FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Decoding visual stimuli from cortical activity using neural networks
Student:	Jan Sobotka
Supervisor:	Mgr. Ján Antolík, Ph.D.
Study program:	Informatics
Branch / specialization:	Artificial Intelligence 2021
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2025/2026

Instructions

Recent years have witnessed a major breakthrough in deep neural network (DNN) models' ability to predict neural population activity in the primary visual cortex (V1) evoked by novel visual stimuli [1,2]. However, the inverse problem of predicting the natural image based on the activity it elicits in a population of V1 neurons remains much less studied and consequently mastered. In this project, the student will implement and test a range of different DNN architectures, including variations of CNN models, generative approaches, and, optionally, other methods if time and space permit. Additionally, the student will quantify the impact of synthetically generated data used to train the models. The best-performing methods will be tested on large datasets acquired from a detailed biologically realistic model of the primary visual cortex developed in our group [3] and on recordings from mice or macaque V1.

[1] Dan A. Butts (2019). Data-driven approaches to understanding visual neuron activity. Annual Review of Vision Science, 5:451-457.
[2] Antolík, J., Hofer, S. B., Bednar, J. A., & Mrsic-Flogel, T. D. (2016). Model Constrained by Visual Hierarchy Improves Prediction of Neural Responses to Natural Scenes. PLoS Computational Biology, 12(6). https://doi.org/10.1371/journal.pcbi.1004927
[3] Ján Antolík, Cyril Monier, Yves Frégnac, and Andrew P. Davison (2019). A comprehensive data-driven model of cat primary visual cortex. BiorXiv.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 4 April 2024 in Prague.

Bachelor's thesis

DECODING VISUAL STIMULI FROM CORTICAL ACTIVITY USING NEURAL NETWORKS

Jan Sobotka

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Mgr. Ján Antolík, Ph.D.
May 15, 2024

Contents

List of Figures

List of Tables

List of code listings

I would like to thank my family for their support throughout my studies, allowing me to devote full attention to my interests and education. I am also profoundly grateful to my supervisor, Mgr. Ján Antolík, Ph.D., for his guidance and expertise provided throughout my exploration of computational neuroscience and the thesis project. Last but not least, a big thank you to Luca Baroni, M.Sc., for his ideas, advice, and readiness to help with every issue.

Declaration

In Prague on May 15, 2024

Abstract

This thesis explores the application of deep learning techniques for reconstructing visual stimuli from neural activity in the primary visual cortex (V1). The focus is placed on overcoming the scarcity of biological data by developing data-efficient architectures, analyzing the impact of synthetic training data, employing adversarial and transfer learning, and introducing novel auxiliary optimization objectives. A series of experiments is conducted using data from in silico simulations of cat V1 and in vivo recordings from mouse V1, highlighting the best-performing decoding approach and offering suggestions for future research. Notably, the methods developed in this thesis outperform some existing state-of-the-art decoding techniques according to several widely used evaluation measures. Overall, the results underscore the potential of machine learning in neural activity decoding and pave the way for future advancements in brain-computer interfaces and neuroscientific research.

Keywords neural activity decoding, neural data analysis, primary visual cortex, applied machine learning, image reconstruction, synthetic data, deep neural networks

Abstrakt

Tato práce zkoumá použití technik hlubokého učení pro rekonstrukci vizuálních podnětů z neuronální aktivity v primární zrakové oblasti (V1). Zaměřuje se na překonání nedostatku biologických dat vývojem úsporných architektur, analýzou dopadu syntetických trénovacích dat, využitím adversariálního a transferového učení a zavedením nových pomocných optimalizačních cílů. Je provedena řada experimentů s daty z in silico simulací V1 u koček a in vivo záznamů z V1 u myší, přičemž je poukázáno na nejlepší přístup k dekódování a jsou navrženy kroky pro budoucí výzkum. Metody vyvinuté v této práci překonávají některé stávající state-of-the-art techniky podle několika široce používaných hodnotících kritérií. Celkově výsledky zdůrazňují potenciál strojového učení pro dekódování neuronální aktivity a otevírají cestu k budoucímu pokroku v oblasti technologií propojující mozek s počítačem a v neurovědeckém výzkumu.

Klíčová slova dekódování neuronální aktivity, analýza neurálních dat, primární zraková oblast, aplikované strojové učení, rekonstrukce obrazu, syntetická data, hluboké neuronové sítě

List of abbreviations

AI	Artificial intelligence
ANN	Artificial neural network
BCI	Brain-computer interface
CNN	Convolutional neural network
DL	Deep learning
DNN	Deep neural network
EM	Encoder matching
fMRI	Functional magnetic resonance imaging
GAN	Generative adversarial network
MEG	Magnetoencephalography
MEI	Most exciting input
ML	Machine learning
MSE	Mean squared error
LGN	Lateral geniculate nucleus
PL	Perceptual loss
SSIM	Structural similarity index measure
SSIML	Structural similarity index measure loss
V1	Primary visual cortex

Introduction

Computational neuroscience has a long-standing goal of deciphering how the brain processes and stores information. Although decades of work have gone into this endeavor, the precise mapping from brain activity to thoughts and perceptions remains a mystery.

To address these unresolved challenges, this thesis tackles the task of reconstructing visual percepts from the population activity of neurons in the early visual system. The motivation for solving this problem is twofold. First, it would shed light on the algorithms that the brain uses to compress the high-dimensional external world. Second, it would advance the development of brain-machine interfaces for restoring vision in blind people.

More specifically, visual prostheses, the technology behind vision restoration, require protocols to artificially stimulate neurons in order to induce the desired percepts. However, the experimental validation and refinement of these protocols is currently limited to subjective human reports in the form of language descriptions and drawings, or to other behavioral responses in the case of animals. A method capable of decoding brain activity into visual stimuli would bypass these limitations, enabling more precise testing and enhancing the reliability of vision restoration technology.

From a scientific perspective, the computational reconstruction of visual stimuli would offer a lower-bound estimate of the information content within specific brain regions. Additionally, such decoding systems could assess the accuracy and credibility of biologically detailed models widely used in computational neuroscience.

Despite these promising applications, significant challenges in the interpretation of cortical population activity still persist, and our limited understanding of the highly complex and non-linear brain processing calls for innovative approaches.

Drawing on recent advances in machine learning (ML), particularly in computer vision and image generation, this thesis aims to leverage learning-based techniques to decode image stimuli from neural activity in the primary visual cortex (V1). Given the inherent challenge of acquiring large biological datasets, the focus is directed towards data-efficient methods and training paradigms.

Specifically, the first objective of this thesis is to implement and test a range of different deep neural network (DNN) architectures in which data efficiency is introduced by additional learning signals, such as adversarial losses, and by parameter-efficient designs. To further alleviate the limited amounts of biological data, the second objective is to introduce synthetically generated datasets during DNN training and quantify their impact on performance.

Together, these two overarching goals will culminate in selecting the best-performing combination of model architecture and training paradigm. Evaluations will be conducted on data acquired from a detailed biologically plausible model of cat V1, as well as on experimental data from mouse V1 recordings.

The first four chapters start by providing background on the modeling of neural activity, generative adversarial networks, synthetic data, and image reconstruction metrics. The fifth chapter describes the datasets used by the methods introduced in chapter 6 during the experiments in chapter 7. Finally, the thesis concludes with a summary of key findings and directions for future work.

Chapter 1

Modeling of neural activity

Neural activity encompasses the electrical and chemical dynamic processes that occur in the brain. In computational neuroscience, researchers investigate both its underlying causal mechanisms and the higher-level statistical patterns through computational modeling. This in silico modeling is employed at multiple levels of abstraction, from precise biologically realistic models of ion channels and neurons to predictions of whole-brain imaging results. While one approach usually prioritizes depth over breadth, capturing intricate details, the other focuses more on large-scale regularities, sacrificing low-level specifics.

In the rest of this work, the term *neural activity* will refer exclusively to the firing rates of individual neurons. That is, to the (average) number of spikes of single neurons in a prespecified time window.

When external stimuli, such as visual, auditory, or tactile, are explicitly considered, two directions of investigation emerge. One is the forward process of modeling the neural activity in response to known stimuli, and the other is the backward (inverse) process of reconstructing the stimuli knowing the neural activity. Both can provide useful findings and applications and will be further introduced in the following sections, together with a brief review of the neuroscience behind vision, the main focus of the thesis.

1.1 Neuroscience behind visual processing

Vision begins in the retina, where light stimulates photoreceptors called rods and cones. The rods provide night vision and are generally good at detecting light, while cones are responsible for color vision and high spatial acuity. This first light detection step creates the visual field in which researchers pay special attention to the receptive fields of neurons along the visual hierarchy. That is, to the areas where visual stimuli influence the activity of specific neurons.

The light information encoded in the stimulated photoreceptors is then sent to bipolar and ganglion cells, which aggregate the local signals and perform initial processing. For example, some ganglion cells are activated when the center of their circular receptive fields detects a higher luminance than the surrounding, while other cells are activated in the opposite case.

From the retina, visual signals travel through the optic nerve to the lateral geniculate nucleus (LGN), which lies deep in the middle of the brain. This evolutionarily old brain structure not only relays but also modulates the passing visual information, influenced by additional non-visual factors such as attention and the task currently performed [1].

The visual information then proceeds to the primary visual cortex (V1), one of the first cortical areas reached by the external stimulus. Neurons in this area need more sophisticated visual patterns to activate compared to the simplicity of patterns detectable in the retina and LGN.

More specifically, it was found that V1 neurons respond most strongly to bars of light moving through their receptive fields and that the responses are selective to the bar's orientation [2].

In addition, V1 is notable for its retinotopic organization, where neighboring neurons in the retina connect to adjacent neurons in the LGN as well as in V1. Retinotopy is particularly important and interesting because, to some degree, it preserves the locality and spatial organization of the visual field in V1.

Subsequently, in addition to numerous other outputs of V1, the information travels through two main streams for further processing: the dorsal stream, often referred to as the "where" pathway, and the ventral stream, often referred to as the "what" pathway. In the ventral stream, the signals from V1 travel down to the inferior temporal lobe, which, among other tasks, is capable of recognizing objects. An example area on this pathway is called V4 and is activated by specific shapes and colors. In the dorsal stream, on the other hand, the information from V1 goes up through the parietal lobe, where motion detection and spatial components of vision take place. For example, neurons in area V5 are preferentially activated by left-to-right or up-to-down movements of objects. As one can see, the further in the visual pathway, the more complex the detected patterns can be, which is also a side-product of increasingly larger receptive fields. [3]

For further details and a broader neuroscientific context, interested readers can refer to [4].

1.2 Encoding neural activity

Models that predict neural activity in response to external stimuli are called encoding models. They learn or simulate the forward process of neural information encoding. The difficulty of their task varies depending on the level of abstraction and the brain region under consideration. For example, due to the complexity of the visual pathway and the differences in encoded information, predicting the average firing rates of individual neurons in the visual area V4 in response to natural images has been shown to be more difficult than predicting responses in V1 [5].

Despite the inherent challenges of this task, extensive research efforts have produced significant successes over the years, particularly in encoding neural activity within the primary visual area V1. The progress started with simple linear-nonlinear and multi-layer models [6, 7, 8] that were heavily based on neurophysiological data and known neuronal properties. Subsequent breakthroughs followed with the advent of deep learning (DL), which has since become dominant in the field [9, 10, 11, 12].

Artificial neural networks (ANNs) [13, 14], the workhorse behind DL, are known for their ability, under mild conditions, to approximate any continuous function with arbitrary precision [15, 16]. This makes them a particularly useful tool for capturing the nonlinear stimulus-activity mappings of neurons. In general, they consist of a series of layers that interleave linear matrix multiplications with nonlinear functions, transforming the input into the output in a nontrivial way. Importantly, the matrices inside the networks are learnable parameters that can be tuned to fit the data. For an extensive background on ANNs and DL, we refer the reader to [17].

The state-of-the-art approaches for encoding neural activity using ANNs consist of two parts [11, 18, 19]. The first is the *core*, which learns a latent representation of images that is common to all neurons and shared between potentially different animals and multiple recordings. The second are recording-specific *readout* modules that predict neural responses given the latent features from the core. This split allows for more efficient reuse of data, which in turn enables more powerful models to be used.

1.2.1 Core

The core is usually implemented as a convolutional neural network (CNN) [20], a variant of neural networks that is highly effective, computationally efficient, and particularly well suited for processing data with a grid-like topology, such as images [17]. CNNs consist of a specialized architecture inspired by the organization of the visual cortex, where neurons in successive layers respond to increasingly complex patterns.

The key components of a typical CNN include convolutional layers, nonlinear activation functions, pooling layers, and fully connected layers. The convolutional layers perform feature extraction by convolving input data with learnable filters, capturing spatial patterns at different scales. The activation functions transform these outputted feature maps which are then downsampled by the pooling layers, reducing the spatial dimensions and extracting dominant features. Finally, fully connected layers integrate the extracted features into a single vector that represents the final prediction of the network. The pooling and fully connected layers are sometimes omitted, and other modules, such as batch normalization [21] and dropout [22], are added, depending on the particular task.

One of the main advantages of CNNs is their ability to automatically learn hierarchical representations of features directly from the data, without the need for manual feature engineering. This makes them highly adaptable to various tasks, including image classification, object detection, and, in the context of neuroscience, encoding (and decoding) neural activity patterns.

The design and architecture of the CNN core can vary depending on the specific requirements of the task and the characteristics of the input data. For example, for neural activity encoding, the authors of [11] use four depth-separable convolutional layers with 64 filters each [23], where each 2D convolution is followed by a batch normalization and an ELU nonlinearity [24].

1.2.2 Readout

Complementary to the core, the readout modules are responsible for mapping the learned latent features to neural responses specific to individual animals and recordings. These modules typically have fewer learnable parameters than the core, which allows them to be trained from smaller recording datasets. They are trained in conjunction with the core CNN, taking advantage of its feature extraction to learn recording-specific mappings between visual stimuli and neural activity. The use of recording-specific readout modules enables the model to adapt to variations in neural response characteristics across different animals, improving the overall performance and robustness of the encoding model. Additionally, by separating the core CNN from the readout modules, the model architecture remains flexible and scalable, allowing easier adaptation to new datasets and experimental conditions.

One proposed implementation of the readout is called a *Gaussian readout* [11]. It significantly reduces the number of parameters compared to previous approaches by learning per-neuron spatial coordinates within the feature representation of the core. Given these positions at which neurons attend, the predicted responses are then linear combinations of feature map channels at their learned locations followed by an ELU + 1 activation function. The name of this readout stems from the fact that during training, the learned location serves as the mean of a multivariate Gaussian distribution with a learned covariance matrix. Its purpose is to facilitate gradient flow during training by sampling and applying the reparametrization trick from [25]. Furthermore, instead of learning the per-neuron positions directly, the Gaussian readout utilizes the neural positions along the cortical surface available from experiments. More specifically, it trains a single fully connected network that transforms the neural positions in the cortex into the positions in the feature representation predicted by the CNN core.

The authors of [19] build on the model architecture of [11], and extend the Gaussian readout by adding a *shifter network*, a fully connected network with three layers of hidden dimension of 5 followed by a Tanh nonlinearity. Its goal is to incorporate also the pupil position of animals into the predicted locations of neurons by predicting their shift. The reason is that the authors employed a free-viewing paradigm for the recorded animals, which resulted in varying receptive field positions after eye movements. This version of the CNN-based encoder, which has been open-sourced[1], is shown in Figure 1.1 and will be used in this work.

[1]`https://github.com/sinzlab/sensorium`

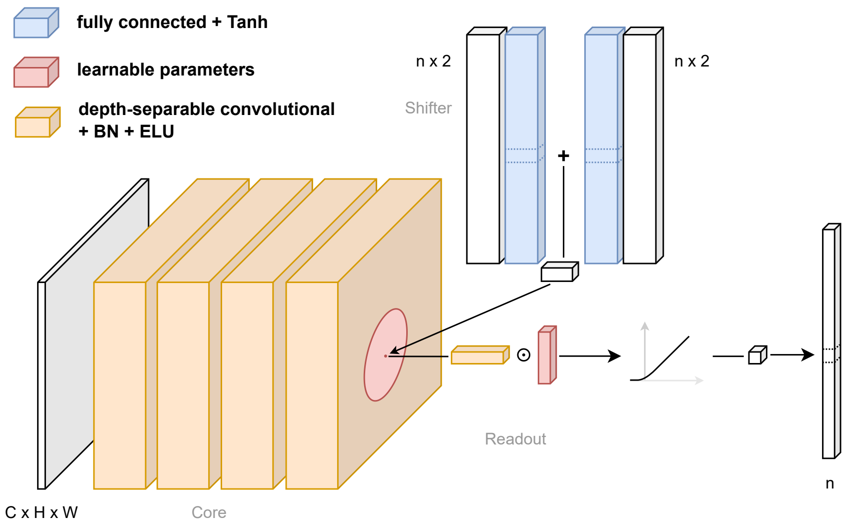


Figure 1.1 Encoder architecture. n denotes the number of neurons, and C, H, and W are the dimensions of the image.

1.2.3 Most exciting inputs

In certain applications, it is desirable to synthesize inputs (images) that maximize the response of a given neuron in the visual system. Such generated image stimuli are in the literature called *most exciting inputs* (MEIs). As with other tasks in computational neuroscience, ANNs have been successfully applied to this task using a range of different approaches [26, 27, 28]. The most common is through direct pixel optimization, a well-established technique in the field of interpretable machine learning for visualizing features learned by a neural network [29, 30].

It works by first training a CNN-based encoder model, randomly initializing a MEI, and then iteratively adding the gradient of the encoder's prediction for the target neuron w.r.t. this image. This gradient ascent is often interleaved with a Gaussian filter applied to the generated image or to its gradient in order to suppress high-frequency noise. The resulting MEIs from this method have been successfully validated in vivo [26], pointing at the promising application of deep learning models as digital twins of the biological brain.

Example MEIs corresponding to randomly sampled neurons from mouse V1 are shown in Figure 1.2.

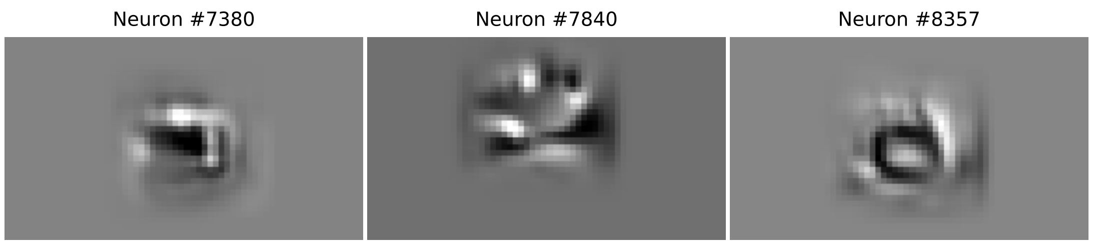


Figure 1.2 Most exciting input images of randomly sampled neurons.

1.3 Decoding neural activity

The reverse process of reconstructing some external variable, for example, the stimulus, from the elicited neural activity is known as decoding. Despite progress in encoding neural activity and in decoding a few variables, such as stimulus classes, decoding of entire image stimuli has seen fewer developments, particularly from responses of neurons in the visual cortex.

However, if one extends the term neural activity to also encompass other more indirect brain signal measurements, such as functional magnetic resonance imaging (fMRI), one can find many attempts at decoding full image stimuli in recent years. For example, [31] used pretrained image embeddings to perform real-time decoding of magnetoencephalography (MEG) signals, while others [32, 33, 34, 35] focused on fMRI data and employed recent advances in generative artificial intelligence (generative AI), such as diffusion models [36, 37]. Most of the works in this area aimed at reconstructing the semantic information in the visual stimuli, such as the categories of objects, but not at capturing low-level features of the images. One reason is that these brain signal modalities do not provide the necessary temporal and spatial resolution for decoding such fine image structures, which also makes them unsuitable for studying the tuning properties of individual neurons or for fine-grained evaluations of stimulation protocols of visual prostheses.

Returning to neural responses, a series of works investigated image reconstruction from the activity of retinal ganglion cells [38, 39, 40], which form one of the first processing stages of the visual system. At this early stage of processing, even linear projections were able to reconstruct natural images from neural responses [38]. The use of nonlinear techniques, such as deep neural networks, further refined the structures in the reconstructed visual stimuli and also led to the ability to reconstruct dynamic videos [40].

