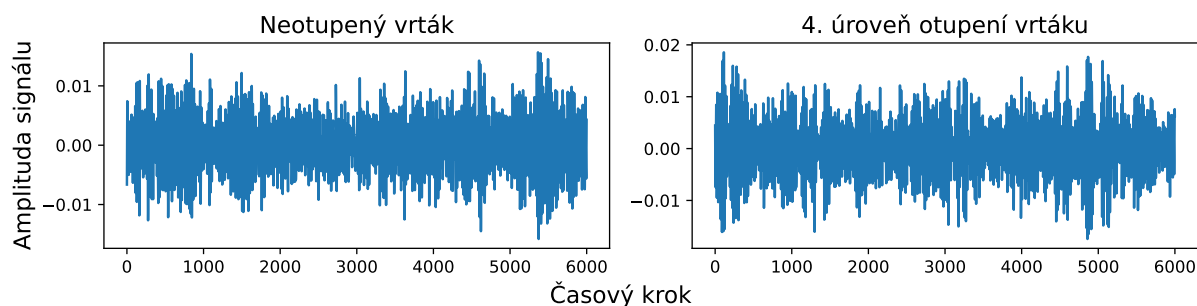
$$x_{RMS} = \sqrt{\frac{1}{n} (x_1^2 + x_2^2 + \dots + x_n^2)}.$$

Ze signálu byla pro další zpracování vybrána pouze ohraničená část, jak je patrné z obrázku 2.1. Měřítka na ose x odpovídá zmíněným půlsekundovým úsekům

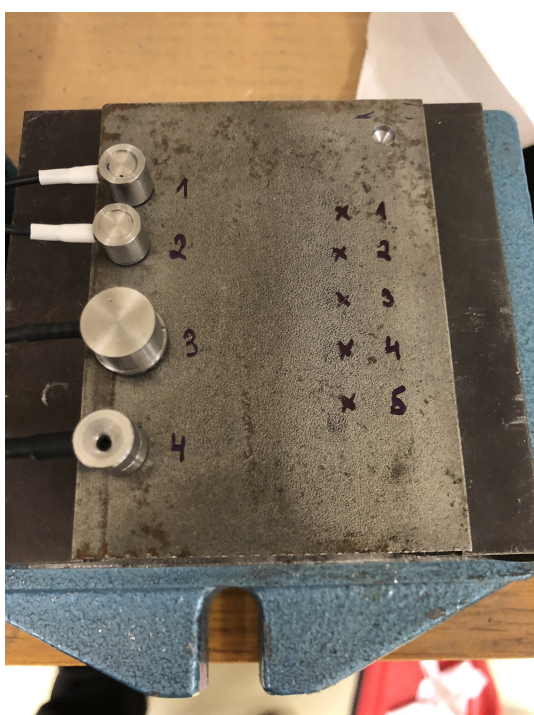


Obrázek 2.1: *Root Mean Square signálu spojité akustické emise po půlsekundových intervalech.*

Po tomto ořezání signálů dostáváme pro každou úroveň tuposti vrtáku pět signálů příslušejících pěti dířům, každý ze signálů má 4 kanály. Délka jednoho signálu je 12,5 milionu hodnot (4 sekundy), což je pro přímé použití v klasifikačních metodách neúnosně dlouhé. Díky spojitému charakteru této akustické emise a relativní homogenitě signálu v celé délce můžeme přistoupit k vybrání většího množství podintervalů o výrazně kratší délce, které budou tvořit data, se kterými budeme dále pracovat. Délka vybíraných vektorů byla po několika pokusech stanovena na 6000, což se ukázalo jako stále dostatečně dlouhé pro zachycení charakteristiky signálu, ale zároveň dostatečně krátké pro rozumnou výpočetní dobu použitých algoritmů. Na obrázku 2.2 Počátek vybíraných vektorů byl volen náhodně rovnoměrně z celé délky signálu. Pro každou úroveň tuposti vrtáku jsme ze signálu příslušejícího každé díře vybrali 500 kratších vektorů, celkem tak máme pro každou z pěti úrovní otupení vrtáku 2500 kratších signálů, každý z nichž má 4 kanály příslušející čtyřem snímačům. To je dataset se kterým budeme dále pracovat.



Obrázek 2.2: Příklad výřezů ze signálů z 2. kanálu.



Obrázek 2.3: Ocelový blok použitý k vrtání se snímači.



Obrázek 2.4: Vrtací soustava s měřicí aparaturou.

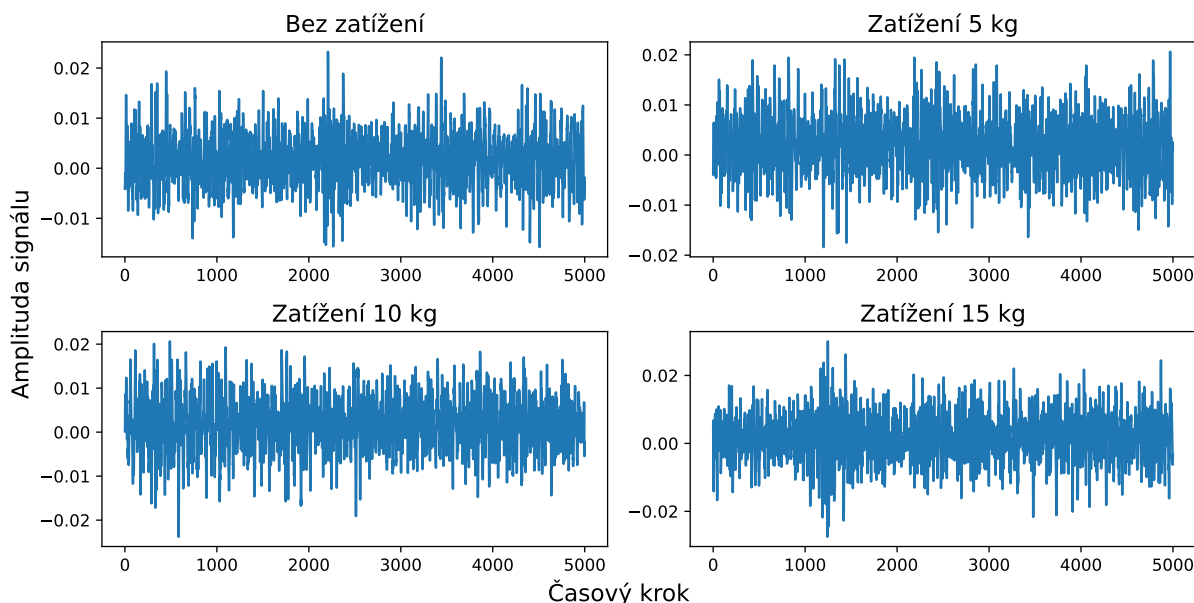
2.2 Dataset 2 - Ložiska

Druhý experiment se zabýval sběrem akustické emise generované chodem kuličkových ložisek. Motivací pro tento experiment je, podobně jako v předchozím případě, potenciální uplatnění výsledků v průmyslu, kde má diagnostika a odhad opotřebení ložiska řadu možných uplatnění. U valivých ložisek dochází postupem času nevyhnutelně k opotřebení a degradaci materiálu, toto opotřebení může urychlit řada faktorů, např. znečištění, nesprávné mazání. Dostupná diagnostická technika může pomoci předejít fatálnímu poškození jak samotného ložiska, tak i součásti mnohem dražší a důležitější, která na ložisku závisí, a ušetřit tak náklady spojené s jejich výměnou.

Za tímto účelem byla sestavena zatěžovací experimentální aparatura zachycená na obrázku 2.5. V levé části můžeme vidět elektromotor typu *m5120-4025GN-5K* o výkonu 120 W a točivém

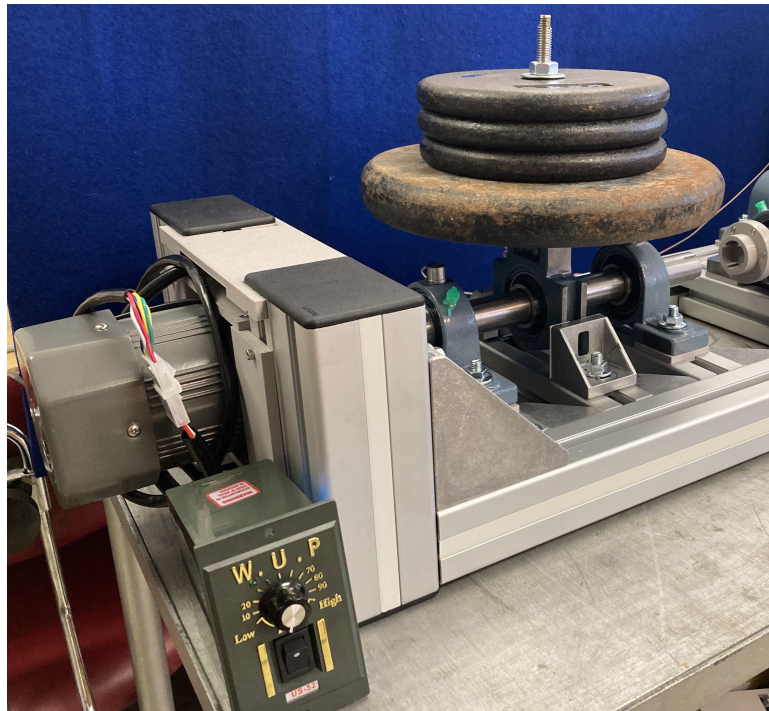
momentu 3,5 Nm. Před ním je potenciometr řídicí chod motoru. Motor je uchycen v hliníkovém rámu za pomoci 3D-tištěných adapterů a poháněn ocelovou hřídelí. Na hřídeli jsou nasazena celkem tři ložiska typu UC 205 šířky 15 mm s vnitřním průměrem 25 mm. První a třetí ložisko v pohledu od motoru jsou uchycena v ložiskových domcích typu UCP 205, které jsou pevně spojeny s hliníkovým rámem, jak je dobře patrné z obrázku 2.6. Prostřední ložisko je uchyceno v domku typu UCT 205, který není s rámem nijak spojen a celá jeho váha tak spočívá na hřídeli. Ocelové vzpěry tvaru L pouze zajišťují, že domek zůstane v napřímené poloze, nijak ho však nepodepírají. K prostřednímu domku je přichycen ocelový trn vyrobený ze závitové tyče, umožňující přímé zatížení prostředního ložiska pomocí ocelových kotoučů.

Samotný experiment spočíval ve snímání akustické emise produkované ložisky při různém zatížení. Během experimentu byly otáčky fixovány na cca 700 ot./min. K záznamu byly použity čtyři snímače typu 2, jejichž rozmístění je patrné z obrázku 2.6. Snímače jsou číslovány zprava, tedy snímače 1 a 3 jsou uchyceny na vrchu krajních ložiskových domků, snímač 2 je uchycen ze strany prostředního ložiskového domku a snímač 4 je uchycen u upevnění ložiskového domku vlevo dole. Akustická emise byla pro každou úroveň zatížení měřena po dobu 30 sekund se vzorkovací frekvencí 1,5625 MHz. Celkem bylo naměřeno 8 úrovní zatížení, od nulového s konzistentními 2,5 kg přírůstky až po maximální zatížení 17,5 kg.

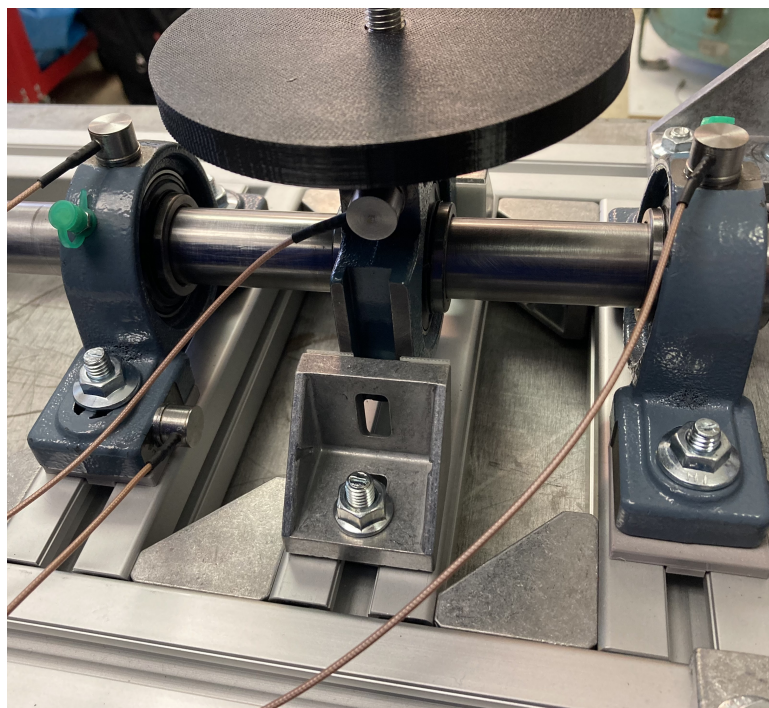


Obrázek 2.7: Ukázka výřezů signálu z druhého snímače pro různé úrovně zatížení.

Výstupem měření tedy byl pro každou z osmi úrovní zatížení signál akustické emise o čtyřech kanálech. Délka 30 sekundového záznamu byla, vzhledem k vysoké vzorkovací frekvenci, přibližně 47 milionů hodnot, což jakoukoliv přímou manipulaci se signálem velmi komplikuje. Obdobně jako u prvního experimentu jsme se vzhledem ke spíše spojitému charakteru AE rozhodli k vybrání sady podintervalů, které budou tvořit dataset pro další zpracování. Po předchozích zkušenostech se zpracováním obdobných signálů byla délka vybíraných podintervalů stanovena na 5000. Příklad úseků vybraných ze signálů pro různé zatížení je na obrázku 2.7. Počátek vybíraných úseků byl volen náhodně rovnoměrně z celé délky původního signálu. Celkem jsme pro každou úroveň zatížení vybrali 2000 podintervalů. Finální dataset má tedy podobu tenzoru reálných čísel o rozměrech (16000, 4, 5000).



Obrázek 2.5: *Zatěžovací aparatura s maximálním závažím.*



Obrázek 2.6: *Umístění snímačů na zatěžovací aparatuře.*

Kapitola 3

Architektura neuronových sítí

Umělé neuronové sítě (Neural Networks) se v posledních letech, i díky pokroku ve výpočetním výkonu počítačů, vynořily jako dominantní a efektivní nástroje pro řešení problémů v široké řadě oblastí. Podoba různých modelů řazených do rodiny neuronových sítí se zásadně liší na základě konkrétního problému, pro který jsou navrženy, společnou charakteristikou je však možnost je rozdělit na jednotlivé vrstvy. Vstupní data postupně procházejí a jsou transformována vrstvami neuronové sítě až po výstupní vrstvu jejíž tvar je dán příslušnou problematikou. V následující části popíšeme jednotlivé běžně používané vrstvy - stavební bloky neuronových sítí.

3.1 Vrstvy neuronové sítě

3.1.1 Hustá vrstva

Základní stavební vrstvou a také nejstarším konceptem v oboru neuronových sítí je tzv. hustá vrstva (*dense layer, fully connected layer*). Hustá vrstva sestává z předdefinovaného počtu neuronů a jako vstup přijímá vektor reálných čísel $\mathbf{x} \in \mathbb{R}^q$. Jeden neuron lze chápat jako jednoduchou procesní jednotku, která spočte vážený průměr složek vstupního vektoru podle vah w_i , $i \in \hat{q}$ s přičtením absolutního členu w_0 . Výstupní hodnota y neuronu tedy je

$$y = w_0 + \sum_{i=1}^q x_i w_i = \bar{\mathbf{x}} \cdot \mathbf{w},$$

kde w_0 je tzv. bias a vektor $\bar{\mathbf{x}} = (1, \mathbf{x}) \in \mathbb{R}^{q+1}$ vznikl přidáním jedničky na začátek vstupního vektoru. Výstupem husté vrstvy o m neuronech je potom vektor $\mathbf{y} \in \mathbb{R}^m$ sdružující výstupy jednotlivých neuronů. Lze psát

$$\mathbf{y} = \mathbf{W}\bar{\mathbf{x}},$$

kde $\mathbf{W} \in \mathbb{R}^{m, q+1}$ je matice vah husté vrstvy a hodnota $W_{i,j}$ představuje váhu mezi i -tým neuronem a j -tou složkou vstupního vektoru.

3.1.2 Aktivační vrstva

Jelikož kombinace více lineárních transformací je stále lineární transformace a od neuronových sítí požadujeme i modelování nelineárních závislostí je třeba představit nelineární prvek. Tím je nelineární aktivační funkce aplikovaná v podobě vrstvy na vybraná místa v neuronové síti. Aktivační vrstva obvykle spočívá v aplikaci skalární aktivační funkce $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ na každý prvek

vstupu zvlášť. V moderních architektúrah se používá řada různých aktivačních funkcí, zmíníme zde pouze ty nejvýznamnější,

$$\begin{aligned}\sigma_{\tanh}(x) &= \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \\ \sigma_{ReLU}(x) &= \max(0, x), \\ \sigma_{GELU}(x) &= x \cdot \Phi(x),\end{aligned}$$

kde Φ je distribuční funkce standardního normálního rozdělení. Tyto aktivační funkce se obvykle používají uvnitř modelu po skrytých vrstvách. ReLU aktivační funkce je dnes standardně používána a nahradila Tanh zejména z důvodu nižších výpočetních nároků. GELU překonává ostatní aktivační funkce na několika problémech [9] a je velmi populární v moderních architektúrah, kde nahrazuje ReLU. Z obrázku 3.1 je patrné že GELU představuje jakési vyhlazení ReLU.

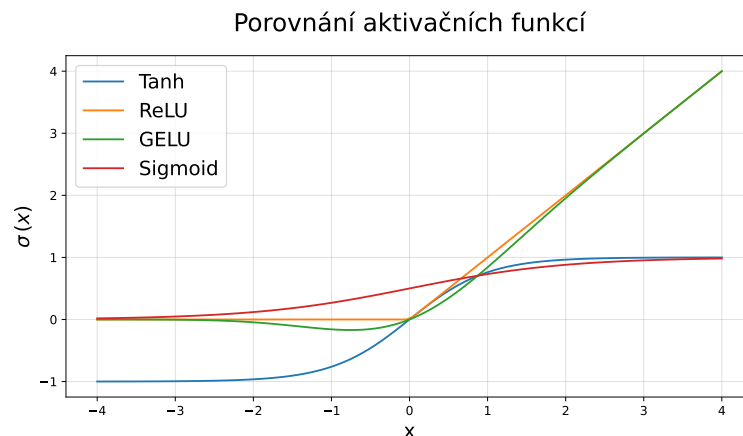
Na výstupní vrstvě neuronové sítě se často používají jiné aktivační funkce. Při regresním nasazení neuronové sítě se obvykle v poslední vrstvě žádná aktivace nepoužívá. V případě klasifikace se používají následující aktivační funkce

$$\begin{aligned}\sigma_{sigmoid}(x) &= \frac{1}{1 + e^{-x}}, \\ \sigma_{softmax}(\mathbf{x})_i &= \frac{e^{x_i}}{\sum_j e^{x_j}}.\end{aligned}$$

Aktivační funkce sigmoid se používá v případě binární klasifikace, protože zobrazuje vstup x na interval $[0, 1]$. Aktivace softmax, na rozdíl od ostatních, pracuje s celým výstupem husté vrstvy v podobě vektoru a používá se v případě klasifikace do více tříd, protože pro něj platí

$$\sum_i \sigma_{softmax}(\mathbf{x})_i = 1, \quad 0 < \sigma_{softmax}(\mathbf{x})_i < 1, \quad \forall i \in \hat{p},$$

tzn., že pro libovolný vstup \mathbf{x} poskytuje pravděpodobnostní rozdělení. U klasifikačních sítí tak má poslední vrstva vždy takový počet neuronů jaký je počet tříd, do kterých klasifikujeme.

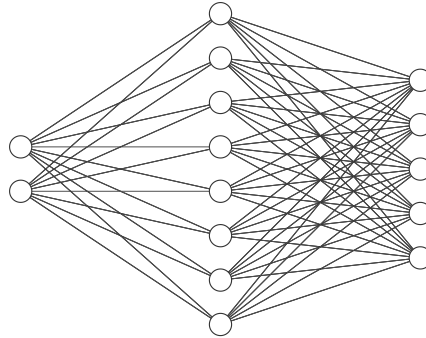


Obrázek 3.1: Porovnání aktivačních funkcí.

Multilayer Perceptron Neuronová síť sestavená pouze z hustých a aktivačních vrstev se nazývá vícevrstvý perceptron (také hustá vrstevnatá síť, *Multilayer Perceptron - MLP, Feedforward neural network*). Tato síť se skládá z vstupní vrstvy, libovolného počtu skrytých vrstev a výstupní vrstvy. Je charakteristická tím, že každý neuron v libovolné vrstvě je skrze příslušnou váhu ovlivněn každým neuronem z předchozí vrstvy. Aktivační vrstva se obvykle aplikuje po každé skryté vrstvě. Pro MLP síť s $m - 1$ skrytými vrstvami, s aktivační funkcí σ a vstupem \mathbf{x}_0 tak můžeme psát

$$\mathbf{x}_n = \sigma \left(\mathbf{W}^{(n)} \mathbf{x}_{n-1} \right), \quad n \in \hat{m},$$

kde $\mathbf{W}^{(n)}$ je matice vah n -té vrstvy a vektor \mathbf{x}_m je požadovaný výstupní vektor. Na obrázku 3.2 je zobrazena MLP síť s jednou skrytou vrstvou. Vstupní dimenze této sítě je 2, výstupní 5. Tato síť by se tedy dala využít například pro klasifikaci do pěti tříd na základě dvou příznaků. V dnešní době se tyto vrstevnaté sítě používají zejména v kombinaci s komplexnějšími metodami. Výzkum využití architektur založených čistě na MLP však stále pokračuje a například v [8] byl úspěšně použit model MLP-mixer pro zpracování obrazu.



Obrázek 3.2: Příklad jednoduché MLP sítě.

3.1.3 Konvoluční vrstva

Vrstevnaté sítě mají obecně problém se zpracováním vícedimenzionálních vstupů a jsou velmi citlivé i na jeho drobné posunutí. Za účelem překonání těchto problémů pro zpracování obrazových dat byly vyvinuty konvoluční neuronové sítě. Zde představíme podobu konvoluční vrstvy pro 1D vstupní data, což odpovídá našim potřebám na zpracování signálů AE.

Konvoluční vrstva sestává z konvolučních jader (filtrů) namísto jednoduchých neuronů, jejím cílem je extrahovat příznaky ze vstupních tenzorů, které jsou následně předány vrstevnaté části sítě. Jako vstup očekává 1D konvoluční vrstva matici $\mathbf{X} \in \mathbb{R}^{c,w}$, kde c je počet kanálů jednoho pozorování (v našem případě počet senzorů, kterými byla AE měřena) a w je délka časových řad (signálů). Konvoluční vrstva se skládá z d konvolučních jader $\mathbf{A}^{(i)} \in \mathbb{R}^{c,w_a}$, $\forall i \in \hat{d}$. Aplikací konvolučních jader \mathbf{A} na vstupní matici \mathbf{X} dostaneme výstupní matici $\mathbf{Y} \in \mathbb{R}^{d,(w-w_a+1)}$ podle předpisu

$$Y_{ij} = \sum_{k=1}^c \sum_{l=1}^{w_a} A_{kl}^{(i)} X_{k,j+l}, \quad \forall i \in \hat{d}, \quad \forall j \in \{1, \dots, w - w_a + 1\}.$$

Jelikož platí $w - w_a + 1 \leq w$, výsledný tenzor \mathbf{Y} má kratší rozměr v poslední dimenzi. Je možné použít tzv. padding pro zachování rozměru poslední dimenze, ten spočívá v přidání vhodného

počtu prvků ke vstupnímu tenzoru. Nejjednodušší formou je zero-padding, který přidává nulové hodnoty.

Jeden konvoluční filtr si lze představit jako posuvné okno o délce w_a , které pomocí vah $\mathbf{A}^{(i)}$ vytváří nový signál kombinací všech kanálů na vstupu. Pro snížení objemu reprezentovaných dat lze zavést tzv. *stride*, který určuje o kolik pozic se toto okno posouvá, standardně je volen jako $stride = 1$.

