IT UNIVERSITY OF COPENHAGEN

Master's Thesis

Domain Adaptation of Convolutional Neural Networks in Neuroimaging through EEG Signal Reconstruction

Authors

Mikkel Rahlff Berggreen
mirb@itu.dk

Simon Brage Petersen
sibp@itu.dk

Supervisor

Paolo Burelli
ITU brAIn lab
pabu@itu.dk

Co-supervisor

Laurits Dixen
ITU brAIn lab
ldix@itu.dk

Master of Computer Science
KISPECI1SE

June 3rd, 2024

Abstract

This thesis explores the intersection of neuroimaging and deep learning, through a domain adaptation of convolutional neural networks (CNNs) that investigates their effectiveness in reconstructing EEG signals from visual stimuli. We employ three different model designs for the image-to-EEG task: MLPEEG, a simple MLP acting as a baseline; ResNetEEG, the ResNet-18 architecture modified to predict EEG directly; and ResNetLatentEEG, a novel approach of combining the image feature extraction of ResNet-18 with the decoding capabilities of a temporal convolutional network autoencoder. We evaluate their performance based on the encoding fidelity with three metrics capturing separate relations, and the encoding similarity with representational similarity analysis. The results highlight significant challenges in EEG reconstruction from visual stimuli using our approach, with marginal encoding fidelity and virtually zero correlation in encoding similarity across all models. Due to the poor encoding performance, we cannot conclude exactly how our different model configurations impact the task of reconstructing EEG signals. However, we do not see any indications that the feature extraction of CNNs specifically poses a limitation in this context. The models display the ability to learn from the training data, but are limited by overfitting and lack of generalization. For future work, we stress the importance of further research into metrics that can provide meaningful and interpretable comparisons of EEG signals, and suggest the need for a more systematic approach to model architecture experimentation.

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Paolo Burelli, and co-supervisor, Laurits Dixen, from the ITU brAIn lab. Thank you for your invaluable guidance throughout this project, and your efforts in creating an inclusive and resourceful environment by integrating us with the ITU brAIn lab. We greatly appreciate your expertise in both machine learning and neuroscience, and the time and effort you invested to help us navigate the intersection of these fields - especially as we were initially unfamiliar with the complexities of the brain. We also want to thank the ITU HPC for providing the resources that have enabled the core experimental phases of this project. Finally, we extend our heartfelt thanks to our family and friends for their support and encouragement.

Mikkel Rahlff Berggreen & Simon Brage Petersen

Table of Contents

List of Acronyms and Abbreviations

AE	Autoencoder
CNN	Convolutional Neural Network
DL	Deep Learning
EEG	Electroencephalography
fMRI	Functional Magnetic Resonance Imaging
MACS	Modified Algonauts Challenge Score
MEG	Magnetoencephalography
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
RDM	Representational Dissimilarity Matrix
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RSA	Representational Similarity Analysis
RSVP	Rapid Series Visual Presentation
TCNAE	Temporal Convolutional Network Autoencoder

1 Introduction

The human brain remains one of the final frontiers of human knowledge. Its complex structure and impressive capabilities continue to fascinate scientists, and understanding the intricate processes of the brain is not only a scientific challenge, but a way to advance in numerous fields such as medicine and artificial intelligence. Machine learning, a subfield of artificial intelligence, has taken inspiration from the human brain since its inception, with neural networks in particular aiming to mimic the brain's structure and function. This intersection between artificial and biological intelligence inspires research that attempts to understand and replicate the remarkable capabilities of the human brain.

One of the many fascinating aspects of the brain is its ability to process visual information through complex neural mechanisms. Neuroimaging techniques, such as *Electroencephalography* (EEG), provide valuable insights into these processes by capturing the brain's electrical activity. Machine learning, and especially deep learning, is well suited for analyzing intricate data like neuroimaging data due to its ability to effectively process complex datasets and uncover patterns that are not immediately clear through manual analysis. Among the various deep learning techniques, *Convolutional Neural Networks* (CNN) have shown great proficiency in computer vision tasks. The intersection between neuroimaging and deep learning enables us to explore the underlying neural processes in unprecedented ways.