One of the first studies investigating decoding from the visual cortex leveraged known retinotopy[2] that allowed them to reconstruct simple visual stimuli and mental imagery [41]. Later, using generative adversarial networks [42] and other deep learning approaches, a series of works [43, 44, 45, 46, 47, 48] showed promising results in decoding more complex stimuli from higher-order areas of the visual system, such as V1 and the inferior temporal cortex.

For example, [47] incorporated known biological properties of neurons in the visual system into their brain-inspired architecture to successfully reconstruct images from sequences of individual spikes, so-called spike trains. This work demonstrated the usefulness of injecting prior knowledge into machine learning models not only by outperforming other methods but also by validating neuroscience theories.

Similarly as for the fMRI data, [48] leveraged a pretrained diffusion model to decode neural responses predicted by an encoder model that was trained on data from the visual area V4. More specifically, their method, Energy Guided Diffusion, modified the so-called encoder inversion approach, explained below, by guiding the iterative inversion process using a frozen generative image model pretrained on a large image dataset. The diffusion model enabled sharper reconstructions but, in some cases, did not accurately capture the structure and layout of the ground-truth images. As the authors also pointed out, the reconstructions by this method are constrained by the manifold of the pretrained diffusion model.

Overall, given new V1 recordings, there is still no simple, data-efficient method capable of reliably providing structurally accurate and detailed reconstructions. For example, [47] captures contrast but lacks fidelity, whereas [48] produces detailed but sometimes structurally inaccurate reconstructions and is constrained to predicted neural responses. Additionally, the encoder inversion approach [45], when used alone without modifications by [48], relies on a relatively computationally expensive process that fails to fully capture natural-looking stimuli. Another drawback of this class of encoder inversion methods is that multiple different images can invoke the same neural responses, thereby reducing the decoding reliability. This thesis aims to address some of these limitations by providing an extensive comparison of relatively simple-to-implement

[2]Retinotopy refers to the arrangement of neurons corresponding to the spatial layout of the retina.

approaches. Furthermore, previous work has not explored the applicability of synthetic data in this specific setting, which is also the focus of this thesis.

1.3.1 Encoder inversion

Just as a pretrained encoder can be used to synthesize MEIs, it can also serve as a powerful tool for decoding neural responses into full image stimuli by inverting its mapping from images to neural activity. More specifically, rather than maximizing the predicted response for a single target neuron, as in MEI generation, this method optimizes the pixel values of the decoded image to minimize the difference between all the predicted and target neural responses [45]. As can be seen, one of the motivations for this method is to reconstruct images that are neurally equivalent to the original visual stimuli and are sometimes called *metamers*.

The specific setup of [45] involved a 3-layer CNN with a linear projection readout serving as an encoding model. During reconstructions, they initialized an image with zero-valued pixels and iteratively optimized it through gradient descent to minimize the mean squared error (4.1) between the predicted and recorded neural responses. To avoid high-frequency noise in the reconstructions, they applied Gaussian blur with a standard deviation of 2.5 px to the gradient at each iteration and ran the optimization for 1,000 steps.

An illustration of this method, which will be referred to as *encoder inversion* in the following chapters, is shown in Figure 1.3.

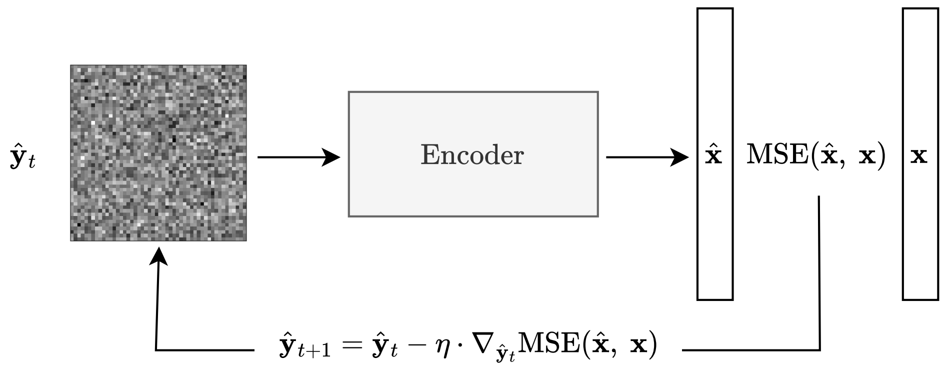

$\hat{\mathbf{y}}_t$
Encoder
$\hat{\mathbf{x}}$
MSE($\hat{\mathbf{x}}$, $\mathbf{x}$)
$\mathbf{x}$
$\hat{\mathbf{y}}_{t+1} = \hat{\mathbf{y}}_t - \eta \cdot \nabla_{\hat{\mathbf{y}}_t}\mathrm{MSE}(\hat{\mathbf{x}}, \mathbf{x})$

Figure 1.3 Encoder inversion. $\hat{\mathbf{y}}_t$ denotes the reconstructed image at timestep t, $\hat{\mathbf{x}}, \mathbf{x}$ refer to predicted and target neural responses, respectively, η is the step size, and MSE stands for mean squared error. Gaussian blur of the image gradient is omitted for clarity.

1.4 Applications

Relatively recently, [49] showed a successful modulation of neural activity in the V1 and V4 regions of the visual cortex in monkeys through invasive brain-computer interfaces (BCIs). Specifically, they encoded a range of visual patterns into a targeted stimulation of neurons, leading to the immediate recognition of various shapes and letters by the monkeys. In another study, a human patient with complete blindness was implanted with a stimulation device for 6 months. During this time, targeted stimulation provided by neural encoding enabled the patient to identify the boundaries of objects as well as some written letters [50].

Various invasive BCIs have also been developed to control robotic arms by decoding the neural activity of the primary motor cortex [51]. By decoding activity in this area, monkeys were able to control cursor movement in real time without any preliminary training [52]. Remarkable results were also achieved in a human patient with paralysis of the upper body who successfully drank coffee using a robotic arm controlled by an invasive BCI [53]. Others used invasive BCI technology and neural decoding to synthesize correct artificial speech when subjects only silently imitated sentences [54]. Instead of synthesizing speech, [55] decoded neural activity from the patient's motor cortex into text, giving their mind a typing speed of 90 characters per minute with around 94% accuracy.

In addition to decoding from the motor cortex, leveraging decoding techniques from visual and other sensory cortices holds promise for restoring various senses, including vision and tactile perception. Of particular interest for this thesis is the potential application in visual prostheses aiming at restoring vision. Currently, these devices rely on subjective human reports or animal behavioral responses for the experimental validation and refinement of stimulation protocols. The development of a robust decoding method could improve this process by reconstructing visual percepts directly, thereby enabling more precise testing and refinement of visual prosthesis stimulation strategies.

As can be seen, modeling of neural activity has many engineering applications, especially in BCIs. However, other applications can also be found in science, where neural modeling can help to understand how certain brain regions relate to the external world. Specifically, researchers have been developing hypothesis-based decoders to test specific structures and functional properties of neural coding. In addition, decoders can also be applied to artificially generated neural activity, testing the validity of in silico models of the brain.

Chapter 2

Generative adversarial networks

Generative adversarial networks (GANs), one of the decoding methods explored in this thesis, belong to major advances in machine learning, revolutionizing the generation of realistic and high-fidelity synthetic images. Proposed by [42] in 2014, GANs have since attracted the attention of many researchers in the field and have sparked a series of developments and extensions to data modalities other than images [56, 57, 58, 59].

More broadly, GANs are a class of artificial neural networks comprising two key components: a generator and a discriminator. The generator aims to generate synthetic (fake) data samples that are indistinguishable from the reference (real) ones, while the discriminator tries to differentiate between the two. This setup leads to a dynamic adversarial process in which the generator and the discriminator engage in a two-player minimax game.

2.1 Generator

The generator, denoted as G, is a neural network that takes a random noise or a latent space vector $\mathbf{z}$ as input and produces synthetic data samples $\hat{\mathbf{x}} = \mathrm{G}(\mathbf{z})$ as output. It aims to generate data samples that closely resemble the real data distribution. Mathematically, the generator is trained to minimize the following value function:

$$\min_{\mathrm{G}} \max_{\mathrm{D}} V\big(\mathrm{D}, \mathrm{G}\big) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}\big[\log \mathrm{D}\big(\mathbf{x}\big)\big] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\big[\log\big(1 - \mathrm{D}\big(\mathrm{G}(\mathbf{z})\big)\big)\big] \tag{2.1}$$

where $p_{\text{data}}(\mathbf{x})$ is the distribution of reference data samples, $p_{\mathbf{z}}(\mathbf{z})$ is the distribution of latent space vectors, and D is the discriminator.

Extending the original GAN formulation, GANs have also been used as conditional models in which both the discriminator and the generator are conditioned on some extra information. This class of models, called conditional GANs, was successfully adopted to generate examples conditioned on class labels [57], text [60], images [61], and many other types of information.

2.2 Discriminator

The discriminator, denoted as D, is another neural network that takes data samples as input and produces a score that indicates the probability that the sample is real. Its goal in the GAN setup is to distinguish between reference and synthetic data samples. More specifically, it is trained to maximize the inner part of 2.1, i.e.

$$\max_{\mathrm{D}} V\big(\mathrm{D}, \mathrm{G}\big) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}\big[\log \mathrm{D}\big(\mathbf{x}\big)\big] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\big[\log\big(1 - \mathrm{D}\big(\mathrm{G}(\mathbf{z})\big)\big)\big] \tag{2.2}$$

The discriminator's loss is minimized when it correctly classifies real data as real and synthetic data as synthetic.

2.3 Training

Training a GAN involves iteratively updating the parameters of the generator and discriminator networks as they try to compete with each other. The process can be summarized as follows:

1. Initialize the parameters of the generator G and the discriminator D.
2. Sample a batch of reference data samples from the training dataset and generate corresponding synthetic samples using the current generator.
3. Update the discriminator by maximizing its objective function with respect to its parameters while keeping the generator parameters fixed.
4. Update the generator by minimizing the discriminator's objective function with respect to its parameters while keeping the discriminator parameters fixed.
5. Repeat steps 2-4 until convergence.

The generator usually performs poorly during the early learning phase, and the generated samples differ substantially from the reference data. Therefore, the task of the discriminator is relatively easy as it can reject these early generated examples with high confidence. This results in one of the well-known problems related to the difficulties in training GANs, where even minor changes in the selected hyperparameters, initializations, or other settings can completely destabilize the training process, leading to very poor performance [62]. For an extensive review of the challenges and remedies proposed in the literature, we refer the reader to an in-depth recent review of GANs by Gui et al. [63].

2.4 Applications

The versatility and effectiveness of GANs have led to a wide range of applications in various domains. However, their most successful applications are in image processing and computer vision, which were also the first [42]. More specifically, in image generation, GANs have been shown to be able to generate highly realistic images of human faces [64, 65], natural scenes [66, 42], and artistic designs [67]. Since the generator can synthesize images from random noise or latent space vectors, which can lead to diverse outputs, GANs have also been used for synthetic data generation [68]. That is, to augment training datasets by generating additional synthetic samples on which other machine learning approaches can be trained. Further applications can be found in style transfer [69, 70], super-resolution [71, 72], as well as drug discovery, electronic health record generation, and molecular design [59, 73, 74], where the adversarial framework of GANs is extended to different data modalities.

Considering the specific task of neural activity decoding, [75] pretrained a GAN on a large image dataset of natural scenes and then separately learned how to map fMRI signals to the latent space of the generator. As can be seen in the article, the imbalance of pretraining and separate training on a small dataset with neural signals resulted in structurally inconsistent but semantically rich reconstructions. In contrast, this thesis uses the formulation of conditional GANs to learn the response-stimulus mapping directly with the goal of higher consistency.

Chapter 3
Synthetic data

Given this thesis' aim to overcome the limited availability of paired neural and image data, this chapter introduces the notion of synthetic data. This type of data refers to artificially generated information that is not obtained by direct measurements but is instead created algorithmically through computational means. In this thesis and in ML in general, synthetic data is used to augment or completely replace real datasets for testing or training data-driven methods.

One of the primary motivations for using synthetic data is the scarcity of data obtained from biological or other real-world experiments and concerns related to privacy, fairness, and regulations. Given the nature of ML approaches and the increasing availability of computing power for large-scale training, issues surrounding data have become even more pronounced and important. In this regard, synthetic data poses a promising direction, as it is generally easier and less expensive to generate, is inexhaustible, and does not directly come from real-world entities, thereby also alleviating some privacy concerns. Notably, the authors of [76] view artificially generated data as a key enabler for the next generation of deep learning models that have a deeper understanding of the world and can continually learn in many different modalities.

Nevertheless, in addition to the challenges mentioned in the section below, [77] pointed out that a major problem with synthetic data, especially in medicine and healthcare, is currently ensuring that the generated data meets all the required quality standards and minimizes potential safety risks. Further challenges mentioned by the authors include the inability of data generation methods to account for corner cases in the real data and potential inherent biases in the generation methods, which in turn translate to the synthetic data and the downstream applications.

3.1 Approaches

Several techniques exist for generating synthetic data, each suitable for different types of data and use cases. Some of the most popular methods include simulation-based generation, generative AI, and (input space) data augmentation.

3.1.1 Simulation-based generation

This method assumes an accurate model of some real-world process, which it then simulates to create new data. It is widely used in domains such as robotics, autonomous vehicles, and medical imaging, where real data can be scarce or expensive to obtain. In robotics, for example, one of the best-known simulation engines is called MuJoCo (multi-joint dynamics with contact), developed by Todorov [78]. It is a physics engine well suited for developing contact-rich behaviors and has become an industry standard for robotics research. In the domain of robotics and deep reinforcement learning, synthetic environments such as MuJoCo are often used even more frequently than the real ones.

The challenge of simulation-based techniques is obtaining the computational model or engine that achieves high levels of realism and accuracy [76]. It can be costly and the complexity of real-world processes can pose significant barriers. In the literature, the difference between real and simulated is often referred to as a *sim-to-real* domain gap [79, 80].

3.1.2 Generative AI

Another notable data-generation technique is generative AI, which has attracted a great deal of attention in recent years. Methods such as GANs, variational autoencoders [25], diffusion models [36], and transformers [81] have been effective in generating realistic images [65, 66, 82], videos [82], and language data [83]. These techniques have been applied to medical imaging tasks [84, 85], pose estimation, fraud detection [86], and many other areas [68, 87].

While not strictly classified as a generative AI technique, this thesis' approach to data generation also uses ML models to create new data. As detailed in 5.1.1, the idea is to learn a neural activity encoding model that can then estimate the ground-truth neural responses for new images, which are much easier to obtain than neural signals. Since most ML tasks can be cast into function approximation with inputs and outputs, this method is widely applicable, as it relies on learning the output-to-input mapping to subsequently generate synthetic data for the original problem.

One of the challenges with generative AI and ML-based approaches is, however, the controllability of the generated data. Unlike physical simulators, DL techniques are black-box models of the data, and precise conditioning remains difficult. Furthermore, it is not clear how much these models can extend the base datasets on which they were initially trained.

3.1.3 Input space data augmentation

Last mentioned here, input space data augmentation is a class of techniques that directly modify the input data to create additional data points. The main advantage of these methods is that they are more intuitive and can be designed specifically to generate the desired variations in new data. Common manipulation techniques include rotating, scaling, cropping, or changing the color properties of images.

Input space data augmentation methods have a relatively long history and have even been incorporated into popular deep learning libraries such as PyTorch [88] as plug-and-play modules. For a detailed review of the various input transformations as well as applications, we refer the reader to [89].

Chapter 4
Metrics

In the domain of image generation and image restoration, evaluating the quality of images is crucial for assessing and comparing the proposed methods. Various metrics have been developed to quantitatively measure different aspects of image quality, and some commonly used ones will be described in this chapter. It should be noted that the term *metrics* is used in a broad sense in this work, and some of the evaluation functions introduced do not strictly adhere to the rigorous mathematical definition of a metric.

4.1 Mean squared error

Mean squared error (MSE) is one of the simplest and most widely used metrics to evaluate image quality. It measures the average squared difference between the pixels of the generated image and the reference image. More formally,

$$\mathrm{MSE}(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \cdot \sum_{i=1}^{d} \big(\mathbf{x}_i - \mathbf{y}_i\big)^2, \tag{4.1}$$

where $d \in \mathbb{N}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ are the flattened reference and generated images, respectively.

Despite its simplicity, MSE has limitations, especially in evaluating perceptual quality, as it does not take into account human perception factors (see 4.3 for an example).

4.2 Structural similarity index measure

Unlike MSE, the structural similarity index measure (SSIM) considers multiple factors of images with more emphasis placed on perceptual quality. More specifically, SSIM [90] computes the similarity between two images $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{h,w}$, $h, w \in \mathbb{N}$ based on three factors: luminance similarity, contrast similarity, and structural similarity. It ranges between -1 and 1, where 1 indicates perfect similarity, and with a particular weighting of the three factors as proposed by the authors, the SSIM index can be expressed as:

$$\mathrm{SSIM}\big(\mathbf{x}, \mathbf{y}\big) = \frac{(2\mu_x + \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{4.2}$$

where μ_x and μ_y are the pixel sample means of $\mathbf{x}$ and $\mathbf{y}$, σ_x, σ_y their standard deviations, and σ_{xy} the covariance between the images. c_1 and c_2 are constants to avoid division by zero and, for images with pixel values between 0 and 1, they are typically set to $c_1 = 0.0001$ and $c_2 = 0.0009$.

Since statistical features of images usually spatially vary, the original authors recommended applying the measure above regionally and combining the individual results afterward. Furthermore, they proposed using 11 × 11 circular-symmetric Gaussian weighting $\mathbf{w} \in \mathbb{R}^{11,11}$, $\sum_{i=1}^{11} \sum_{j=1}^{11} w_{ij} = 1$ with a standard deviation of 1.5 to calculate the local statistics. The specific calculation can be expressed as:

$$\mu_x^{(k)} = \sum_{i=1}^{11} \sum_{j=1}^{11} w_{ij} \mathbf{x}_{ij}^{(k)} \tag{4.3}$$

$$\sigma_x^{(k)} = \left(\sum_{i=1}^{11} \sum_{j=1}^{11} w_{ij} \big(\mathbf{x}_{ij}^{(k)} - \mu_x^{(k)}\big)^2 \right)^{\frac{1}{2}} \tag{4.4}$$

$$\sigma_{xy}^{(k)} = \sum_{i=1}^{11} \sum_{j=1}^{11} w_{ij} \big(\mathbf{x}_{ij}^{(k)} - \mu_x^{(k)}\big)\big(\mathbf{y}_{ij}^{(k)} - \mu_y^{(k)}\big), \tag{4.5}$$

where the superscript $^{(k)}$ indexes local windows. The final (mean) SSIM index is obtained by averaging $\mathrm{SSIM}\big(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\big)$ over all K local windows.

4.3 Perceptual loss

Rather than measuring the differences between the pixels of two images directly, perceptual loss functions consider the differences between high-level image feature representations extracted from pretrained neural networks. A motivating example behind these perceptual loss functions was given in [91]: Consider two identical images, one of which is offset from the other by one pixel. Traditional per-pixel losses, such as MSE, might report a large error, even though the images are almost indistinguishable to the human eye.

The procedure for calculating the perceptual loss (PL) typically involves the following steps. Both the reference image and the generated image are fed through a pretrained neural network, and as the images progress through the network, the intermediate feature maps of both images are extracted and compared using standard pixel-level measures such as MSE. This approach allows for the capture of low-level differences when comparing image representations at earlier layers, while comparisons at later layers capture higher-level perceptual dissimilarities.

More formally, given a function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $d \in \mathbb{N}$ for comparing $k \in \mathbb{N}$ intermediate feature maps $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)})$ and $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)})$ corresponding to two images $\mathbf{x} \in \mathbb{R}^{h,w}$ and $\mathbf{y} \in \mathbb{R}^{h,w}$, the perceptual loss can be formulated as follows:

$$\mathrm{PL}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{k} w_i \cdot f(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}). \tag{4.6}$$

Above, $\{w_i \in \mathbb{R}\}_{i=1}^{k}$ are arbitrary weights for individual feature maps. In this thesis for instance, f is the L1 loss function, all w_i are set to $(4 \cdot k)^{-1}$, and the features are extracted from a 16-layer VGG CNN [92] pretrained for classification on the ImageNet dataset [93].

An example application of perceptual loss can be found in [91], where the authors used the same 16-layer VGG CNN as in this thesis and used MSE as the feature reconstruction loss function. After training with perceptual loss on a style transfer task, the authors observed that the semantic content of the generated images was significantly improved. This enhancement was interpreted as an inherent property of the pretrained network, which has learned a feature-extraction mechanism selective for recognizing features relevant to people and animals. Altogether, perceptual losses have shown great promise over the last few years as more human-aligned comparison functions for images.