3.1.4 Poolingové vrstvy

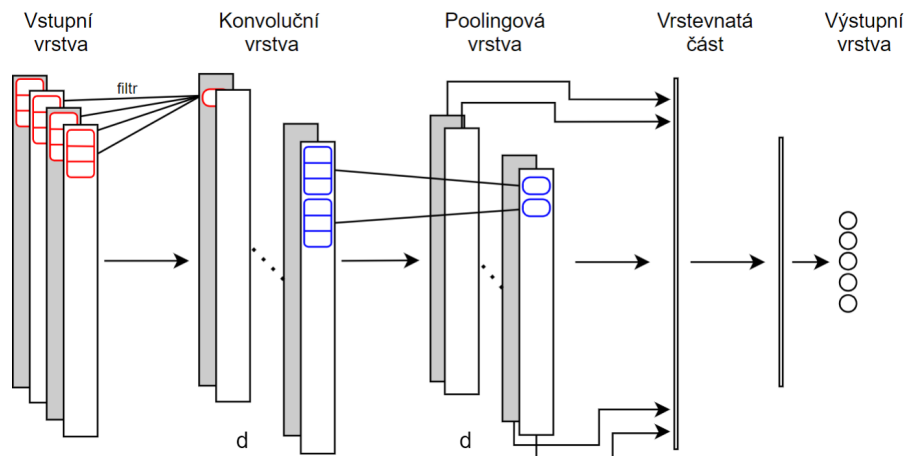
Další možností, jak snížit velikost reprezentovaných dat v síti a zároveň potlačit šum a nepotřebnou informaci obsaženou ve vstupních datech, jsou tzv. poolingové vrstvy. Ty lze pro vstupní data $\mathbf{X} \in \mathbb{R}^{c,w}$ definovat jako

$$Y_{ij} = f(X_{i,s(j-1)+1}, X_{i,s(j-1)+2}, \dots, X_{i,s(j-1)+w_a}), \quad \forall i \in \hat{c}, \quad \forall j \in \{1, \dots, (w - w_a)/s + 1\},$$

kde s je obdobně jako u konvoluční vrstvy *stride*, w_a je délka pooling filtru a f je funkce shrnující statistiky vstupu. Nejčastěji používanými funkcemi jsou maximum a průměr, které vedou na standardní názvosloví max-pooling a average-pooling. Obvykle volíme *stride* stejný jako délku filtru.

CNN. S využitím konvolučních a poolingových vrstev se už v roce 1998 v [10] podařilo vyvinout konvoluční neuronovou síť pro rozpoznávání ručně psaných číslic, která našla využití v automatizovaném zpracování šekových poukázek. Principem této sítě, nazvané LeNet5, bylo nejprve několikrát využít tandem konvoluční a poolingové vrstvy pro extrakci vhodných příznaků z obrázku a zásadní snížení jeho velikosti. Následně byla data zpracována vrstevnatou sítí. Tímto přístupem se řídila řada dalších autorů a představuje nejjednodušší pojetí implementace konvoluční neuronové sítě.

Na obrázku 3.3 je vyobrazena jednoduchá konvoluční neuronová síť, odpovídající naší problematice zpracování dat v podobě 1D časových řad. Vstupem je pozorování o čtyřech kanálech. Konvoluční vrstva s d jádry o délce 3 (v obrázku červeně) vstupní data transformuje do d nových časových řad. Následně je aplikována poolingová vrstva (modře) s filtrem délky 3 a stejnou hodnotou *stridu*, která zkrátí délku dat na třetinu. Poté je proveden *flattening*, tzn. že jsou data převedena z výsledné mapy charakteristik do jednoho dlouhého vektoru, který slouží jako vstup do vrstevnaté sítě. Ta charakteristiky extrahované konvoluční částí sítě zpracuje. V případě neuronové sítě z obrázku 3.3 by se mohlo jednat např. o klasifikaci do pěti tříd. Obvykle konvoluční neuronová síť obsahuje více než jednu konvoluční i poolingovou vrstvu, ty také není nutné používat pouze v tandemu tak jako na našem příkladu.

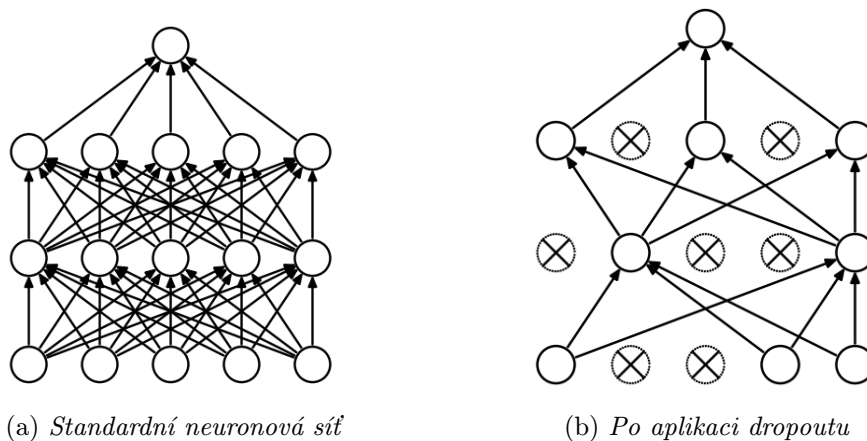


Obrázek 3.3: Vizualizace architektury jednoduché konvoluční neuronové sítě.

3.1.5 Regularizační vrstvy

Mezi regularizační vrstvy lze zařadit řadu různých přístupů, které mají pomoci k rychlejšímu a stabilnějšímu trénování neuronové sítě a k předejití přetrénování. Přetrénování neuronové sítě je problém, ke kterému dochází, když se neuronová síť příliš přizpůsobí trénovacím datům a ztrácí generalizační schopnosti pro nová data.

Dropout. Mezi nepoužívanější regularizační vrstvy používané k potlačení přetrénování patří bezpochyby tzv. *dropout*. Ten spočívá ve vypínání výpočetních jednotek (neuronů, konvolučních filtrů) v neuronové síti. Vypnutím jednotky je myšleno její dočasné odstranění ze sítě společně se všemi jejími příchozími a odchozími spojeními. Příklad neuronové sítě po použití dropoutu je na obrázku 3.4.



Obrázek 3.4: Ukázka efektů aplikace dropoutu s pravděpodobností 0,5 na hustou vrstevnatou síť se dvěma skrytými vrstvami [19].

Výběr jednotek, které mají být vypnuty probíhá náhodně a je prováděn znovu každou trénovací epochou. Obvykle je každá jednotka nezávisle vypnuta s předdefinovanou pravděpodobností $p \in (0, 1)$, té se říká dropout rate. Jednotky jsou vypínány pouze při učení sítě, při inferenci jsou

všechny ponechány zapnuté. Jelikož je při trénování sítě v každé vrstvě aktivní pouze p -násobek jejích jednotek, je zapotřebí při inferenci přenásobit všechny váhy vycházející z této vrstvy hodnotou $(1 - p)$, tak aby součet přes všechny vstupy v následující vrstvě zůstal stejný jako při trénování.

Náhodný dropout zabraňuje neuronovým sítím příliš spoléhat na několik málo neuronů s vysokými hodnotami vah, což je jeden z hlavních problémů způsobujících přetrénování. V [19] je ukázáno, že dropout významně zlepšuje úspěšnost neuronových sítí v řadě aplikací.

Batch-normalizace. Batch-normalizace je normalizační technika, která se snaží řešit problém proměnlivosti rozdělení vstupních dat pro danou vrstvu. Tvar rozdělení vstupních dat je dán předchozími vrstvami, jejichž parametry se během tréninku mění. To zpomaluje učení sítě a vyžaduje volit nižší učící krok. Jak název napovídá, je batch-normalizace založena na práci s celými mini-batchemi, viz sekce 3.2.4. V článku [20], představujícím batch-normalizaci, je navrženo ji využívat pro úpravu vstupů před vrstvami. V praxi ji v této práci využíváme pouze před vybranými aktivačními vrstvami. Ukazuje se, že její použití přispívá k regularizaci učícího procesu sítě.

Batch-normalizace vstupní matice $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(d)}) \in \mathbb{R}^{N,d}$, kde N je velikost mini-batche a d dimenze jednoho pozorování, provede normalizaci zvlášť pro každý kanál

$$\hat{\mathbf{x}}^{(k)} = \frac{\mathbf{x}^{(k)} - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}}, \quad (3.1)$$

kde v čitateli odčítáme skalár μ_k od vektoru po složkách a $\epsilon = 10^{-5}$ je přičteno pro prevenci dělení nulou. Hodnoty μ_k , resp. σ_k^2 jsou odhady střední hodnoty, resp. rozptylu hodnot z daného minibatche, definujeme je tedy jako

$$\mu_k = \frac{1}{N} \sum_{i=1}^N x_i^{(k)},$$

$$\sigma_k^2 = \frac{1}{N} \sum_{i=1}^N (x_i^{(k)} - \mu_k)^2.$$

Pouhou normalizací dat vstupujících do vrstvy bychom potenciálně omezili, co může daná vrstva reprezentovat. Proto je batch-normalizace rozšířena tak, aby mohla za určitých podmínek představovat i identickou transformaci. Za tímto účelem použijeme dvojici parametrů $\gamma^{(k)}, \beta^{(k)}$, které škálují a posouvají normalizované hodnoty

$$\mathbf{y}^{(k)} = \gamma^{(k)} \hat{\mathbf{x}}^{(k)} + \beta^{(k)}, \quad \forall k \in \hat{d}.$$

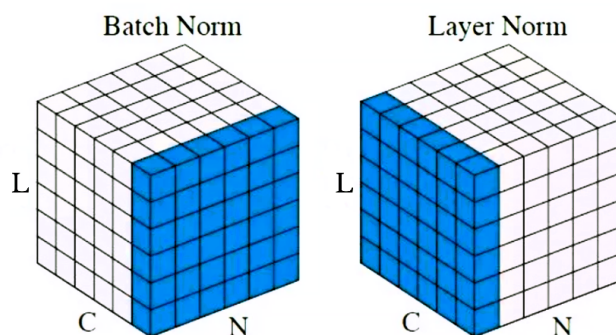
Parametry $\gamma^{(k)}, \beta^{(k)}$, jsou trénovatelné a jejich hodnota je stanovena v průběhu učení sítě spolu s ostatními volnými parametry. Označme $\mathbf{Y} = (\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(d)}) \in \mathbb{R}^{N,d}$, potom batch-normalizaci tenzoru \mathbf{X} můžeme psát jako $\text{BatchNorm}(\mathbf{X}) = \mathbf{Y}$.

Pro batch-normalizaci dat v 1D konvolučních vrstvách, která jsou tenzorového tvaru $\mathbf{X} \in \mathbb{R}^{N,C,L}$, kde N je velikost mini-batche, C je počet kanálů a L je délka signálů, postupujeme obdobně. Data normalizujeme společně přes celý minibatch a všechny pozice signálu. Tzn. že normalizujeme přes všech $N \cdot L$ hodnot v daném kanálu napříč celou délkou všech signálů z mini-batche.

Při inferenčním použití sítě nemusí být vždy možné a žádoucí normalizovat data na základě statistik závislých na mini-batchi. Proto se pro odhady střední hodnoty a rozptylu v transformacích použijí všechna dostupná trénovací data.

Layer-normalizace. Layer-normalizace byla představena nedlouho po batch-normalizaci, se stejným záměrem urychlit a regularizovat učení sítě. Efekt batch-normalizace byl silně závislý na velikosti batche a nebylo možné ji použít v rekurentních neuronových sítích. Layer-normalizace, představená v [21], odbourává tyto problémy díky transpozici batch-normalizace, jak je naznačeno na obrázku 3.5. Výpočet odhadu střední hodnoty a rozptylu zde tedy probíhá napříč kanály přes celou délku signálu, avšak pro každé pozorování z mini-batche zvlášť.

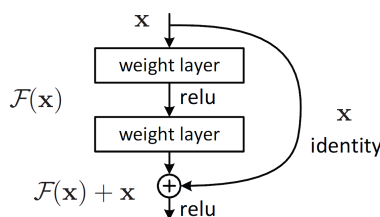
Stejně jako u batch-normalizace se po normalizování pozorování použijí trénovatelné parametry γ a β pro škálování a posunutí. Na rozdíl od batch-normalizace zde není rozdíl ve fungování mezi trénováním sítě a inferencí. Zároveň je snadné ji aplikovat do rekurentních sítí, kde pracuje zvlášť na každém časovém kroku.



Obrázek 3.5: Rozdíl mezi batch-normalizací a layer-normalizací na třírozměrném tenzoru.

Residuální spoje. Residuální spoj je obměna přístupu k použití ostatních vrstev v rámci neuronové sítě. Byl představen v [22] jako řešení problému trénování hlubších a hlubších neuronových sítí. Při klasickém přístupu, kdy data procházejí sítí z jedné vrstvy do druhé se při příliš velké hloubce musíme potýkat s problémy mizejícího/explodujícího gradientu. Dalším v článku popsaným problémem je degradace trénovací přesnosti při navyšování hloubky sítě.

Tyto problémy jsou adresovány residuálním učícím přístupem, kdy namísto abychom od několika vrstev očekávali, že se naučí aproximovat konkrétní přímé zobrazení $\mathcal{H}(\mathbf{x})$, necháme je aproximovat residuální zobrazení $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. Původní zobrazení je poté nahrazeno tvarem $\mathcal{F}(\mathbf{x}) + \mathbf{x}$, jak je přiblíženo na obrázku 3.6. Nabízená intuice ospravedlňující tento krok říká, že při modelování identického zobrazení je pro síť s nelineárními komponentami jednodušší modelovat nulový výstup, než identitu.



Obrázek 3.6: Příklad residuálního spoje [22].

3.2 Učení neuronové sítě

V předchozí sekci jsme popsali stavební bloky - vrstvy, ze kterých se neuronové sítě skládají. V této sekci se zaměříme na to, jak se neuronové sítě učí, tzn., jak na základě dostupných trénovacích dat upravují své volné parametry tak, aby minimalizovaly specifikovanou ztrátovou funkci. Volnými parametry myslíme zejména matice vah \mathbf{W} v hustých vrstvách a konvoluční jádra \mathbf{A} v konvolučních vrstvách.

Pro trénování neuronových sítí (stejně jako dalších supervised metod) je třeba mít trénovací dataset obsahující vedle množiny pozorování $\mathbf{X} = \{\mathbf{x}_i, i \in \hat{n}\}$ také množinu jim odpovídajících očekávaných výstupů sítě $\mathbf{Y} = \{\mathbf{y}_i, i \in \hat{n}\}$. Pozorování \mathbf{x}_i je poté vstupem do neuronové sítě a label \mathbf{y}_i požadovaným výstupem. Běžně se dostupná data rozdělují na tři části: trénovací, validační a testovací. Trénovací slouží k samotnému učení sítě, validační k optimalizaci trénovacího procesu a testovací data slouží k závěrečnému vyhodnocení úspěšnosti sítě.

3.2.1 Ztrátová funkce

Účelem ztrátové funkce je porovnávat míru chyby neuronové sítě v jejích predikcích oproti známým labelům. Volba ztrátové funkce tak musí odpovídat typu dat i účelu neuronové sítě.

Pro úlohu binární klasifikace předpokládáme labely tvaru $\mathbf{y} = (y_1, \dots, y_n), y_i \in \{0, 1\}, \forall i \in \hat{n}$, a výstupy neuronové sítě ve tvaru $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n), \hat{y}_i \in (0, 1), \forall i \in \hat{n}$. Potom zavádíme tzv. *binární křížovou entropii* (BCE) jako

$$L_{BCE}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{n} \sum_{i=1}^n [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)].$$

Pro úlohu klasifikace do $m > 2$ tříd předpokládáme labely $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)^T$ ve tvaru matice, pro jejíž prvky y_{ij} platí

$$y_{ij} = \begin{cases} 1 & \text{pokud } i\text{-té pozorování náleží do } j\text{-té třídy,} \\ 0 & \text{jinak.} \end{cases}$$

Jedná se o tzv. *one-hot encoding* příslušnosti ke třídám. Výstup neuronové sítě očekáváme ve tvaru matice $\hat{\mathbf{Y}} \in \mathbb{R}^{n,m}$, pro jejíž prvky platí $\hat{y}_{ij} \in (0, 1)$, a zároveň $\sum_{j=1}^m \hat{y}_{ij} = 1, \forall i \in \hat{n}$. Potom zavádíme *kategorickou křížovou entropii* (CE) jako

$$L_{CE}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m y_{ij} \log \hat{y}_{ij}.$$

Pro regresní aplikaci neuronové sítě využijeme jako ztrátovou funkci střední kvadratickou odchylku. V tomto případě nahrazuje labely závislá proměnná, kterou se snažíme modelovat, označme ji pro i -té pozorování $\mathbf{y}_i \in \mathbb{R}^d, d \geq 1$. Matice hodnot závislé proměnné $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)^T$ bude stejně jako výstup sítě $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_n)^T$ ležet v prostoru $\mathbb{R}^{n,d}$. Ztrátovou funkci střední kvadratické odchylky (MSE) zavádíme jako

$$L_{MSE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2.$$

Další možnou ztrátovou funkcí pro regresní úlohu je střední absolutní chyba (MAE). Tu s využitím stejného značení jako pro MSE zavádíme jako

$$L_{MAE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_1.$$

3.2.2 Inicializace vah

Před začátkem trénování neuronové sítě je zapotřebí inicializovat všechny volné parametry. Nevhodná inicializace parametrů může způsobit problémy při učení sítě [11], např. při konstantní inicializaci by síť byla symetrická a učení by kvůli symetrickému gradientu nebylo možné. Navíc je z důvodu ztrácejícího se gradientu (vanishing gradient problem) nutné zajistit, aby hodnota vah nebyla příliš vysoká nebo nízká. To by mohlo vést k téměř nulovému gradientu použité aktivační funkce, což by znemožnilo další efektivní učení. Dále může být problematické náhodně inicializovat váhy v celé síti ze stejného rozdělení, v takovém případě se některé vrstvy mohou učit rychleji než jiné, což snižuje přínosy použití více vrstev.

Představíme dvě techniky inicializace parametrů. Pro hustou vrstvu označím N_{in} počet neuronů v předcházející vrstvě a N_{out} počet neuronů v následující vrstvě. Pro konvoluční vrstvu označím jako N_{in} součin délky konvolučního jádra a počtu kanálů na vstupu a N_{out} součin délky konvolučního jádra a počtu kanálů na výstupu.

Glorot inicializace. Pro neuronové sítě s aktivačními funkcemi *sigmoid* a *tanh* byla v [12] odvozena inicializace parametrů, pro níž je rozdělení aktivací v dané vrstvě blízké rozdělení gradientů při zpětném průchodu sítí, což pomáhá vyváženému trénování napříč vrstvami hluboké sítě. Pro náhodnou inicializaci vah W_{ij} je doporučeno rozdělení *Glorot Uniform* a *Glorot Normal*

$$W_{ij} \sim \mathcal{U} \left[-\frac{\sqrt{6}}{\sqrt{N_{in} + N_{out}}}, \frac{\sqrt{6}}{\sqrt{N_{in} + N_{out}}} \right],$$
$$W_{ij} \sim \mathcal{N} \left(0, \frac{2}{N_{in} + N_{out}} \right).$$

He inicializace. Glorot inicializace funguje dobře pro aktivační funkce, které lze okolo nuly lineárně aproximovat, pro aktivační funkce podobné ReLU se však ukazuje jako neoptimální [13]. Z toho důvodu He et al. ukázali, že je vhodné přeskálovat rozptyl určitou konstantou závislou na aktivační funkci (2 pro ReLU), zároveň vypustili z výpočtu hodnotu N_{out} , která se ukázala jako nepotřebná. Výsledná rozdělení pro ReLU aktivaci *He Uniform* a *He Normal* mají tvar

$$W_{ij} \sim \mathcal{U} \left[-\frac{\sqrt{6}}{\sqrt{N_{in}}}, \frac{\sqrt{6}}{\sqrt{N_{in}}} \right] \quad \text{a} \quad W_{ij} \sim \mathcal{N} \left(0, \frac{2}{N_{in}} \right).$$

3.2.3 Backpropagation

Zpětné šíření chyby (backpropagation) je fundamentálním algoritmem pro trénování neuronových sítí. Umožňuje jim se "učit" z trénovacích dat iterativním upravováním volných parametrů (vah). Backpropagation se skládá ze dvou hlavních částí - dopředný průchod a zpětný průchod. Každá neuronová síť se skládá z vrstev, které tvoří výpočetní graf.

Během dopředného průchodu jsou vstupní trénovací data postupně zpracována vrstvu po vrstvě až po vytvoření výstupu sítě. Tento proces je možné vyjádřit jako sérii maticových násobení, díky čemuž je možné ho paralelizovat a provádět velmi rychle. Na počátku trénování jsou parametry (matice vah v hustých vrstvách a jádra v konvolučních vrstvách) inicializovány náhodně, od výstupu ze sítě tedy nelze očekávat že by se blížil požadované hodnotě dané cílovým výstupem Z trénovacího datasetu. Za pomoci ztrátové funkce je vyhodnocena chyba mezi výstupem sítě a cílovou hodnotou.

Zpětný průchod sítí dále šíří tuto chybu nazpět sítí, opět vrstvu po vrstvě. Pro každý parametr každé vrstvy je spočten gradient ztrátové funkce za pomoci řetězového pravidla. Bližší popis způsobu výpočtu gradientů lze nalézt např. v [11]. Záporná hodnota tohoto gradientu vyjadřuje směr změny parametrů sítě, která povede k největšímu poklesu ztrátové funkce.

3.2.4 Algoritmy učení

Jakmile jsou gradienty spočteny, optimalizační algoritmus upraví parametry sítě ve směru daném gradientem. Iterativní úpravou parametrů na základě spočtených gradientů sítí postupně zlepšuje své predikce. Cílem optimalizace je dosáhnout globálního minima ztrátové funkce, to však v praxi není téměř nikdy dosažitelné. Stabilní dosažení lokálního minima je ale pro reálné aplikace obvykle dostačující.

Nejjednodušší variantou gradientního sestupu je tzv. *batch gradient descent* (BGD), který v každé iteraci upravuje parametry za použití všech pozorování v trénovacím datasetu. Jeden krok BGD můžeme pro učící krok (learning rate) η zapsat jako

$$\theta_{n+1} = \theta_n - \eta \nabla_{\theta} L(\theta_n),$$

kde θ_n jsou parametry sítě v n -té iteraci a $\nabla_{\theta} L(\theta_n)$ představuje průměrný gradient ztrátové funkce ze všech trénovacích pozorování. Výhodou tohoto přístupu je, že zaručeně konverguje do globálního minima pro konvexní ztrátovou funkci a do lokálního minima pro nekonvexní ztráty [14]. Pro praktické použití je však BGD často nepoužitelný, protože může být velmi pomalý a je nutné, aby se všechna trénovací data vešla do paměti.

Upravená varianta BGD, která adresuje jeho hlavní problémy se nazývá mini-batch gradient descent (obvykle se používá označení stochastic gradient descent - SGD). Spočívá v náhodném rozdělení datasetu na podmnožiny (mini-batche) o velikosti b trénovacích pozorování. Následně upravuje váhy podle průměrného gradientu pro každý mini-batch. Tento přístup vede k snížení rozptylu úprav parametrů oproti čistému SGD (pro $b = 1$) a umožňuje využít velmi efektivní výpočetní optimalizace pro výpočet gradientu. Ukázalo se, že ačkoliv SGD obecně nemusí konvergovat stejným způsobem jako BGD, tak při postupném snižování učícího kroku vykazuje téměř stejné chování [14].

Ani SGD však sám o sobě nezaručuje dobrou konvergenci a trpí řadou problémů, např. učící krok má stejný pro všechny parametry, což může znevýhodnit vzácně se vyskytující příznaky v našich datech. Dále má SGD zásadní problém uniknout ze sedlového bodu ztrátové funkce, jelikož gradient ve všech směrech je téměř nulový.

Adam. Existuje řada algoritmů optimalizujících gradientní sestup, které se snaží adresovat výše zmíněné problémy, zde popíšeme pouze Adaptive Moment Estimation (Adam) algoritmus. Ten pro každý parametr volí vlastní adaptivní velikost učícího kroku. Označíme proto $g_{n,i}$ gradient ztrátové funkce vzhledem k parametru θ_i v n -té iteraci:

$$g_{n,i} = \nabla_{\theta_n} L(\theta_{n,i}),$$

které vytvoří vektor $\mathbf{g}_n = (g_{n,1}, \dots, g_{n,N})$, kde N je celkový počet parametrů sítě. Adam si uchovává exponenciálně klesající průměr minulých gradientů m_n a minulých kvadrátů gradientů v_n ,

$$\begin{aligned} m_n &= \beta_1 m_{n-1} + (1 - \beta_1) g_n, \\ v_n &= \beta_2 v_{n-1} + (1 - \beta_2) g_n^2, \end{aligned}$$

m_n a v_n jsou odhady prvního a druhého momentu gradientů, což dává metodě její jméno. Na počátku byly m_n a v_n inicializovány jako nulové vektory, což vedlo k jejich vychýlení v prvních iteracích [15]. Proto jsou použity korigované hodnoty

$$\hat{m}_n = \frac{m_n}{1 - \beta_1^n},$$

$$\hat{v}_n = \frac{v_n}{1 - \beta_2^n}.$$

Iterační krok pro úpravu vah s učícím krokem η je potom ve tvaru

$$\theta_{n+1} = \theta_n - \frac{\eta}{\sqrt{\hat{v}_n} + \epsilon} \hat{m}_n,$$

kde ϵ je malé kladné číslo přičtené pro prevenci dělení nulou. Standardně používané hodnoty parametrů zde jsou $\beta_1 = 0,9$, $\beta_2 = 0,999$ a $\epsilon = 10^{-8}$.