This thesis follows an experimental approach to adapt deep learning model architectures, like convolutional neural networks, to a new domain of EEG encoding models. We attempt to reconstruct EEG signals from visual stimuli, which – to the best of our knowledge – is a novel area that has not been extensively studied at the time of this study. We aim to explore the feasibility and effectiveness of this approach, contributing to our understanding of the relationship between visual stimuli and neural responses. Specifically, we seek to answer the following research questions:

> *RQ1: How effective are convolutional neural networks in reconstructing EEG responses from perceived images?*
>
> *RQ2: What challenges does EEG as a neuroimaging technique pose for signal reconstruction?*

The thesis will be structured as follows:

1. **Background and Theoretical Foundations:** This chapter covers essential knowledge on the visual system of the brain and neuroimaging techniques, and delves into the concepts of deep learning that are central for this study. Additionally, we offer an overview of the application of deep learning in neuroimaging-based tasks and review related work at the intersection of deep learning and visual processing in the brain.
2. **Proposed method:** Here, we describe the characteristics of the dataset used in our study. We detail our preprocessing approach for both images and EEG signals and justify our choice of evaluation metrics. Furthermore, we present the model architectures we developed and explain our training setup.
3. **Results**: This chapter presents the results achieved following our proposed method. We analyze the performance of our models across multiple metrics and assess the effectiveness of different model configurations.
4. **Discussion:** We discuss the findings of our study and address the limitations of our approach. The discussion points also incorporate proposals for future work, suggesting potential improvements and avenues for further research.
5. **Conclusion:** Based on our research questions, we summarize the key findings of our research.

2 Background and Theoretical Foundations

In this section, we outline and explain the theoretical background for this study. We cover basic concepts related to how the brain processes visual information, types of brain signals, and how brain signals are collected. Next, we explain the deep learning concepts we have used in our experiments and the challenges of performance evaluation that come with time series data. Lastly, we present related work from which our study is inspired.

2.1 The Visual System

Understanding how the visual cortex of the human brain processes information is crucial when dealing with computer vision tasks that rely on brain data. As per Kruger et al. (2012), the visual cortex is commonly divided into three parts: the occipital region, the ventral pathway, and the dorsal pathway. The three parts are visualized in Figure 2.1.

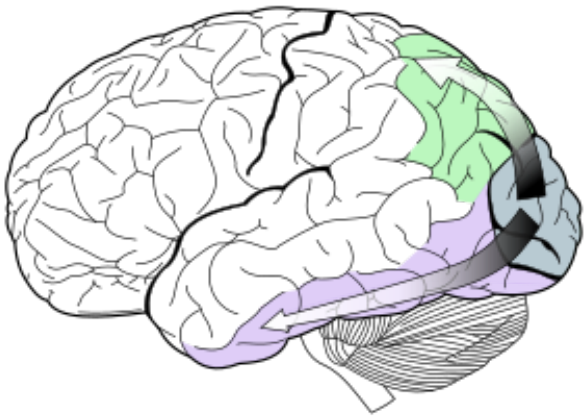


Figure 2.1: Visualization of the human visual cortex. The occipital region is the far right area. The arrow pointing downwards and to the left shows the ventral pathway. The arrow pointing upwards and to the left shows the dorsal pathway. Source: Selket, Wikimedia Commons, licensed under CC BY-SA 3.0.

The processing of visual information starts with *precortical processing* performed in the retina of the eyes and the visual area called LGN. Information is then passed to the occipital region at the back of the head. This region is considered to do generic feature processing with increased complexity depending on the level of hierarchy within the region. Early layers may capture features such as edges, gratings, and disparity, while later layers may capture more sophisticated texture-defined contours, curvatures, or motion indicators. From the occipital region, information is passed in parallel to the ventral and dorsal pathways, with all three parts continuously sharing information. The areas of the ventral pathway are involved in object recognition and categorization, which comes from simple combinations of features or more complex features that correspond to object-level features. The areas of the dorsal pathway are concerned with analyzing space and motion, creating a bridge between the visual and motor systems.