Chapter 5
Datasets

In this chapter, we describe the two datasets used to train and evaluate the models implemented in this thesis. For the first dataset (5.1), we also introduce an additional synthetic set of samples and describe its generation procedure.

When combining multiple datasets for training or evaluation, we mix the data points sampled independently from each of them within individual mini-batches. Subsequently, we report the percentage of the mini-batch that the data points from a particular dataset comprise (i.e. 25% of synthetic data would correspond to a mini-batch of 5 synthetic and 15 original data points). When the sizes of the combined datasets differ, we truncate the datasets to the size of the smallest one. The motivation behind the truncation is to balance the number of samples from each dataset in the combined one such that the trained decoder performs well on all of them instead of overfocusing on a single one that has more data samples available.

Unless stated otherwise, the image stimuli are normalized to have zero mean and unit variance, and the neural responses are rescaled by the inverse of the standard deviation. More specifically, given the substantial variability in activity levels among different neurons, responses are divided by the standard deviations of individual neuron activity. The neuron coordinates are rescaled to lie between -1 and 1.

5.1 SENSORIUM 2022 dataset

Introduced in 2022, the SENSORIUM competition [19] aimed to benchmark artificial neural networks in predicting neural activity in mouse V1. The organizers of this competition provided participants with a large-scale data corpus of recordings that contained responses from more than 28,000 neurons from seven mice. Data from five of these mice were intended as *training recordings* and we will use them for our decoding task.

The image stimuli were sampled from the ImageNet dataset [93], converted to grayscale, and downsampled to a width and height of 64 and 36 px, respectively. Upon presentation of the stimuli to awake and head-fixed mice, the authors recorded the neural responses of excitatory neurons in layer 2/3 of the mice right V1 using calcium imaging. More specifically, neural activity was accumulated during the 50 and 550 ms time window after each stimulus onset using a boxcar window. To reduce noise during evaluation, neural activity in the test datasets was obtained by averaging over 10 repetitions of stimulus presentation.

During our experiments, we observed that neither the inverted encoder nor our decoding models were able to capture the surroundings of the central 22×36 px image patch. We attribute this to the extensive overlap of receptive fields of the recorded neurons, which results in limited coverage of the visual field. Hence, we consider only this image window in training and evaluation.

Mouse ID	Number of neurons	Number of samples		
		Training	Validation	Testing
1	8,372	4,473	523	100
2	7,344	4,498	500	100
3	7,334	4,466	496	100
4	8,107	4,477	496	100
5	8,098	4,490	499	100

Table 5.1 Details of the mice datasets from SENSORIUM 2022.

The number of neurons and the sizes of the datasets from the five selected mice are given in Table 5.1. Note that our numbering of mice aligns with the original done by the authors after adding two to our mouse IDs. For instance, their mouse ID 3 corresponds to our mouse ID 1. In chapter 7, we will use a shorthand notation in which **M-1** refers to the dataset of the mouse with an ID of 1, and **M-All** corresponds to datasets from all mice combined.

In addition to pairs of responses and image stimuli, which are key to our work, the original data samples contained other variables such as, for example, mouse eye movements and running speeds. To increase generality and reduce the complexity of our developed models, we limit ourselves to incorporating only the neuron x and y coordinates as additional information.

5.1.1 Synthetic dataset from CNN-based encoding model

We employed the CNN-based encoding model introduced in section 1.2 to generate additional synthetic samples for the SENSORIUM 2022 dataset. The primary motivation was to train decoding models with improved generalization capability compared to methods trained solely on the relatively small biological dataset.

The specific procedure for generating this synthetic data corpus was as follows. First, we sampled images from ImageNet, converted them to grayscale, and cropped patches of size 36 $\times$ 64 px. We then presented these previously unseen images to the CNN-based encoder to obtain the predicted neural responses. The image stimuli, predicted responses, and the x and y coordinates of the neurons then formed the data samples in our synthetic dataset.

By applying the above steps and acquiring predicted responses for each mouse, we generated five datasets, each containing 50,000 data points. Since the purpose of the synthetic data lies only in training of the decoding models, we did not leave any of the data points for validation or testing.

Analogously to the base SENSORIUM 2022 dataset, we will use **S-1** and **S-All** to refer to mouse 1 and all mice synthetic datasets, respectively.

Additionally, as mentioned in the previous section 5.1, we used only the central image patch of size 22 $\times$ 36 px to train the decoding methods due to the limited coverage of the visual field.

5.2 Dataset from biologically realistic encoding model

The second base dataset was generated using the spiking model of a cat V1 introduced in [94]. This biologically realistic model represents cortical layers 4 and 2/3, corresponding to a 5×5 mm patch of V1. It contains a total of 108,150 excitatory and inhibitory neurons that are modeled as single-compartment exponential integrate-and-fire units [95]. For further details and validation results of this in silico model, we refer the reader to the original work [94].

When generating the samples, we first presented this encoding model with an image stimulus and then measured the evoked mean firing rates of neurons over the following 560 ms time window. For this thesis, we used only responses from 46,875 neurons in layer 2/3 and their corresponding x and y coordinates, which this spiking model provides.

Similarly to the previous dataset (5.1), the image stimuli were sampled from ImageNet, converted to grayscale, downsampled, and then cropped to a size of 50×50 px. Importantly, when we subsequently used the data to train and evaluate the decoding models, we only considered the central 20×20 px patch of the images. The reason is that the neurons in the encoding model have overlapping receptive fields that do not cover the whole visual field; therefore, their induced responses contain information only about the central patch of the presented images.

Finally, we split the generated 50,250 data points into training, validation, and testing datasets of size 45,000, 5,000, and 250, respectively. To reduce the impact of noise on the evaluation, the image stimuli of the test set were presented 100 times and the corresponding neural responses were averaged to obtain the final neural activity. In the chapter where we present the experiments and results (7), we will refer to this dataset as the **C** dataset.

Chapter 6
Methods

In this chapter, we will describe our approach to tackling the problem of decoding visual stimuli from neural activity. We will start by introducing the building blocks and training objective of our CNN-based decoding method and then provide details of how we use GANs for this task. Finally, we will explain our novel auxiliary training signal called encoder matching.

All our implementation was done in the Python programming language which is widely used in the machine learning community. Of similar prevalence in this field is our chosen deep learning library called PyTorch [88].

6.1 CNN

Our first approach to decoding is inspired by neural network encoders that have been studied and improved over the past few years. Analogously to the encoder architecture divided into a *core* and a *readout*, we build our decoder from a *core* and a *readin*. While the task of the core is to learn a single general mapping from a latent representation onto images, the readins are trained for each recording dataset separately and act as embedding functions from the neural activity into the shared latent space of the core. The purpose is to reuse as much information and a learning signal across different animal recordings even when their number of neurons and response-stimulus mapping differ.

In the following sections, we will introduce our implementation of the core, detail three readin variants, and describe the training procedure. We will refer to this decoding method as CNN-FC (6.1.2), CNN-Conv (6.1.3), or CNN-MEI (6.1.4) depending on the particular readin used.

The hyperparameters of these building blocks that we used in our experiments (7) are provided in Table 6.1. The specific values were found by manual search.

6.1.1 Core

We model the core as a CNN with five layers of transposed convolution [96], batch normalization, ReLU activation function, and dropout. With the MEI readin described below, we use the standard convolutional layer instead of its transposed version. Lastly, we apply a convolutional layer with a single filter to turn the feature map into an image with one channel.

6.1.2 Fully connected readin

The fully connected readin (FC readin) is composed of a linear layer, batch normalization, Leaky ReLU activation function, and dropout. It transforms the vector of neural responses into a latent representation that is then unflattened into a three-dimensional tensor of size [number of channels × height × width] suitable for the core to operate on.

As can be seen, this readin takes as input only the neural activity, not the coordinates of the neurons. It serves as a simple-to-implement baseline upon which we can build with additional structure and input information.

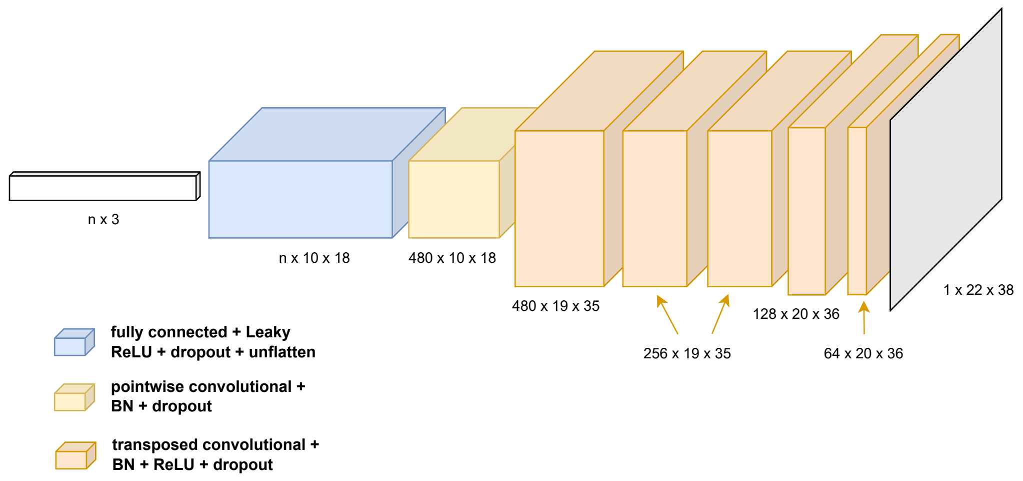
6.1.3 Convolutional readin

The convolutional readin (Conv readin) has two building blocks. The first is a *grid network* that transforms individual neuron activity together with its x and y coordinates into a latent 2D map. It is implemented as a fully connected three-layer neural network with the Leaky ReLU activation function and dropout, followed by an unflattening operation. By applying the grid network for each neuron independently, we get a 3D tensor of size [number of neurons × height × width], where the last two dimensions correspond to the sizes of the neuron maps.

During our experiments, we observed that a nonlinear transformation of the neural responses speed up training of this readin. Specifically, we preprocess the grid network's input by applying a logarithm with base 10 to the neuron activity.

The second part of the convolutional readin is a pointwise convolutional layer. It takes the 3D tensor from the grid network and applies a set of learned 1×1 kernels to produce a feature map for the core. The purpose of this second part is to linearly combine the individual neuron maps and lower the channel dimension using a small number of parameters.

The architecture of this readin is depicted in Figure 6.1. Note that when the dimensions of the output of the model do not match the target sizes, we crop only the middle patch with the right size from the predicted image.


n x 3
n x 10 x 18
480 x 10 x 18
480 x 19 x 35
256 x 19 x 35
128 x 20 x 36
64 x 20 x 36
1 x 22 x 38
fully connected + Leaky ReLU + dropout + unflatten
pointwise convolutional + BN + dropout
transposed convolutional + BN + ReLU + dropout

■ **Figure 6.1** CNN-Conv decoder with sizes of intermediate representations corresponding to the SENSORIUM 2022 dataset. n denotes the number of neurons, and the left-most block represents the neural responses concatenated with the coordinates.

6.1.4 MEI readin

To combat the relatively small amount of available data that may not contain enough information for models to learn from, we introduce an additional prior in the form of MEIs. Since MEIs

	Mouse V1 datasets		Cat V1 dataset
Core			
Channels	[480, 256, 256, 128, 64]	Channels	[480, 256, 256, 128, 64]
Kernel sizes	[7, 5, 4, 3, 3]	Kernel sizes	[7, 5, 5, 4, 3]
Strides	[2, 1, 1, 1, 1]	Strides	[2, 1, 1, 1, 1]
Paddings	[3, 2, 2, 1, 1]	Paddings	[3, 2, 2, 1, 1]
Dropout prob.	0.35	Dropout prob.	0.35
FC readin			
Latent dimension	432	Latent dimension	512
Unflattened size	[3, 9, 16]	Unflattened size	[8, 8, 8]
Dropout prob.	0.15	Dropout prob.	0.15
Conv readin			
Grid net layer sizes	[32, 86, 180]	Grid net layer sizes	[64, 128, 100]
Neuron map shape	10 × 18	Neuron map shape	10 × 10
Channels	480	Channels	480
Dropout prob.	0.1	Dropout prob.	0.15
MEI readin			
Grid net layer sizes	[32, 128, 792]	Grid net layer sizes	[64, 128, 400]
Neuron map shape	22 × 36	Neuron map shape	20 × 20
Channels	480	Channels	480
Dropout prob.	0.1	Dropout prob.	0.15

■ **Table 6.1** Hyperparameters used for the CNN decoder.

encode the receptive fields of neurons and the shapes that most stimulate them, one can easily imagine an effective decoding algorithm that overlays the MEIs of all neurons, weighted by their corresponding neural responses, onto each other. This approach intuitively suggests that MEIs could provide valuable guidance to the decoding models, informing them about what features to include and where to place them in the reconstructed image.

In particular, we implement this readin by precomputing and concatenating the MEIs of all neurons, arriving at a 3D tensor of shape [number of neurons × height × width]. Note that the generation of MEIs needs to be done only once, thus incurring only a one-time cost. These MEIs then form the initial feature maps of the MEI readin, which is an extension of the convolutional readin described above.

More specifically, we apply the grid network as previously (6.1.3) but pointwise multiple its outputted 3D tensor onto the precomputed MEIs as a form of contextualization. The second part of this readin, which contains pointwise convolution, batch normalization, and dropout, is the same as in the convolutional readin.

6.1.5 Training

Since our aim is to closely capture the structure of the restored images while taking into account human perception factors, we modified the standard SSIM to act as the training loss. That is, let $d \in \mathbb{N}$ and $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^d$ be the flattened target and restored images, respectively, and $\mathrm{SSIM}(\cdot, \cdot)$ the structural similarity index measure described in 4.2, then the training reconstruction loss is given by:

$$L_{\text{REC}}(\mathbf{y}, \hat{\mathbf{y}}) = -\log\left(\frac{\text{SSIM}(\mathbf{y}, \hat{\mathbf{y}}) + 1}{2} + \epsilon\right), \tag{6.1}$$

where ϵ is introduced for numerical stability and set to 10^{-6}. Based on the first term inside the logarithm in 6.1, we also define the SSIM loss (SSIML) for evaluation purposes as follows:

$$\text{SSIML}(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{\text{SSIM}(\mathbf{y}, \hat{\mathbf{y}}) + 1}{2}. \tag{6.2}$$

Furthermore, in the case of the convolutional readin (6.1.3), we add an L1 regularization loss to combat overfitting of the grid network. More specifically, if $\lambda_{\text{REG}} \in \mathbb{R}$ is the regularization factor and $\mathbf{z} \in \mathbb{R}^{n,h,w},\ n, h, w \in \mathbb{N}$ are the neuron maps produced by the grid network, then the regularization loss becomes:

$$L_{\text{REG}}(\mathbf{z}) = \lambda_{\text{REG}} \cdot \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{h} \sum_{j=1}^{w} |\mathbf{z}_{kij}|. \tag{6.3}$$

Training proceeds using the AdamW optimizer [97] with a learning rate 3^{-4}, weight decay 3^{-2}, and other hyperparameters set as the optimizer's default in PyTorch [88]. Similarly to early stopping [98], we pick the best model found during training based on the sum of the perceptual loss (4.3) and the SSIM loss measured on the validation dataset.

6.2 Generative adversarial network

To address the problem of data scarcity, we explore different strategies to incorporate additional prior information into our models, as discussed in the MEI readin above. Another approach we hypothesize might help is to leverage additional learning signals, such as the adversarial loss commonly found in GANs. Our reasoning stems from the fact that, while the limited amount of data might provide enough coverage of the space of neural responses, it might not suffice for pushing the models onto a manifold of natural-looking images that reside in much higher-dimensional space.

It is important to note that, while we draw inspiration from the literature on GANs, our formulation deviates from the standard implementation. We adapt ideas from GANs that we find effective for our specific task. For instance, we inject label noise into discriminator training, remove logarithms from the value function formulation (2.1), and generate synthetic images from neural responses rather than generating them from random noise or latent space vectors not corresponding to the data. Furthermore, we consider the adversarial loss only as an auxiliary task to reconstructing the image stimuli, weighing it less in the combined loss of the generator.

In the following sections, we provide a detailed description of the training objectives for both the discriminator and the generator, and describe their architecture. Specific hyperparameters can be found in Table 6.2. Identically to the CNN decoder implementation, we use the AdamW optimizer with a learning rate 3^{-4} and weight decay 3^{-2}, and perform a model selection based on the same criterion. Additionally, during training, we clip the gradients of the losses w.r.t. the parameters to a range $[-1, 1]$. This adjustment helped us mitigate early training instabilities and ensure smoother convergence.

6.2.1 Discriminator

The discriminator D is implemented as a CNN with five layers of convolution, batch normalization, ReLU activation function, and dropout. The output of the last layer is flattened into a one-dimensional vector and transformed by a linear layer followed by the sigmoid activation function. The result is a predicted probability that the given input is a reference image from the dataset (i.e. not generated by the generator).

Let $\lambda_{\mathrm{R}}, \lambda_{\mathrm{G}} \in \mathbb{R}_+$ be the loss weighting factors, $\epsilon_{\mathrm{R}} \in [0, \xi_{\mathrm{R}} \in \mathbb{R}_+], \epsilon_{\mathrm{G}} \in [0, \xi_{\mathrm{G}} \in \mathbb{R}_+]$ be the target noising components, and $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote the reference and generator image, respectively. Then, our implementation of the disciminator loss L_{D} is as follows:

$$L_{\mathrm{D}}\big(\mathbf{y}, \hat{\mathbf{y}}\big) = \lambda_{\mathrm{R}} \cdot \big(\mathrm{D}(\mathbf{y}) - 1 - \epsilon_{\mathrm{R}}\big)^2 + \lambda_{\mathrm{G}} \cdot \big(\mathrm{D}(\hat{\mathbf{y}}) - \epsilon_{\mathrm{G}}\big)^2. \tag{6.4}$$

The target noising component ϵ_{R} for the reference image part is sampled from a uniform distribution between 0 and ξ_{R}, while the noising component ϵ_{G} comes from a uniform distribution between 0 and ξ_{G}. We found that by introducing noise into the discriminator training, we were able to better balance the generator and discriminator, and thereby stabilize the training. We note that this is reminiscent of *one-sided label smoothing* as introduced in [62], where the discriminator's positive targets are smoothed from 1 to 0.9, making its task harder.

6.2.2 Generator

The generator follows the exact architecture of the CNN-based decoding approach, including the separation into a core and a readin. The only difference comes in the loss.

Specifically, the generator training loss L_{G} consists of two parts. The first is an image reconstruction loss L_{REC} derived from the standard SSIM as described for the CNN decoder (6.1). The second part L_{D} comes from the discriminator being able to discern the generated image $\hat{\mathbf{y}}$ from the reference image $\mathbf{y}$. Therefore, the final loss is:

$$L_{\mathrm{G}}\big(\mathbf{y}, \hat{\mathbf{y}}\big) = \lambda_{\mathrm{REC}} \cdot L_{\mathrm{REC}}\big(\mathbf{y}, \hat{\mathbf{y}}\big) + \lambda_{\mathrm{D}} \cdot L_{\mathrm{D}}\big(\hat{\mathbf{y}}\big), \tag{6.5}$$

where $\lambda_{\mathrm{REC}}, \lambda_{\mathrm{D}} \in \mathbb{R}_+$ are the weighting factors, $\mathbf{y}$ and $\hat{\mathbf{y}}$ are as in the previous section, and the loss from the discriminator is given by

$$L_{\mathrm{D}}\big(\hat{\mathbf{y}}\big) = \big(\mathrm{D}(\hat{\mathbf{y}}) - 1\big)^2. \tag{6.6}$$

6.3 Encoder matching

Analogously to the previous section, we introduce another auxiliary training loss to address the overly weak constraints on the decoding models imposed by the scarcity of data. The name of this loss, encoder matching (EM), stems from the use of a pretrained CNN-based encoder. The reason for its use is to minimize the effects of noisy neural responses in the data and make the set of response-stimulus mappings learned by the decoder more densely sampled. More specifically, we penalize the decoder for differences between its image reconstructions from the neural responses in the data and its reconstructions from the neural responses predicted by the encoder. Intuitively, the encoder predicts neural responses with less heavy-tailed noise than observed in the data, resulting in a more compact and smooth manifold of neural responses seen by the decoder. We hypothesize that this might, in turn, lead to the decoder learning smoother response-stimulus mappings, which could be more robust to noisy measurements of neural responses in the data.