EPOCHOU v trénovacím procesu nazveme jeden průchod všemi dostupnými trénovacími daty. Ta jsou na začátku epochy rozdělena do $\lceil T/b \rceil$ mini-batchů, kde T je počet pozorování v trénovacím datasetu a b je velikost mini-batche, takže během epochy dojde právě k $\lceil T/b \rceil$ úpravám parametrů sítě. K řízení trénovacího procesu dále v rámci této práce využíváme tzv. *early stopping callback*. Ten sleduje vývoj hodnoty ztrátové funkce vyhodnocené na datech z validačního datasetu po každé epoše. Validační dataset nebyl použit k předchozí úpravě parametrů sítě. Pokud se sledovaná ztráta na validačním datasetu nesníží po předdefinovaný počet epoch je trénování ukončeno a parametry sítě se vrátí do stavu z epochy s nejnižší zaznamenanou ztrátou na validačním datasetu.

3.3 Transformer

V roce 2017 byl v přelomovém článku *Attention Is All You Need* představen nový model nazvaný Transformer. Ten spoléhá pouze na tzv. *attention* mechanismus a vynechává, do té doby nepostradatelné, konvoluční a rekurentní vrstvy. Původní představená podoba transformeru se zabývá řešením problému transkuce sekvencí (*sequence-to-sequence modelling*) v oboru zpracování přirozeného jazyka (NLP), konkrétně strojovým překladem jazyka. Od jejího uvedení bylo představeno množství upravených a vylepšených architektur na bázi transformeru se zaměřením na nejrůznější problematiku. V této části popíšeme původní, tzv. *vanilla transformer*, upravená verze transformeru použitá pro zpracování signálů AE je popsána v kapitole 4.

3.3.1 Attention mechanismus

Začneme popisem klíčové součásti používané v rámci tranformeru - attention mechanismu. Jak název napovídá, tato metoda dovoluje modelu soustředit se na vzájemně relevantní části analyzovaných sekvencí dat přiřazením různých úrovní významnosti různým částem sekvence. To umožňuje zachycení kontextu v datech a pomáhá zpracování vzdálenějších závislostí v sekvenci, což byl hlavní problém rekurentních sítí.

Attention mechanismus pracuje s posloupnostmi n vektorů *values*, *queries* a *keys* uspořádanými do matic

$$\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_n)^T \in R^{n, d_v},$$

$$\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_n)^T \in R^{n, d_q},$$

$$\mathbf{K} = (\mathbf{k}_1, \dots, \mathbf{k}_n)^T \in R^{n, d_q}.$$

V praxi se obvykle dimenze *values* volí rovna dimenzi *queries* a *keys*, proto i zde dále předpokládáme $d_v = d_q$. Existuje řada funkcí použitelných pro výpočet *attention*, stejně jako původní autoři budeme vždy používat škálovaný skalární součin.

Definice 3.3.1 (Attention podle škálovaného skalárního součinu). *Nechť $\mathbf{V}, \mathbf{Q}, \mathbf{K} \in \mathbb{R}^{n, d_q}$ jsou matice values, queries a keys s řádky $\mathbf{v}_i^T, \mathbf{q}_i^T, \mathbf{k}_i^T \in \mathbb{R}^{d_q}, \forall i \in \hat{n}$. Potom attention podle škálovaného skalárního součinu je dáno jako*

$$z_i = \text{attention}(\mathbf{v}_1, \dots, \mathbf{v}_n, \mathbf{q}_i, \mathbf{k}_1, \dots, \mathbf{k}_n) = \sum_{j=1}^n \text{softmax} \left(\frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d_q}} \right) \mathbf{v}_j, \quad (3.2)$$

což lze přepsat do maticového zápisu

$$\mathbf{Z} = \text{attention}(\mathbf{V}, \mathbf{Q}, \mathbf{K}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_q}} \right) \mathbf{V},$$

za dodefinování funkce softmax pro matice jako

$$\text{softmax}(\mathbf{Z})_{i,j} = \frac{e^{z_{i,j}}}{\sum_{l=1}^{d_q} e^{z_{i,l}}}, \quad \forall i \in \hat{n}, \forall j \in \hat{d}_q.$$

Na tomto místě je zapotřebí se podívat, jak dochází v transformeru k získání matic $\mathbf{Q}, \mathbf{K}, \mathbf{V}$. Mějme vstupní sekvenci dat $(\mathbf{x}_1, \dots, \mathbf{x}_n) = \mathbf{X} \in \mathbb{R}^{n, d}$, kde d nazýváme dimenzí modelu. Matice $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ dostáváme třemi různými lineárními transformacemi vstupu \mathbf{X} za použití příslušných matic vah

$$\mathbf{V} = \mathbf{X}\mathbf{W}_v, \quad \mathbf{Q} = \mathbf{X}\mathbf{W}_q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_k, \quad (3.3)$$

kde $\mathbf{W}_v, \mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d, d_q}$ jsou matice vah transformací, které jsou trénovatelné a jejich hodnoty jsou tak určeny až během učení sítě.

Aby se modelu umožnilo více specializovat a obsáhnout různé kontextuální závislosti v datech, provádí se výpočet *attention* paralelně a nezávisle v tzv. *hlavách*. Matice vah transformací se tak mohou v každé hlavě učit jinak a soustředit se na jiný aspekt dat. Tento proces se nazývá multi-head self-attention.

Definice 3.3.2 (Multi-head self-attention). *Nechť $\mathbf{X} \in \mathbb{R}^{n, d}$ jsou vstupní sekvenci data, $h \in \mathbb{N}$ je počet hlav, $d \in \mathbb{N}$ je dimenze modelu a $d_q \in \mathbb{N}$ je dimenze hlavy. Potom výstup i -té hlavy získáme jako*

$$\mathbf{H}_i = \text{attention} \left(\mathbf{X}\mathbf{W}_v^{(i)}, \mathbf{X}\mathbf{W}_q^{(i)}, \mathbf{X}\mathbf{W}_k^{(i)} \right),$$

kde $\mathbf{W}_v^{(i)}, \mathbf{W}_q^{(i)}, \mathbf{W}_k^{(i)} \in \mathbb{R}^{d, d_q}$ jsou trénovatelné matice vah i -té hlavy. Označíme $\widehat{\mathbf{H}} = (\mathbf{H}_1, \dots, \mathbf{H}_h) \in \mathbb{R}^{n, h d_q}$ spojení (concatenation) výstupů jednotlivých hlav. Výsledná multi-head self-attention funkce je konečně dána jako

$$\text{MultiHead}(\mathbf{X}) = \widehat{\mathbf{H}}\mathbf{W}_o,$$

kde $\mathbf{W}_o \in \mathbb{R}^{h d_q, d}$ je trénovatelná matice transformace výstupů.

Multi-head self-attention je pouze jedním ze tří různých bloků založených na principu attention, které jsou v originální struktuře transformeru použity. Jedná se však o nejdůležitější z nich, zejména vzhledem k upravené podobě transformeru, který dále využíváme pro analýzu

časových řad, proto zde zbylé dva popíšeme pouze ve stručnosti. Zevrubnější popis lze nalézt např. v [17].

Masked multi-head self-attention pracuje velmi obdobně jako popsaný self-attention, vzhledem k jeho použití v dekodéru (viz obrázek 3.7) je však nutné zajistit kauzalitu modelu, tzn. zajistit aby data nevytvářela kontext s následujícími členy sekvence a tím se neučila z budoucích dat. Za tímto účelem se využívá maskování všech naddiagonálních prvků matice v argumentu funkce *softmax* při výpočtu *attention* podle rovnice (3.2). Maskování spočívá v nahrazení těchto prvků hodnotou $-\infty$.

Multi-head cross-attention je druhou obměnou známého self-attention, který je opět použit pouze v dekodéru transformeru. Cross-attention pracuje na rozdíl od předchozích dvou variant se dvěma vstupy dat, nazvěme je $\mathbf{X}_{enc} \in \mathbb{R}^{n,d}$, $\mathbf{X}_{dec} \in \mathbb{R}^{m,d}$ v souladu s jejich původem jakožto výstupem z enkodéru, resp. vstupem do dekodéru. Jediným rozdílem oproti multi-head self-attention je způsob výpočtu matic \mathbf{V} , \mathbf{Q} , \mathbf{K} , oproti (3.3), zde je vytváříme jako

$$\mathbf{V} = \mathbf{X}_{enc} \mathbf{W}_v, \quad \mathbf{Q} = \mathbf{X}_{dec} \mathbf{W}_q, \quad \mathbf{K} = \mathbf{X}_{enc} \mathbf{W}_k.$$

Rozdílná dimenze matice \mathbf{Q} se propíše do dimenze finálního výstupu multi-head cross-attention, který bude ležet v $\mathbb{R}^{m,d}$.

3.3.2 Poziční kódování

V rámci transformeru doposud nebyla nikde představena komponenta, který by brala v potaz časovou souslednost v sekvenčních datech. Při výpočtu *attention* nehraje pořadí pozorování v matici vstupních dat roli. Jelikož časové pořadí je pro většinu aplikací naprosto zásadní, bylo představeno poziční kódování (*positional encoding* - PE). To vytváří vektory jejichž přičtení vstupní data obohacuje o informaci o jejich relativní pozici v rámci sekvence.

V rámci původního článku [16] byl navržen výpočet vektoru PE za použití sinusoidálních funkcí následujícím způsobem.

Definice 3.3.3 (Poziční kódování). *Mějme vstupní sekvenci dat $\mathbf{X} \in \mathbb{R}^{n,d}$, kde n je délka vstupní sekvence, d je sudé číslo značící dimenzi modelu a $\tau \in \mathbb{N}$ je konstanta. Potom definujeme vektor pozičního kódování $\mathbf{p}_i \in \mathbb{R}^d$ pro pozici $i \in \hat{n}$ jako*

$$(\mathbf{p}_i)_j = \begin{cases} \sin(\omega_l \cdot i) & \text{pro } j = 2l, \\ \cos(\omega_l \cdot i) & \text{pro } j = 2l - 1, \end{cases}$$

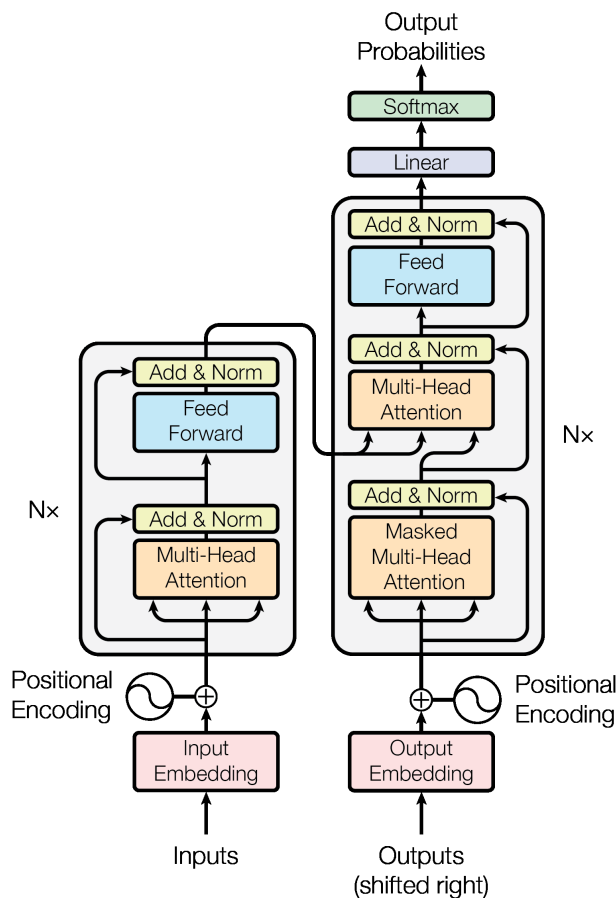
kde

$$\omega_l = \frac{1}{\tau^{\frac{2(l-1)}{d}}}, \quad \forall l \in \left\{1, \dots, \frac{d}{2}\right\}.$$

Hodnota konstanty τ je zpravidla volena jako $\tau = 10000$.

3.3.3 Encoder - Decoder

Konečně, na obrázku 3.7 můžeme vidět vizualizaci celé architektury původního transformeru. Obdobně jako předešlé architektury zabývající se strojovým překladem (např. na bázi rekurentních sítí) jej lze rozdělit na enkodér (levá část) a dekodér (pravá část). Obecně enkodér zpracovává potenciálně různě dlouhé vstupní sekvence a vytváří jejich abstraktní reprezentaci v podobě vektorů o fixní délce. Dekodér potom z této abstraktní reprezentace vytváří různě dlouhé výstupní sekvence.



Obrázek 3.7: *Architektura původního transformeru z [16].*

Vstupem do enkodéru je text v jednom jazyce v podobě sekvence tokenů z předdefinovaného slovníku o délce d_s . Tu je třeba nejprve předzpracovat. Každý token je na počátku reprezentován tzv. *one-hot* vektorem, tj. vektorem nul délky d_s , kde pouze člen vektoru na pozici slovníku odpovídající danému tokenu má hodnotu 1. Tuto sekvenci označíme jako $\mathbf{X}_{in} \in \{0, 1\}^{n, d_s} \subset \mathbb{R}^{n, d_s}$. Dalším krokem je aplikace embeddingu pro zobrazení jednotlivých tokenů z diskrétního prostoru do spojitého prostoru \mathbb{R}^d . Role embeddingu je klíčová, protože je třeba, aby zachoval informaci o sémantickém a syntaktickém významu jednotlivých tokenů tak, aby následně mohl transformer hodnotit kontextuální význam mezi tokeny. Příkladem modelu použitelného pro účely embeddingu je i *word2vec* [18] s českou stopou. V původní podobě transformeru je však pro embedding využita jednoduchá lineární transformace za použití matice vah $\mathbf{W}_{in} \in \mathbb{R}^{d_s, d}$. Tato matice je na počátku trénování inicializována náhodně a spolu s ostatními volnými parametry je upravována během trénování sítě. Po embeddingu přičteme k sekvenci poziční kódování, finální podobu dat $\mathbf{X}_0 \in \mathbb{R}^{n, d}$ vstupujících do enkodéru tak můžeme dostat jako

$$\mathbf{X}_0 = \mathbf{X}_{in} \mathbf{W}_{in} + \mathbf{P},$$

kde $\mathbf{P} = (\mathbf{p}_1, \dots, \mathbf{p}_n)^T$ je sada vektorů pozičního kódování sestavená podle definice 3.3.3.

Samotný enkodér se skládá z N -krát opakované, na sebe navazující enkodérové vrstvy. Každá enkodérová vrstva se skládá ze dvou částí. V první části se provede multi-head self-attention, který je obtečen residuálním spojem a normalizován pomocí *layer normalization*, tzn. že výstup

první části $\mathbf{X}_1 \in \mathbb{R}^{n,d}$ lze zapsat jako

$$\mathbf{X}_1 = \text{LayerNorm}(\mathbf{X}_0 + \text{MultiHead}(\mathbf{X}_0)).$$

V druhé části enkodérové vrstvy procházejí data hustou vrstevnatou sítí s jednou skrytou vrstvou s aktivací ReLU, která je aplikována na každou pozici v sekvenci zvlášť a identicky. Tato vrstva je opět obtečena residuálním spojem a po sečtení normalizována. To můžeme celkem pro výstup z druhé části $\mathbf{X}_2 \in \mathbb{R}^{n,d}$ zapsat jako

$$\mathbf{X}_2 = \text{LayerNorm}\left(\mathbf{X}_1 + \max\{0, \mathbf{X}_1 \mathbf{W}_1 + \mathbf{b}_1\} \mathbf{W}_2 + \mathbf{b}_2\right),$$

kde $\mathbf{W}_1 \in \mathbb{R}^{d,d_f}$, $\mathbf{W}_2 \in \mathbb{R}^{d_f,d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_f}$, $\mathbf{b}_2 \in \mathbb{R}^d$, jsou trénovatelné parametry vrstevnaté sítě. Dimenze skryté vrstvy je obvykle volena jako $d_f = 4d$.

Vstupem do dekodéru při trénování původního transformeru je překlad vstupního textu do kýženého jazyka. Tento text je stejně jako pro enkodér převeden na tokeny, embedován pomocí stejné matice \mathbf{W}_{in} a obohacen o poziční kódování.

Blok dekodéru je konstruován obdobným stylem z N -krát opakované dekodérové vrstvy. Ta se skládá celkem ze tří částí, které jsou všechny jednotlivě obtečeny residuálním spojem a normalizovány pomocí *layer* normalizace, stejně jako v enkodéru. Nejprve vstup projde vrstvou masked multi-head self-attention. Následuje vrstva multi-head cross-attention, ve které se *queries* vytvářejí použitím výstupu první části dekodérového bloku, ale *values* a *keys* jsou spočteny za použití výstupu z poslední vrstvy enkodéru. Tato část dekodéru jako jediná zprostředkovává spojení s enkodérem. Poslední částí dekodérové vrstvy je, stejně jako v enkodéru, hustá vrstevnatá síť stejného typu.

Výstup z dekodéru je předán lineární vrstvě, která provede tzv. *unembedding*, tedy převede vnitřní interpretaci členů sekvence o rozměru d zpět na vektor mapující slovník tokenů o délce d_s . K tomu je použita transformační matice $\mathbf{W}_{out} = \mathbf{W}_{in}^T$. Poslední vrstvou je aktivační funkce *softmax* pro získání pravděpodobností tokenů ze slovníku. Ztrátová funkce je spočtena porovnáním těchto pravděpodobností s očekávaným výstupem a na jejím základě se při tréninku upravují parametry.

Při inferenčním použití transformeru, kdy máme k dispozici pouze sekvenci tokenů překládaného jazyka, se práce enkodéru nijak nemění. Změna nastává v práci dekodéru, který na začátku dostane jako vstup pouze token značící začátek věty a provádí predikce iterativně. Po provedení každé predikce je výsledný token přidán na konec sekvence, která vstupovala do dekodéru a v další iteraci je použita jako vstup tato prodloužená sekvence. Tento proces je ukončen pokud je predikován token značící konec sekvence nebo dosažena maximální délka sekvence.

Kapitola 4

Klasifikační metody

V rámci této práce představíme celkem čtyři různé typy neuronových sítí, za účelem porovnání jejich využitelnosti pro analýzu dat pocházejících z měření AE. Zmíněné čtyři typy sítí se liší koncepčně, využívají jiné mechanismy ke zpracování dat nebo výrazně rozšiřují využitelnost těch stávajících. V rámci každého z typů jsou konkrétní implementované neuronové sítě závislé na řadě hyperparametrů (hloubka neuronové sítě, délky filtrů, apod.), optimalizace těchto hyperparametrů v rámci každé třídy v závislosti na konkrétních datech je rozsáhlý netriviální problém. V praxi jsou hyperparametry často voleny pouze na základě intuice.

4.1 Metriky úspěšnosti klasifikace

Pro porovnání jednotlivých klasifikátorů mezi sebou i pro posouzení jejich celkové využitelnosti je potřebné zavést několik metrik úspěšnosti klasifikace. Pro úlohu klasifikace do $p \in \mathbb{N}$ tříd předpokládáme že máme k dispozici označovaný dataset, tedy pozorování $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ pocházející z p tříd (C_1, \dots, C_p) a jejich příslušné labely (y_1, \dots, y_n) , $y_i \in \hat{p}$. Tento dataset si obvykle náhodně rozdělíme na trénovací, validační a testovací množinu v poměru 7 : 2 : 1. Data z trénovací a validační množiny jsou použita v rámci učení dané sítě, jejich využití pro posouzení úspěšnosti sítě tedy není vhodné, neboť síť během trénovacího procesu využila znalosti jejich labelů. Úspěšnost sítě na datech z trénovací množiny je typicky větší než na neznámých datech, je to jedním z ukazatelů přetrénování metody, tomu se však někdy nelze vyhnout.

Posouzení úspěšnosti dané klasifikační metody tedy bude vycházet zejména z její úspěšnosti na testovací části datasetu $\mathbf{X}^t = (\mathbf{x}_1^t, \dots, \mathbf{x}_{n_t}^t) \subset \mathbf{X}$, kde se pro každé pozorování \mathbf{x}_i^t bude snažit odhadnout jeho třídu \hat{y}_i . Tento odhad následně můžeme porovnat se skutečným labelem pozorování y_i , ke kterému síť neměla přístup. Pro zaznamenání výsledku klasifikace se zavádí matice záměn, na níž jsou poté definovány skalární metriky kvality klasifikace. Matice záměn $\mathbf{Z} \in \mathbb{R}^{p,p}$, sčítá v prvku Z_{ij} počet všech pozorování pocházejících ze třídy C_i klasifikovaných do třídy C_j .

$$Z_{ij} = \sum_{l=1}^{n_t} \mathbb{1}_{\{y_l=i\}} \mathbb{1}_{\{\hat{y}_l=j\}}.$$

Pro bezchybnou klasifikaci by tedy měla matice záměn tvar $\mathbf{Z} = \text{diag}(n_1, \dots, n_p)$ kde n_i je počet pozorování ve třídě C_i .

Metriky kvality klasifikace pro problém více tříd zavádíme podle [23] jako zobecnění metrik pro binární klasifikační problém. Metriky se zobecňují jako průměr binárních metrik při zjednodušení na klasifikaci jedna třída vs. ostatní. S tímto zjednodušením zavádíme pro každou třídu

C_i klasifikační čítecí hodnoty TP_i (True positive), TN_i (True negative), FP_i (False positive) a FN_i (False negative), spočtené z matice \mathbf{Z} následovně

$$\begin{aligned} TP_i &= Z_{ii}, & TN_i &= \sum_{\substack{j,k=1 \\ j,k \neq i}}^p Z_{jk}, \\ FP_i &= \sum_{\substack{j=1 \\ j \neq i}}^p Z_{ji}, & FN_i &= \sum_{\substack{j=1 \\ j \neq i}}^p Z_{ij}. \end{aligned}$$

Metriky lze poté zavést dvěma přístupy k průměrování. Macro přístup (zn. M) průměruje binární metriky spočtené pro každou třídu zvlášť, zatímco micro přístup (zn. μ) počítá příslušné čítecí hodnoty a z nich počítá metriky. Macro přístup dává každé třídě stejnou váhu, micro přístup bere v potaz množství pozorování v dané třídě. Základní metriky pro problém více tříd zavádíme následovně

$$\begin{aligned} \text{Průměrná přesnost:} & \quad \frac{1}{p} \sum_{i=1}^p \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}, \\ \text{Průměrná chybovost:} & \quad \frac{1}{p} \sum_{i=1}^p \frac{FP_i + FN_i}{TP_i + TN_i + FP_i + FN_i}, \end{aligned}$$

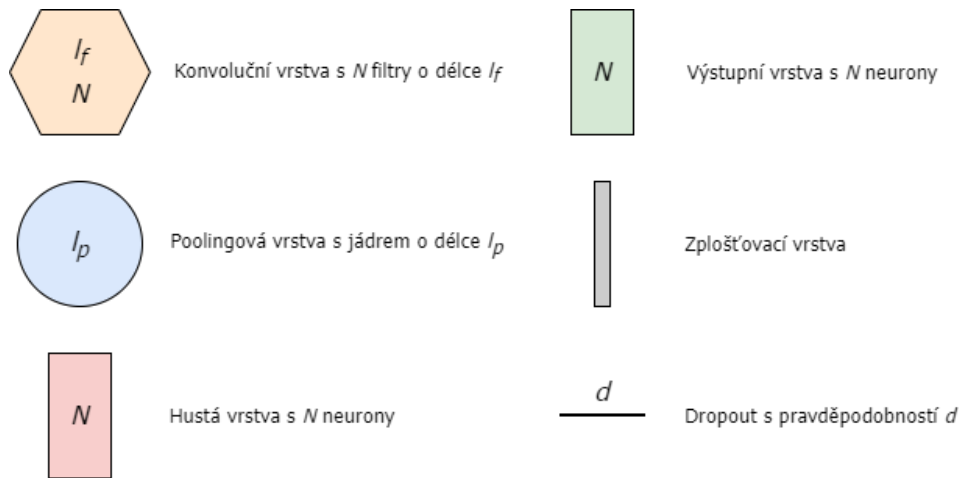
$$\begin{aligned} \text{Precision}_M &= \frac{1}{p} \sum_{i=1}^p \frac{TP_i}{TP_i + FP_i}, & \text{Precision}_\mu &= \frac{\sum_{i=1}^p TP_i}{\sum_{i=1}^p (TP_i + FP_i)}, \\ \text{Recall}_M &= \frac{1}{p} \sum_{i=1}^p \frac{TP_i}{TP_i + FN_i}, & \text{Recall}_\mu &= \frac{\sum_{i=1}^p TP_i}{\sum_{i=1}^p (TP_i + FN_i)}, \\ \text{F}_1\text{score}_M &= \frac{1}{p} \sum_{i=1}^p \frac{2TP_i}{2TP_i + FP_i + FN_i}, & \text{F}_1\text{score}_\mu &= \frac{\sum_{i=1}^p 2TP_i}{\sum_{i=1}^p (2TP_i + FP_i + FN_i)}. \end{aligned}$$

I vzhledem k faktu, že ve zde používaných datasetech máme vyvážené velikosti jednotlivých tříd pracujeme obvykle pouze s nejnázne interpretovatelnou metrikou průměrné přesnosti (*accuracy*).