2.2 Neuroimaging

With an idea of the physical mechanisms through which the brain processes information, the next challenge is quantifying the brain's activities into measurable data for further analysis and processing. Following Hans Berger's first human trial in 1924 (Haas, 2003), *electroencephalography* (EEG) has emerged as a fundamental neuroimaging technique for measuring and recording electrical activity in the brain. Renowned for its non-invasive nature, cost-effectiveness, ease of use, and high temporal resolution, EEG provides an exceptional means for capturing the dynamics of neurocognitive processes to study human behavior and cognitive functions (Farnsworth, 2019).

The process of collecting EEG data requires electrodes that are placed at precise locations on the scalp surface, typically adhering to the international standard 10-20 system (visualized in Figure 2.2) or variations of it, such as the 10-10 or 10-5 system (Jasper, 1958; Oostenveld and Praamstra, 2001), to decide on the positions. This system correlates with specific regions of interest in the brain and ensures consistency of electrode positions across multiple studies.

Since the brain is constantly active, the recording of EEG essentially samples data as discrete snapshots over time at a certain *sampling rate*. The sampling rate indicates the number of samples recorded per second, denoted by the unit Hertz (Hz). Each voltage signal detected by the electrodes is amplified and converted into a digital format for analysis. The oscillations in EEG signals are typically classified based on certain frequency bands: delta (0.1 - 4Hz), theta (4 - 8Hz), alpha (8 - 12Hz), beta (13 - 30Hz), or gamma (over 30Hz) – as specified by Gable, Miller and Bernat (2022).

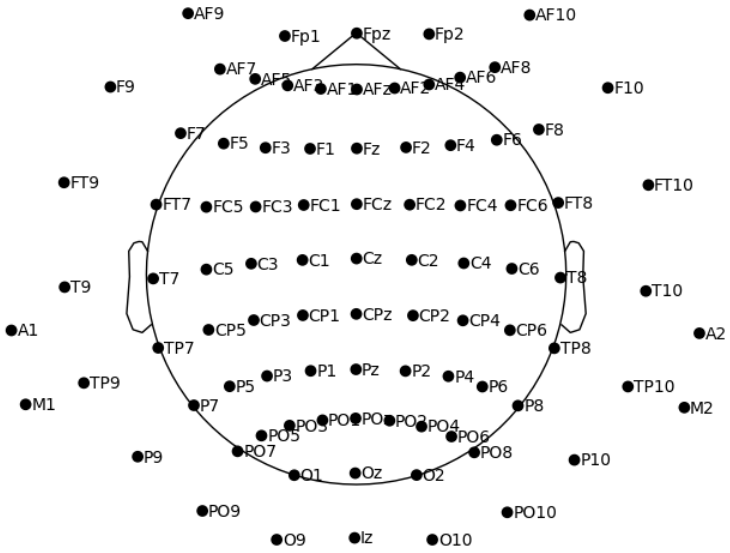


Figure 2.2: 2D representation of the international standard 10-20 system. Each dot represents an electrode that captures EEG signals. The electrodes' names refer to the brain region from which they capture information. In a 3D setting, the electrodes wrap around the scalp of the human subject.

However, before data collection and analysis can begin, it is important to ensure clean data to achieve the best possible signal-to-noise ratio. In practice, this is not an easy task, since electrodes also register electrical activity from external sources and events. These physiological *artifacts* are reflected in the collected data. The most common artifacts when collecting EEG data are muscle activity, which generates electrical currents; eye movements affecting the electrical fields picked up by the electrodes; and blinks, which affect the processing in the cortex (Farnsworth, 2019). Additional noise can also appear in the form of environmental interference or intrinsic fluctuations in brain activity unrelated to the experiment. While noise and artifacts can be minimized and controlled with proper EEG paradigms before or during data collection, it is still necessary to preprocess the collected EEG data to reduce the impact of noise on analytical results.

While EEG is instrumental in providing high-temporal resolution insights into brain dynamics, it is part of a broader spectrum of neuroimaging techniques that enhance our understanding of the brain's functions. Other popular techniques include (Farnsworth, 2019):

1. *Magnetoencephalography* (MEG), which measures the electric fields generated by the electrical activity in the brain. It has a high temporal resolution similar to EEG and is often considered to capture deeper and more nuanced neural activity compared to EEG, but at the cost of versatility and ease of use.