More formally, let $\mathrm{M}(\cdot)$ be the decoding model, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{q} \in \mathbb{R}^{n,2}$ denote the reference neural responses and neuron coordinates, respectively, and $\hat{\mathbf{x}} \in \mathbb{R}^n$ refer to the predicted responses from the reference stimulus by the encoder. Then, we formulate the EM loss L_{EM} as:

$$L_{\mathrm{EM}}\big(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{q}\big) = \lambda_{\mathrm{EM}} \cdot L_{\mathrm{REC}}\Big(\mathrm{M}\big(\mathbf{x}, \mathbf{q}\big), \mathrm{M}\big(\hat{\mathbf{x}}, \mathbf{q}\big)\Big), \tag{6.7}$$

where $\lambda_{\mathrm{EM}} \in \mathbb{R}$ is the multiplication factor for the EM loss, and L_{REC} is the same as in 6.1. This auxiliary loss is added in each training iteration of the decoder to its base loss. Unless stated otherwise, all of our experiments with EM use $\lambda_{\mathrm{EM}} = 1$.

	Mouse V1 datasets		Cat V1 dataset	
Training				
	λ_{REC}	0.9	λ_{REC}	0.9
	λ_{D}	0.1	λ_{D}	0.1
	λ_{R}	0.5	λ_{R}	0.5
	λ_{G}	0.5	λ_{G}	0.5
	ξ_{R}	0.05	ξ_{R}	0.05
	ξ_{G}	0.05	ξ_{G}	0.05
CNN (Discriminator)				
	Channels	[256, 256, 128, 64, 64]	Channels	[256, 256, 128, 64, 64]
	Kernel sizes	[7, 5, 3, 3, 3]	Kernel sizes	[7, 5, 3, 3, 3]
	Strides	[2, 1, 1, 1, 1]	Strides	[2, 1, 1, 1, 1]
	Paddings	[2, 2, 1, 1, 1]	Paddings	[2, 2, 1, 1, 1]
	Dropout prob.	0.3	Dropout prob.	0.3
Core (Generator)				
	Channels	[480, 256, 256, 128, 64]	Channels	[480, 256, 256, 128, 64]
	Kernel sizes	[7, 5, 5, 4, 3]	Kernel sizes	[7, 5, 5, 4, 3]
	Strides	[2, 1, 1, 1, 1]	Strides	[2, 1, 1, 1, 1]
	Paddings	[3, 2, 2, 1, 1]	Paddings	[3, 2, 1, 1, 1]
	Dropout prob.	0.35	Dropout prob.	0.35
FC readin				
	Latent dimension	432	Latent dimension	512
	Unflattened size	[3, 9, 16]	Unflattened size	[8, 8, 8]
	Dropout prob.	0.15	Dropout prob.	0.15
Conv readin				
	Grid net layer sizes	[32, 86, 180]	Grid net layer sizes	[64, 128, 64]
	Neuron map shape	10×18	Neuron map shape	8×8
	Channels	480	Channels	480
	Dropout prob.	0.1	Dropout prob.	0.15
MEI readin				
	Grid net layer sizes	[32, 128, 792]	Grid net layer sizes	[64, 128, 400]
	Neuron map shape	22×36	Neuron map shape	20×20
	Channels	480	Channels	480
	Dropout prob.	0.1	Dropout prob.	0.15

Table 6.2 Hyperparameters used for the GAN decoder.

Chapter 7
Experiments

Here, we describe our conducted experiments and discuss the obtained results. This chapter is divided according to the datasets introduced in chapter 5 and concludes with a section on transfer learning and a summary of key results.

Our goal for the experimental work was to come up with the best-performing combination of model design and training paradigm for decoding visual percepts. Additionally, we aimed to better understand the impact of synthetic data in this otherwise data-constrained setting.

As mentioned in the encoder readout section 1.2.2, we used the implementation of the CNN-based encoder model provided by [19] and generated MEIs using the code from the featurevis repository[1]. The inverted encoder, as explained in 1.3.1, is, to the best of our knowledge, the state-of-the-art decoding approach and served as a baseline for our methods. Its hyperparameters were selected from a grid search based on the same model selection criterion as used for the other decoding methods evaluated on the validation dataset.

For the quantitative evaluation of the reconstructed visual stimuli, we chose the mean squared error, the loss based on the structural similarity index measure[2], and the perceptual loss[3]. Our selection was based on the popularity of these evaluation measures with the exception of SSIML, which is, nonetheless, also very closely related to the widely used SSIM (see 6.2). We prefer to report SSIML instead of SSIM since its lower values indicate better results as for the remaining two measures. The reference images shown in the qualitative comparisons were sampled from the test datasets.

Lastly, for reporting purposes, we adopt the following naming convention for our methods: All names begin with [core]-[readin] where [core] is either "CNN" or "GAN" and [readin] is "FC", "Conv", or "MEI". When using the auxiliary encoder matching objective during training, we append "w/ EM" to the name. When necessary, we also add a specification of the training setup in round brackets. For example, "(87.5% S-All)" corresponds to training on a combined dataset of 87.5% **S-All** and 12.5% **M-All** data, and "(M-1)" refers to training only on **M-1**.

7.1 Biologically realistic encoding model dataset

For an initial evaluation of the decoding approaches, we leveraged the dataset generated from the in silico model of cat V1. It provided us with a less data-constrained setting compared to the SENSORIUM 2022 dataset, allowing us to inspect the performance in a more ideal setting.

[1]`https://github.com/ecobost/featurevis.git`
[2]Implementation from the TorchMetrics library: `https://lightning.ai/torchmetrics`
[3]Implementation based on `https://gist.github.com/alper111/8233cdb0414b4cb5853f2f730ab95a49`

For the encoder inversion hyperparameter search, we used only 1,280 samples from the validation dataset instead of all 4,500 due to the computational burden of the method. The best-performing hyperparameter values were as follows:

- 100 optimization steps (tested 100, 200, 500, and 1,000);
- learning rate (step size) of 10 (tested 10, 20, 50);
- standard deviation of 1.5 for the Gaussian blur of the image gradient (tested 1, 1.5, 2, 2.5).

Interestingly, only for this dataset, we found that the encoder inversion approach works better when we do not rescale the neural responses but initialize the bias parameters in the encoder's readout with the average neural responses.

From the methods introduced in the last chapter, we compared all combinations of the core and the readin, with and without the encoder matching auxiliary objective.

From the quantitative results reported in Table 7.1, we can observe that some of the CNN and GAN decoders outperform the baseline inverted encoder in terms of SSIML, but are generally worse measured by MSE and PL. The difference between GANs and plain CNNs appears to be more nuanced. Nevertheless, if we consider only the cases with the convolutional readin, GAN achieves better performance than the CNN and is one of the best overall.

Interestingly, the encoder matching objective clearly degrades the methods when used in combination with the FC or MEI readins, but provides a significant improvement with the Conv readin. Although EM produces the best of our methods in terms of quantitative measures, it leads to fewer details in the reconstructions, as can be seen in Figure 7.1. A reasonable initial explanation arises from our earlier hypothesis that EM could mitigate the effects of noise in the neural responses present in the data. Namely, since the **C** dataset has been generated artificially, there might not be much of this assumed neural noise and therefore the inaccuracies in the trained encoder used for EM might outweigh the EM benefits.

Lastly, comparing the best-performing methods, GAN-Conv with and without EM generates more contrast in the reconstructions than the inverted encoder but at the expense of lower fidelity.

Method	SSIML	MSE	PL
Inverted encoder [45]	0.125	**0.032**	**0.238**
CNN			
CNN-FC	0.103	0.040	0.287
CNN-FC w/ EM	0.340	0.070	0.370
CNN-Conv	0.137	0.040	0.291
CNN-Conv w/ EM	0.105	0.033	0.288
CNN-MEI	0.112	0.041	0.265
CNN-MEI w/ EM	0.166	0.063	0.314
GAN			
GAN-FC	0.114	0.039	0.278
GAN-FC w/ EM	0.294	0.111	0.404
GAN-Conv	0.126	0.042	0.289
GAN-Conv w/ EM	**0.097**	**0.032**	0.267
GAN-MEI	0.114	0.038	0.269
GAN-MEI w/ EM	0.182	0.045	0.305

■ **Table 7.1** Quantitative comparison of different decoding methods on the **C** dataset. Bold values signify the lowest (best) results.

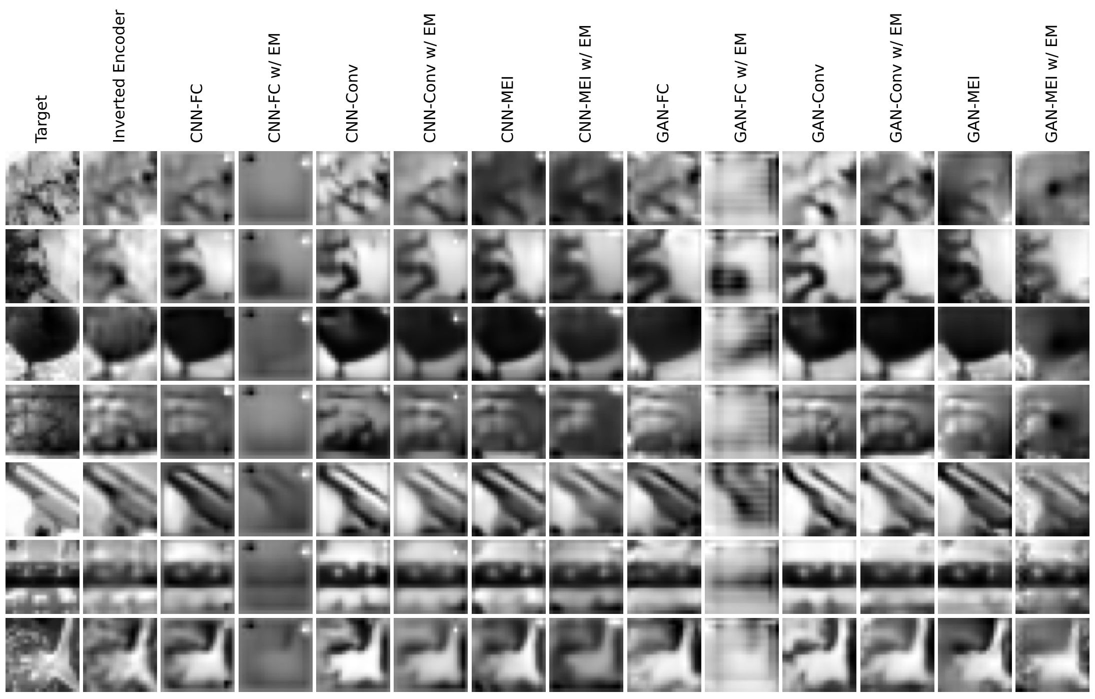


Figure 7.1 Decoded samples from the **C** dataset.

In summary, for this comparison alone, the convolutional readin in combination with EM and a GAN produces the best results. It outperforms the baseline from the literature in terms of the evaluation measure based on the SSIM, is on par with it in MSE, and performs worse from the perspective of PL. The advantages of EM are not completely clear-cut from the current results, and it is up to the experiments on biological data in the next section to validate or invalidate our initial hypothesis about its benefits.

7.2 SENSORIUM 2022 dataset

Next, we turned to the biological dataset recorded in the mouse V1 and made similar comparisons as in the previous section with an extension to synthetic data and transfer learning. It is important to note that some of the datasets here are an order of magnitude smaller than the previous dataset from the biologically realistic encoding model. On the one hand, it is a more challenging setting, and, on the other hand, it does not require as much computational resources for training, allowing for more experiments.

As a baseline decoding approach, we used the inverted encoder trained on **M-All**. The subsequent hyperparameter search performed on the validation dataset of **M-1** as well as **M-All** resulted in the same best-performing hyperparameter values, namely:

- 200 optimization steps (tested 100, 200, 500, and 1,000);
- learning rate (step size) of 150 (tested 50, 150, 500, and 1,000);
- standard deviation of 2 for the Gaussian blur of the image gradient (tested 1, 1.5, 2, 2.5).

The quantitative and qualitative results are shown in Table 7.2 and Figure 7.2, respectively. Here, we provide visual comparison only for the best-performing methods and the baseline for clarity of presentation; the decoded samples from the other decoders can be found in the appendix (A.1 and A.2). GAN-FC is omitted due to its poor performance and frequent training divergence.

From the **M-1** evaluation column of the table, the first observation is that training on all mouse datasets (**M-All**) degrades the performance compared to training on single mouse data (**M-1**) in terms of PL. The comparison measured by SSIML and MSE is less clear, but the decoders with the overall best results on **M-1** are achieved by training on **M-All** (bold values in the table).

The second observation arises from the fact that, with the exception of CNN-Conv trained and evaluated on **M-All**, the best results are obtained from methods incorporating the auxiliary EM objective. Moreover, in all but one case, EM results in lower SSIML and MSE, but usually produces worse PL. From the visual comparison in Figure 7.2, we can see that low SSIML correlates with less sharp but structurally similar reconstructions with a well-matching contrast, while PL favors higher-frequency details.

The third observation that we can make here is that, from our three introduced readin modules, the convolutional and MEI types are significantly better than the fully connected one. It is important to note here that since the FC readin does not incorporate the neural coordinates, it also has slightly fewer parameters (3,618,000) than the MEI (4,126,360) and the Conv (4,038,146) readins.

Method	**M-1**			**M-All**		
	SSIML	MSE	PL	SSIML	MSE	PL
Inverted encoder [45]	0.335	0.057	0.358	0.336	0.057	0.357
Trained on M-1						
CNN-FC	0.355	0.080	0.387	–	–	–
CNN-FC w/ EM	0.356	0.077	0.388	–	–	–
CNN-Conv	0.327	0.058	0.339	–	–	–
CNN-Conv w/ EM	0.291	0.047	0.350	–	–	–
CNN-MEI	0.324	0.056	0.341	–	–	–
CNN-MEI w/ EM	0.304	0.047	**0.337**	–	–	–
GAN-Conv	0.326	0.058	0.342	–	–	–
GAN-Conv w/ EM	0.296	0.047	0.350	–	–	–
GAN-MEI	0.321	0.056	0.338	–	–	–
GAN-MEI w/ EM	0.287	0.052	0.357	–	–	–
Trained on M-All						
CNN-FC	0.357	0.082	0.392	0.355	0.080	0.390
CNN-FC w/ EM	0.326	0.069	0.400	0.325	0.068	0.398
CNN-Conv	0.329	0.057	0.355	0.326	0.056	**0.349**
CNN-Conv w/ EM	0.288	**0.043**	0.353	0.288	**0.044**	0.359
CNN-MEI	0.312	0.064	0.378	0.323	0.055	0.355
CNN-MEI w/ EM	0.284	0.050	0.374	0.285	0.049	0.372
GAN-Conv	0.328	0.064	0.354	0.329	0.063	0.350
GAN-Conv w/ EM	0.284	0.048	0.369	**0.283**	0.047	0.364
GAN-MEI	0.334	0.064	0.358	0.329	0.062	0.354
GAN-MEI w/ EM	**0.280**	0.047	0.362	0.286	0.048	0.362

■ **Table 7.2** Quantitative comparison of decoding methods and training setups. Columns **M-1** and **M-All** refer to evaluation datasets. Bold values signify the lowest (best) results in each of the evaluation settings.

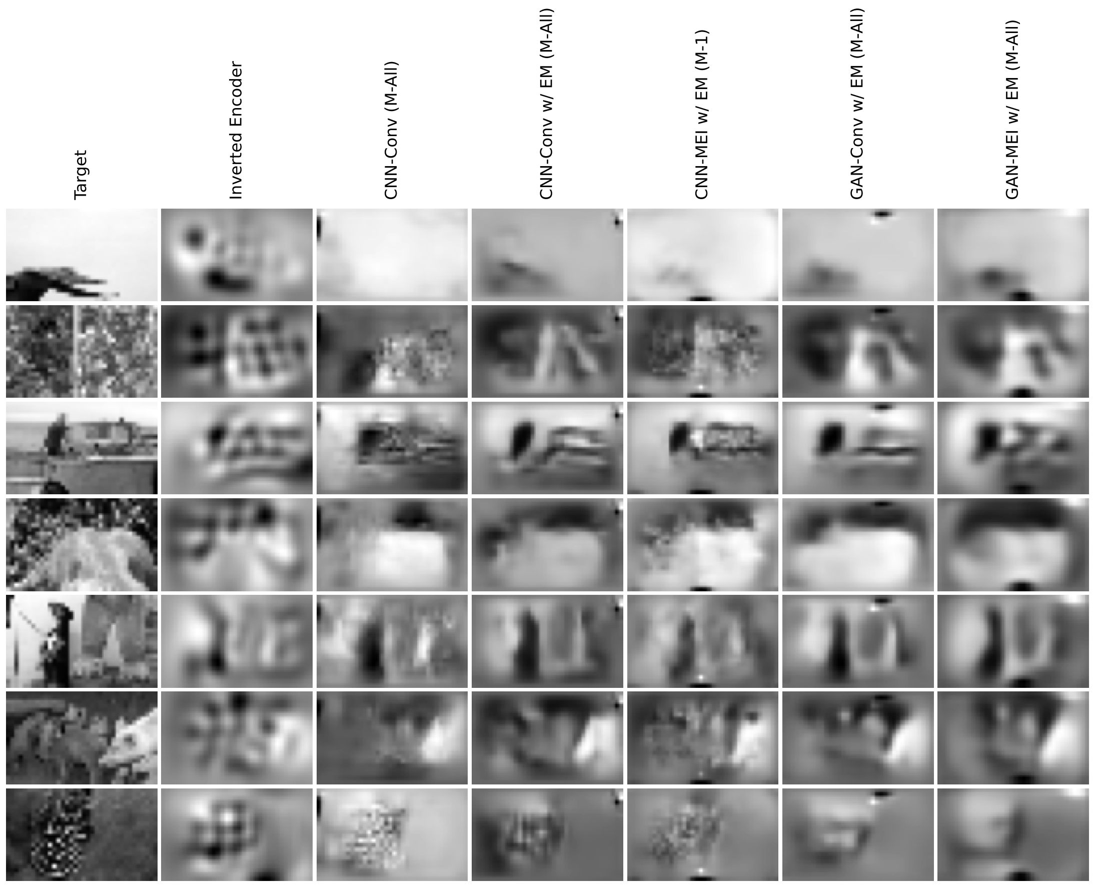


Figure 7.2 Decoded samples from the **M-1** dataset by the best-performing methods and the baseline. Dataset names in brackets refer to the datasets on which the methods were trained.

Lastly, there seem to be only minor differences between the **M-1** and **M-All** performance of methods trained on **M-All**. It tells us that the methods did not overfocus on a single mouse data but can perform well across all mice. This would be a particularly important observation in visual prosthesis applications because there, the methods need to be accurate enough for all patients instead of placing all of their expressive power on datasets with stimulus-activity mappings that are easier to learn.

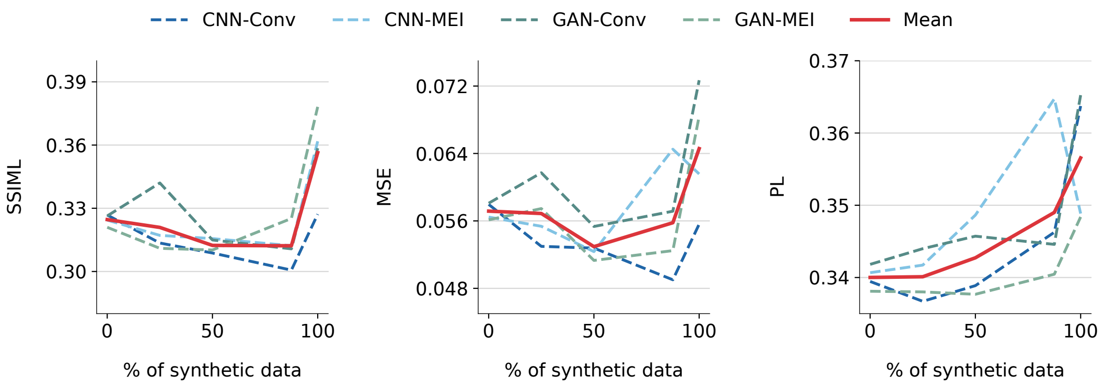
7.2.1 Impact of synthetic data

To combat the scarcity of real biological data, we experimented with synthetic data generated by a pretrained encoder as described in 5.1.1. Since the overarching goal was to improve the performance on the biological data, we used the artificially generated datasets **S-1** and **S-All** only for training. The specific aim was to understand the impact of synthetic training data for neural decoding and to shed light on the right balance between artificial and biological data.