4.2 Conv2Net

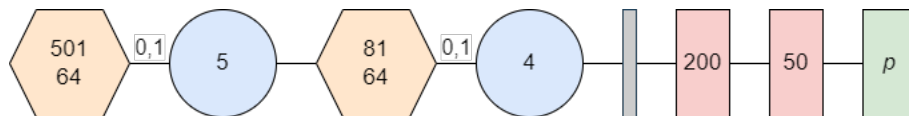
První z typů neuronové sítě, které využíváme je jednoduchá dopředná konvoluční síť inspirovaná architekturou pro zpracování obrazových dat LeNet-5 [10]. Ta využívá dvou konvolučních vrstev v tandemu s max-poolingovými vrstvami pro extrakci příznaků ze vstupních signálů, které následně po zploštění dat zpracuje hustá vrstevnatá část sítě. Max-pooling je zapotřebí využít pro snížení objemu dat při průchodu sítí vzhledem k relativně velké délce zpracovávaných signálů.

Pro vizualizaci budeme na příhodných místech využívat piktogramy popsané v obrázku 4.1.



Obrázek 4.1: Popis piktogramů využívaných pro vizualizaci neuronových sítí.

Délka filtrů v konvolučních vrstvách musí být adaptována v závislosti na délce zpracovávaných signálů. Pro naše data se osvědčily délky filtrů 501 a 81. Po zploštění využíváme dvě skryté husté vrstvy s 200, resp. 50 neurony před výstupní vrstvou, která má počet neuronů odpovídající počtu tříd, do kterých klasifikujeme. Při klasifikaci do p tříd tedy můžeme síť Conv2Net charakterizovat pomocí schematického obrázku 4.2.

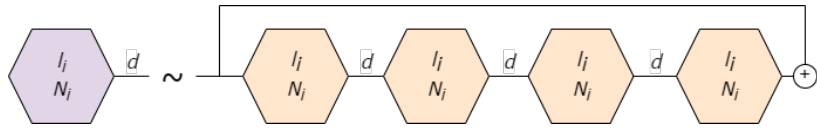


Obrázek 4.2: Architektura klasifikační sítě Conv2Net pro klasifikaci do p tříd.

4.3 ResNet-16

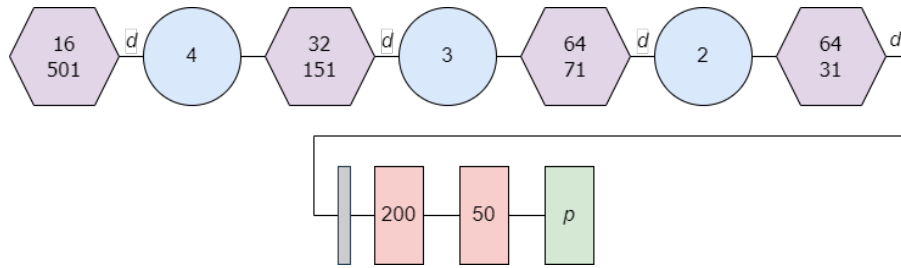
Druhým typem sítě, který využíváme, je hluboká konvoluční síť založená na residuálních blocích. Síť tohoto druhu byly pro efektivní využití představeny v článku [22] z roku 2015, který empiricky dokázal, že za využití residuálních spojů je možné efektivně trénovat i velmi hluboké neuronové sítě. Článek zároveň na několika veřejných obrazových datasetech (ImageNet, COCO) demonstroval zlepšení v klasifikačních a detekčních úlohách při využití velmi hlubokých residuálních sítí oproti do té doby špičkovým architektuám. Navrhované architektury pracovaly až se 152 konvolučními vrstvami.

Pro naše účely se spokojíme se sítí využívající celkem 16 konvolučních vrstev. Na obrázku 4.3 je náčrt jednoho residuálního bloku, který tvoří základ výsledné konvoluční sítě. Blok se skládá ze čtyř konvolučních vrstev se stejným počtem filtrů N_i a délkou jádra l_i . Po každé konvoluční vrstvě je použita batch-normalizace a následně aktivační ReLU vrstva. V případě poslední konvoluční vrstvy je aktivace aplikována až po přičtení residuálního spoje. Dropout pro regularizaci sítě je aplikován po prvních třech konvolučních vrstvách. V případě, že v rámci residuálního bloku dojde ke změně počtu kanálů v datech, tzn. N_i není stejné jako počet kanálů ve vstupních datech, je zapotřebí zohlednit tuto změnu dimenze i u netransformovaných dat obtékajících konvoluční vrstvy residuálním spojem. Za tímto účelem se na residuálním spoji využívá tzv. bottleneck, což je konvoluční vrstva s délkou jádra a stridem hodnoty 1 a počtem filtrů N_i , který umožňuje přímou zpětnou propagaci gradientu.



Obrázek 4.3: Jeden blok v síti ResNet-16 s residuálním spojem.

Síť ResNet-16 je tvořena celkem čtyřmi residuálními bloky v kombinaci s max-poolingovými vrstvami, které snižují délku transformovaných signálů, což nám umožňuje zvyšovat počet kanálů s hloubkou sítě za zachování rozumného objemu průchozích dat. Po zpracování vstupních signálů všemi konvolučními vrstvami je výsledný tensor dat zploštěn a zpracován dvěma skrytými hustými vrstvami se stejným počtem neuronů jako v případě sítě Conv2Net, jak je patrné z obrázku 4.4. Výstupní vrstva pro účely klasifikace má opět takový počet neuronů p jaký je počet tříd do kterých signály klasifikujeme.



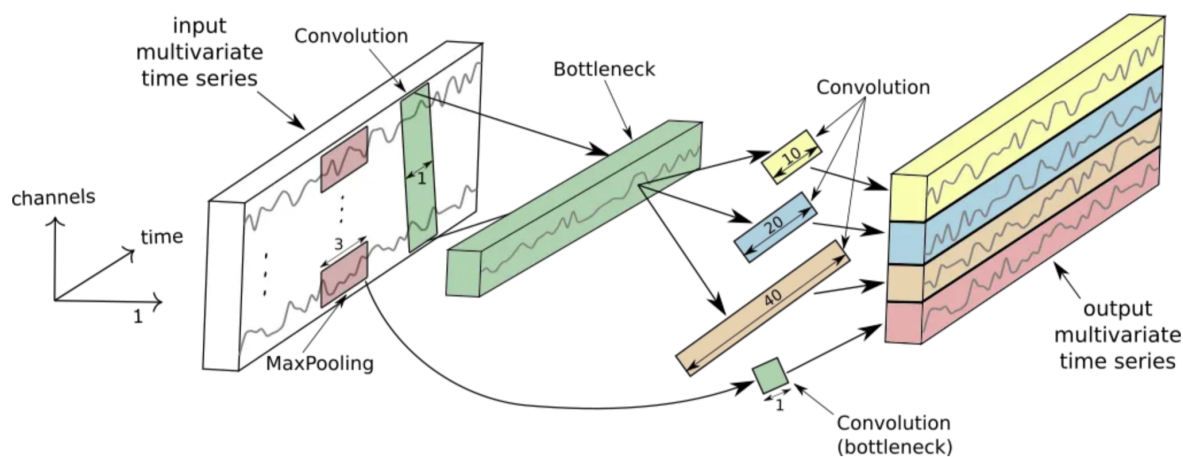
Obrázek 4.4: Struktura celé sítě ResNet-16.

4.4 Inception Time

Další využívanou architekturou je Inception Time představený v [24] v roce 2020. Průnik metod hlubokého učení do oboru klasifikace časových řad (*Time Series Classification - TSC*) byl oproti jiným odvětvím (zpracování obrazu, NLP) značně opožděn. V době publikace článku byl tak *state of the art* přístupem v oboru TSC algoritmus HIVE-COTE [26], ensemble metoda využívající řadu tradičnějších klasifikátorů. Inception Time, jakožto hluboká konvoluční neuronová síť, dokázal vyrovnat přesnost HIVE-COTE algoritmu na benchmarkových datasetech a nabídnout oproti němu nepoměrně lepší škálovatelnost. To je pro naše využití naprosto zásadní, časová složitost algoritmu HIVE-COTE je totiž $O(M^2 \cdot \tau^2)$ pro dataset s M časovými řadami o délce τ , což ho dělá pro většinu praktických aplikací nepoužitelným. Navíc Inception Time dle [25] konzistentně překonává metody založené na ResNet architektuře.

Při popisu Inception Time architektury neuronové sítě začneme popisem tzv. Inception modulu, který tvoří její základ a de facto zde nahrazuje tradiční konvoluční vrstvu. Visualizace tohoto modulu z [24] je na obrázku 4.5, oproti původní podobě architektury jsme provedli několik drobných změn adaptujících síť pro práci se signály AE, ty v průběhu popisu zmíníme. V levé části nákresu 4.5 máme vstup do Inception modulu v podobě vícerozměrné časové řady s d kanály. První hlavní částí modulu je tzv. *bottleneck*, který je implementován jako konvoluční vrstva s m filtry délky 1 s jednotkovým stridem (na obrázku pro jednoduchost $m = 1$). Ten transformuje vstupní časové řady z d kanálů do m kanálů. Jak název *bottleneck* napovídá je obvykle $m < d$, což zdatelně snižuje dimenzionalitu časových řad, vede ke snížení komplexity modelu a omezuje riziko přetrénování sítě. Zároveň to umožňuje následně použít výrazně delší

filtry v konvolučních vrstvách při zachování počtu trénovatelných parametrů oproti přístupu bez bottlenecku.

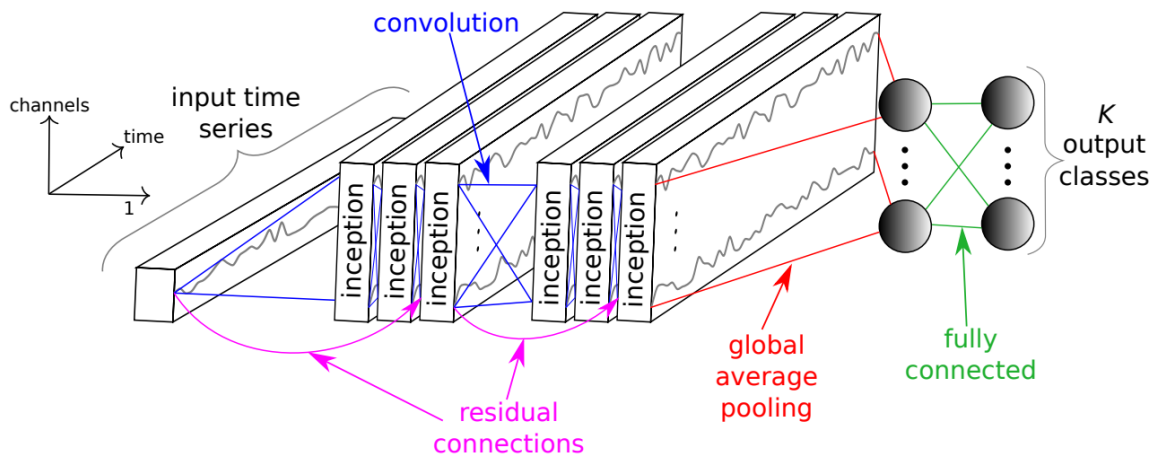


Obrázek 4.5: Struktura Inception modulu využívaného v Inception Time architektuře [24].

V případě dat z měření AE s nízkým počtem kanálů je v prvním inception modulu $m > d$ a bottleneck tak efektivně zvyšuje dimenzionalitu dat. Po průchodu *bottleneckem* jsou data souběžně zpracována třemi konvolučními vrstvami s různými délkami jader. V původním článku byla použita jádra s délkami $l \in \{10, 20, 40\}$, pro aplikaci na signálech akustické emise bylo nutné délku filtrů znatelně zvýšit. Po prozkoumání chování a na základě zkušeností s dříve implementovanými architekturami byly zvoleny délky $l \in \{41, 101, 301\}$. U všech konvolučních vrstev je využíván zero-padding, tak aby délka signálů zůstala stejná a počet filtrů je pro každou z vrstev roven počtu kanálů *bottlenecku*, tedy m . Navíc, paralelně s celým dosavadním tokem dat, je použit max-pooling s délkou okna 3 a stridem 1 na vstupních datech, jehož výstup je zpracován druhým *bottleneckem* obdobným způsobem jaký byl popsán výše, opět s výsledným počtem kanálů m . Konečně, výstup z Inception modulu je vytvořen spojením výstupů ze tří konvolučních vrstev a druhého *bottlenecku* podle kanálové dimenze. Data mají v tuto chvíli $4m$ kanálů, tento tenzor na závěr prochází batch-normalizací a aktivační vrstvou.

Na obrázku 4.6 převzatém z původního článku je vykreslena struktura celé Inception Time sítě. Ta vždy shlukuje tři Inception moduly do reziduálního bloku. V případě, že počet kanálů v datech na vstupu do reziduálního bloku není roven počtu kanálů v datech na výstupu z něj, jsou data procházející reziduálním spojem transformována do příslušné dimenze za využití principu *bottlenecku* tak, jak je popsán v předchozím odstavci. Síť navržená v původním článku se skládá ze dvou reziduálních bloků, tedy celkem ze šesti Inception modulů. Je důležité zmínit, že až doposud se v rámci sítě nijak nesnížila délka transformovaných časových řad. To jednak klade vysoké nároky na paměť během výpočtu a nutí nás k hledání kompromisu mezi hloubkou sítě, počtem kanálů uvnitř sítě a velikostí batche, za druhé to prakticky vylučuje využití klasického zploštění pro převedení transformovaných dat z tenzorové podoby do podoby vektoru, který by mohla zpracovat hustá vrstevnatá síť. Jako řešení druhého z problémů je v [24] navrženo využít tzv. global average pooling (GAP), ten zprůměruje tenzor vystupující z posledního Inception modulu podél celé časové dimenze. Tedy z tenzoru tvaru (N, C, L) dostaneme zprůměrováním podle poslední dimenze výrazně redukovaný výstup (N, C) , který už můžeme rozumně zpracovat vrstevnatou sítí. N je velikost batche, C počet kanálů, L délka časových řad. Po aplikaci GAP

používáme, navíc oproti síti v původním článku, jednu skrytou hustou vrstvu s počtem neuronů rovným počtu kanálů ve výstupním tenzoru před výstupní vrstvou neuronové sítě s p neurony při klasifikaci do p tříd. Pokud nebude uvedeno jinak, předpokládá se, že dimenze *bottlenecku* je stejná ve všech použitých Inception modulech.



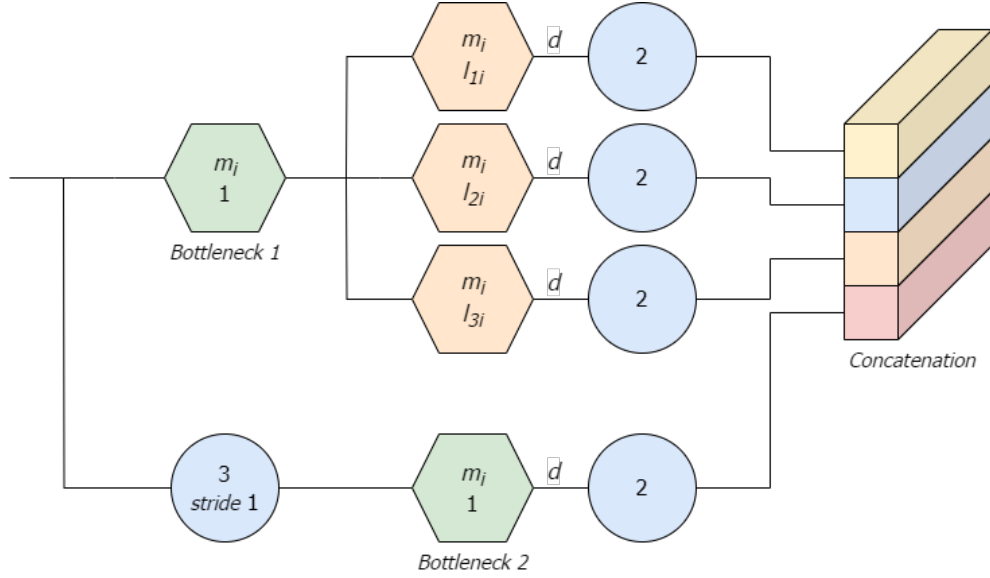
Obrázek 4.6: Vizualizace celé klasifikační Inception Time sítě [24].

4.5 Pooled Inception Time

Vzhledem k problémům, kterým jsme čelili při využívání originální architektury sítě Inception Time jsme se rozhodli upravit podobu této sítě tak, aby lépe odpovídala potřebám klasifikace relativně dlouhých signálů akustické emise. Hlavním problémem bylo, že při využití velmi dlouhých filtrů, vhodných pro signály AE, se velmi výrazně zvyšovalo množství trénovatelných parametrů v síti, což v kombinaci s délkou používaných signálů, která při průchodu dat sítí nepodléhala redukci, vedlo k opakovanému překračování dostupné paměti na používaných grafických kartách.

Vnější struktura samotné neuronové sítě zůstává beze změny oproti standardní Inception Time architektuře, tak jak je vyobrazena na obrázku 4.6. Provedené změny se týkají především samotného Inception modulu a také různého nastavení těchto modulů v rámci celé sítě. Jako nejsnazší způsob pro předejití problémům s přetečením dostupné paměti jsme vyhodnotili využití poolingové vrstvy pro zkrácení délky signálů v každém Inception modulu, odtud také název metody. Používáme avg-pooling s oknem délky 2 přímo po každé ze tří konvolučních vrstev a po druhém *bottlenecku*, jak je patrné ze schématu Pooled Inception modulu z obrázku 4.7. Při průchodu sítí se tak délka transformovaných signálů zkrátí na polovinu po průchodu každým modulem.

Při ponechání filtrů konstantní délky a dostatečné hloubce sítě bychom se takto nevyhnutelně dostali do situace, kdy by délka konvolučního jádra byla větší než délka časové řady, kterou má zpracovat. Je tedy nutné se zkracující se délkou časových řad, vstupujících do jednotlivých Inception vrstev, vhodně upravit i délku filtrů používaných v dané vrstvě tak, aby se filtry zkracovaly společně s daty a stále mohly zachycovat charakteristiky časových řad v okně relevantní délky. Délku filtrů by bylo možné nějakou funkcí přímo svázat s délkou dat vstupujících do příslušného Pooled Inception modulu tak, aby byla zajištěna smysluplnost konvolucí pro libovolnou hloubku sítě. My však, vzhledem k výrazně jednodušší implementaci a výpočetním výkonem efektivně



Obrázek 4.7: Schéma upraveného Inception modulu - Pooled Inception module.

omezené hloubce sítě, uvažujeme maximální hloubku sítě $d_{max} = 9$. Délku konvolučních filtrů v i -tém Pooled Inception modulu v rámci sítě s d moduly uvažujeme pak jako

$$\{l_{1i}, l_{2i}, l_{3i}\} = \{2^{d-i+1} + 2, 2^{d-i+2} + 2, 2^{d-i+4} + 2\}. \quad (4.1)$$

Pro názornost v tabulce 4.8 udáváme vyčíslené délky konvolučních jader v jednotlivých modulech Pooled Inception Time sítě s hloubkou $d = 6$ a $d = 9$.

S klesající délkou dat při průběhu sítě si můžeme dovolit postupně zvyšovat počet kanálů m_i v i -té vrstvě. Nechť m_1 je předem definovaný počet kanálů v prvním Pooled Inception bloku, potom počet kanálů v i -tém bloku volíme jako $m_i = m_1 \cdot 2^{(i-1)}$. Stejně jako u původní Inception Time architektury je i zde před aplikací hustých vrstev využít global avg-pooling. V této fázi už ale mají data v naší podobě sítě výrazně kratší délku, a proto při průměrování podél časové dimenze nedochází k takové ztrátě informace.

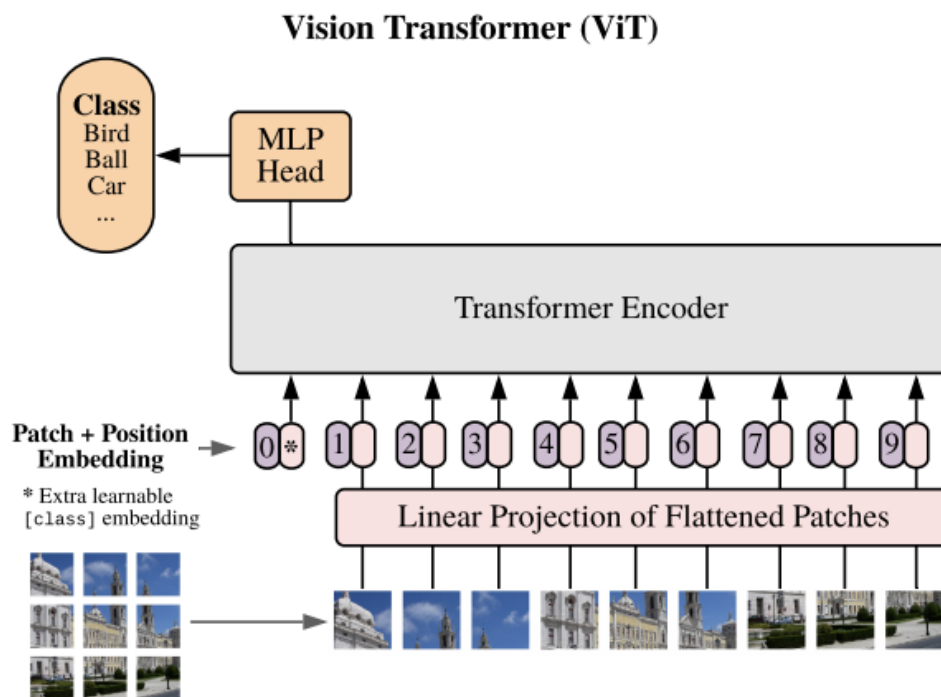
$d = 9$	$d = 6$	Délka filtrů		
		l_{1i}	l_{2i}	l_{3i}
1		514	1026	4098
2		258	514	2050
3		130	258	1026
4	1	66	130	514
5	2	34	66	258
6	3	18	34	130
7	4	10	18	66
8	5	6	10	34
9	6	4	6	18

Tabulka 4.8: Vyčíslené délky filtrů v i -tém modulu v rámci sítě Pooled Inception Time s hloubkou $d = 9$ a $d = 6$.

4.6 TSiT

Posledním uvažovaných typem sítě je tzv. TSiT transformer, implementovaný v [28]. TSiT pouze triviální obměnou v přípravě dat zaměňuje časové řady za obrázky v populární architektuře Vision Transformer (ViT). Ta byla představena v roce 2021 v [27]. Jedná se o první široce rozšířenou architekturu určenou pro zpracování obrazu, která úplně upouští od využívání konvolučních vrstev a nahrazuje je transformerem. Tvůrci ViT ukázali, že při předtrénování této sítě na obrovském množství dat dosahuje následně po fine-tuningu na menších obrazových datasetech (ImageNet, CIFAR-100...) výborných výsledků i v porovnání s dosavadními nejlepšími konvolučními architekturami. Popíšeme dále jak pracuje ViT s obrazovými daty, přechod na časové řady je potom triviální.

ViT, podobně jako například BERT [30], kterým je inspirován, využívá z originálně navržené podoby transformera z [16] pouze enkodérovou část a drží se co nejvěrněji jeho původní podoby. Jak je naznačeno ve schématu ViT na obrázku 4.9, vstupní obrazová data jsou nejprve rozdělena na části (patche) fixních rozměrů, které jsou následně zploštěny a seřazeny do sekvence. Mějme 2D obrázek $\mathbf{x} \in \mathbb{R}^{C,H,W}$, kde C je počet kanálů, H výška obrázku a W jeho šířka. Po jeho rozdělení na části a zploštění dostáváme sekvenci $\mathbf{x}_P = (\mathbf{x}_P^1, \dots, \mathbf{x}_P^N) \in \mathbb{R}^{N,P^2,C}$, kde (P, P) je rozměr jednotlivých částí a $N = HW/P^2$ je celkový počet částí na které byl obrázek rozdělen. Tatáž hodnota N je efektivně také délka vstupní sekvence pro enkodér transformera. Obrázek je tak reprezentován maticí v souladu s popisem původního transformera v sekci 3.3, obdobně tedy využíváme trénovatelnou matici $\mathbf{W}_{in} \in \mathbb{R}^{P^2,C,d}$ pro transformaci sekvence \mathbf{x}_P do vnitřní dimenze transformera d . Výstup transformace nazýváme patch embeddingem.