2. *Functional Magnetic Resonance Imaging* (fMRI), which measures changes in blood

flow associated with changes in neural activity. It utilizes the fact that activity in the brain's neurons depends on oxygen, which is delivered by blood - and oxygenated blood has different magnetic properties compared to non-oxygenated blood. Compared to EEG, fMRI has excellent spatial resolutions, but a greatly reduced temporal resolution.

2.3 Deep Learning

The limitations of EEG and other neuroimaging techniques lie in the complexity and volume of the data they generate, which often exceed traditional analytical capabilities. Here, deep learning has emerged as an essential tool for processing raw brain data and providing actionable insights, due to its excellent capabilities in extracting patterns from large, complex datasets. This section will cover some of the basic concepts of deep learning and a selection of popular model types relevant to this study.

2.3.1 Multilayer Perceptron

Multilayer Perceptron (MLP) neural networks are a foundational architecture in deep learning, originally presented by Rosenblatt (1958). As demonstrated by Figure 2.3, an MLP consists of multiple layers of interconnected neurons, where each neuron in a given layer receives input from all neurons in the preceding layer through weighted connections in a feedforward structure. All layers between the visible input and output layers are considered hidden layers. The neurons transform the input with their respective learned parameter weights and biases. This interconnected structure allows for the propagation and transformation of information through the network.

The input layer is the first layer of an MLP, where the initial data is fed into the network. This data can be of various forms such as numerical values, pixel intensities for images, word embeddings for text, etc. Each input neuron corresponds to one feature of the input data, and the information is passed to the subsequent hidden layers of the network. For example, in an image recognition task, each input neuron corresponds to one pixel value of the input image.

The output layer is the final layer of an MLP, where the transformed data from the last hidden layer is used for the model predictions. The output layer depends on the specific task of the MLP. For example, in a classification task, the output layer could consist of neurons corresponding to the number of classes, where each neuron represents the probability of the input belonging to a specific class. In a regression task, the output layer could consist of a single neuron that produces a continuous value prediction.

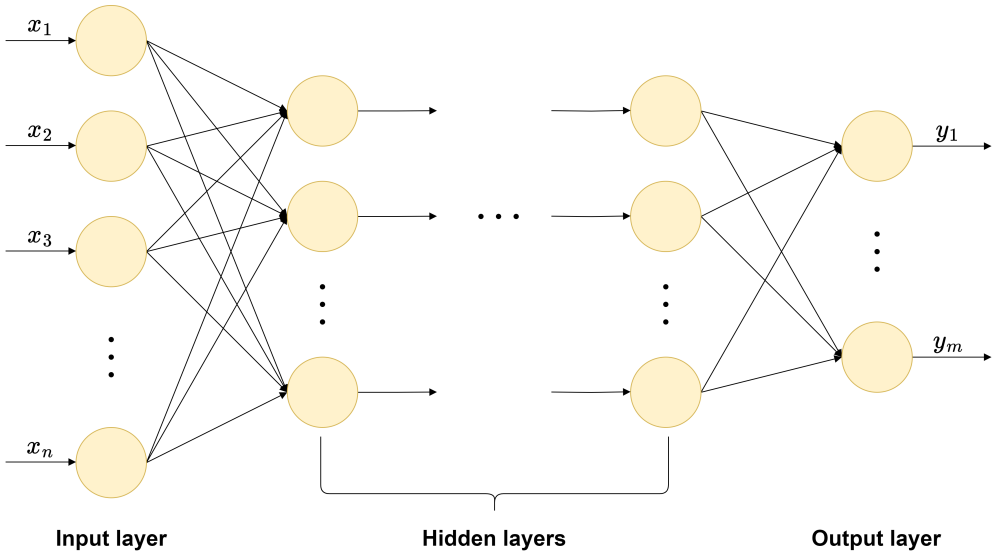


Figure 2.3: Vertices represent neurons, and edges represent the weighted connections between them. The vectorized data in each neuron of the input layer (left) is passed to every neuron in the first hidden layer. Each neuron in the hidden layers is fully connected to every neuron in the subsequent layer, continuing this pattern toward the output layer.