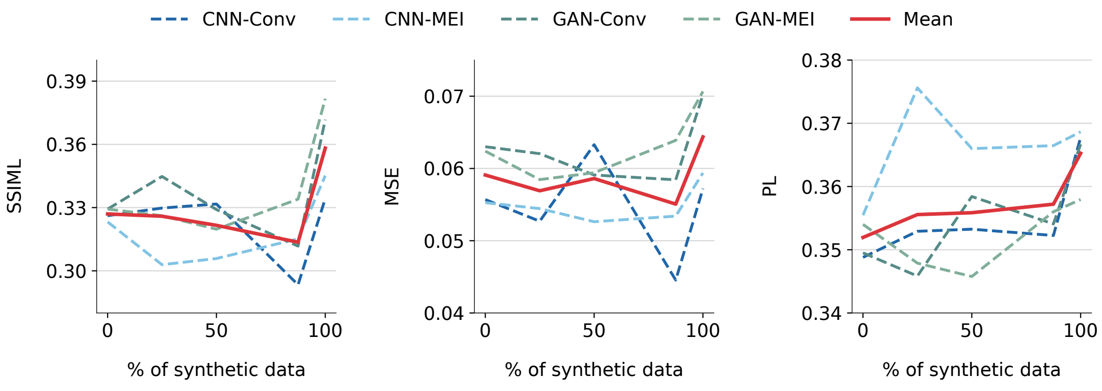
For this purpose, we chose the two best readins, Conv and MEI, from the initial results, and trained CNNs and GANs with different ratios of synthetic and biological data. More specifically, for training on **M-1** + **S-1**, we fixed the batch size at 16 or 20 and altered the portion of synthetic data **S-1** within a batch to be 0% (no synthetic data), 25%, 50%, 87.5%, or 100% (no biological data). Similarly, for training on **M-All** + **S-All**, each batch contained 7 or 8 samples from each of the 5 mice, and we only modified the contribution of synthetic samples from **S-All**. Importantly, for all, except for the case with 100% synthetic data, the models leveraged the entire biological datasets, only the amount of additional synthetic data changed. Note that the readin modules for individual mice were shared between the real and synthetic data.

To have an easier time interpreting and comprehending the results, we opt for a graphical depiction of the impact coming from the synthetic data. Interested readers can refer to Table B.1 in the appendix for specific values achieved by all the considered methods. Lastly, given the range of possible combinations of model design and training setups, together with limited computational resources, we report results only from single training runs. We believe that it is an important first step to start answering some of the questions we posed and leave a more rigorous and large-scale study for future work.

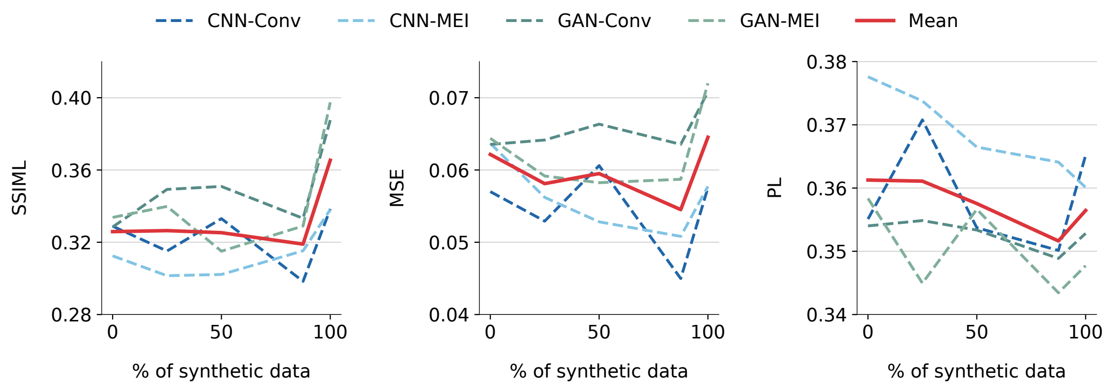
From Figures 7.3, 7.4, and 7.5, we can see that the average SSIML across the four decoding methods decreases with more synthetic data up to and including the 87.5% mark. For this evaluation metric and the range of percentage values considered, 87.5% of the synthetic data appears to be the right spot. Similar characteristics hold for MSE with the exception of training on **M-1** + **S-1** followed by **M-1** evaluation. Notably, CNN with the convolutional readin performs the best of all the methods considered in this section, and the 87.5% of synthetic data is clearly its perfect spot in terms of SSIML and MSE.



■ **Figure 7.3** Performance impact of using additional synthetic data **S-1** during training on the base dataset **M-1**. Evaluations done on the **M-1** dataset.



■ **Figure 7.4** Performance impact of using additional synthetic data **S-All** during training on the base dataset **M-All**. Evaluations done on the **M-All** dataset.



■ **Figure 7.5** Performance impact of using additional synthetic data **S-All** during training on the base dataset **M-All**. Evaluations done on the **M-1** dataset.

However, from the perspective of perceptual loss alone, synthetic data degrades performance in two out of three evaluation settings (7.3 and 7.4). But in this case, it is important to disentangle the averaged results (the red lines). Namely, while CNN-MEI performs poorly with synthetic data, GAN-MEI always benefits from it, and its optimal amount is 50% or 87.5%. This points to the importance of carefully designing the right combination of architecture and training data.

The visual comparison between the best-performing methods is available in Figure 7.6 and the reconstructions of all the remaining methods can be found in the appendix. Note that although not specified in full in the method names, the **S-1** and **S-All** datasets are always implicitly accompanied by **M-1** and **M-All**, respectively, filling the rest of the combined dataset to 100%.

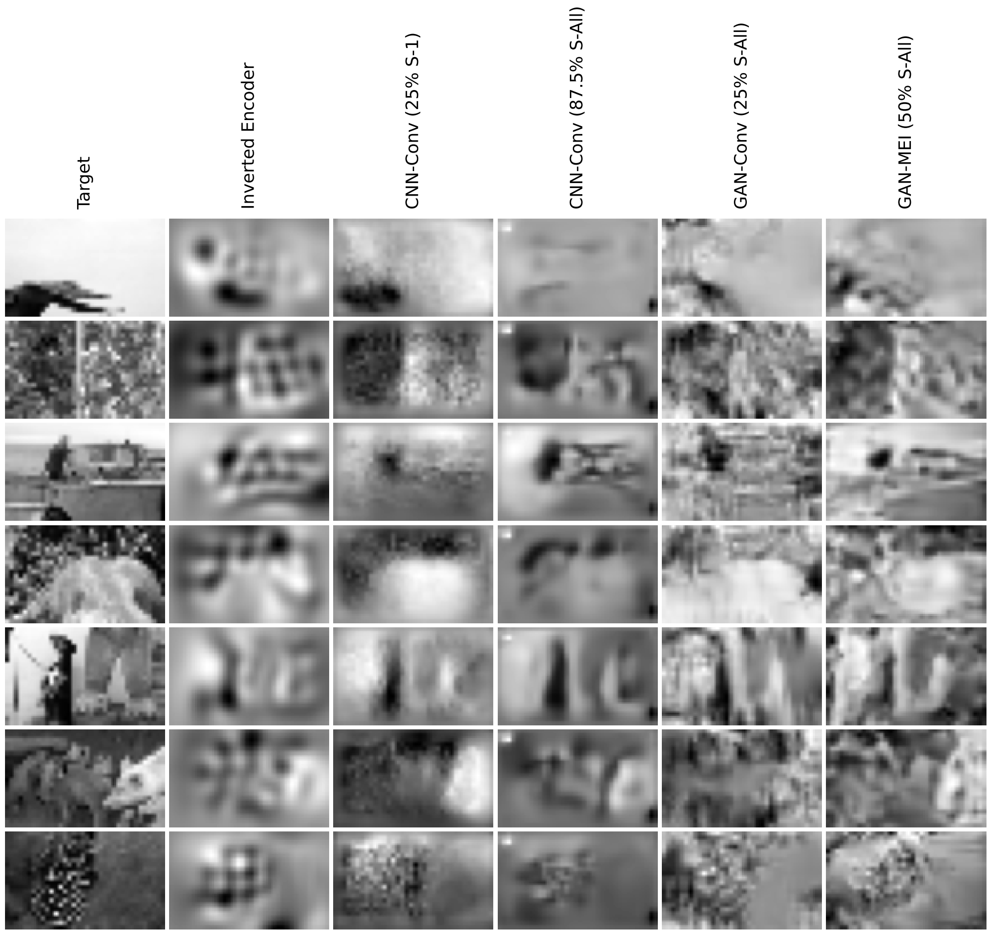
One can notice that the plain CNN decoders do not capture the fine-grained structure of images but are relatively noise-free, while GANs decode very detailed reconstructions but sometimes produce hallucinated content. Subjectively speaking, the latter is preferred by us. Hence, we think that perceptual loss, according to which GANs perform the best, is more suitable than SSIML and MSE. Future work could incorporate PL as a substitute for the SSIM-based training loss function, as the only criterion for model selection, or both.

In conclusion, if SSIML or MSE are metrics that one cares about the most, they should opt for around 87.5% of synthetic training data. That is, for each data point in the base (real) dataset, they should add 7 additional synthetic samples. Since the best methods, according to PL, are with 25% and 50% of synthetic data (see B.1), it is the range that should be aimed at if one focuses more on detailed reconstructions with occasional unmatched contrast or noise. Nevertheless, considering the variation in the method performance from the three figures, especially in terms of PL, the amount of synthetic data should be tuned to the specific decoding architecture.

7.3 Transfer learning

Lastly, we studied whether pretraining on a larger dataset followed by separate training, so-called fine-tuning, only on the target dataset could improve the results. For the pretraining datasets, we considered all available data we had, including data from the biologically realistic encoding model (**C**), SENSORIUM 2022 data (**M-1** and **M-All**), and synthetic datasets obtained from the CNN-based encoder (**S-1** and **S-All**). **M-1** then served as the target (fine-tuning) dataset.

In the fine-tuning process, we used the pretrained core of the decoder architecture, discarded the readin modules, and freshly initialized a single readin. Subsequently, we trained the pretrained core in combination with the new readin end-to-end on the **M-1** dataset. For the method naming convention, we specify transfer learning by writing the pretraining dataset to the left of a right-arrow symbol ($\rightarrow$), followed by the fine-tuning dataset.



■ **Figure 7.6** Decoded samples from the **M-1** dataset by the baseline and the best-performing methods from the synthetic data section 7.2.1.

The baselines included the inverted encoder, as in previous sections, as well as methods trained solely on the target dataset **M-1**. Since the aim was to evaluate the benefits of transfer learning compared to standard training, and given the limited time and computational resources available for the thesis, we omitted experiments with GANs and focused only on the CNN architecture.

From the quantitative results reported in Table 7.3, it is clear that fine-tuning after **M-All** + **S-All** pretraining provides significant improvements. In particular, for the CNN-Conv architecture, all considered amounts of pretraining data **S-All** results in lower SSIML, MSE, and PL. For CNN-MEI, the benefits are less pronounced but still demonstrate that transfer learning is a desirable training paradigm. The ideal amount of synthetic data **S-All** appears to be architecture-dependent. Specifically, CNN-Conv achieves the best performance with around 87.5% of synthetic data, while CNN-MEI does not have an optimal amount that works best across all measures. A visual comparison can be found in the appendix (A.7 and A.8).

Pretraining on the **C** dataset alone provides similar improvements as pretraining on **M-All** + **S-All** for CNN-MEI, but significantly degrades the performance of CNN-Conv. One plausible explanation for this is that the MEI readin forces the core network to work with inputs that encode the receptive fields of individual neurons. This may result in the pretrained core being more adaptable to novel response-to-stimulus mappings during fine-tuning because it has been trained to just map contextualized receptive fields of individual neurons to natural-looking

Method	SSIML	MSE	PL
Inverted encoder [45]	0.335	0.057	0.358
CNN-Conv (M-1)	0.327	0.058	0.339
CNN-Conv (0% S-All → M-1)	0.306	0.049	0.336
CNN-Conv (50% S-All → M-1)	0.306	0.044	**0.328**
CNN-Conv (87.5% S-All → M-1)	0.301	**0.042**	**0.328**
CNN-Conv (100% S-All → M-1)	**0.300**	0.048	0.337
CNN-Conv (C → M-1)	0.337	0.062	0.347
CNN-MEI (M-1)	0.324	0.056	0.341
CNN-MEI (0% S-All → M-1)	0.323	0.050	0.338
CNN-MEI (50% S-All → M-1)	0.321	0.053	0.343
CNN-MEI (87.5% S-All → M-1)	0.321	0.054	0.340
CNN-MEI (100% S-All → M-1)	0.323	0.054	0.342
CNN-MEI (C → M-1)	0.323	0.052	0.336

Table 7.3 Evaluation of transfer learning performance on the **M-1** dataset. The pretraining dataset **S-All** is accompanied by **M-All**. Bold values signify the lowest (best) results.

images. In contrast, the convolutional readin is not restricted to MEIs and receptive fields, allowing more freedom in encoding the individual neuron responses. This could make the core network much harder to transfer because the representations coming from the pretraining readin and the new fine-tuning readin can vary more significantly and arbitrarily.

Intuitively, the core network has learned a specific *neural language* during pretraining and, in the case of CNN-Conv, is forced to learn a completely new language with different words and meanings, akin to mastering a foreign language from scratch. Meanwhile, in CNN-MEI, the core is merely tasked with understanding a novel ordering of the same words as in the original language, somewhat akin to transitioning from infix to prefix notation in mathematics, where the fundamental elements remain the same but their arrangement changes.

7.4 Summary of key results

Based on the results presented in this chapter, the best-performing model designs and training paradigms are as follows:

- For the biologically realistic encoding model dataset (5.2), the GAN-Conv with our novel auxiliary encoder matching objective (6.3) performs the best in terms of SSIM and MSE. A similar MSE is also achieved by one of the state-of-the-art baseline approaches from the literature, called the inverted encoder. Moreover, none of our methods is able to achieve a lower perceptual loss than this baseline.

- For the biological dataset **M-1** (5.1), CNN core with the convolutional readin pretrained on **M-All** + **S-All** and subsequently fine-tuned on **M-1** achieves the best MSE and PL as well as very competitive SSIM. The optimal amount of the synthetic data **S-All** during pretraining is around 87.5% of the combined dataset size. In terms of SSIM alone, the best results are obtained by GAN with our novel MEI readin and encoder matching objective.

- For the entire biological dataset **M-All** (5.1), CNN-Conv and GAN-Conv with EM perform best from the perspective of SSIM and MSE, respectively, while GAN-Conv with 25% of synthetic data **S-All** achieves the lowest PL. Notably, CNN-Conv with 87.5% of synthetic data achieves SSIM and MSE close to the models that incorporate EM but not synthetic data.

However, given the benefits of transfer learning observed for **M-1**, we expect even better results for **M-All** after **M-All** + **S-All** pretraining. Due to limited time and resources, we leave this for future work.

Based on the findings presented above and our extensive experimentation, we would approach the neural activity decoding task on a new recording dataset with CNN-Conv and transfer learning. More specifically, we would pretrain a CNN-Conv on a dataset consisting of 12.5% real and 87.5% synthetic data generated by a CNN-based encoder, and subsequently fine-tune on the available biological data.

In Figure 7.7, columns 3 to 5 show the reconstructions generated by the methods with the lowest SSIML, MSE, and PL obtained on the dataset **M-1**. Specifically, from left to right:

- GAN-MEI with EM trained on **M-1** achieved the lowest SSIML of 0.280;
- CNN-Conv pretrained on a combined dataset 50% **M-All** + 50% **S-All** and fine-tuned on **M-1** reached the lowest PL of 0.328;
- CNN-Conv pretrained on a combined dataset 12.5% **M-All** + 87.5% **S-All** and fine-tuned on **M-1** achieved the best MSE of 0.042 as well as the lowest PL of 0.328.

Additionally, since we noticed during our experiments that the quantitative measures used for our model selection (6) do not fully capture perceptual quality, we also tried training some of the methods for longer and then picked the best manually based on reconstructed samples from the validation dataset. Reconstructions of these decoders are shown in the last four columns in Figure 7.7, and their names are marked with an asterisk (*). Although such a model selection is not optimal, it shows the achievable performance when a more suitable criterion is in place.

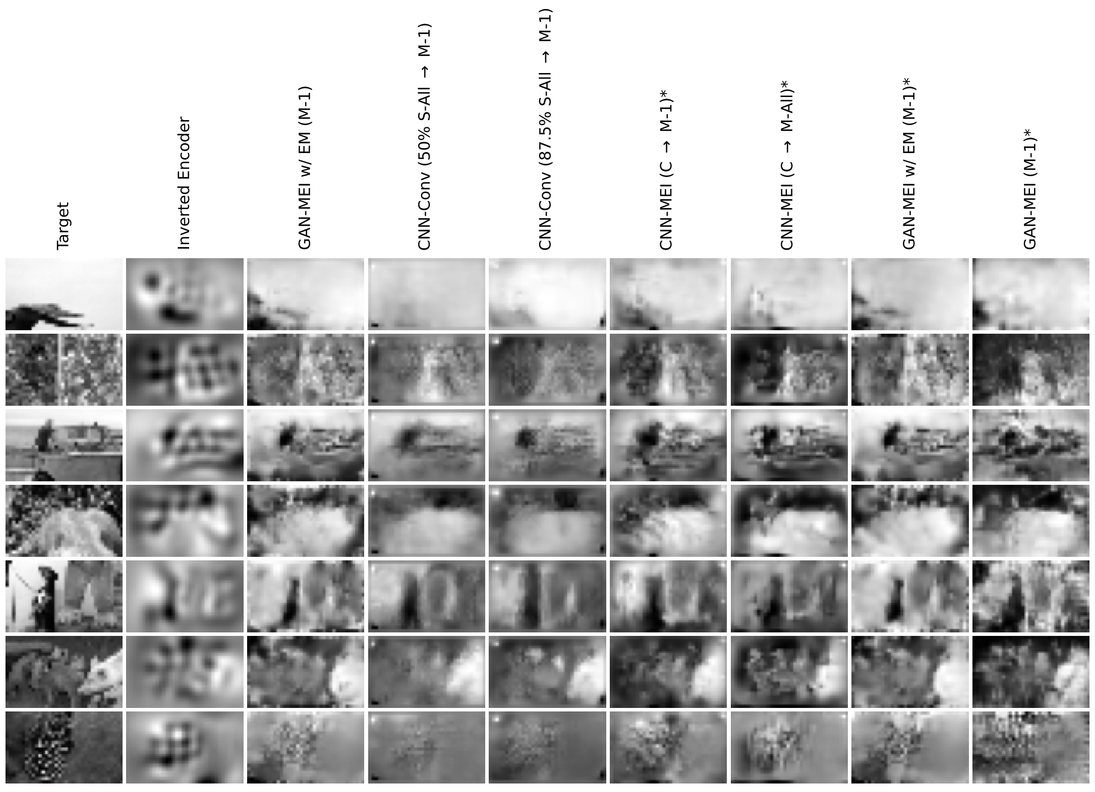


Figure 7.7 Decoded samples from the **M-1** dataset by the best-performing methods.

Conclusions

In this thesis, our objective was to test a range of deep neural network architectures and training paradigms for the data-constrained task of decoding neural activity into images. For this overarching goal, we introduced novel decoding architectures derived from convolutional neural networks and generative adversarial networks, developed an auxiliary training loss called encoder matching, and quantified how synthetic data and transfer learning influence the final performance. We performed extensive comparisons and selected the best-performing decoding method from experiments on data from a biologically plausible model of cat V1, as well as in vivo recordings of V1 from mice.

More specifically, we (1) demonstrated how data-constrained settings can benefit from incorporating priors in the form of most exciting inputs, (2) found the optimal amount of synthetic training data, and (3) outperformed the state-of-the-art baseline in the setting of limited data using our decoding architectures and novel encoder matching objective. Despite the improvements and findings of this thesis, making neural decoding accurate and reliable enough for real-world applications in visual prostheses still requires further work.

Firstly, our decoded reconstructions still lack the finest details that might be paramount in use cases such as fine motor tasks of visual prosthesis users. This is closely related to the fact that we tested our methods only on relatively small images, and it is unclear how the methods would scale to larger image stimuli in this data-constrained setting.

Secondly, invasive neural recording devices can be costly, difficult to maintain, and pose risks; therefore, having a better understanding of the required number of recorded neurons in advance could be very beneficial. In this regard, future work could explore correlating the amount of decoded neural responses with the quality of the reconstructions.