Obrázek 4.9: Schéma modelu Vision Transformer [27].

Následně je k embedovaným datům přičteno poziční kódování. V původní podobě ViT, na rozdíl od originálního transformeru, není využíváno fixní sinusoidální kódování, ale trénovatelné poziční kódování. To spočívá v přičtení matice, jejíž prvky jsou náhodně inicializovány a jsou upravovány v průběhu trénování sítě. Volba typu pozičního kódování mezi sinusoidálním a trénovatelným podle [29] významně neovlivňuje úspěšnost výsledného modelu. V tomto článku zároveň však na úkolu klasifikace obrázků ukazují, že model úplně bez pozičního kódování podává výrazně horší výsledky než model s jedním ze dvou zmíněných kódování. Nakonec také navrhuji nový způsob pozičního kódování - *conditional positional encoding* (CPE), které umožňuje lepší adaptaci na zpracování sekvencí jiné délky, potom co byla síť natrénována. Vzhledem ke konstantní délce signálů akustické emise však dále používáme trénovatelné poziční kódování jako v původní podobě ViT.

Obdobně jako pro NLP model BERT [30], se před sekvenci embedovaných patchů předradí tzv. [class] token. Jedná se o token, který na počátku nenesou žádnou informaci o daném obrázku, v průběhu průchodu enkodérem transformeru je obohacován skrze self-attention mechanismus informací ze všech patchů a následně je to pouze výstup z enkodéru náležející k tomuto tokenu, na který je připojena závěrečná hustá vrstevnatá síť (MLP Head ve schématu), která provede samotnou klasifikaci.

Je důležité zmínit, že ViT má oproti neuronovým sítím založeným na konvolučních vrstvách do základu danou daleko nižší úroveň apriorního porozumění obrazovým datům (totéž platí i pro náš případ 1D dat). To je dáno tím, že na rozdíl od CNN, kde je lokalita, struktura sousedních hodnot a translační ekvivariance zakořeněna v podobě konvoluční vrstvy, působí self-attention vrstvy globálně a veškeré prostorové vazby mezi jednotlivými patchemi se síť musí naučit. I z tohoto důvodu ViT potřebuje pro trénink obrovské množství dat, v [27] ukázali, že finální úspěšnost klasifikace v porovnání s ResNet sítěmi je značně ovlivněna velikostí trénovacího datasetu.

Konečně, při zpracování časových řad modelem TSiT, jsou jednotlivé časové řady tvaru $\mathbf{x} \in \mathbb{R}^{C,L}$, kde C je počet kanálů a L je délka signálů. Sekvenci dat, která vstupuje do transformeru, dostaneme obdobně jako pro obrazová data rozdělením na patche $(\mathbf{x}_P^1, \dots, \mathbf{x}_P^N) \in \mathbb{R}^{N,C \cdot P}$, kde P je délka patche a $N = L/P$ je počet patchů vybraných ze signálů. Poté už jednotlivé členy sekvence embedujeme do vnitřní dimenze transformeru d stejně jako v případě ViT a pokračujeme tak, jak bylo popsáno výše.

Kapitola 5

Výsledky analýzy AE

V této kapitole představíme výsledky analýzy signálů AE a porovnáme jednotlivé implementované architektury neuronových sítí. Vzhledem k faktu, že trénování neuronové sítě je ze své podstaty náhodný proces, nemůžeme očekávat, že by dvě nezávisle natrénované realizace téže architektury podávaly totožné výsledky. S cílem omezit vliv této variability na předkládané výsledky je každá zkoumaná neuronová síť nezávisle inicializována a natrénována celkem třikrát. Předkládané výsledky klasifikace a průběhu trénování sítě jsou poté průměrem z těchto tří realizací, k průměrné hodnotě přesnosti daného klasifikátoru je přidán i odhad její směrodatné odchylky. Pravděpodobně vlivem silného zašumění analyzovaných signálů pozorujeme napříč architekturami během trénování silné fluktuace metrik, jak na trénovacím, tak zejména na validačním datasetu. Pro lepší přehlednost grafů zachycujících průběh trénování využíváme vyhlazování naměřených metrik za pomoci *Exponential Moving Average*. Nechť $\{u_t\}$ je sekvence dat, kterou chceme vykreslit. Její EMA vyhlazení $\{s_t\}$ získáme jako

$$s_t = \alpha u_t + (1 - \alpha)s_{t-1}, \quad t \in \mathbb{N},$$

kde α je vyhlazovací parametr a $s_0 = u_0$. Vyhlazovací parametr volíme pro všechny případy jako $\alpha = 0,5$.

5.1 Úspěšnost klasifikace podle délky filtrů

V první části analýzy signálů AE jsme se zaměřili na vliv délky konvolučních jader na přesnost klasifikace při použití architektur Conv2Net a Inception Time. Používáme zde pouze data pocházející z 1. experimentu, tedy klasifikujeme jednotlivá pozorování do pěti tříd odpovídajících úrovni otupení vrtáku. Vzhledem k charakteru dat a délce zpracovávaných signálů se ukázalo jako nezbytné pracovat s výrazně většími konvolučními jádry než je běžné u obdobných neuronových sítí, které pracují s obrázky.

5.1.1 Conv2Net

V nejjednodušší představené architektuře Conv2Net využíváme pouze dvě konvoluční vrstvy. Délku konvolučního jádra v první vrstvě volíme výrazně delší než ve druhé, protože mezi vrstvami je aplikován max-pooling a druhá vrstva tak zpracovává výrazně kratší signál. Celkem jsme natrénovali architekturu Conv2Net se sedmi různými volbami délek filtrů, konkrétně se jednalo o dvojice $\{31, 9\}$, $\{51, 15\}$, $\{101, 31\}$, $\{201, 51\}$, $\{351, 65\}$, $\{501, 81\}$ a $\{801, 121\}$. Všechny ostatní hyperparametry sítě byly fixovány tak, aby se ve výsledcích klasifikace projevila pouze

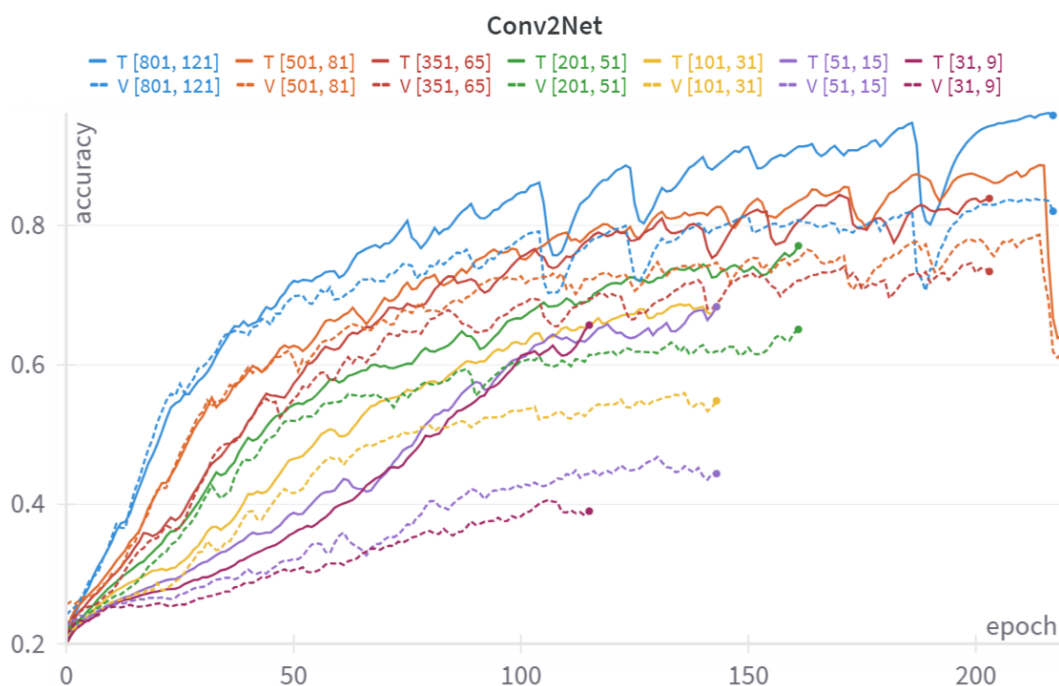
změna délek filtrů. Sítě byly trénovány s velikostí batche 512, hodnotou dropoutu 0,1 a za pomoci early stopping s čekací lhůtou 40 epoch.

V tabulce 5.1 jsou vypsané výsledky klasifikace pro jednotlivé varianty architektury Conv2Net. Jak můžeme vidět, s rostoucí délkou konvolučních filtrů roste přesnost dosažené klasifikace a nejlepších výsledků bylo dosaženo za použití filtrů délky {801, 121}. Při použití příliš dlouhých filtrů se však zvyšují požadavky trénovacího algoritmu na dostupnou paměť a prodlužuje se výpočetní doba. Nárůst v přesnosti mezi posledními dvěma variantami sítě už není tak významný, obě varianty se tak jeví jako přijatelné pro využití se signály AE.

Tabulka 5.1: Střední přesnost (Mean acc) klasifikace sítě Conv2Net na datech z 1. experimentu v závislosti na délce filtrů.

	Délka filtrů						
	{31, 9}	{51, 15}	{101, 31}	{201, 51}	{351, 65}	{501, 81}	{801, 121}
Mean acc	36,4%	45,1%	55,6%	66,1%	74,1%	79,7%	82,2%
Std dev	2,06	5,27	1,18	0,52	0,98	1,41	0,72

Na obrázku 5.2 je vykreslen průběh přesnosti sítě na trénovacím (plnou čarou) a validačním (přerušovaně) datasetu během trénování sítě. Jak už bylo zmíněno, jsou tyto průběhy vyhlazeny pomocí EMA. U nevyhlazených průběhů pozorujeme výrazně vyšší fluktuaci v přesnosti a využití early stopping se tak stává naprosto zásadním pro identifikaci epochy s vysokou hodnotou přesnosti na validačním datasetu, kterou si přejeme uchovat. Pozorované přetrénování sítě je pro delší filtry s vyšší celkovou přesností ještě v únosné míře.



Obrázek 5.2: Průběh přesnosti klasifikace na trénovacím (T) a validačním (V) datasetu během trénování architektury Conv2Net na datech z 1. experimentu v závislosti na délce použitých filtrů.

5.1.2 Inception Time

Jak již bylo zmíněno při popisu architektury Inception Time v sekci 4.4, v původním článku [24] byl jeden Inception modul sítě koncipován se třemi filtry o délkách 10, 20 a 40. Opět předpokládáme, že pro zachycení významných charakteristik v signálu AE bude vhodnější využít delší filtry. Implementujeme proto šest variant sítě Inception Time hloubky 6 s vnitřní dimenzí bottlenecku 8. Jednotlivé varianty se liší pouze délkou konvolučních filtrů použitých v rámci jejich Inception modulů, jednotlivé délky filtrů volíme jako {5, 10, 20}, {10, 20, 40}, {21, 45, 101}, {31, 71, 191}, {41, 101, 301} a {55, 151, 501}. Jelikož data při průchodu sítí nesnižují svoji délku, používáme stejné filtry v každém ze šesti Inception modulů, ze kterých je síť složena. Všechny sítě trénujeme s velikostí batche 64 a s čekací lhůtou 60 epoch.

V tabulce 5.3 jsou pro každou z variant sítě vypsány výsledky. Pozorujeme obdobný trend jako u Conv2Net, kdy delší filtry vedou k lepším výsledkům klasifikace. Oproti Conv2Net je však přesnost mnohem vyrovnanější a i varianta sítě s nejkratšími filtry klasifikuje se solidní úspěšností přes 70%. Předpokládáme, že toto je dáno větší hloubkou sítě Inception Time a tedy širším vjemovým polem (tzv. receptive field) v jejích posledních konvolučních vrstvách. Růst přesnosti mezi posledními dvěma variantami sítě s nejdelšími filtry je už téměř zanedbatelný a vzhledem k značnému nárůstu ve velikosti modelu při použití delších signálů volíme jako optimální délky filtrů hodnoty {41, 101, 301}.

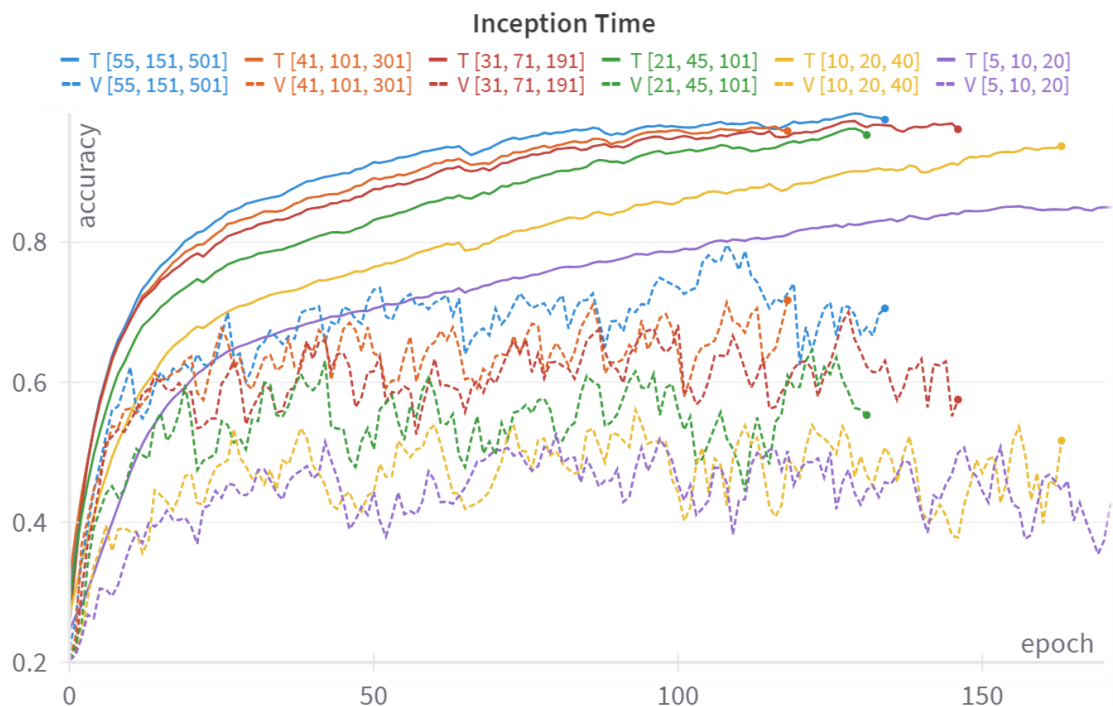
Tabulka 5.3: Přesnost klasifikace sítě Inception Time na datech z 1. experimentu v závislosti na délce filtrů.

	Délka filtrů		
	{5, 10, 20}	{10, 20, 40}	{21, 45, 101}
Mean accuracy	70,3%	73,1%	79%
Std dev	2,69	0,53	1,19
	{31, 71, 191}	{41, 101, 301}	{55, 151, 501}
Mean accuracy	82,3%	85%	85,9%
Std dev	1,11	0,34	0,85

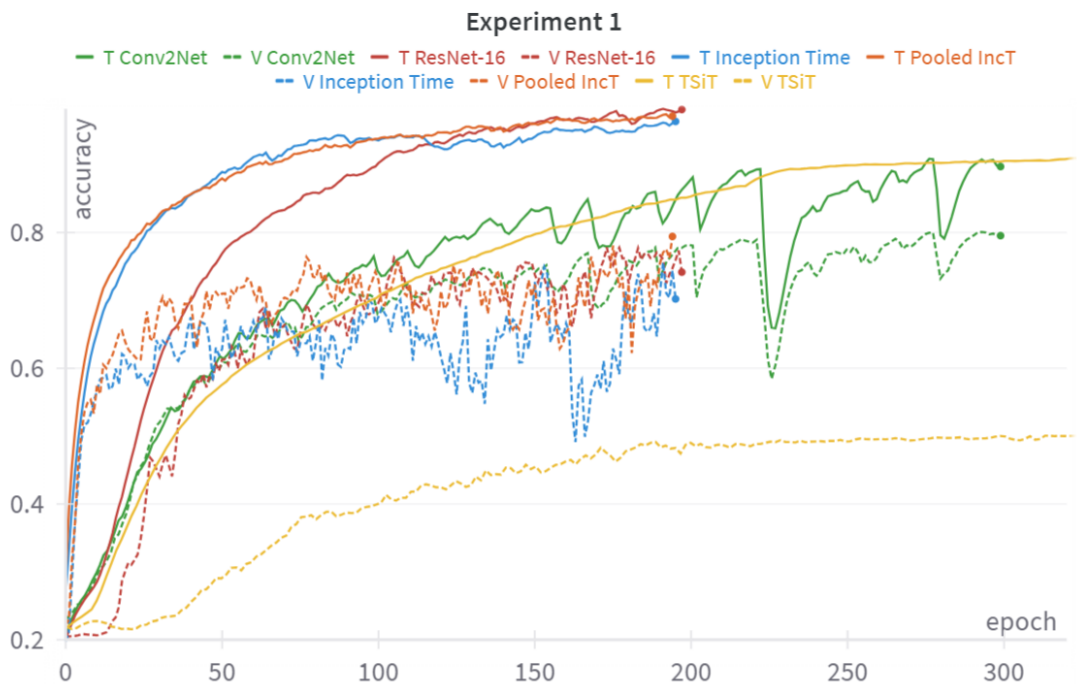
Průběh trénování jednotlivých variant sítě je vykreslen na obrázku 5.4. Oproti průběhu trénování sítě Conv2Net zde pozorujeme znatelně vyšší míru přetrénování, kdy pro tři varianty sítě s nejdelšími filtry se přesnost na trénovacím datasetu dostává nad hladinu 95%, zatímco přesnost na validačním datasetu silně osciluje s maximy do 85%. Vnitřní dimenze sítě je už v tomto nastavení relativně nízká a další techniky pro odstranění přetrénování vedly k výsledné nižší hodnotě validační přesnosti. Proto zůstáváme u této podoby sítě Inception Time i pro další experimenty a spoléháme na homogenost rozdělení dat v trénovacím, validačním a testovacím datasetu, která nám obhájí využití early stopping callback k nalezení optimální epochy při tréninku.

5.2 Experiment 1

V této části se zaměříme na porovnání jednotlivých architektur představených neuronových sítí pro klasifikaci dat z 1. experimentu. Připomeňme, že se jedná o měření z vrtací soustavy, která jsou rozdělena do pěti tříd podle úrovně otupení vrtáků. Hyperparametry jednotlivých neuronových sítí byly optimalizovány a naladěny manuálně zvlášť pro každou síť.



Obrázek 5.4: Průběh přesnosti klasifikace na trénovacím (T) a validačním (V) datasetu během trénování architektury Inception Time na datech z 1. experimentu v závislosti na délce použitých filtrů.



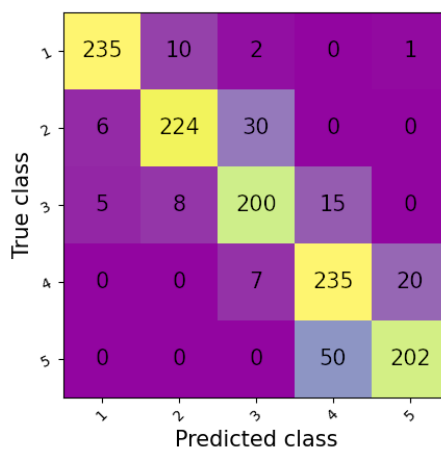
Obrázek 5.5: Průběh přesnosti klasifikace na trénovacím (T) a validačním (V) datasetu během trénování různých architektur na datech z 1. experimentu.

Celkem máme k dispozici pět různých architektur popsaných v kapitole 4. Použili jsme síť Conv2Net s filtry délky {501, 81} a dropoutem 0,1. Síť ResNet-16 zde odpovídá výše zmíněnému popisu, tedy využívá 4x4 konvoluční vrstvy s délkami filtrů {501, 151, 71, 31} v jednotlivých blocích a hodnotou dropoutu 0,2. Jak Conv2Net tak ResNet-16 pracoval s velikostí batche 512, ta je pro další architektury příliš velká a z důvodu omezené paměti při výpočtu bylo třeba ji snížit. Použitá podoba sítě Inception Time pracovala s filtry délky {41, 101, 301} a byla složená z devíti Inception modulů s vnitřní dimenzí 8. Síť Pooled Inception Time měla pro tuto úlohu hloubku 6, vnitřní dimenze prvního Pooled Inception modulu byla 8 s hodnotou dropoutu 0,3. Proměnlivá délka filtrů v jednotlivých modulech je popsána v tabulce 4.8. Obě sítě založené na Inception modulu pracovaly s velikostí batche 64, což byla experimentálně určená maximální mocnina čísla 2, pro kterou nedocházelo k problémům s přetečením paměti. Poslední použitou klasifikační sítí byl transformer TSiT s devíti bloky v enkodéru, využívající 16 hlav v každém multi-head self-attention modulu s vnitřní dimenzí 256. Hodnota dropoutu byla 0,3 v attention modulu a klasifikační hlavě a 0,5 ve vrstevnatých částech sítě následujících po výpočtu attention. Vstupní signály byly rozděleny na tokeny s délkou okna 40 a TSiT pracoval s velikostí batche 32. Čekací lhůta pro early stopping byla volena specificky pro každou architekturu v závislosti na rychlosti trénování.

Tabulka 5.6: Přesnost klasifikace dat z 1. experimentu podle použité klasifikační neuronové sítě.

	Typ sítě				
	Conv2Net	ResNet-16	InceptionTime	Pooled IncT	TSiT
Mean accuracy	80,4%	81,5%	83,2%	86,4%	56,2%
Std dev	1,96	1,57	4,73	2,15	1,11

V tabulce 5.6 jsou vypsány výsledné přesnosti klasifikace. Nejlepších výsledků dosáhla síť Pooled Inception Time, s ostatními CNN klasifikačními metodami zaostávajícími o jednotky procent. Klasifikace pomocí transforméru TSiT dopadla s velkým odstupem nejhůře. Vzhledem k poznatkům o trénování metod založených na transformerech pro příbuzné úlohy [27] předpokládáme, že výsledek klasifikace s pomocí TSiT by bylo možné zlepšit využitím výrazně většího datasetu v kombinaci se semi-supervised předtrénováním sítě. Tato procedura však byla mimo rámec této práce.



Obrázek 5.7: Matice záměn zobrazující výsledky klasifikace metodou Pooled Inception Time.

Na obrázku 5.5 jsou vyobrazeny průběhy přesnosti na trénovacím a validačním datasetu pro jednotlivé použité klasifikační sítě. Jak je z něj patrné, chování sítí ResNet-16, Inception Time a Pooled Inception Time je podobné v tom, že přibližně po epoše 150 se přesnost na trénovacích datech dostane nad úroveň 95% a už nemá prostor se významně zlepšit, zatímco přesnost na validačních datech silně osciluje s maximy o 10-15% níže. Vidíme zde podobnou úroveň přetrénování jako pro síť trénované v sekci 5.1.2. U sítě Conv2Net nedochází k tak silnému přetrénování pravděpodobně proto, že síť nemá kapacitu na přílišné přizpůsobení se trénovacím datům, přesnost na validačním datasetu však nedosahuje tak vysokých maxim. Tato skutečnost neodpovídá průběhu křivek v obrázku 5.5, to je dáno vyhlazením průběhu validační přesnosti. Trénování TSiT trvá déle a probíhá stabilně až do epochy 702, kde trénovací přesnost dosáhla hodnoty 91% a validační 50%. Průběh byl oříznut pro lepší čitelnost. Klasifikační metoda TSiT je tedy zdaleka nejvíce přetrénovaná a nedosahuje výsledků sítí založených na konvolučních vrstvách, navzdory značně delšímu trénování.

Pro neúspěšnější klasifikační metodu Pooled Inception Time je na obrázku 5.7 vykreslena kompletní matice záměn popisující výsledky klasifikace. Z matice lze vyzorovat, že nejspolehlivěji oddělí klasifikátor pozorování z 1. třídy, která odpovídá neotupenému vrtáku. To je v souladu s očekáváním, protože míru otupení vrtáku mezi ostatními třídami nebylo možné v experimentu zcela přesně kontrolovat. K největšímu překryvu dochází mezi třídami 4 a 5, celkově je však většina pozorování v matici záměn koncentrována v hlavní a prvních vedlejších diagonálách, což naznačuje že se neuronová síť naučila vnitřně interpretovat určitou spojitou míru otupení vrtáku.

Výsledky klasifikace dosažené pomocí hlubokých neuronových sítí můžeme porovnat s výsledky z [1], kde byla klasifikace stejného datasetu provedena jiným přístupem. Ze signálů bylo nejprve extrahováno celkem 1161 předdefinovaných příznaků pomocí softwarového balíku TSFresh [36]. Klasifikace byla následně provedena buď přímo na těchto příznacích, nebo na jejich projekci do prostoru snížené dimenze. V tabulce 5.8 jsou výsledky klasifikace pro metody snížení dimenze LDA (Linear discriminant analysis), GDA (Generalized discriminant analysis) a klasifikační metody QDA (Quadratic discriminant analysis), RF (Random Forrest) a MLP (Multilayer perceptron). Použitím metod hlubokého strojového učení, zejména konvolučních vrstev, a s přímou klasifikací celých syrových výřezů ze signálů AE jsme dosáhli výrazně lepší přesnosti klasifikace dat z 1. experimentu do 5 tříd tuposti vrtáku.

Tabulka 5.8: Přesnost klasifikace dat z 1. experimentu přístupem extrakce příznaků z [1].

	Metoda redukce dimenze		
	LDA	GDA (rbf)	Žádná
QDA	73.9%	75.5%	x
RF	72.8%	74.2%	67.4%
MLP	73.9%	75.7%	76%

5.3 Experiment 2

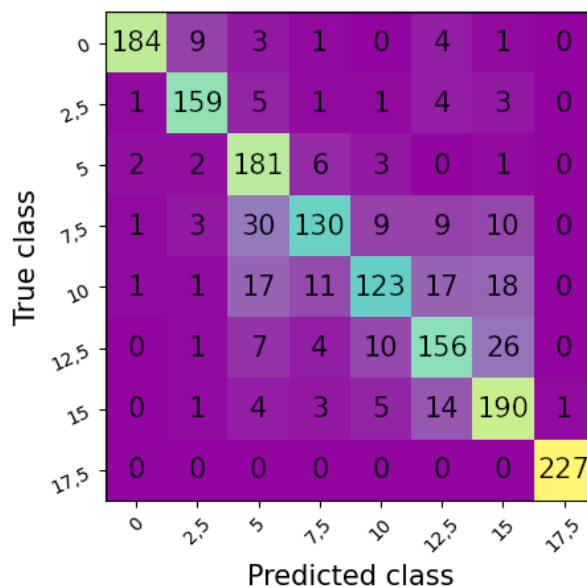
V této sekci porovnáme jednotlivé architektury při klasifikaci dat z druhého experimentu. Připomeňme, že se jedná o signály akustické emise snímané z povrchu ložiskových domků při různém zatížení středního ložiska při chodu zatěžovací aparatury popsané v sekci 2.2. Celkem byla naměřena AE pro 8 úrovní zatížení, v této části se zabýváme pouze problémem klasifikace pozorování do těchto osmi tříd.

Výběr hyperparametrů opět probíhal manuálně zvlášť pro každou z implementovaných architektur a vycházel ze zkušeností získaných při analýze dat z 1. experimentu. Síť Conv2Net a ResNet-16 zůstaly ve stejném tvaru jako pro 1. experiment (sekce 5.2), pouze došlo, tak jako u všech sítí, ke změně počtu neuronů ve výstupní vrstvě na 8, aby odpovídal počtu tříd. Síť InceptionTime byla pro tuto úlohu složená ze šesti Inception modulů s vnitřní dimenzí 32 a využívala filtry délek {41, 101, 301}. Zde použitá instance Pooled InceptionTime měla hloubku 9, vnitřní dimenze prvního modulu byla 8 a hodnota dropoutu 0,3. Obě tyto sítě stejně jako v předchozí úloze pracovaly s velikostí batche 64. Posledním implementovaným klasifikátorem byla síť TSiT s devíti bloky v enkodéru, využívající 8 hlav v každém multi-head self-attention modulu a s vnitřní dimenzí 64. Hodnota dropoutu byla 0,3 v attention modulu a klasifikační hlavě a 0,5 ve vrstevnatých částech sítě následujících po výpočtu attention. Vstupní signály byly stejně jako pro data z 1. experimentu rozděleny na tokeny s délkou okna 40 a velikost batche byla 32.

Tabulka 5.9: Přesnost klasifikace dat z 2. experimentu podle použité klasifikační neuronové sítě.

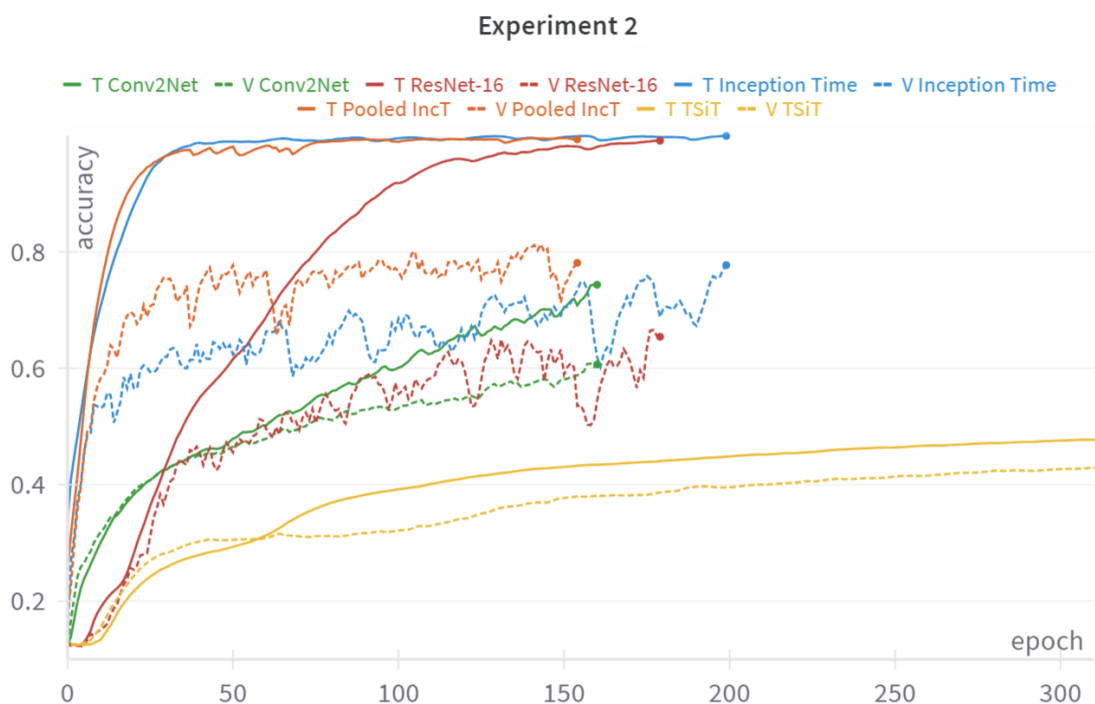
	Typ sítě				
	Conv2Net	ResNet-16	InceptionTime	Pooled IncT	TSiT
Mean accuracy	57%	63,9%	78,1%	84,8%	49,8%
Std dev	4,46	7,68	1,41	0,57	0,44

Výsledné hodnoty přesnosti klasifikace na testovacím datasetu jsou zapsány v tabulce 5.9. Nejlepšího výsledku dosáhla opět architektura Pooled InceptionTime s přesností klasifikace 84,8%. Matice záměn získaná touto metodou je na obrázku 5.10. Oproti výsledkům 1. experimentu je však přesnost ostatních klasifikačních metod v porovnání s Pooled IncT výrazně nižší, její využití je zde tedy opodstatněné a vede k významnému zvýšení přesnosti klasifikace. Za zmínku stojí vysoká směrodatná odchylka u sítě ResNet-16, která je způsobena nestabilním trénováním sítě silně ovlivněným náhodnou inicializací parametrů.



Obrázek 5.10: Matice záměn zobrazující výsledky klasifikace metodou Pooled Inception Time.

Vyobrazení průběhu přesnosti na trénovacím a validačním datasetu na obrázku 5.11 ukazuje, obdobně jako pro první experiment, že sítě ResNet-16, Inception Time a Pooled Inception Time dosahují téměř perfektní přesnosti na trénovacím datasetu za cenu silně fluktuující přesnosti na validačním datasetu. Hlavní změnou zde je, že Pooled Inception Time dosahuje znatelně lepší generalizace a stabilně se svojí validační přesností drží nad ostatními klasifikačními metodami. Architektura Conv2Net v tomto případě nedosahuje konkurenceschopných výsledků, stejně jako TSiT, který opět i za cenu dlouhého trénovacího času až po epochu 748 dosahuje přesnosti pouze 55% na trénovacím datasetu a 45% na validačním datasetu při společné klasifikaci do 8 tříd.



Obrázek 5.11: Průběh přesnosti klasifikace na trénovacím (T) a validačním (V) datasetu během trénování různých architektur na datech z 2. experimentu.

5.4 Úspěšnost klasifikace podle délky vybíraných signálů

Velký objem dostupných dat daný vysokou vzorkovací frekvencí při snímání signálů AE je jednou z výzev, kterou je při jejich analýze třeba vzít v potaz. Při zpracování signálů metodami hlubokého učení přistupujeme k výběru kratších výřezů z původního kontinuálního signálu. Délka těchto výřezů zásadním způsobem ovlivňuje objem dat, které je nutné zpracovat a dobu pro toto zpracování potřebnou. V této sekci se tedy budeme zabývat vlivem délky vyříznutého úseku signálu na přesnost klasifikace, za účelem ověření využitelnosti kratších výřezů. Využijeme k tomu síť Pooled Inception Time, která se ukázala jako nejlepší z implementovaných architektur.

5.4.1 Experiment 1

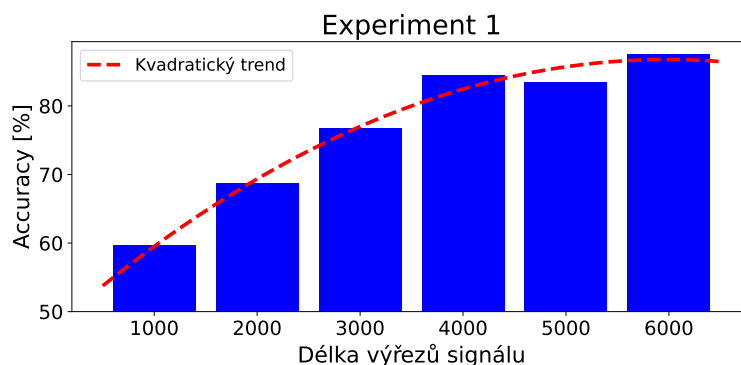
Pro data z prvního experimentu jsme při předzpracování původních kontinuálních signálů spojité akustické emise náhodně vybírali výřezy o délce 6000. Tato délka byla zvolena už v předchozí

práci [1] jako kompromis mezi snahou použít co nejdelší výřezy, které by v signálu zachycovaly trendy s nižší frekvencí, a omezeními plynoucími z výpočetní náročnosti práce s dlouhými časovými řadami. Pro tuto studii jsme využili dostupné výřezy, ze kterých jsme použili postupně prvních 1000, 2000, 3000, 4000, 5000 a 6000 hodnot. Celkem jsme tak natrénovali klasifikační metodu na šesti datasetech s různou délkou signálů.

Abychom omezili vliv dalších faktorů na výsledky klasifikace, použili jsme ke klasifikaci pro všechny délky signálů architekturu Pooled Inception Time se stejnými zafixovanými hyperparametry. Konkrétně se jednalo o model o hloubce 6 s vnitřní dimenzí prvního modulu 4, regularizovaný dropoutem 0,3, který pracoval s velikostí batche 64. Čekací lhůta pro early stopping byla ve všech případech nastavena na hodnotu 50.

Tabulka 5.12: Přesnost klasifikace dat z 1. experimentu v závislosti na délce vybraných úseků signálu.

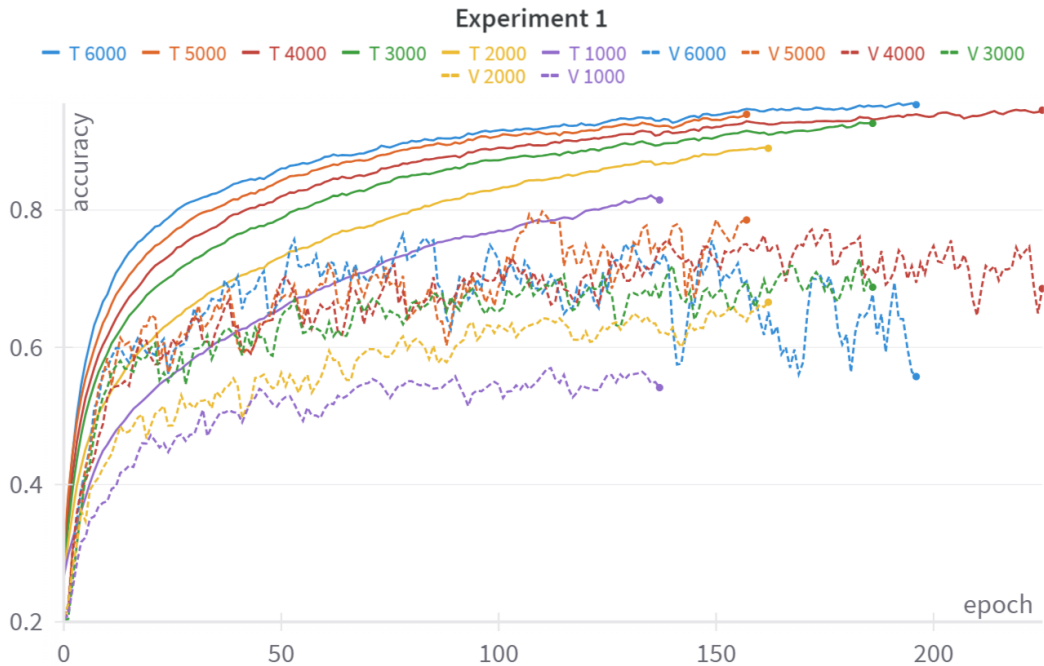
	Délka vybraných signálů					
	1000	2000	3000	4000	5000	6000
Mean accuracy	59,7%	68,8%	76,8%	84,5%	83,7%	87,5%
Std dev	0,51	1,77	1,37	1,08	2,75	0,33



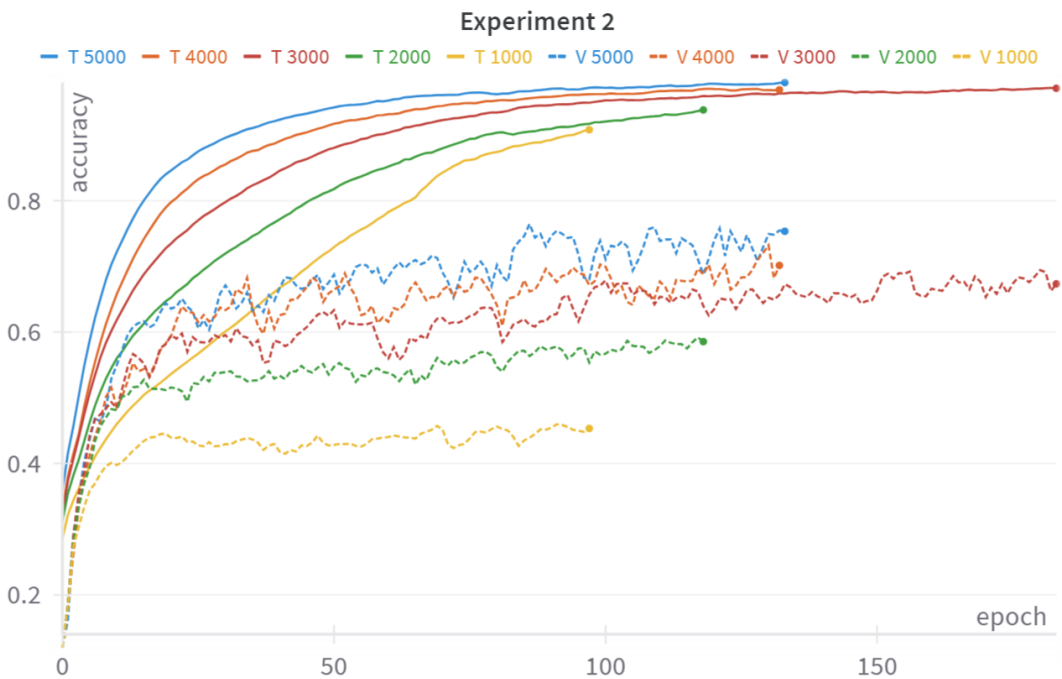
Obrázek 5.13: Přesnost klasifikace dat z 1. experimentu v závislosti na délce vybraných úseků signálu.

Výsledky klasifikace jsou popsány v tabulce 5.12. V souladu s očekáváním bylo dosaženo nejlepších výsledků při použití celé délky dostupných výřezů. Pokles přesnosti klasifikace při použití výřezů o délce 5000 lze přisoudit náhodnému průběhu trénovacího procesu a i s přihlédnutím k relativně vysoké směrodatné odchylce mezi přesností jednotlivých opakování nepovažujeme tento pokles za statisticky významný. Na obrázku 5.13 jsou tyto výsledky vyneseny v podobě sloupcového grafu, který je navíc proložen kvadratickou křivkou modelující vývoj přesnosti klasifikace. Jak je patrné, přibližně od hodnoty 4000 pro délku výřezů signálu dochází při jejím navyšování už jen k mírnému zlepšení výsledné přesnosti. Pro určité aplikace vyžadující nižší výpočetní náročnost nebo rychlejší analýzu lze tedy uvažovat o snížení délky výřezů signálu bez výrazné ztráty přesnosti klasifikace.

Průběh přesnosti během trénování je pro všechny použité modely vykreslen na obrázku 5.14. Nezdá se, že by míra přetrénování sítě nějak souvisela s použitou délkou výřezů.



Obrázek 5.14: Průběh přesnosti klasifikace na trénovacím (T) a validačním (V) datasetu během trénování architektury *Pooled Inception Time* na datech z 1. experimentu v závislosti na délce vybraného úseku signálu.



Obrázek 5.15: Průběh přesnosti klasifikace na trénovacím (T) a validačním (V) datasetu během trénování architektury *Pooled Inception Time* na datech z 2. experimentu v závislosti na délce vybraného úseku signálu.

5.4.2 Experiment 2

Pro data z druhého experimentu byly, po zkušenostech s analýzou signálů vrtáku, při předzpracování z původních kontinuálních signálů spojité akustické emise vybrány výřezy o délce 5000. Obdobně jako v předchozí sekci jsme zde z dostupných výřezů použili postupně prvních 1000, 2000, 3000, 4000 a 5000 hodnot. Dostali jsme tak pět datasetů s různou délkou signálu na nichž jsme natrénovali klasifikační síť.

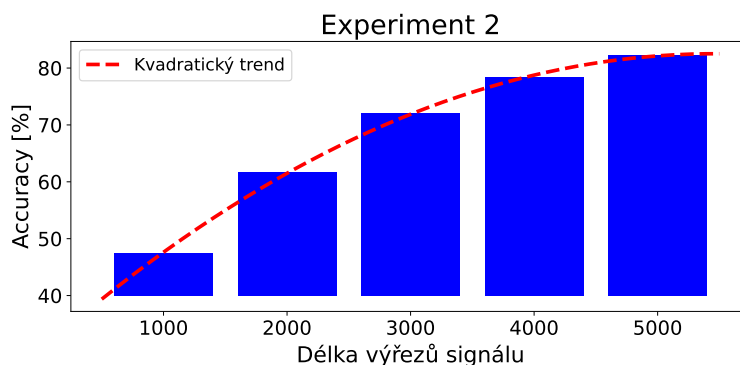
Pro samotnou klasifikaci jsme využili Pooled Inception Time se stejným nastavením hyperparametrů jako pro data z prvního experimentu. Tedy model o hloubce 6 s vnitřní dimenzí prvního modulu 4, regularizovaný dropoutem 0,3 a pracující s velikostí batche 64. Trénování je zastaveno early stopping callbackem s čekací lhůtou 50 epoch.

Výsledky klasifikace jsou zapsány v tabulce 5.16. Pozorujeme monotonní nárůst přesnosti s rostoucí délkou použitých výřezů. Jak je patrné i z obrázku 5.17, kde jsou výsledky vykresleny jako sloupcový graf a proloženy kvadratickým odhadem závislosti, je nárůst významný i mezi délkami výřezů 4000 a 5000. Z toho vyvozujeme, že při použití výřezů delších než 5000 bychom mohli za cenu zvýšené výpočetní náročnosti dosahovat ještě o něco lepších výsledků. Nelze však očekávat vyšší nárůsty v přesnosti klasifikace než o nízké jednotky procent, proto se v rámci této práce do dalšího předzpracování surových signálů AE nepouštíme.

Průběh přesnosti na trénovacím a validačním datasetu je pro jednotlivé modely vykreslen na obrázku 5.15. Oproti průběhu přesnosti při stejné úloze na datech 1. experimentu z obrázku 5.14 pozorujeme nižší fluktuace na validačním datasetu a znatelnější závislost úrovně přetrénování na délce použitých výřezů. Zatímco trénovací přesnost se i pro výřezy o délce 1000 dostane nad hranici 90%, validační přesnost zůstává pod 50%. Naopak pro výřezy o délce 5000, kdy se trénovací přesnost pohybuje okolo 97%, se validační přesnost v některých epochách dostává přes 80%.

Tabulka 5.16: Přesnost klasifikace dat z 2. experimentu v závislosti na délce vybraných úseků signálu.

	Délka vybraných signálů				
	1000	2000	3000	4000	5000
Mean accuracy	47,5%	61,7%	72%	78,4%	82,3%
Std dev	1,17	0,23	1,30	1,43	1,26



Obrázek 5.17: Přesnost klasifikace dat z 2. experimentu v závislosti na délce vybraných úseků signálu.

5.5 Experiment 2 - citlivost snímačů

Pro experiment 2 - zatěžování ložisek, jsme provedli studii informační výtěžnosti jednotlivých snímačů. Cílem úlohy je určit, které umístění snímačů je vhodné pro zachycení signálů AE nesoucích co největší množství informace o zatížení. Připomeňme popis úlohy ze sekce 2.2, kde na obrázku 2.6 je zachyceno umístění snímačů na zatěžovací aparatuře. Snímače a jim náležející kanály jsou na obrázku číslovány směrem zprava dolů.

Pro porovnání jednotlivých snímačů jsme použili síť Pooled Inception Time, která podává nejlepší klasifikační výsledky. Kompletní předzpracovaný dataset z druhého experimentu jsme rozdělili na 4 části příslušející jednotlivým kanálům a nezávisle jsme na nich natrénovali klasifikátory se shodnými hyperparametry. Konkrétní podoba sítě byla stejná jako v předchozí sekci, tedy hloubka 6 s vnitřní dimenzí prvního modulu 4, hodnota dropoutu 0,3 a velikost batche 64.

Tabulka 5.18: Přesnost klasifikace dat z 2. experimentu v závislosti na použitém kanálu.

	Použitý kanál			
	1	2	3	4
Mean accuracy	36,8%	85%	42,3%	29,5%
Std dev	0,80	0,82	1,69	2,73

Výsledky klasifikace podle jednotlivých kanálů jsou uvedeny v tabulce 5.18. Ukazuje se, že kanál 2 obsahuje v této konfiguraci experimentu zásadně více využitelné informace o zatížení ložiska než všechny ostatní. Druhý snímač je umístěn přímo na domku zatěžovaného ložiska, fakt že se jedná o nejlepší zdroj pro klasifikaci je tedy v souladu s intuicí. Odstup přesnosti klasifikace na ostatních kanálech je však překvapivý. Přesnost klasifikace pouze na druhém kanálu dokonce vyrovnává nejvyšší přesnost klasifikace při použití všech čtyř kanálů zaznamenanou v sekci 5.3. Možnost využívat pouze druhý kanál pro snížení objemu zpracovávaných dat je tak nasnadě.

5.6 Experiment 2 - regresní úloha

V této sekci předkládáme výsledky odlišného přístupu k analýze signálů naměřených v rámci 2. experimentu. Naším cílem zde bylo, namísto klasifikace pozorování do osmi tříd na základě úrovně zatížení, vytvořit neuronovou síť, která bude provádět tzv. *time series extrinsic regression*. Tedy predikovat spojitou skalární proměnnou (v našem případě hmotnost zatěžující ložisko) na základě zaznamenaných signálů AE. Za tímto účelem je třeba upravit dataset a namísto labelu označujícího třídu, do které dané pozorování spadá poskytnout přímo hodnotu zatížení, se kterým bylo dané pozorování naměřeno.