Third, a broader consideration concerns the image quality measures, among which we observed striking disagreements. Given that many of these measures target perceptual quality from a human perspective, it would be interesting to incorporate neural responses evoked by the reference and reconstructed images, thereby comparing their neural equivalence.

Lastly, considering the improvements coming from the use of synthetic data and transfer learning, future work could explore other pretraining datasets and training paradigms. Furthermore, we see great promise in merging our structurally accurate reconstructions with the semantic information available from the currently trending image generation models trained on large-scale datasets.

In conclusion, our results demonstrate the potential of deep learning-based methods for decoding visual percepts from neural responses and highlight the importance of careful architecture and dataset design. Since neural decoding is a relatively nascent field, it presents an exciting opportunity for researchers to develop novel methods and tackle new challenges. Moreover, given the potential applications that range from visual prostheses to robot control and brain-to-brain communication, neural decoding is of significant importance beyond the academic setting.

Appendix A

Qualitative results on the SENSORIUM 2022 dataset

Here we show example reconstructions by all methods presented in the main part of the thesis. The brackets in the names of the decoding methods contain information on the specific datasets on which the models were trained.

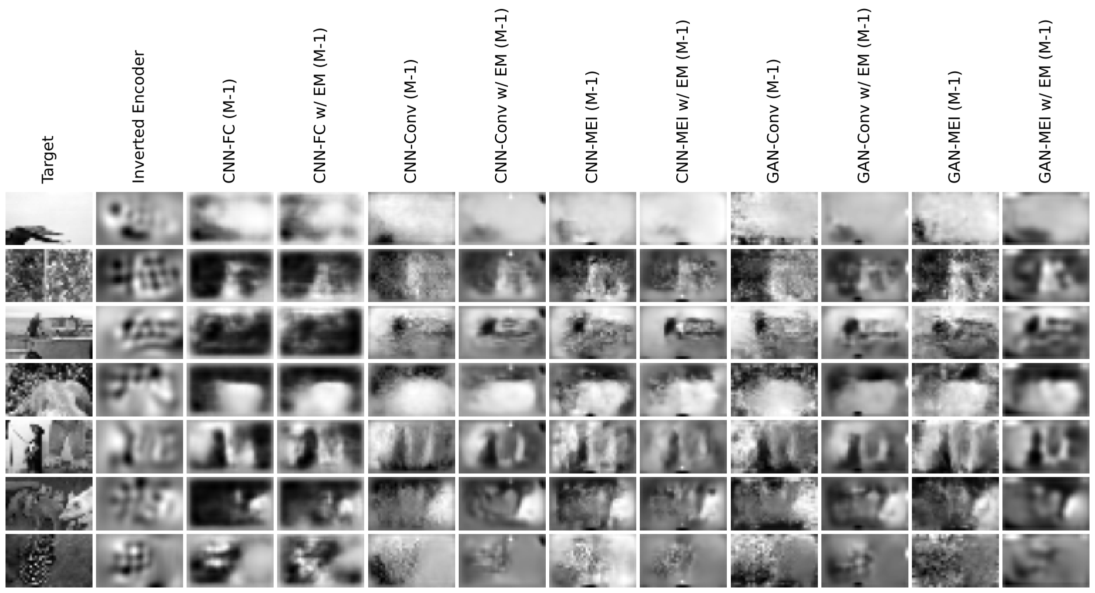
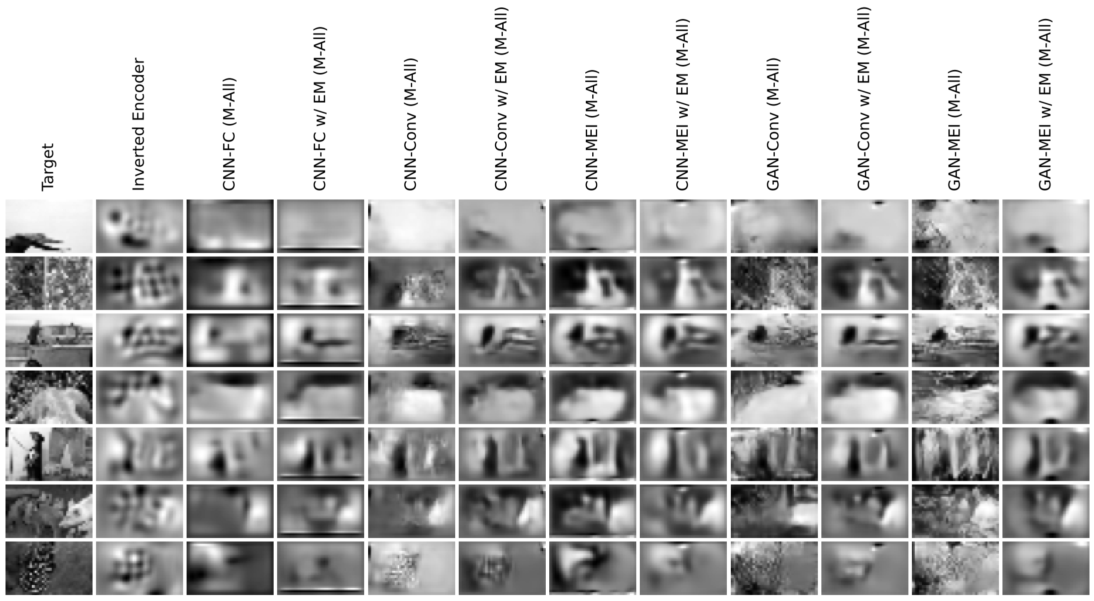
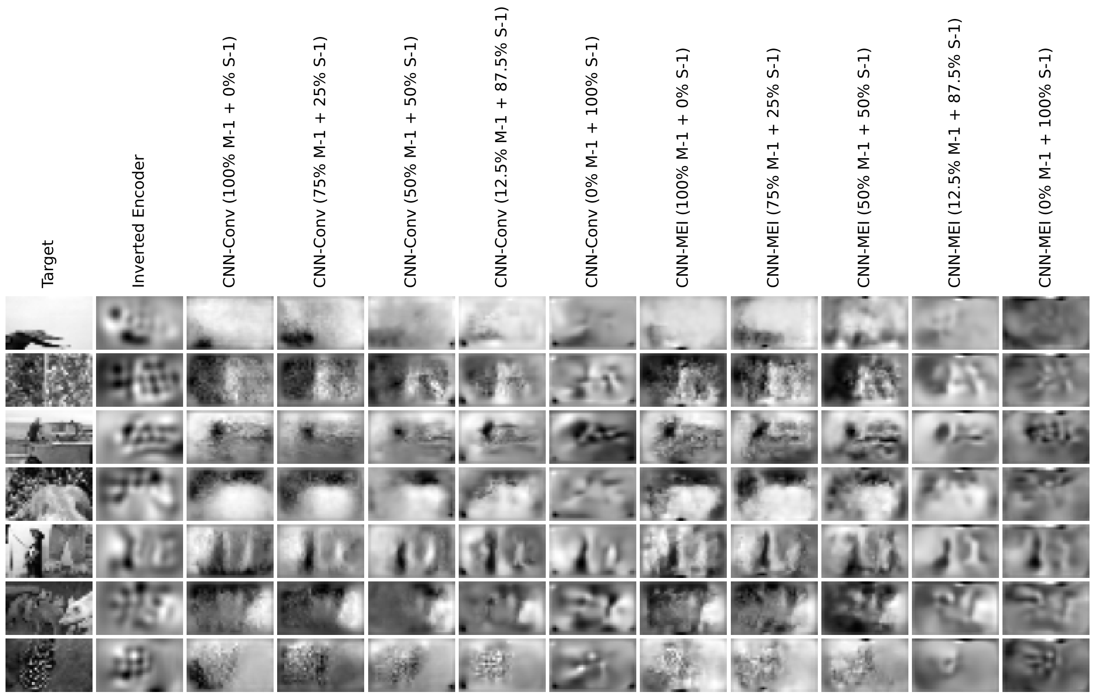


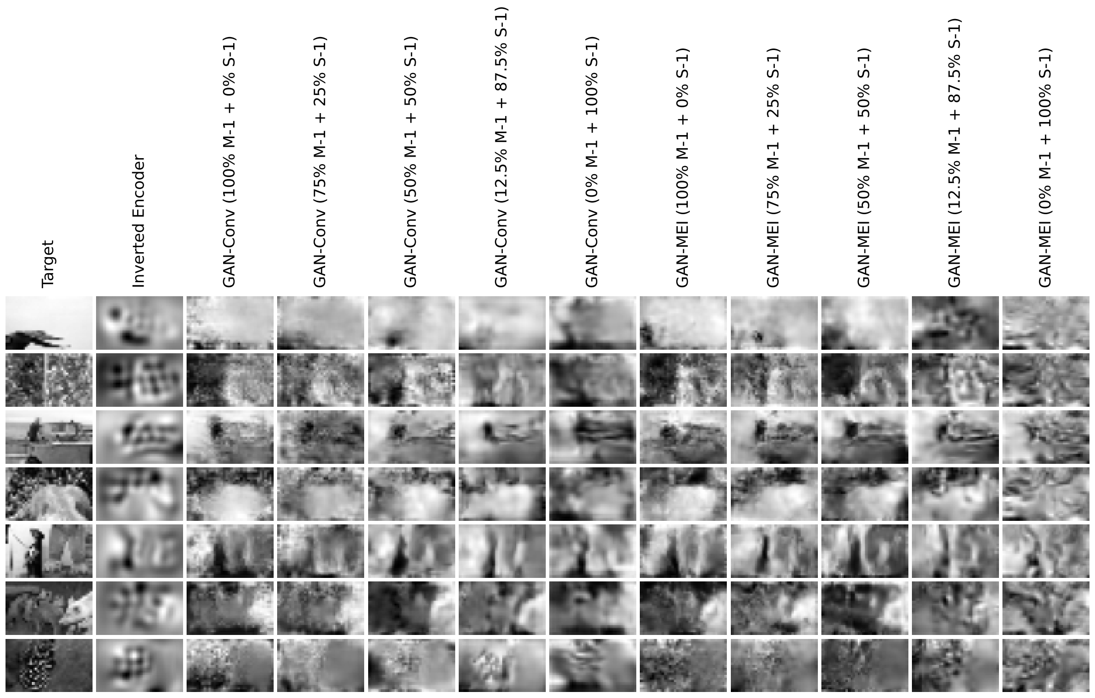
Figure A.1 Decoded samples from the **M-1** dataset by methods trained on **M-1**.



■ **Figure A.2** Decoded samples from the **M-1** dataset by methods trained on **M-All**.

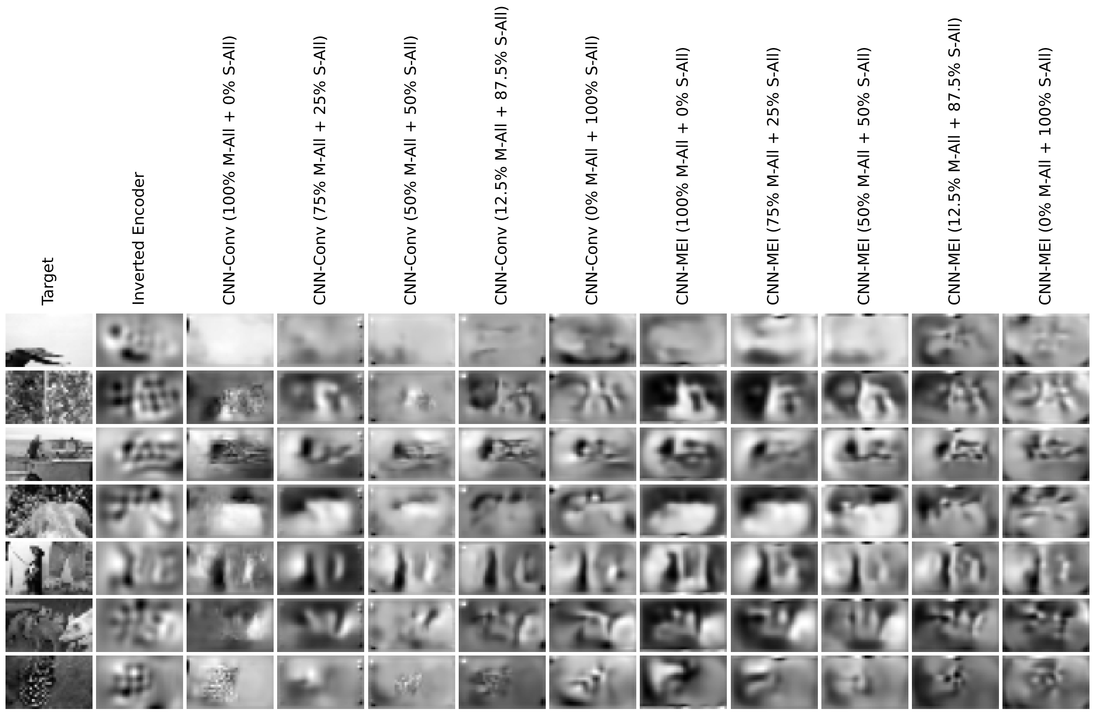

Target
Inverted Encoder
CNN-Conv (100% M-1 + 0% S-1)
CNN-Conv (75% M-1 + 25% S-1)
CNN-Conv (50% M-1 + 50% S-1)
CNN-Conv (12.5% M-1 + 87.5% S-1)
CNN-Conv (0% M-1 + 100% S-1)
CNN-MEI (100% M-1 + 0% S-1)
CNN-MEI (75% M-1 + 25% S-1)
CNN-MEI (50% M-1 + 50% S-1)
CNN-MEI (12.5% M-1 + 87.5% S-1)
CNN-MEI (0% M-1 + 100% S-1)

■ **Figure A.3** Decoded samples from the **M-1** dataset by CNN-based methods trained on **M-1** and **S-1**.



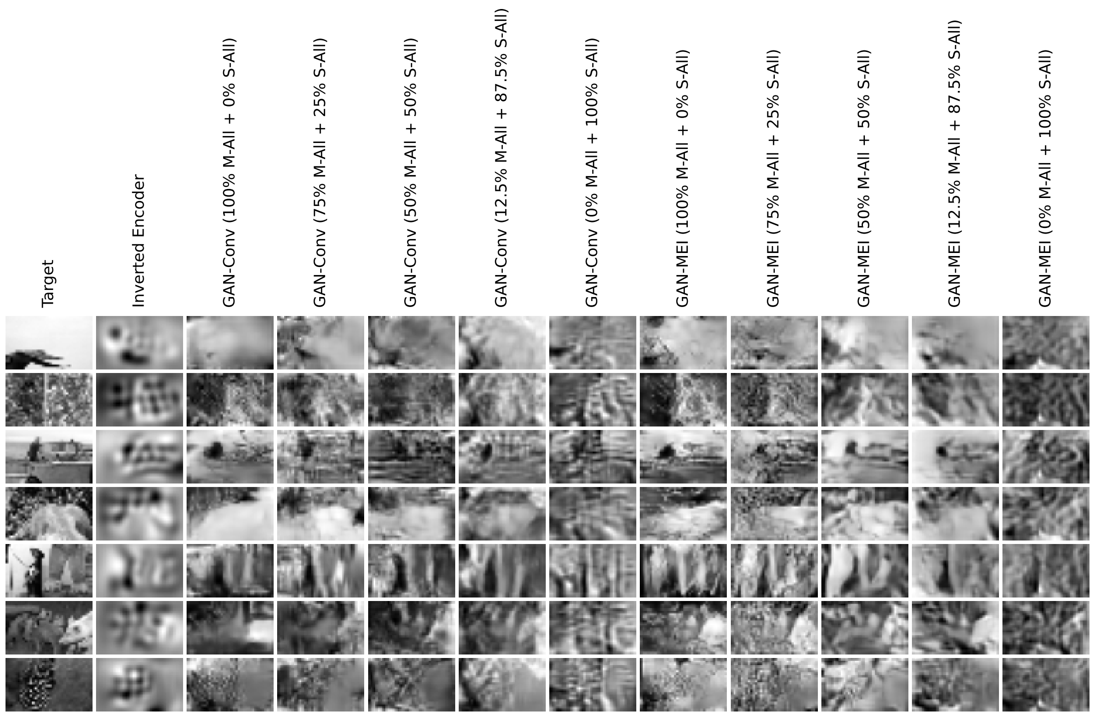
Target
Inverted Encoder
GAN-Conv (100% M-1 + 0% S-1)
GAN-Conv (75% M-1 + 25% S-1)
GAN-Conv (50% M-1 + 50% S-1)
GAN-Conv (12.5% M-1 + 87.5% S-1)
GAN-Conv (0% M-1 + 100% S-1)
GAN-MEI (100% M-1 + 0% S-1)
GAN-MEI (75% M-1 + 25% S-1)
GAN-MEI (50% M-1 + 50% S-1)
GAN-MEI (12.5% M-1 + 87.5% S-1)
GAN-MEI (0% M-1 + 100% S-1)

Figure A.4 Decoded samples from the **M-1** dataset by GAN-based methods trained on **M-1** and **S-1**.

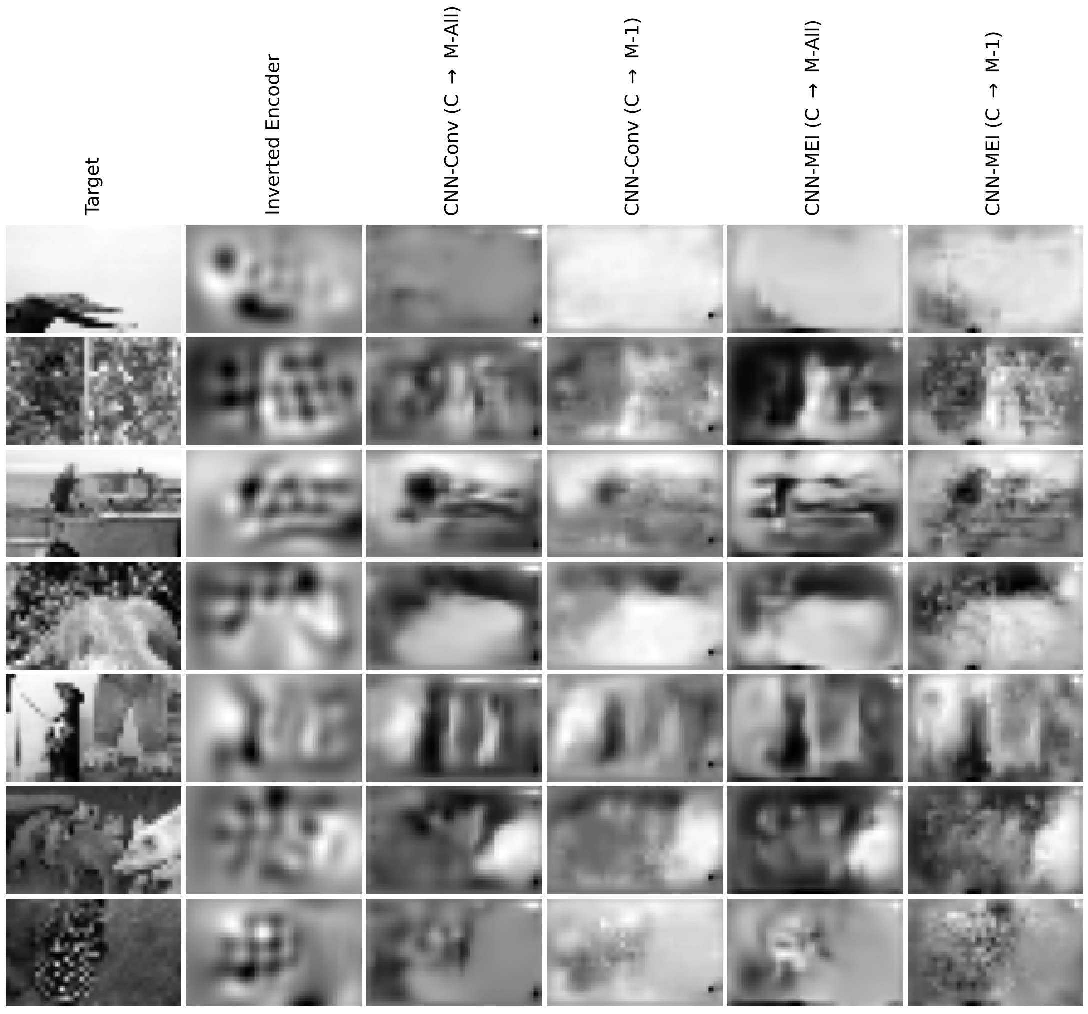


Target
Inverted Encoder
CNN-Conv (100% M-All + 0% S-All)
CNN-Conv (75% M-All + 25% S-All)
CNN-Conv (50% M-All + 50% S-All)
CNN-Conv (12.5% M-All + 87.5% S-All)
CNN-Conv (0% M-All + 100% S-All)
CNN-MEI (100% M-All + 0% S-All)
CNN-MEI (75% M-All + 25% S-All)
CNN-MEI (50% M-All + 50% S-All)
CNN-MEI (12.5% M-All + 87.5% S-All)
CNN-MEI (0% M-All + 100% S-All)