Pro regresní úlohu je také zapotřebí upravit podobu neuronové sítě. Použili jsme pouze architekturu Pooled Inception Time, která se osvědčila pro klasifikační úlohu. V poslední vrstvě sítě je třeba snížit počet neuronů na jeden, ten bude obsahovat samotnou predikci. Aktivační funkce na tomto neuronu je nežádoucí a je proto odstraněna. Nabízí se sice varianta využití aktivace ReLU, čímž by všechny záporné predikce byly nahrazeny nulou, to by ale negativně ovlivnilo proces trénování, při kterém by byla uměle snížena hodnota ztrátové funkce pro záporné predikce. Hlavní změnou v podobě neuronové sítě je volba ztrátové funkce, pro regresní úlohu budeme využívat ztrátové funkce MSE nebo MAE. Tato změna je nutná vzhledem k tomu, že výstupy neuronové sítě nejsou v tomto případě vázány podmínkami danými aplikací softmax aktivace a využití kategoričké křížové korelace by tak vedlo k nulovým predikcím, které by minimalizovaly

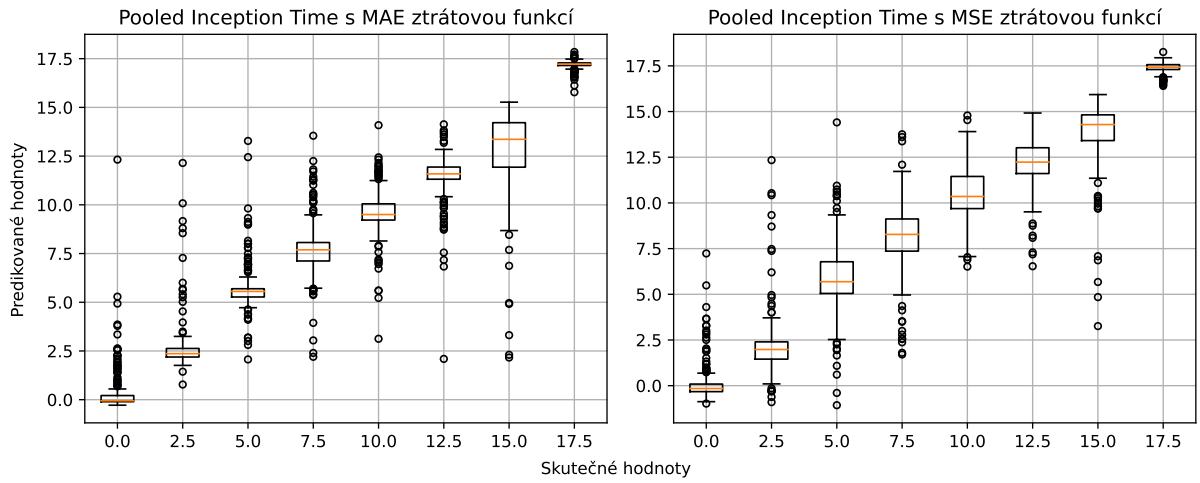
ztrátovou funkci. Ostatní hyperparametry sítě Pooled Inception Time volíme stejně jako v sekci 5.3, tedy síť je složená z devíti modulů s vnitřní dimenzí prvního modulu 8, hodnotou dropoutu 0,3 a velikostí batche 64.

Celkem jsme architekturu Pooled Inception Time natrénovali třemi způsoby. Nejprve jsme váhy v síti inicializovali náhodně a trénovali jsme ji standardním způsobem za pomoci trénovacího a validačního datasetu. Takto jsme natrénovali dvě sítě, jednu za použití ztrátové funkce MSE, druhou za použití MAE. Druhý způsob trénování spočíval ve využití sítě stejné architektury předtrénované na klasifikační úloze. K tomu jsme využili jeden z modelů natrénovaných v sekci 5.3. U tohoto natrénovaného modelu jsme ponechali zafixované všechny váhy až po vrstvu, ve které dochází ke zploštění vnitřní reprezentace dat a jejich převedení do vektoru vnitřních příznaků. Závěrečnou klasifikační vrstevnatou část sítě jsme nahradili regresní vrstevnatou hlavou sestávající z jedné skryté vrstvy se 100 neurony a výstupní vrstvy s jedním neuronem. Následně jsme za využití ztrátové funkce MSE dotrénovali pouze tuto regresní hlavu. Tomuto přístupu se obecně říká *transfer learning*.

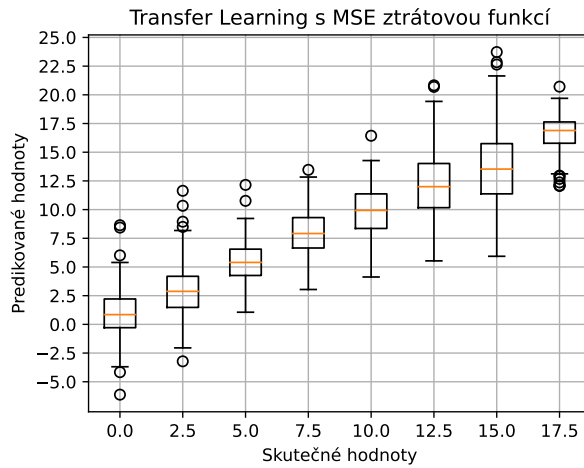
Tabulka 5.19: Výsledné hodnoty ztrátových funkcí na testovacím datasetu pro regresní úlohu odhadující zatížení v 2. experimentu.

	MSE	MAE
Síť se ztrátovou funkcí MSE	2,83	1,05
Síť se ztrátovou funkcí MAE	2,84	0,97
Síť s převzatými vahami	5,97	1,86

V tabulce 5.19 jsou uvedeny výsledky všech tří sítí v podobě hodnot MSE a MAE na testovacím datasetu. Síť trénovaná plnohodnotně na regresní úlohu mají podobné výsledky v obou metrikách přesnosti predikce a dosahují znatelně lepších hodnot než síť s vahami přejatými z klasifikační úlohy. Je obtížné hodnotit kvalitu predikce pouze na základě těchto skalárních metrik, proto jsou na obrázcích 5.20 a 5.21 vykreslené boxploty predikcí z testovacího datasetu rozdělených podle skutečné hodnoty. V boxplotu značí oranžová linka medián predikcí a spodní, resp. horní hrana obdélníku značí první, resp. třetí kvartil. Tzv. vousy vystupují z obdélníku až po poslední pozorování ležící do vzdálenosti 1,5 IQR (*inter quartile range*) od hran obdélníku. Ostatní pozorování jsou brána jako outliersy a jsou vykreslena samostatně. Na obrázku 5.20 vidíme, že pro zatížení 0 až 10 kilogramů poskytuje lepší predikce s nižším rozptylem síť trénovaná pomocí MAE ztráty. Pro vyšší hodnoty zatížení však predikuje lépe síť trénovaná s MSE ztrátou. Převzetí vah ze sítě natrénované pro klasifikační úlohu vede obecně ke většímu rozptylu v predikcích, jak lze nahlédnout na obrázku 5.21. Stále však poskytuje smysluplné výsledky.



Obrázek 5.20: *Boxploty odhadů zatížení rozdělených podle reálných hodnot pro síť Pooled Inception Time v regresním módu, která byla trénována s MAE a MSE ztrátovou funkcí.*



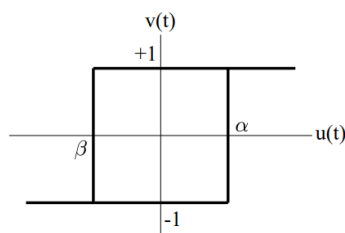
Obrázek 5.21: *Boxplot odhadů zatížení rozdělených podle reálných hodnot pro síť Pooled Inception Time v regresním módu s převzatými vahami z klasifikační úlohy a doučenou regresní hlavou.*

Kapitola 6

Hystereze v materiálech

Další problematikou, kterou se v rámci této práce zabýváme je hledání hustoty pravděpodobnosti, resp. směsi distribucí v Preisach-Mayergoyzově modelu struktury materiálu na základě naměřených hysterezních křivek. Zde se zabýváme mechanickou elastickou hysterezí a využíváme ji k vyhodnocení elasticity a poškození materiálů. Cílem je zjistit, zda metody strojového učení vyvinuté pro analýzu signálů AE mohou být použity i v tomto příbuzném oboru. Motivací je vyhodnocení reálných dat z testování disipativních tlumičů, určených k absorpci otřesů v rámci nosné struktury budovy během zemětřesení. Podobné pasivní ochranné prostředky jsou v tektonicky aktivnějších oblastech velmi užitečné, protože snižují riziko kolapsu budovy a náklady na opravu. Analytické metody vyhodnocující stupeň poškození těchto tlumičů jsou klíčové pro jejich efektivní výměnu a údržbu. Reálná data z nedestruktivního testování tlumičů pochází z experimentu provedeného univerzitou v Granadě, podrobněji je experiment popsán v [31].

Hystereze označuje takové chování dynamického systému, kdy výstupní veličina závisí nejen na nezávisle proměnné vstupní veličině, ale i na předchozím stavu systému, vyskytuje se tedy tzv. paměťový efekt. Toto chování lze pozorovat v řadě odvětvích, od elektromagnetismu až po ekonomii. Graf popisující takové chování nazýváme hysterezní křivkou, ta tvoří obvykle téměř uzavřenou smyčku. Hysterezní chování můžeme popsat za pomoci relé na obrázku (6.1). Pokud je na počátku hodnota vstupního signálu $u(t)$ nižší než α , výstup $v(t)$ nabývá hodnotu -1. Na ní zůstává až do chvíle, než vstup přesáhne hranici α , kdy se přepne do stavu 1, ve kterém zůstává až do případného poklesu $u(t)$ pod spodní hranici β . Pokud tedy máme k dispozici pouze informaci $\beta < u(t) < \alpha$ nejsme schopni určit hodnotu $v(t)$ bez znalosti o předchozím vývoji $u(t)$. Navíc se v této práci omezujeme pouze na případ tzv. *rate-independent* hystereze, pro kterou hodnoty výstupní proměnné závisí pouze na minimech a maximech vstupní proměnné. Dále uvažujeme skalární model hystereze, tedy $u(t)$ je vždy jednorozměrná vstupní veličina a $v(t)$ jednorozměrný výstup.



Obrázek 6.1: Hysterezní přepínací relé [32].

6.1 Preisach-Mayergoyzův (PM) model

Pro matematický popis hystereze je zapotřebí uvažovat model, který je díky své struktuře schopen uchovávat informaci o dosažených extrémních hodnotách vstupního signálu, tak aby mohl predikovat průběhy hysterezních křivek. Zde využíváme Preisach-Mayergoyzův (PM) model [34], který je založen na složení jednoduchých hysterezních operátorů. Ty jsou definovány obdobně jako relé z obrázku 6.1. Hysterezní operátor (hysteron) $\gamma_{\alpha,\beta} \in \{-1, 1\}$, $-\infty < \beta < \alpha < \infty$, na vstupním signálu $u(t)$ definujeme jako

$$\gamma_{\alpha,\beta} = \begin{cases} -1, & u(t) \leq \beta, \\ 1, & u(t) \geq \alpha, \\ \xi, & u(t) \in (\beta, \alpha), \end{cases}$$

$$\xi = \begin{cases} 1, & \text{pokud } \exists t^* < t : u(t^*) > \alpha, \\ -1, & \text{pokud } \exists t^* < t : u(t^*) < \beta, \end{cases} \quad \text{přičemž } \forall \tau \in (t^*, t) : u(\tau) \in (\beta, \alpha).$$

Tedy hodnota hysteronu je jednoznačně dána pro vstupní signál mimo interval (β, α) a pro hodnoty z tohoto intervalu je třeba znát vstupní signál v předchozích časech. Uvažujeme-li množinu jednoduchých hysterezních operátorů, definujeme Preisach-Mayergoyzův model jako jejich superpozici

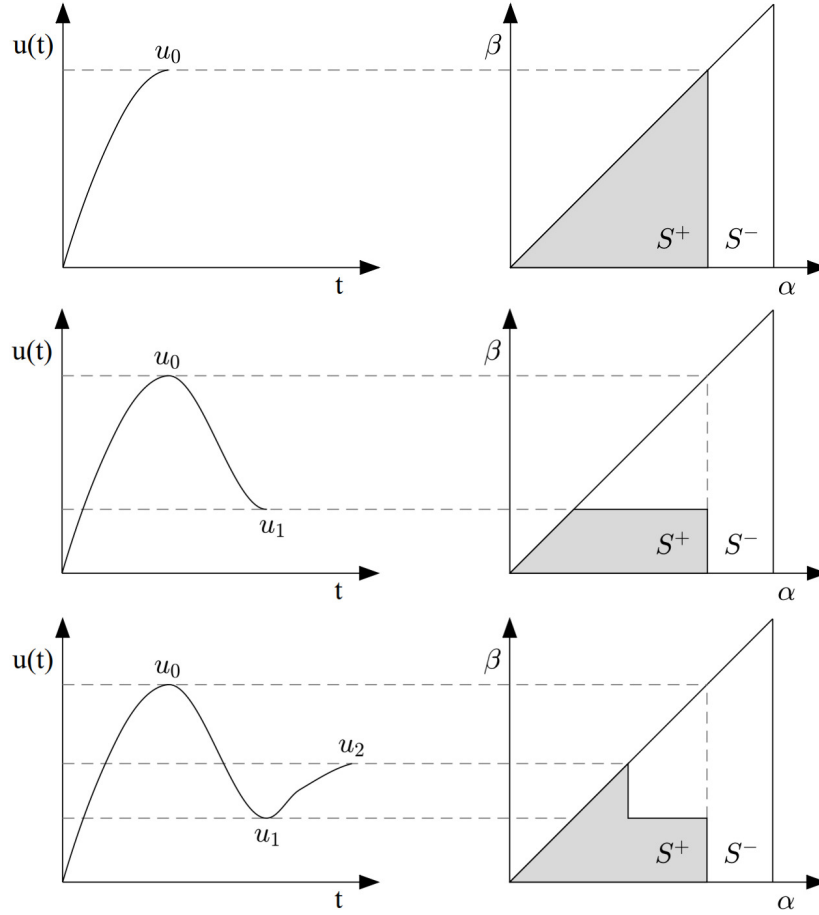
$$v(t) = \Gamma(u(t)) = \iint_{\beta < \alpha} \mu(\alpha, \beta) \gamma_{\alpha,\beta}(u(t)) d\alpha d\beta, \quad (6.1)$$

kde $t \in \mathbb{R}_0^+$ a $\mu(\alpha, \beta)$ označuje nedegenerovanou hustotu pravděpodobnosti na prostoru parametrů charakterizujících hysterony. Tuto hustotu nazýváme Preisachovou funkcí.

Pro vizualizaci PM modelu vykreslujeme jednotlivé hysterony, z nichž superpozice Γ sestává, podle jim příslušejících hodnot α, β do parametrického prostoru. Vzhledem k podmínce $\beta < \alpha$ a v praxi platnému předpokladu omezených hodnot α, β budou navíc všechny hysterony ležet uvnitř pravoúhlého rovnoramenného trojúhelníku, jak lze nahlédnout z obrázku 6.2. Hustota pravděpodobnosti μ (Preisachova funkce) je mimo tento limitní trojúhelník dodefinována nulou. Vliv vstupního signálu $u(t)$ na libovolný PM model je znázorněn na obrázku 6.2 převzatém z [32]. Předpokládejme, že na počátku v čase $t = 0$ jsou všechny hysterony otevřené (na úrovni -1). Při monotónním nárůstu $u(t)$ až na hodnotu u_0 dojde k uzavření (přechodu na hodnotu 1) u všech hysteronů s parametrem $\alpha < u_0$, to znázorňujeme šedou oblastí S^+ . Při následném monotónním poklesu vstupního signálu na úroveň u_1 dojde k opětovnému otevření všech hysteronů s parametrem $\beta > u_1$, a opět při nárůstu $u(t)$ na hladinu u_2 se uzavřou hysterony s $\alpha < u_2$. Tímto způsobem se Preisachův trojúhelník rozdělil na dvě části: S^+ - uzavřené hysterony a S^- - otevřené hysterony. Dělicí hranice $L(t)$ těchto dvou množin má podobu lomené čáry a je závislá pouze na lokálních extrémech vstupního signálu. Díky tomuto rozdělení můžeme přepsat rovnici (6.1) do podoby

$$v(t) = \Gamma(u(t)) = \iint_{S^+(t)} \mu(\alpha, \beta) d\alpha d\beta - \iint_{S^-(t)} \mu(\alpha, \beta) d\alpha d\beta.$$

Identifikace PM prostoru se tedy omezila na hledání rozdělení pravděpodobnosti $\mu(\alpha, \beta)$ v Preisachově trojúhelníku. Vstupní signál a jeho aplikace na konkrétní PM prostor dohromady vytvoří hysterezní křivku. V praxi je však k dispozici pouze vstupní signál a empiricky naměřená hysterezní křivka. Naším cílem je tak na základě těchto měřených veličin co nejlépe odhadnout příslušné rozdělení hysteronů, ze kterého hysterezní křivka pochází.



Obrázek 6.2: Vliv vstupního signálu $u(t)$ na PM prostor [32].

V našem případě modelování elasticity, resp. poškození materiálu, považujeme hysteron za základní elastickou jednotku, ze které je materiál tvořen. Hodnoty α a β udávají jeho zavírací a otevírací tlak. Uzavřený hysteron v našem případě ztrácí schopnost pohlcovat zatížení a snižuje tak celkovou elasticitu materiálu. Hysterony blízko tzv. dokonale elastické diagonály $\alpha = \beta$ pohlcují zatížení téměř dokonale (pružně), na rozdíl od vzdálenějších hysteronů, které vyžadují značný pokles zatížení pro jejich regeneraci.

6.2 Identifikace PM prostoru

Klíčovou úlohou při modelování PM prostoru je identifikace Preisachovy funkce, tedy hustoty rozdělení hysteronů μ , pouze na základě znalosti vstupního zatížení $u(t)$ a příslušné hysterezní křivky. Za tímto účelem volíme obdobný přístup jako v [32, 33], tedy uvažujeme μ v podobě distribuční směsi. Distribuční směsí myslíme každou konvexní kombinaci komponent p_i , tedy

$$p(\mathbf{x}|\Theta) = \sum_{i=1}^M \lambda_i p_i(\mathbf{x}|\theta_i), \quad \sum_{i=1}^M \lambda_i = 1, \quad \lambda_i > 0, \quad \forall i \in \widehat{M}, \quad (6.2)$$

kde M je počet komponent, $\Theta = (\lambda_1, \dots, \lambda_M, \theta_1, \dots, \theta_M)$ je množina parametrů distribuční směsi a $p_i(\mathbf{x}|\theta_i)$ jsou komponentní hustoty pravděpodobnosti. Jako jednotlivé komponenty směsi budeme uvažovat celkem 8 různých rozdělení, která v další části textu popíšeme.

Roli vstupního signálu má v našem případě zatížení (tlak) P působící na materiál. Pro popis jednotlivých hysteronů tedy přecházíme od parametrů α, β k parametrům P_c - zavírací (closing) a P_o - otevírací (opening) tlak. Nechť dále ω značí maximum dosaženého vstupního tlaku a $r_c, r_o \sim \mathcal{U}(0, 1)$ jsou realizace náhodné proměnné s rovnoměrným rozdělením na intervalu $(0, 1)$, které slouží pouze pro generování P_c a P_o . Potom zavádíme celkem čtyři rozdělení, která byla navržena pro popis nohomogenních materiálů v [35].

Rozdělení Guyer 1

$$P_c = \omega \cdot r_c^\eta, \quad P_o = P_c \cdot r_o^\kappa,$$

kde $\eta, \kappa \geq 0$ jsou parametry rozdělení.

Rozdělení Guyer 2

$$P_c = \omega \cdot r_c^\eta, \quad P_o = P_c \cdot r_o^{\frac{1}{4} + \frac{3}{4}\kappa},$$

kde $\eta, \kappa \geq 0$ jsou parametry rozdělení.

Rozdělení Koen

$$P_c = \omega \cdot r_c, \quad P_o = \left(\frac{P_c}{\lambda}\right)^\kappa \cdot r_o,$$

kde $\lambda, \kappa \geq 0$ jsou parametry rozdělení.

Rozdělení Guyer 3

$$P_c = \omega \cdot r_c^\eta, \quad P_o = P_c \cdot (\lambda r_o)^\kappa,$$

kde $\eta, \kappa \geq 0, \lambda \in (0, 1)$ jsou parametry rozdělení.

Dále využíváme běžná dvourozměrná rozdělení, která jsou omezena na Preisachův obdélník. To znamená, že jejich definiční obor splňuje $\mathcal{D} = \{(x_1, x_2) : x_1, x_2 \in [0, \omega] \wedge x_1 \geq x_2\}$. Toto omezení porušuje normalizaci hustot, vzhledem k charakteru našeho problému, kdy hustoty využíváme pouze pro generování bodů v Preisachově trojúhelníku nám postačí nenormalizovaná rozdělení.

Normální rozdělení

$$(P_c, P_o)^T \sim \frac{1}{2\pi\sqrt{|\mathbb{C}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbb{C}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right),$$

kde parametry rozdělení jsou vektor středních hodnot $\boldsymbol{\mu} \in \mathbb{R}^+ \times \mathbb{R}^+$ a symetrická kovarianční matice $\mathbb{C} \succeq 0 \in \mathbb{R}^{2,2}$.

Exponenciální rozdělení

$$P_c = \mu_1 \exp(-\mu_1 x_1), \quad P_o = \mu_2 \exp(-\mu_2 x_2),$$

kde μ_1, μ_2 jsou parametry rozdělení.

Weibullovo rozdělení

$$P_c = \frac{\kappa_1}{\lambda_1} \left(\frac{x_1}{\lambda_1} \right)^{\kappa_1-1} \exp \left[- \left(\frac{x_1}{\lambda_1} \right) \right], \quad P_o = \frac{\kappa_2}{\lambda_2} \left(\frac{x_2}{\lambda_2} \right)^{\kappa_2-1} \exp \left[- \left(\frac{x_2}{\lambda_2} \right) \right],$$

kde $\kappa_1, \kappa_2, \lambda_1, \lambda_2 > 0$ jsou parametry rozdělení.

Laplaceovo rozdělení

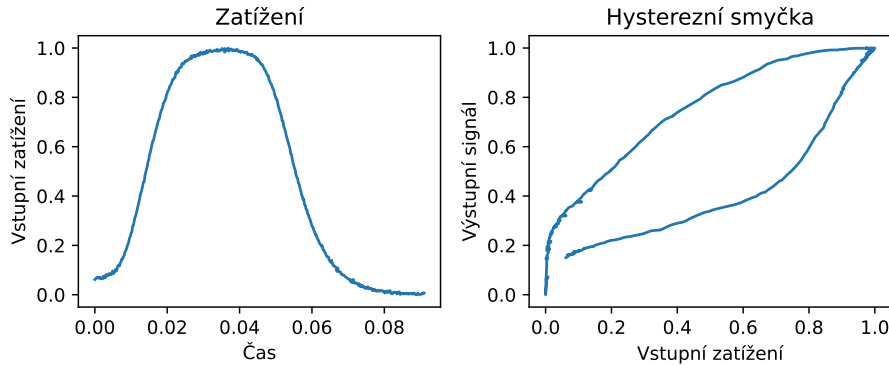
$$P_c = \frac{1}{2\lambda_1} \exp \left(- \frac{|x_1 - \mu_1|}{\lambda_1} \right), \quad P_o = \frac{1}{2\lambda_2} \exp \left(- \frac{|x_2 - \mu_2|}{\lambda_2} \right)$$

kde $\lambda_1, \lambda_2, \mu_1, \mu_2 > 0$ jsou parametry rozdělení.

V [32, 33] byla úloha hledání vhodného pravděpodobnostního rozdělení řešena za pomoci meta-heuristických optimalizačních algoritmů Jaya, aDE-Jaya a simulovaného žihání. V rámci této práce navazujeme na první zmíněný zdroj ve snaze výpočetně zefektivnit tyto optimalizační algoritmy za pomoci metod hlubokého strojové učení, které by identifikovaly typ příslušné distribuční směsi. Zde se za tímto účelem pokusíme využít metody, které byly představeny pro analýzu AE. Za tímto účelem je nutné vytvořit vhodný dataset pro trénování těchto neuronových sítí.