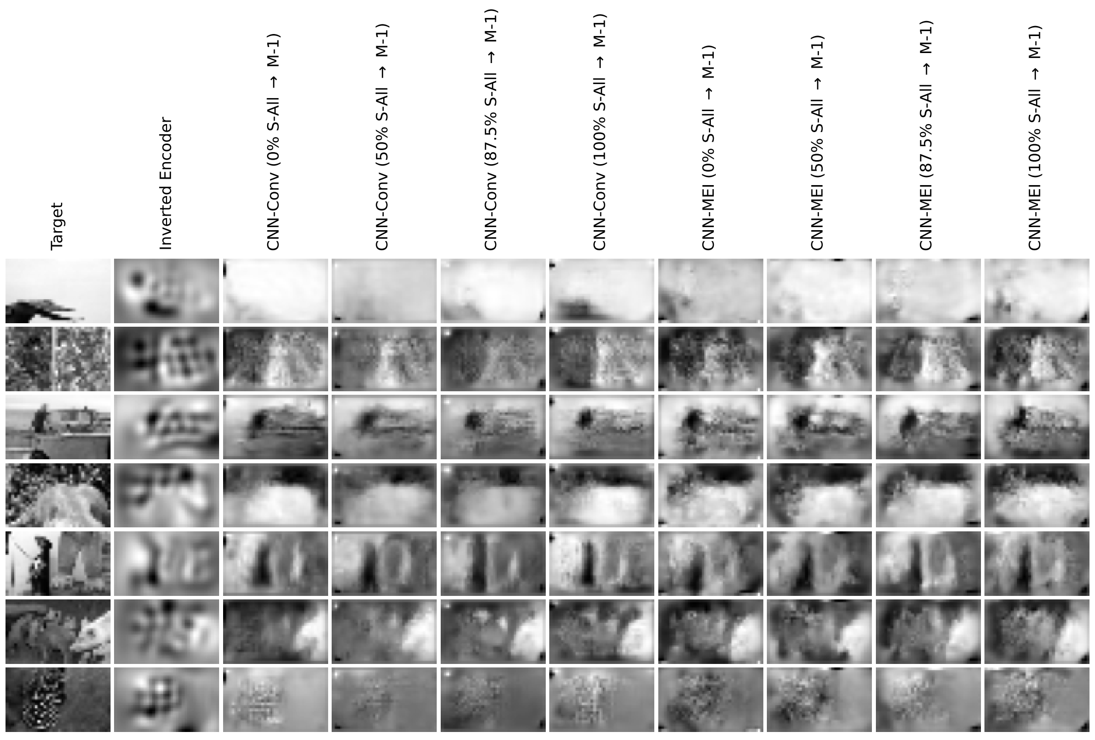
Figure A.5 Decoded samples from the **M-1** dataset by CNN-based methods trained on **M-All** and **S-All**.


Target
Inverted Encoder
GAN-Conv (100% M-All + 0% S-All)
GAN-Conv (75% M-All + 25% S-All)
GAN-Conv (50% M-All + 50% S-All)
GAN-Conv (12.5% M-All + 87.5% S-All)
GAN-Conv (0% M-All + 100% S-All)
GAN-MEI (100% M-All + 0% S-All)
GAN-MEI (75% M-All + 25% S-All)
GAN-MEI (50% M-All + 50% S-All)
GAN-MEI (12.5% M-All + 87.5% S-All)
GAN-MEI (0% M-All + 100% S-All)

■ **Figure A.6** Decoded samples from the **M-1** dataset by GAN-based methods trained on **M-All** and **S-All**.



■ **Figure A.7** Decoded samples from the **M-1** dataset by CNN-based methods pretrained on **C** and fine-tuned on **M-1** or **M-All**.


Target
Inverted Encoder
CNN-Conv (0% S-All → M-1)
CNN-Conv (50% S-All → M-1)
CNN-Conv (87.5% S-All → M-1)
CNN-Conv (100% S-All → M-1)
CNN-MEI (0% S-All → M-1)
CNN-MEI (50% S-All → M-1)
CNN-MEI (87.5% S-All → M-1)
CNN-MEI (100% S-All → M-1)

■ **Figure A.8** Decoded samples from the **M-1** dataset by methods pretrained on **M-All** + **S-All** and fine-tuned on **M-1**.

Appendix B

Quantitative results after training on synthetic data

Here we show the quantitative evaluation of methods trained with additional synthetic data.

Method	M-1			M-All		
	SSIML	MSE	PL	SSIML	MSE	PL
Inverted encoder [45]	0.335	0.057	0.358	0.336	0.057	0.357
Trained on M-1 + S-1						
CNN-Conv (0% S-1)	0.327	0.058	0.339	–	–	–
CNN-Conv (25% S-1)	0.314	0.053	**0.337**	–	–	–
CNN-Conv (50% S-1)	0.309	0.053	0.339	–	–	–
CNN-Conv (87.5% S-1)	0.301	0.049	0.346	–	–	–
CNN-Conv (100% S-1)	0.327	0.056	0.364	–	–	–
CNN-MEI (0% S-1)	0.324	0.056	0.341	–	–	–
CNN-MEI (25% S-1)	0.317	0.055	0.342	–	–	–
CNN-MEI (50% S-1)	0.316	0.052	0.349	–	–	–
CNN-MEI (87.5% S-1)	0.312	0.065	0.365	–	–	–
CNN-MEI (100% S-1)	0.362	0.062	0.349	–	–	–
GAN-Conv (0% S-1)	0.326	0.058	0.342	–	–	–
GAN-Conv (25% S-1)	0.342	0.062	0.344	–	–	–
GAN-Conv (50% S-1)	0.315	0.055	0.346	–	–	–
GAN-Conv (87.5% S-1)	0.311	0.057	0.345	–	–	–
GAN-Conv (100% S-1)	0.358	0.073	0.365	–	–	–
GAN-MEI (0% S-1)	0.321	0.056	0.338	–	–	–
GAN-MEI (25% S-1)	0.311	0.057	0.338	–	–	–
GAN-MEI (50% S-1)	0.310	0.051	0.338	–	–	–
GAN-MEI (87.5% S-1)	0.325	0.052	0.340	–	–	–
GAN-MEI (100% S-1)	0.378	0.068	0.348	–	–	–
Trained on M-All + S-All						
CNN-Conv (0% S-All)	0.329	0.057	0.355	0.326	0.056	0.349
CNN-Conv (25% S-All)	0.315	0.053	0.371	0.330	0.053	0.353
CNN-Conv (50% S-All)	0.333	0.061	0.354	0.332	0.063	0.353
CNN-Conv (87.5% S-All)	**0.298**	**0.045**	0.350	**0.293**	**0.045**	0.352
CNN-Conv (100% S-All)	0.338	0.058	0.365	0.334	0.057	0.368
CNN-MEI (0% S-All)	0.312	0.064	0.378	0.323	0.055	0.355
CNN-MEI (25% S-All)	0.301	0.056	0.374	0.303	0.054	0.376
CNN-MEI (50% S-All)	0.302	0.053	0.366	0.306	0.053	0.366
CNN-MEI (87.5% S-All)	0.312	0.065	0.365	0.315	0.053	0.366
CNN-MEI (100% S-All)	0.338	0.058	0.360	0.345	0.059	0.369
GAN-Conv (0% S-All)	0.328	0.064	0.354	0.329	0.063	0.350
GAN-Conv (25% S-All)	0.349	0.064	0.355	0.345	0.062	**0.346**
GAN-Conv (50% S-All)	0.351	0.066	0.353	0.329	0.059	0.358
GAN-Conv (87.5% S-All)	0.333	0.064	0.349	0.312	0.058	0.354
GAN-Conv (100% S-All)	0.388	0.071	0.353	0.372	0.070	0.367
GAN-MEI (0% S-All)	0.334	0.064	0.358	0.329	0.062	0.354
GAN-MEI (25% S-All)	0.340	0.059	0.345	0.326	0.058	0.348
GAN-MEI (50% S-All)	0.315	0.058	0.357	0.320	0.059	**0.346**
GAN-MEI (87.5% S-All)	0.329	0.059	0.343	0.334	0.064	0.356
GAN-MEI (100% S-All)	0.397	0.072	0.348	0.382	0.071	0.358

■ **Table B.1** Performance after training on synthetic data. Columns **M-1** and **M-All** refer to evaluation datasets. Bold measurement values signify the lowest (best) results in each of the evaluation settings.

Bibliography

1. O'CONNOR, Daniel H; FUKUI, Miki M; PINSK, Mark A; KASTNER, Sabine. Attention modulates responses in the human lateral geniculate nucleus. *Nature Neuroscience*. 2002, vol. 5, no. 11, pp. 1203–1209. Available from DOI: `10.1038/nn957`.
2. HUBEL, D H; WIESEL, T N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol*. 1962, vol. 160, no. 1, pp. 106–154. Available from DOI: `10.1113/jphysiol.1962.sp006837`.
3. HEDGES, V. *Introduction to Neuroscience*. Michigan State University Libraries, 2022. ISBN 9781626101227. Available also from: `https://openbooks.lib.msu.edu/introneuroscience1/`.
4. BEAR, M.; CONNORS, B.; PARADISO, M.A. *Neuroscience: Exploring the Brain, Enhanced Edition: Exploring the Brain, Enhanced Edition*. Jones & Bartlett Learning, 2020. ISBN 9781284211283. Available also from: `https://books.google.cz/books?id=m-PcDwAAQBAJ`.
5. CADENA, Santiago A.; WILLEKE, Konstantin F.; RESTIVO, Kelli; DENFIELD, George; SINZ, Fabian H.; BETHGE, Matthias; TOLIAS, Andreas S.; ECKER, Alexander S. Diverse task-driven modeling of macaque V4 reveals functional specialization towards semantic tasks. *bioRxiv*. 2023. Available from DOI: `10.1101/2022.05.18.492503`.
6. JONES, J P; PALMER, L A. The two-dimensional spatial structure of simple receptive fields in cat striate cortex. *J Neurophysiol*. 1987, vol. 58, no. 6, pp. 1187–1211. Available from DOI: `10.1152/jn.1987.58.6.1187`.
7. LAU, Brian; STANLEY, Garrett B; DAN, Yang. Computational subunits of visual cortical neurons revealed by artificial neural networks. *Proc Natl Acad Sci U S A*. 2002, vol. 99, no. 13, pp. 8974–8979. Available from DOI: `10.1073/pnas.122173799`.
8. PRENGER, Ryan; WU, Michael C-K; DAVID, Stephen V; GALLANT, Jack L. Nonlinear V1 responses to natural scenes revealed by neural network analysis. *Neural Netw*. 2004, vol. 17, no. 5-6, pp. 663–679. Available from DOI: `10.1016/j.neunet.2004.03.008`.
9. ANTOLÍK, Ján; HOFER, Sonja B.; BEDNAR, James A.; MRSIC-FLOGEL, Thomas D. Model Constrained by Visual Hierarchy Improves Prediction of Neural Responses to Natural Scenes. *PLOS Computational Biology*. 2016, vol. 12, no. 6, pp. 1–22. Available from DOI: `10.1371/journal.pcbi.1004927`.
10. ZHANG, Yimeng; LEE, Tai Sing; LI, Ming; LIU, Fang; TANG, Shiming. Convolutional neural network models of V1 responses to complex patterns. *J Comput Neurosci*. 2018, vol. 46, no. 1, pp. 33–54. Available from DOI: `10.1007/s10827-018-0687-7`.

11. LURZ, Konstantin-Klemens; BASHIRI, Mohammad; WILLEKE, Konstantin; JAGADISH, Akshay K.; WANG, Eric; WALKER, Edgar Y.; CADENA, Santiago A.; MUHAMMAD, Taliah; COBOS, Erick; TOLIAS, Andreas S.; ECKER, Alexander S.; SINZ, Fabian H. Generalization in data-driven models of primary visual cortex. *bioRxiv.* 2021. Available from DOI: `10.1101/2020.10.05.326256`.

12. LI, Bryan M.; CORNACCHIA, Isabel Maria; ROCHEFORT, Nathalie; ONKEN, Arno. V1T: large-scale mouse V1 response prediction using a Vision Transformer. *Transactions on Machine Learning Research.* 2023. ISSN 2835-8856. Available also from: `https://openreview.net/forum?id=qHZs2p4ZD4`.

13. IVAKHNENKO, A.G.; LAPA, V.G.; ENGINEERING., PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL. *Cybernetic Predicting Devices.* Joint Publications Research Service [available from the Clearinghouse for Federal Scientific and Technical Information], 1965. JPRS 37, 803. Available also from: `https://books.google.cz/books?id=l38DHQAACAAJ`.

14. ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review.* 1958, vol. 65, no. 6, pp. 386–408. Available from DOI: `10.1037/h0042519`.

15. CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems.* 1989, vol. 2, no. 4, pp. 303–314. ISSN 1435-568X. Available from DOI: `10.1007/BF02551274`.

16. ZHOU, Ding-Xuan. Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis.* 2020, vol. 48, no. 2, pp. 787–794. ISSN 1063-5203. Available from DOI: `10.1016/j.acha.2019.06.004`.

17. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

18. CADENA, Santiago A.; DENFIELD, George H.; WALKER, Edgar Y.; GATYS, Leon A.; TOLIAS, Andreas S.; BETHGE, Matthias; ECKER, Alexander S. Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLOS Computational Biology.* 2019, vol. 15, no. 4, pp. 1–27. Available from DOI: `10.1371/journal.pcbi.1006897`.

19. WILLEKE, Konstantin F; FAHEY, Paul G; BASHIRI, Mohammad; PEDE, Laura; BURG, Max F; BLESSING, Christoph; CADENA, Santiago A; DING, Zhiwei; LURZ, Konstantin-Klemens; PONDER, Kayla, et al. The Sensorium competition on predicting large-scale mouse primary visual cortex activity. *arXiv preprint arXiv:2206.08666.* 2022. Available also from: `https://arxiv.org/abs/2206.08666`.

20. FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics.* 1980, vol. 36, no. 4, pp. 193–202. ISSN 1432-0770. Available from DOI: `10.1007/BF00344251`.

21. IOFFE, Sergey; SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: BACH, Francis; BLEI, David (eds.). *Proceedings of the 32nd International Conference on Machine Learning.* Lille, France: PMLR, 2015, vol. 37, pp. 448–456. Proceedings of Machine Learning Research. Available also from: `https://proceedings.mlr.press/v37/ioffe15.html`.

22. SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research.* 2014, vol. 15, no. 56, pp. 1929–1958. Available also from: `http://jmlr.org/papers/v15/srivastava14a.html`.

23. CHOLLET, François. Xception: Deep Learning with Depthwise Separable Convolutions. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807. Available from DOI: `10.1109/CVPR.2017.195`.

24. CLEVERT, Djork-Arné; UNTERTHINER, Thomas; HOCHREITER, Sepp. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In: BENGIO, Yoshua; LECUN, Yann (eds.). *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016. Available also from: `http://arxiv.org/abs/1511.07289`.

25. KINGMA, Diederik P; WELLING, Max. *Auto-Encoding Variational Bayes*. 2022. Available from DOI: `10.48550/arXiv.1312.6114`.

26. WALKER, Edgar Y; SINZ, Fabian H; COBOS, Erick; MUHAMMAD, Taliah; FROUDARAKIS, Emmanouil; FAHEY, Paul G; ECKER, Alexander S; REIMER, Jacob; PITKOW, Xaq; TOLIAS, Andreas S. Inception loops discover what excites neurons most using deep predictive models. *Nat Neurosci*. 2019, vol. 22, no. 12, pp. 2060–2065. Available from DOI: `10.1038/s41593-019-0517-x`.

27. PONCE, Carlos R; XIAO, Will; SCHADE, Peter F; HARTMANN, Till S; KREIMAN, Gabriel; LIVINGSTONE, Margaret S. Evolving Images for Visual Neurons Using a Deep Generative Network Reveals Coding Principles and Neuronal Preferences. *Cell*. 2019, vol. 177, no. 4, 999–1009.e10. Available from DOI: `10.1016/j.cell.2019.04.005`.

28. BARONI, Luca; BASHIRI, Mohammad; WILLEKE, Konstantin F.; ANTOLÍK, Ján; SINZ, Fabian H. Learning invariance manifolds of visual sensory neurons. In: SANBORN, Sophia; SHEWMAKE, Christian; AZEGLIO, Simone; DI BERNARDO, Arianna; MIOLANE, Nina (eds.). *Proceedings of the 1st NeurIPS Workshop on Symmetry and Geometry in Neural Representations*. PMLR, 2023, vol. 197, pp. 301–326. Proceedings of Machine Learning Research. Available also from: `https://proceedings.mlr.press/v197/baroni23a.html`.

29. ERHAN, Dumitru; BENGIO, Y.; COURVILLE, Aaron; VINCENT, Pascal. Visualizing Higher-Layer Features of a Deep Network. *Technical Report, Univeristé de Montréal*. 2009. Available also from: `https://www.researchgate.net/publication/265022827_Visualizing_Higher-Layer_Features_of_a_Deep_Network`.

30. OLAH, Chris; MORDVINTSEV, Alexander; SCHUBERT, Ludwig. Feature Visualization. *Distill*. 2017. Available from DOI: `10.23915/distill.00007`.

31. BENCHETRIT, Yohann; BANVILLE, Hubert; KING, Jean-Remi. Brain decoding: toward real-time reconstruction of visual perception. In: *The Twelfth International Conference on Learning Representations*. 2024. Available also from: `https://openreview.net/forum?id=3y1K6buO8c`.

32. NISHIMOTO, Shinji; VU, An T; NASELARIS, Thomas; BENJAMINI, Yuval; YU, Bin; GALLANT, Jack L. Reconstructing visual experiences from brain activity evoked by natural movies. *Curr Biol*. 2011, vol. 21, no. 19, pp. 1641–1646. Available from DOI: `10.1016/j.cub.2011.08.031`.

33. SEELIGER, K; GÜÇLÜ, U; AMBROGIONI, L; GÜÇLÜTÜRK, Y; GERVEN, M A J van. Generative adversarial networks for reconstructing natural images from brain activity. *Neuroimage*. 2018, vol. 181, pp. 775–785. Available from DOI: `10.1016/j.neuroimage.2018.07.043`.

34. SCOTTI, Paul Steven; BANERJEE, Atmadeep; GOODE, Jimmie; SHABALIN, Stepan; NGUYEN, Alex; ETHAN, Cohen; DEMPSTER, Aidan James; VERLINDE, Nathalie; YUNDLER, Elad; WEISBERG, David; NORMAN, Kenneth; ABRAHAM, Tanishq Mathew. Reconstructing the Mind's Eye: fMRI-to-Image with Contrastive Learning and Diffusion Priors. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. Available also from: `https://openreview.net/forum?id=rwrblCYb2A`.

35. TAKAGI, Yu; NISHIMOTO, Shinji. High-resolution image reconstruction with latent diffusion models from human brain activity. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 14453–14463. Available from DOI: `10.1109/CVPR52729.2023.01389`.

36. SOHL-DICKSTEIN, Jascha; WEISS, Eric; MAHESWARANATHAN, Niru; GANGULI, Surya. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In: BACH, Francis; BLEI, David (eds.). *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015, vol. 37, pp. 2256–2265. Proceedings of Machine Learning Research. Available also from: `https://proceedings.mlr.press/v37/sohl-dickstein15.html`.

37. HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising diffusion probabilistic models. *Advances in neural information processing systems*. 2020, vol. 33, pp. 6840–6851. Available from DOI: `10.48550/arXiv.2006.11239`.

38. BRACKBILL, Nora; RHOADES, Colleen; KLING, Alexandra; SHAH, Nishal P; SHER, Alexander; LITKE, Alan M; CHICHILNISKY, E J. Reconstruction of natural images from responses of primate retinal ganglion cells. *Elife*. 2020, vol. 9. Available from DOI: `10.7554/eLife.58516`.