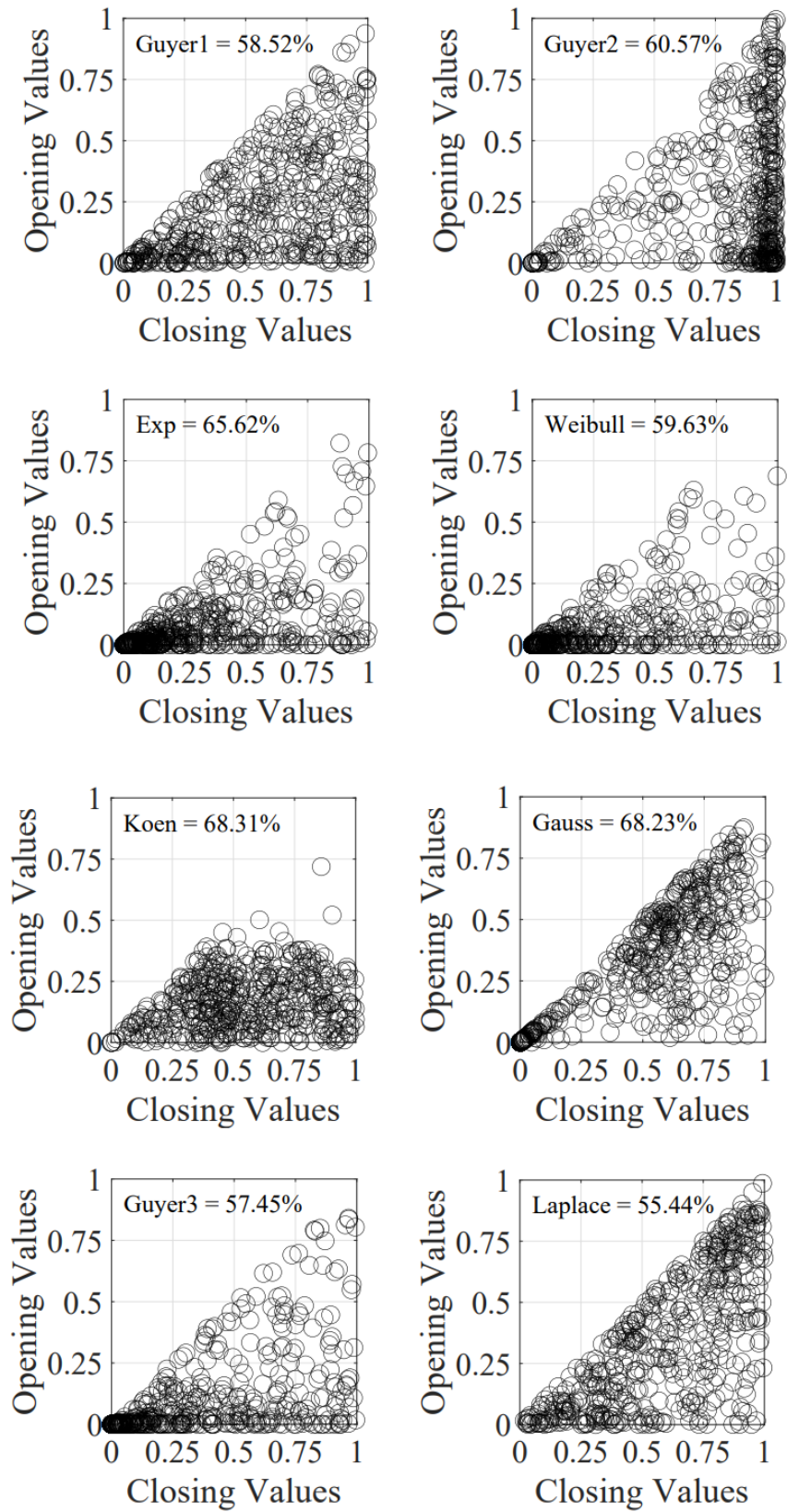
6.3 Hysterezní data

Předpokládáme, že rozdělení hysteronů na Preisachově prostoru má podobu distribuční směsi s až 8 komponentami, kde pravděpodobnostní rozdělení představená v minulé kapitole tvoří jednotlivé komponenty. Problém hledání vhodného rozdělení za použití neuronových sítí zjednodušíme a definujeme dvěma přístupy. Prvním z nich je určit, na základě hysterezní křivky a zatěžovacího signálu, která komponenta je v distribuční směsi dominantní, distribuční směs by potom bylo možné modelovat pouze tímto jedním rozdělením. Druhým přístupem, který se nedopouští tak hrubého zjednodušení, je provést nad hysterezní křivkou regresní úlohu a predikovat koeficienty zastoupení λ_i jednotlivých komponent ve směsi. Po této výpočetně nenákladné identifikaci vhodného typu distribuční směsi za pomoci neuronové sítě lze využít výše zmíněné heuristické algoritmy k dohledání konkrétních parametrů jednotlivých distribucí.



Obrázek 6.3: Normovaný zatěžovací profil pocházející z reálných měření a jemu příslušná hysterezní křivka z generovaného PM prostoru s dominantním zastoupením Laplaceovy distribuce.

Pro natrénování neuronových sítí za tímto účelem využíváme dataset vytvořený v rámci [32]. Ten vznikl vytvořením směsí distribučních funkcí představených v předešlé části. V každé vygenerované směsi byla jedna komponenta v dominantním zastoupení, tedy s koeficientem λ_i větším než $\frac{1}{2}$. Zastoupení zbylých komponent bylo voleno náhodně a tvoří formu šumu. Při tvorbě datasetu bylo zajištěno, aby byly jednotlivé distribuce použity jako dominantní obdobně často a dataset byl v tomto směru vyvážený. Počet komponent zastoupených ve směsi není fixní, byly tak vytvořeny směsi obsahující od jedné po osm komponent. Pro rozdělení závislá na parametrech byly tyto parametry vygenerovány náhodně, aby byla simulována realita, kdy zpravidla konkrétní distribuci neznáme. Celkem jsme pracovali s datasetem obsahujícím 12800 takto vygenerovaných směsí. Z každé z nich byl vytvořen odpovídající PM prostor, vybrané ukázky těchto prostorů jsou na obrázku 6.4. Následně byl využit normovaný fixní vstupní zatěžovací profil, pocházející z reálných měření při testování zemětřesných tlumičů, pro vytvoření příslušných hysterezních křivek. Tento profil je vykreslen na obrázku 6.3 společně s hysterezní křivkou odpovídající PM prostoru s dominantní Laplaceovou distribucí. Hysterezní smyčky jsou finální podobou dat, která zpracováváme za použití hlubokých neuronových sítí. V [32] bylo představeno několik metod pro předzpracování smyček, jejich převod na obrázek a následné zpracování pomocí 2D konvoluční sítí. Zde využíváme architektury představené pro analýzu AE, které jsou navrženy přímo pro zpracování 1D dat, a proto využíváme přímo vektor výstupního signálu definující danou hysterezní křivku.



Obrázek 6.4: Vygenerované PM prostory směsí s příslušnou dominantní komponentou [32].

6.4 Výsledky pro hysterezní data

Pro analýzu hysterezních křivek využíváme architekturu InceptionTime. Délka časových řad popisujících hysterezní křivku je o řád nižší než výřezy signálů AE, proto zde nenarážíme na problém s přetečením dostupné paměti. Jak již bylo zmíněno, pro hysterezní data máme dva cíle. Prvním je klasifikace jednotlivých hysterezních křivek do osmi tříd podle dominantního rozdělení zastoupeného v distribuční směsi na Preisachově prostoru, z něhož daná křivka pochází. Druhým cílem je predikce vah jednotlivých rozdělení zastoupených v distribuční směsi. Stejně jako v případě analýzy AE, nynější dataset náhodně rozdělujeme na trénovací, validační a testovací množinu v poměru 7:2:1.

6.4.1 Klasifikace dominantního rozdělení ve směsi

Při klasifikaci se snažíme pouze na základě výstupního signálu definujícího hysterezní křivku rozdělit jednotlivé realizace do osmi tříd odpovídajících dominantnímu rozdělení ve směsi, ze které pocházejí. Vzhledem k tomu, že všechny hysterezní křivky byly vytvořeny z jednoho zatěžovacího profilu, není v tomto nastavení nutné předávat klasifikační metodě vektor popisující průběh zatížení. V případě budoucí práce s rozmanitějším datasetem, který bude využívat více různých zatěžovacích profilů, lze triviální úpravou předat klasifikační metodě i průběh zatížení jako další kanál vstupu.

Po manuální optimalizaci hyperparametrů jsme zvolili podobu sítě Inception Time se třemi Inception moduly, vnitřní dimenzí 256 a originální délkou filtrů {10, 20, 40}. Trénování probíhalo s velikostí batche 32 a s čekací lhůtou pro early stopping 60. Poslední vrstva sítě má 8 neuronů, z nichž každý dává po aplikaci softmax aktivace pravděpodobnost, že daná hysterezní křivka pochází z příslušné třídy. Jako ztrátovou funkci používáme, stejně jako pro všechny klasifikační úlohy, kategorickou křížovou entropii.

True class	Predicted class							
	Guyer 1	Guyer 2	Koen	Gauss	Exp	Weibull	Guyer 3	Laplace
Guyer 1	110	32	3	0	1	11	17	5
Guyer 2	50	74	2	0	3	12	8	1
Koen	11	11	103	5	6	3	8	1
Gauss	7	3	2	113	12	0	0	24
Exp	4	1	3	5	153	3	0	5
Weibull	9	5	1	0	16	86	8	0
Guyer 3	11	8	5	1	3	5	157	0
Laplace	7	6	7	30	11	1	1	90

Obrázek 6.5: Matice záměn zobrazující výsledky klasifikace metodou Inception Time.

Výše popsaná architektura sítě Inception Time po natrénování trvajícím 270 epoch dosáhla na testovacím datasetu přesnosti klasifikace do osmi tříd 69,3%. Na obrázku 6.5 je matice záměn

této klasifikace. Z té lze vyčíst, pro které třídy byly hysterezní křivky nejčastěji klasifikovány správně, jmenovitě třídy s dominantním exponenciálním nebo Guyer 3 rozdělením. Také lze nahlédnout, že klasifikátor má největší problém od sebe odlišit třídy Guyer 1 a Guyer 2, což je vzhledem k jejich podobné definici očekávatelné. Další často zaměňované třídy jsou Laplace a Gauss, případně Guyer 1 a Guyer 3. Vzhledem k tomu, že při praktickém použití tohoto přístupu k určení dominantní komponenty směsi rozdělení bychom následně modelovali Preisachovu funkci pouze tímto jedním rozdělením, není dosažená úspěšnost správné klasifikace příliš uspokojivá. Pro zvýšení přesnosti klasifikace by bylo potřebné se spíše než na pokročilejší klasifikační metody zaměřit na rozšíření a zobecnění trénovacího datasetu.

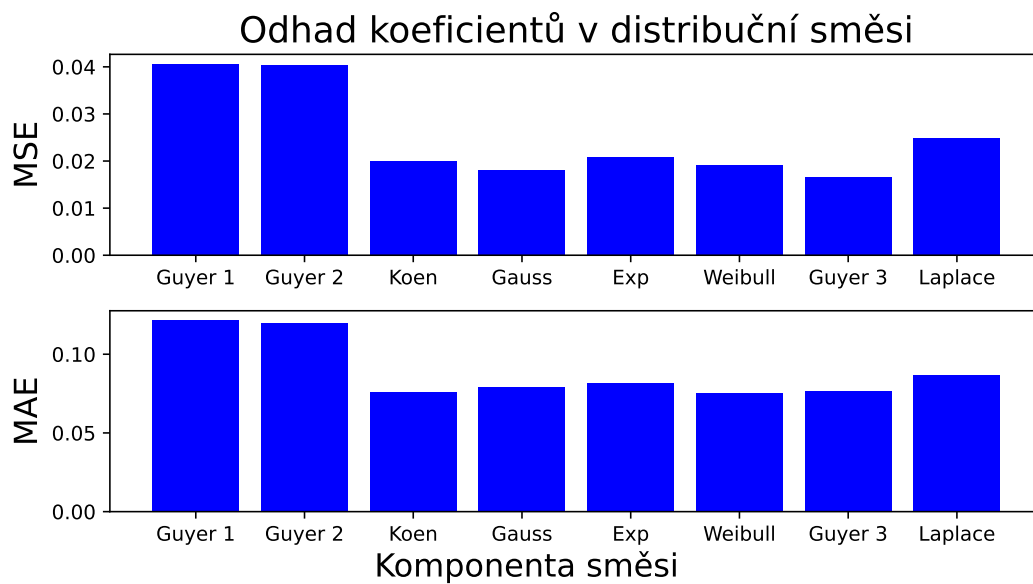
Výsledky klasifikace můžeme porovnat s výsledky z práce [32], která se klasifikaci hysterezních dat věnovala do větší hloubky a pracovala s rozsáhlejším datasetem. Tamní nejlepší klasifikační metoda měla úspěšnost 78,3%. Naše klasifikace metodou Inception Time sice nedosáhla tak vysoké přesnosti, potvrdila však koncept využití neuronových sítí pracujících pouze s vektorem popisujícím hysterezní křivku.

6.4.2 Odhad koeficientů v distribuční směsi

Druhým přístupem k analýze hysterezních smyček a odhadu tvaru distribuční směsi, ze které hystereze pochází, je regresní úloha podobná úloze provedené pro odhad zatížení ložiska na základě signálů AE v sekci 5.6. V tomto případě na základě hysterezních smyček predikujeme koeficienty zastoupení jednotlivých rozdělení v distribuční směsi, tedy vektor λ z definice (6.2). Za tímto účelem je třeba v datasetu k jednotlivým hysterezním křivkám přiřadit celý vektor λ distribuční směsi, ze které pocházejí, namísto pouhého labelu určujícího dominantní rozdělení, jako to bylo v předchozí klasifikační úloze.

Pro samotnou regresí jsme využili síť Inception Time s upravenou strukturou oproti klasifikační úloze. Jednalo se o síť složenou z 9 Inception modulů, s vnitřní dimenzí 64 a délkou filtrů 20, 50, 100. Poslední vrstva sítě měla 8 neuronů, na něž byla aplikována aktivační funkce softmax, proto aby výsledné predikce splňovaly požadavky na vektor λ dané v (6.2). Jako ztrátovou funkci jsme použili MSE, ta je v tomto případě počítána jako průměr přes všechna pozorování v mini-batchi a přes všechny složky predikovaného vektoru.

Regresní neuronovou síť jsme natrénovali standardním způsobem s čekací lhůtou pro early stopping callback 90 epoch. Síť predikovala koeficienty λ pro testovací dataset s MSE 0,025. Lépe interpretovatelná metrika MAE dosáhla na testovacím datasetu hodnoty 0,089. Z důvodu obtížnosti posouzení predikce pouze na základě skalární metriky jsme na obrázku 6.6 vykreslili metriky MSE a MAE zvlášť pro jednotlivé komponenty směsi. Podobně jako u úlohy klasifikace dominantní komponenty pozorujeme nejhorší výsledky pro rozdělení Guyer 1 a Guyer 2. Pro další výzkum by proto mohlo být vhodné prozkoumat variantu vypuštění jednoho z těchto rozdělení z uvažované distribuční směsi. Výsledky opět můžeme porovnat s prací [32], kde nejlepší regresní metoda dosáhla na testovacím datasetu hodnot 0,017 MSE a 0,075 MAE.



Obrázek 6.6: Výsledné dosažené metriky MSE a MAE při regresním použití Inception Time sítě na testovacím datasetu hystrezních smyček, rozdělené zvlášť pro jednotlivé predikované míxovací koeficienty, $\lambda = \{\lambda_1, \dots, \lambda_8\}$, definující váhy jednotlivých 8 komponent ve směsi distribucí generovaných PM prostorů.

Závěr

Hlavním cílem této práce bylo nalezení, implementace a následné porovnání moderních metod hlubokého učení, které by byly aplikovatelné ve zpracování signálů akustické emise. Dalším úkolem bylo otestovat využitelnost těchto metod pro analýzu tvaru rozdělení na PM prostoru popisujícím hysterezní chování v materiálech.

Po stručném popisu fenoménu akustické emise jsou v druhé kapitole představeny dva experimenty, z nichž pocházejí signály AE, které byly použity k naladění a porovnání metod hlubokého učení. Jednalo se o experiment sledující podobu signálu AE při vrtání vrtákem se vzrůstajícím opotřebením hrotu a o experiment snímající akustickou emisi vydávanou ložisky při jejich různém stupni zatížení.

Ve třetí kapitole byla popsána teorie na pozadí neuronových sítí. Ta byla následně využita pro popis celkem pěti implementovaných architektur neuronových sítí. Konvoluční síť Conv2Net, ResNet-16 a transformer TSiT byly inspirovány existujícími sítěmi pro zpracování obrazu a adaptovány pro použití na signálech AE. Síť Inception Time byla přejata v původní podobě a poslední síť Pooled Inception Time, vycházející z předem jmenované, byla vyvinuta v rámci práce a její podoba adresuje problémy s přílišnou paměťovou náročností jejího předchůdce.

Představené architektury byly následně využity pro klasifikační úlohu, jejímž cílem bylo rozdělit výřezy signálů do pěti, resp. osmi tříd, pro data z prvního, resp. druhého experimentu. Nejlepších výsledků dosáhla v obou případech naše modifikace sítě Pooled Inception Time, konkrétně 86,4% a 84,8% správně klasifikovaných pozorování. Tato přesnost zásadně převyšuje výsledky dosažené přístupem extrakce příznaků v [1]. Velmi špatné výsledky klasifikace pomocí TSiT dokazují náročnost trénování neuronových sítí založených na architektuře transformeru a poskytují prostor pro další výzkum. Součástí práce byla studie vlivu délky filtrů v konvolučních vrstvách na úspěšnost dané neuronové sítě, která ukázala obecně lepší výsledky při použití delších filtrů. Studie vlivu délky výřezu signálu na výsledky klasifikace nabídla možnost využití kratších výřezů bez výrazné ztráty přesnosti pro signály z 1. experimentu a potvrdila volbu délky pro data z 2. experimentu. Konečně, analýza výtěžnosti snímačů z 2. experimentu odhalila velmi nevyvážené rozdělení využitelné informace, která je z naprosté většiny obsažena v signálech pocházejících ze snímače akustické emise číslo 2.

Pro data z druhého experimentu byl za použití architektury Pooled Inception Time vytvořen regresní model, který na základě signálu AE predikoval spojité zatížení ložiska. Model dosáhl na testovacím datasetu kritériální hodnoty MAE 0,97kg, což považujeme za přijatelné a i pro praxi dostatečně jemné. V praxi se totiž nejspíše budou vyskytovat řádově větší a výrazně oddělenější stupně zatížení ložiskových náprav, robotických kloubových spojů, převodových stupňů, apod.

V poslední kapitole byla představena problematika modelování hysterezního chování za použití Preisach-Mayergoyzova modelu. Následně byly na syntetickém datasetu natrénovány neuronové sítě využité pro analýzu AE s cílem predikovat podobu distribuční směsi pouze na základě

vektoru popisujícího hysterezní smyčku. Síť Inception Time se ukázala jako využitelná, nedosáhla však až tak dobrých výsledků jako metody použité v práci [32], na kterou jsme navazovali, přesto se pohybujeme relativně blízko v rámci 5% odchylky identifikace distribuční směsi. Limity v této aplikaci jsou však spíše na straně dostupného trénovacího datasetu, než na straně klasifikační/predikční metody.

V rámci diplomové práce jsme navíc zpracovali data pocházející z dalšího experimentu provedeného na půdě Ústavu termomechaniky AV ČR. Jednalo se o experiment zkoumající pomalou dynamiku relaxace bloku pískovce po vyvedení z rovnovážného stavu mechanickým zatížením. Průběh relaxace byl sledován pomocí změny rychlosti šíření pulsu pískovcem a cílem bylo odhalit, zda je proces relaxace rozpoznatelný ze zaznamenané akustické emise. Vzhledem k úvodním negativním výsledkům nebyl experiment v rámci práce podrobněji rozebrán a tato problematika bude pravděpodobně vyžadovat další vývoj a výzkum z potenciálně jiného úhlu pohledu.

Literatura

- [1] J. Zavadil: *Selekce příznaků a klasifikace signálů akustické emise*. Výzkumný úkol, KM FJFI ČVUT, Praha, 2023.
- [2] J. Zavadil: *Statistická a strojová klasifikace signálů akustické emise pro detekci defektů v materiálech*. Bakalářská práce, KM FJFI ČVUT, Praha, 2021.
- [3] Benavides, Samuel: *Corrosion control in the aerospace industry*. Boca Raton: FL/CRC Press, 2009.
- [4] J. Žižka: *Použití akustické emise ke sledování stavu řezného nástroje*. Technická Univerzita Liberec, 2003. ISBN: 8070836881
- [5] pacuk.co.uk website December 27, 2011, at the Wayback Machine. Retrieved 2011-12-05.
- [6] V. Kratochvíl: *Studium signálu akustické emise ve vztahu k opotřebení nástroje*. Diplomová práce, Katedra technologie obrábění, Fakulta strojní, Západočeská Univerzita v Plzni, 2020.
- [7] T. Fucsová: *Strojové učení pro klasifikaci zdrojů spojitě akustické emise*. Bakalářská práce, KM FJFI ČVUT, Praha, 2022.
- [8] I. Tolstikhin et al.: *MLP-Mixer: An all-MLP Architecture for Vision*. arXiv:2105.01601, 2021. url: <https://doi.org/10.48550/arXiv.2105.01601>
- [9] D. Hendrycks, K. Gimpel: *Gaussian Error Linear Units (GELUs)*. arXiv:1606.08415, 2016. url: <https://doi.org/10.48550/arXiv.1606.08415>
- [10] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner: *Gradient-based learning applied to document recognition*. In Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
- [11] M. Kovanda: *Signal Event List Generation Using Neural Networks*. Diplomová práce, KM FJFI ČVUT, Praha, 2022.
- [12] X. Glorot, Y. Bengio: *Understanding the difficulty of training deep feedforward neural networks*. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, vol. 9, 2010. url: <https://proceedings.mlr.press/v9/glorot10a.html>
- [13] K. He, X. Zhang, S. Ren, J. Sun: *Delving Deep into Rectifiers: Surpassing Human Level Performance on ImageNet Classification*. arXiv:1502.01852. url: <http://arxiv.org/abs/1502.01852>.
- [14] . Ruder: *An overview of gradient descent optimization algorithms*. arXiv:1609.04747, 2017. url: <https://doi.org/10.48550/arXiv.1609.04747>

- [15] D. P. Kingma, J. L. Ba: *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980, 2017. url: <https://doi.org/10.48550/arXiv.1412.6980>
- [16] A. vaswani et al.: *Attention Is All You Need*. arxiv:1706.03762, 2017. url: <https://doi.org/10.48550/arXiv.1706.03762>
- [17] V. Obhlídál: *Application of transformers for system imbalance prediction in electric power transmission system*. Diplomová práce, KM FJFI ČVUT, Praha, 2024.
- [18] T. Mikolov: *Efficient Estimation of Word Representations in Vector Space*. arXiv:1301.3781, 2013. url: <https://doi.org/10.48550/arXiv.1301.3781>
- [19] N. Srivastava: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. In *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.
- [20] S. Ioffe, Ch. Szegedy: *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv:1502.03167, 2015. url: <https://doi.org/10.48550/arXiv.1502.03167>
- [21] J. L. Ba, J. R. Kiros a G. E. Hinton: *Layer Normalization*. arXiv:1607.06450, 2016. url: <https://doi.org/10.48550/arXiv.1607.06450>
- [22] K. He et al.: *Deep Residual Learning for Image Recognition*. arXiv:1512.03385, 2015. url: <https://doi.org/10.48550/arXiv.1512.03385>
- [23] M. Sokolova, G. Lapalme: *A systematic analysis of performance measures for classification tasks*. *Information Processing & Management*, Volume 45, Issue 4, 2009, pp. 427-437.
- [24] I. Fawaz et al.: *InceptionTime: Finding AlexNet for time series classification*. In *Data Mining and Knowledge Discovery*, vol. 34/6, pp. 1936-1962, 2020. Springer Science and Business Media LLC. url: <http://dx.doi.org/10.1007/s10618-020-00710-y>
- [25] N. M. Foumani et al.: *Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey*. arXiv:2302.02515, 2023. url: <https://doi.org/10.48550/arXiv.2302.02515>
- [26] J. Lines, S. Taylor a A. Bagnall: *HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles for Time Series Classification*. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, Barcelona, Spain, 2016, pp. 1041-1046.
- [27] A. Dosovitskiy et al.: *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929, 2021. url: <https://doi.org/10.48550/arXiv.2010.11929>
- [28] I. Oguiza: *tsai - A state-of-the-art deep learning library for time series and sequential data*. Github repository, 2023. url: <https://github.com/timeseriesAI/tsai>
- [29] X. Chu, et al.: *Conditional Positional Encodings for Vision Transformers*. arXiv: 2102.10882, 2023. url: <https://doi.org/10.48550/arXiv.2102.10882>
- [30] J. Devlin et al.: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805, 2019. url: <https://doi.org/10.48550/arXiv.1810.04805>

- [31] C. Abarkane: *Monitoring through acoustics techniques of RC earthquake-resistant structures subjected to dynamic loads on shaking table*. Disertační práce, Technical School of Building Engineering, University of Granada, 2019.
- [32] E. Dolejš: *Vývoj a srovnání PM metod pro odhady elasticity materiálů*. Diplomová práce, KM FJFI ČVUT, Praha, 2022.
- [33] C. Kožená, V. Kůs, A. Gallego a A. Benavent-Climent: *Damage assessment of earthquake dampers based on Preisach-Mayergoyz model*. 2018 16th Biennial Baltic Electronics Conference (BEC), Tallinn, Estonia, 2018, pp. 1-4, doi: 10.1109/BEC.2018.8600971.
- [34] I. Mayergoyz: *Mathematical Models of Hysteresis*. In *Magnetics*, IEEE Transactions on, vol. 22, pp 603 - 608, 1986.
- [35] K. R. McCall, R. A. Guyer: *Equation of state and wave propagation in hysteretic nonlinear elastic materials*. *Journal of Geophysical Research*, Vol. 90(B12), 1994.
- [36] M. Christ, N. Braun, J. Neuffer, A. W. Kempa-Liehr: *Time Series Feature Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)*, *Neurocomputing*, Volume 307, 2018, pp. 72-77.