39. KIM, Young Joon; BRACKBILL, Nora; BATTY, Eleanor; LEE, Jinhyung; MITELUT, Catalin; TONG, William; CHICHILNISKY, E J; PANINSKI, Liam. Nonlinear Decoding of Natural Images From Large-Scale Primate Retinal Ganglion Recordings. *Neural Comput.* 2021, vol. 33, no. 7, pp. 1719–1750. Available from DOI: `10.1162/neco_a_01395`.

40. BENSTER, Tyler; BABINO, Darwin; THICKSTUN, John; HUNT, Matthew; LIU, Xiyang; HARCHAOUI, Zaid; OH, Sewoong; GELDER, Russell N. Van. Reconstruction of visual images from mouse retinal ganglion cell spiking activity using convolutional neural networks. *bioRxiv*. 2022. Available from DOI: `10.1101/2022.06.10.482188`.

41. THIRION, Bertrand; DUCHESNAY, Edouard; HUBBARD, Edward; DUBOIS, Jessica; POLINE, Jean-Baptiste; LEBIHAN, Denis; DEHAENE, Stanislas. Inverse retinotopy: inferring the visual content of images from brain activation patterns. *Neuroimage*. 2006, vol. 33, no. 4, pp. 1104–1116. Available from DOI: `10.1016/j.neuroimage.2006.06.062`.

42. GOODFELLOW, Ian; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua. Generative adversarial nets. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680. Available also from: `https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

43. HAYASHI, Ryusuke; KAWATA, Hayaki. Image Reconstruction from Neural Activity Recorded from Monkey Inferior Temporal Cortex Using Generative Adversarial Networks. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2018, pp. 105–109. Available from DOI: `10.1109/SMC.2018.00028`.

44. RAN, Xuming; ZHANG, Jie; YE, Ziyuan; WU, Haiyan; XU, Qi; ZHOU, Huihui; LIU, Quanying. Deep auto-encoder with neural response. *arXiv preprint arXiv:2111.15309*. 2021. Available from DOI: `10.48550/arXiv.2111.15309`.

45. COBOS, Erick; MUHAMMAD, Taliah; FAHEY, Paul G.; DING, Zhiwei; DING, Zhuokun; REIMER, Jacob; SINZ, Fabian H.; TOLIAS, Andreas S. It takes neurons to understand neurons: Digital twins of visual cortex synthesize neural metamers. *bioRxiv*. 2022. Available from DOI: `10.1101/2022.12.09.519708`.

46. ZHANG, Yijun; BU, Tong; ZHANG, Jiyuan; TANG, Shiming; YU, Zhaofei; LIU, Jian K; HUANG, Tiejun. Decoding Pixel-Level Image Features From Two-Photon Calcium Signals of Macaque Visual Cortex. *Neural Comput.* 2022, vol. 34, no. 6, pp. 1369–1397. Available from DOI: `10.1162/neco_a_01498`.

47. LI, Wenyi; ZHENG, Shengjie; LIAO, Yufan; HONG, Rongqi; HE, Chenggang; CHEN, Weiliang; DENG, Chunshan; LI, Xiaojian. The brain-inspired decoder for natural visual image reconstruction. *Frontiers in Neuroscience*. 2023, vol. 17. ISSN 1662-453X. Available from DOI: 10.3389/fnins.2023.1130606.

48. PIERZCHLEWICZ, Paweł A.; WILLEKE, Konstantin Friedrich; NIX, Arne; ELUMALAI, Pavithra; RESTIVO, Kelli; SHINN, Tori; NEALLEY, Cate; RODRIGUEZ, Gabrielle; PATEL, Saumil; FRANKE, Katrin; TOLIAS, Andreas S.; SINZ, Fabian H. Energy Guided Diffusion for Generating Neurally Exciting Images. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. Available also from: https://openreview.net/forum?id=1moStpWGUj.

49. CHEN, Xing; WANG, Feng; FERNANDEZ, Eduardo; ROELFSEMA, Pieter R. Shape perception via a high-channel-count neuroprosthesis in monkey visual cortex. *Science*. 2020, vol. 370, no. 6521, pp. 1191–1196. Available from DOI: 10.1126/science.abd7435.

50. FERNÁNDEZ, Eduardo; ALFARO, Arantxa; SOTO-SÁNCHEZ, Cristina; GONZALEZ-LOPEZ, Pablo; LOZANO, Antonio M.; PEÑA, Sebastian; GRIMA, Maria Dolores; RODIL, Alfonso; GÓMEZ, Bernardeta; CHEN, Xing; ROELFSEMA, Pieter R.; ROLSTON, John D.; DAVIS, Tyler S.; NORMANN, Richard A. Visual percepts evoked with an intracortical 96-channel microelectrode array inserted in human occipital cortex. *The Journal of Clinical Investigation*. 2021, vol. 131, no. 23. Available from DOI: 10.1172/JCI151331.

51. HOCHBERG, Leigh R; BACHER, Daniel; JAROSIEWICZ, Beata; MASSE, Nicolas Y; SIMERAL, John D; VOGEL, Joern; HADDADIN, Sami; LIU, Jie; CASH, Sydney S; SMAGT, Patrick van der; DONOGHUE, John P. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*. 2012, vol. 485, no. 7398, pp. 372–375. Available from DOI: 10.1038/nature11076.

52. CHAPIN, John K; MOXON, Karen A; MARKOWITZ, Ronald S; NICOLELIS, Miguel A L. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature Neuroscience*. 1999, vol. 2, no. 7, pp. 664–670. Available from DOI: 10.1038/10223.

53. DONOGHUE, John P.; NURMIKKO, Arto; BLACK, Michael; HOCHBERG, Leigh R. Assistive technology and robotic control using motor cortex ensemble-based neural interface systems in humans with tetraplegia. *The Journal of Physiology*. 2007, vol. 579, no. 3, pp. 603–611. Available from DOI: 10.1113/jphysiol.2006.127209.

54. ANUMANCHIPALLI, Gopala K; CHARTIER, Josh; CHANG, Edward F. Speech synthesis from neural decoding of spoken sentences. *Nature*. 2019, vol. 568, no. 7753, pp. 493–498. Available from DOI: 10.1038/s41586-019-1119-1.

55. WILLETT, Francis R; AVANSINO, Donald T; HOCHBERG, Leigh R; HENDERSON, Jaimie M; SHENOY, Krishna V. High-performance brain-to-text communication via handwriting. *Nature*. 2021, vol. 593, no. 7858, pp. 249–254. Available from DOI: 10.1038/s41586-021-03506-2.

56. METZ, Luke; POOLE, Ben; PFAU, David; SOHL-DICKSTEIN, Jascha. Unrolled Generative Adversarial Networks. In: *International Conference on Learning Representations*. 2017. Available also from: https://openreview.net/forum?id=BydrOIcle.

57. ODENA, Augustus; OLAH, Christopher; SHLENS, Jonathon. Conditional Image Synthesis with Auxiliary Classifier GANs. In: PRECUP, Doina; TEH, Yee Whye (eds.). *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, vol. 70, pp. 2642–2651. Proceedings of Machine Learning Research. Available also from: https://proceedings.mlr.press/v70/odena17a.html.

58. ZHANG, Han; GOODFELLOW, Ian; METAXAS, Dimitris; ODENA, Augustus. Self-Attention Generative Adversarial Networks. In: CHAUDHURI, Kamalika; SALAKHUTDINOV, Ruslan (eds.). *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, vol. 97, pp. 7354–7363. Proceedings of Machine Learning Research. Available also from: `https://proceedings.mlr.press/v97/zhang19d.html`.

59. KADURIN, Artur; NIKOLENKO, Sergey; KHRABROV, Kuzma; ALIPER, Alex; ZHAVORONKOV, Alex. druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico. *Molecular pharmaceutics*. 2017, vol. 14, no. 9, pp. 3098–3104. Available from DOI: `10.1021/acs.molpharmaceut.7b00346`.

60. ZHANG, Han; XU, Tao; LI, Hongsheng; ZHANG, Shaoting; WANG, Xiaogang; HUANG, Xiaolei; METAXAS, Dimitris N. StackGAN: Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017. Available also from: `https://openaccess.thecvf.com/content_ICCV_2017/papers/Zhang_StackGAN_Text_to_ICCV_2017_paper.pdf`.

61. ISOLA, Phillip; ZHU, Jun-Yan; ZHOU, Tinghui; EFROS, Alexei A. Image-To-Image Translation With Conditional Adversarial Networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. Available also from: `https://openaccess.thecvf.com/content_cvpr_2017/papers/Isola_Image-To-Image_Translation_With_CVPR_2017_paper.pdf`.

62. SALIMANS, Tim; GOODFELLOW, Ian; ZAREMBA, Wojciech; CHEUNG, Vicki; RADFORD, Alec; CHEN, Xi. Improved techniques for training GANs. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2234–2242. NIPS'16. ISBN 9781510838819. Available also from: `https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf`.

63. GUI, Jie; SUN, Zhenan; WEN, Yonggang; TAO, Dacheng; YE, Jieping. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *IEEE Transactions on Knowledge and Data Engineering*. 2023, vol. 35, no. 4, pp. 3313–3332. Available from DOI: `10.1109/TKDE.2021.3130191`.

64. KARRAS, Tero; AILA, Timo; LAINE, Samuli; LEHTINEN, Jaakko. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In: *International Conference on Learning Representations*. 2018. Available also from: `https://openreview.net/forum?id=Hk99zCeAb`.

65. KARRAS, Tero; LAINE, Samuli; AILA, Timo. A Style-Based Generator Architecture for Generative Adversarial Networks. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4396–4405. Available from DOI: `10.1109/CVPR.2019.00453`.

66. REGMI, Krishna; BORJI, Ali. Cross-view image synthesis using geometry-guided conditional GANs. *Computer Vision and Image Understanding*. 2019, vol. 187, p. 102788. ISSN 1077-3142. Available from DOI: `10.1016/j.cviu.2019.07.008`.

67. ELGAMMAL, Ahmed; LIU, Bingchen; ELHOSEINY, Mohamed; MAZZONE, Marian. Can: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*. 2017. Available from DOI: `10.48550/arXiv.1706.07068`.

68. SANDFORT, Veit; YAN, Ke; PICKHARDT, Perry J.; SUMMERS, Ronald M. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. *Scientific Reports*. 2019, vol. 9, no. 1, p. 16884. ISSN 2045-2322. Available from DOI: `10.1038/s41598-019-52737-x`.

69. CHEN, Xinyuan; XU, Chang; YANG, Xiaokang; SONG, Li; TAO, Dacheng. Gated-gan: Adversarial gated networks for multi-collection style transfer. *IEEE Transactions on Image Processing*. 2018, vol. 28, no. 2, pp. 546–560. Available from DOI: 10.1109/TIP.2018.2869695.

70. ZHU, Jun-Yan; PARK, Taesung; ISOLA, Phillip; EFROS, Alexei A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232. Available from DOI: 10.1109/ICCV.2017.244.

71. LEDIG, Christian; THEIS, Lucas; HUSZÁR, Ferenc; CABALLERO, Jose; CUNNINGHAM, Andrew; ACOSTA, Alejandro; AITKEN, Andrew; TEJANI, Alykhan; TOTZ, Johannes; WANG, Zehan; SHI, Wenzhe. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 105–114. Available from DOI: 10.1109/CVPR.2017.19.

72. WANG, Xintao; YU, Ke; WU, Shixiang; GU, Jinjin; LIU, Yihao; DONG, Chao; QIAO, Yu; LOY, Chen Change. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. In: LEAL-TAIXÉ, Laura; ROTH, Stefan (eds.). *Computer Vision – ECCV 2018 Workshops*. Cham: Springer International Publishing, 2019, pp. 63–79. ISBN 978-3-030-11021-5. Available from DOI: 10.1007/978-3-030-11021-5_5.

73. KADURIN, Artur; ALIPER, Alexander; KAZENNOV, Andrey; MAMOSHINA, Polina; VANHAELEN, Quentin; KHRABROV, Kuzma; ZHAVORONKOV, Alex. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*. 2017, vol. 8, no. 7, p. 10883. Available from DOI: 10.18632/oncotarget.14073.

74. SANCHEZ-LENGELING, Benjamin; OUTEIRAL, Carlos; GUIMARAES, Gabriel L; ASPURU-GUZIK, Alan. Optimizing distributions over molecular space. An objective-reinforced generative adversarial network for inverse-design chemistry (ORGANIC). 2017. Available from DOI: 10.26434/chemrxiv.5309668.v3.

75. SEELIGER, K.; GÜÇLÜ, U.; AMBROGIONI, L.; GÜÇLÜTÜRK, Y.; VAN GERVEN, M.A.J. Generative adversarial networks for reconstructing natural images from brain activity. *NeuroImage*. 2018, vol. 181, pp. 775–785. ISSN 1053-8119. Available from DOI: https://doi.org/10.1016/j.neuroimage.2018.07.043.

76. DE MELO, Celso M.; TORRALBA, Antonio; GUIBAS, Leonidas; DICARLO, James; CHELLAPPA, Rama; HODGINS, Jessica. Next-generation deep learning based on simulators and synthetic data. *Trends in Cognitive Sciences*. 2022, vol. 26, no. 2, pp. 174–187. ISSN 1364-6613. Available from DOI: 10.1016/j.tics.2021.11.008.

77. LU, Yingzhou; SHEN, Minjie; WANG, Huazheng; WANG, Xiao; RECHEM, Capucine van; WEI, Wenqi. *Machine Learning for Synthetic Data Generation: A Review*. 2024. Available from DOI: 10.48550/arXiv.2302.04062.

78. TODOROV, Emanuel; EREZ, Tom; TASSA, Yuval. MuJoCo: A physics engine for model-based control. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. Available from DOI: 10.1109/IROS.2012.6386109.

79. ZHAO, Wenshuai; QUERALTA, Jorge Peña; WESTERLUND, Tomi. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 737–744. Available from DOI: 10.1109/SSCI47803.2020.9308468.

80. CHEN, Xiaoyu; HU, Jiachen; JIN, Chi; LI, Lihong; WANG, Liwei. Understanding Domain Randomization for Sim-to-real Transfer. In: *International Conference on Learning Representations*. 2022. Available also from: https://openreview.net/forum?id=T8vZHIRTrY.

81. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is All you Need. In: GUYON, I.; LUXBURG, U. Von; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30. Available also from: `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

82. LIU, Ming-Yu; HUANG, Xun; YU, Jiahui; WANG, Ting-Chun; MALLYA, Arun. Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications. *Proceedings of the IEEE*. 2021, vol. 109, no. 5, pp. 839–862. Available from DOI: `10.1109/JPROC.2021.3049196`.

83. BROWN, Tom B.; MANN, Benjamin; RYDER, Nick; SUBBIAH, Melanie; KAPLAN, Jared; DHARIWAL, Prafulla; NEELAKANTAN, Arvind; SHYAM, Pranav; SASTRY, Girish; ASKELL, Amanda; AGARWAL, Sandhini; HERBERT-VOSS, Ariel; KRUEGER, Gretchen; HENIGHAN, Tom; CHILD, Rewon; RAMESH, Aditya; ZIEGLER, Daniel M.; WU, Jeffrey; WINTER, Clemens; HESSE, Christopher; CHEN, Mark; SIGLER, Eric; LITWIN, Mateusz; GRAY, Scott; CHESS, Benjamin; CLARK, Jack; BERNER, Christopher; MCCANDLISH, Sam; RADFORD, Alec; SUTSKEVER, Ilya; AMODEI, Dario. *Language Models are Few-Shot Learners*. 2020. Available from DOI: `10.48550/arXiv.2005.14165`.

84. CHADEBEC, Clément; THIBEAU-SUTRE, Elina; BURGOS, Ninon; ALLASSONNIÈRE, Stéphanie. Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder. *IEEE Trans. Pattern Anal. Mach. Intell.* 2023, vol. 45, no. 3, pp. 2879–2896. Available from DOI: `10.1109/TPAMI.2022.3185773`.

85. GHORBANI, Amirata; NATARAJAN, Vivek; COZ, David; LIU, Yuan. DermGAN: Synthetic Generation of Clinical Skin Images with Pathology. In: DALCA, Adrian V.; MCDERMOTT, Matthew B.A.; ALSENTZER, Emily; FINLAYSON, Samuel G.; OBERST, Michael; FALCK, Fabian; BEAULIEU-JONES, Brett (eds.). *Proceedings of the Machine Learning for Health NeurIPS Workshop*. PMLR, 2020, vol. 116, pp. 155–170. Proceedings of Machine Learning Research. Available also from: `https://proceedings.mlr.press/v116/ghorbani20a.html`.

86. POTLURU, Vamsi K.; BORRAJO, Daniel; COLETTA, Andrea; DALMASSO, Niccolò; EL-LAHAM, Yousef; FONS, Elizabeth; GHASSEMI, Mohsen; GOPALAKRISHNAN, Sriram; GOSAI, Vikesh; KREAČIĆ, Eleonora; MANI, Ganapathy; OBITAYO, Saheed; PARAMANAND, Deepak; RAMAN, Natraj; SOLONIN, Mikhail; SOOD, Srijan; VYETRENKO, Svitlana; ZHU, Haibei; VELOSO, Manuela; BALCH, Tucker. *Synthetic Data Applications in Finance*. 2024. Available from DOI: `10.48550/arXiv.2401.00081`.

87. NIKOLENKO, Sergey I. *Synthetic Data for Deep Learning*. 2019. Available from DOI: `10.48550/arXiv.1909.11512`.

88. PASZKE, Adam; GROSS, Sam; MASSA, Francisco; LERER, Adam; BRADBURY, James; CHANAN, Gregory; KILLEEN, Trevor; LIN, Zeming; GIMELSHEIN, Natalia; ANTIGA, Luca; DESMAISON, Alban; KÖPF, Andreas; YANG, Edward; DEVITO, Zach; RAISON, Martin; TEJANI, Alykhan; CHILAMKURTHY, Sasank; STEINER, Benoit; FANG, Lu; BAI, Junjie; CHINTALA, Soumith. PyTorch: an imperative style, high-performance deep learning library. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019. Available also from: `https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`.

89. MUMUNI, Alhassan; MUMUNI, Fuseini. Data augmentation: A comprehensive survey of modern approaches. *Array*. 2022, vol. 16, p. 100258. ISSN 2590-0056. Available from DOI: `10.1016/j.array.2022.100258`.

90. WANG, Zhou; BOVIK, A.C.; SHEIKH, H.R.; SIMONCELLI, E.P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*. 2004, vol. 13, no. 4, pp. 600–612. Available from DOI: `10.1109/TIP.2003.819861`.

91. JOHNSON, Justin; ALAHI, Alexandre; FEI-FEI, Li. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In: LEIBE, Bastian; MATAS, Jiri; SEBE, Nicu; WELLING, Max (eds.). *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016, pp. 694–711. ISBN 978-3-319-46475-6. Available from DOI: `10.1007/978-3-319-46475-6_43`.

92. SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 2014. Available also from: `https://arxiv.org/abs/1409.1556`.

93. DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255. Available from DOI: `10.1109/CVPR.2009.5206848`.

94. ANTOLÍK, Ján; CAGNOL, Rémy; RÓZSA, Tibor; MONIER, Cyril; FRÉGNAC, Yves; DAVISON, Andrew P. A comprehensive data-driven model of cat primary visual cortex. *BioRxiv*. 2018, p. 416156. Available from DOI: `10.1101/416156`.

95. BRETTE, Romain; GERSTNER, Wulfram. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J Neurophysiol*. 2005, vol. 94, no. 5, pp. 3637–3642. Available from DOI: `10.1152/jn.00686.2005`.

96. ZEILER, Matthew D.; KRISHNAN, Dilip; TAYLOR, Graham W.; FERGUS, Rob. Deconvolutional networks. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2528–2535. Available from DOI: `10.1109/CVPR.2010.5539957`.

97. LOSHCHILOV, Ilya; HUTTER, Frank. Decoupled Weight Decay Regularization. In: *International Conference on Learning Representations*. 2019. Available also from: `https://openreview.net/forum?id=Bkg6RiCqY7`.

98. MORGAN, N.; BOURLARD, H. Generalization and Parameter Estimation in Feedforward Nets: Some Experiments. In: TOURETZKY, D. (ed.). *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1989, vol. 2. Available also from: `https://proceedings.neurips.cc/paper_files/paper/1989/file/63923f49e5241343aa7acb6a06a751e7-Paper.pdf`.

List of attachments

```
 readme.txt ......... description of the structure of the attachment
 src ......... directory containing the source code of the implementation
 thesis ......... directory containing the text of the thesis
   src ......... directory containing the source code of this thesis in the LaTeX format
   thesis.pdf ......... thesis text in the PDF format
```