Within each neuron, non-linear transformations, called activation functions, are applied to the input. These functions introduce non-linearity throughout the network, enabling it to model complex relationships within the training data more effectively. Many different activation functions exist, with examples being *rectified linear unit* (ReLU), sigmoid, and tanh, each providing unique properties for different tasks (Goodfellow, Bengio and Courville, 2016). In tasks such as image recognition, where the underlying patterns are often highly non-linear, this inclusion of non-linearity within the fully connected architecture is essential to the effectiveness of the model capturing the intricate patterns.

The transformation within a neuron can be mathematically expressed as the function in Equation (2.1), where z is the neuron output from using activation function h over the weighted input a:

$$z = h(a) \tag{2.1}$$

In this, the weighted input, a, is defined by Equation (2.2), where the total length of the neuron input is represented by D, w represents the input weights, x the input values, and b the bias.

$$a = \sum_{i=1}^{D} w_i x_i + b \tag{2.2}$$

2.3.2 Training Deep Learning Models

The training of many supervised neural networks relies on the backpropagation algorithm (Rumelhart, G. E. Hinton and Williams, 1986) to minimize the difference between the network's predictions and the target results, specified as a cost function. Supervised learning specifically refers to the process of having expected output targets for the inputs. Through a backward iterative process from the last layer in the network, the algorithm repeatedly adjusts the parameter weights and biases of the neuron connections in the network. In short, these adjustments are done by computing the gradient of the cost function, one layer at a time, with respect to each model parameter and optimizing their values through various optimization strategies, like gradient descent.

When training neural networks, numerous *hyperparameters* need to be considered. These parameters are external to the model and defined by the developer to control the learning process. Hyperparameters range from the overall architecture of the model to specific settings, such as the number of iterations, or epochs, for which a model is trained. The architectural design of a model can vary across many dimensions, such as its network structure and how its layers are configured. This encompasses decisions like the number of hidden layers and the number of nodes within each layer. The choice of activation functions within the nodes also influences the model's ability to learn complex patterns from data.

Learning rate (LR) is a hyperparameter that controls the magnitude of changes made to the model parameters during the optimization process. High learning rates can lead to faster and sometimes more optimal convergence by avoiding local minima through larger steps in the optimization space. However, it also poses the risk of instability and oscillating behavior when searching for the optimal solution. Conversely, low learning rates can improve stability during training, but may unnecessarily slow down convergence time or get stuck in a local minima in the optimization space.

During model training, the training data is typically divided into batches, which are then processed by the model. Batch size is another common hyperparameter, specifying the number of data samples processed by the model before a parameter optimization is performed. Depending on the size, the amount of noise in the optimization process can vary, affecting both the performance and computational speed of the model. Smaller batches are more sensitive to noise due to fewer data points, leading to more fluctuations in the optimization. Conversely, larger batches can result in a smoother and more stable optimization.

The choice of optimization strategy can also greatly impact model performance. Different optimization algorithms, like Adam (Kingma and Ba, 2014) or stochastic gradient descent (SGD) (Robbins and Monro, 1951), update the model parameters through various means, for

example with adaptive learning rates specific to each parameter, or by making more frequent updates to the parameters using smaller batches of data. A better choice of optimization strategy is highly dependent on the task at hand and is influenced by things like the model architecture, complexity of the data, and other hyperparameter configurations.

The overwhelming amount of different hyperparameters can make it difficult to decide on the optimal settings for a model's performance. Systematic procedures, like random or grid search, can assist with comparing a model's behavior across different hyperparameter configurations. With random search, hyperparameter combinations are randomly sampled from a predefined distribution. This approach helps identify a general understanding of which hyperparameters impact the model performance the most, by exploring a wide combination space. Grid search offers a fully systematic approach, where the model performance is evaluated thoroughly across all hyperparameter combinations. While this approach can guarantee finding an optimal setting, it also becomes computationally expensive when the number of hyperparameter choices and their distribution sizes increase. For this reason, random search is often the preferred choice of model architects, because you rarely have a complete idea of what the ideal configuration is.

Fine-tuning refers to the process of adapting a pretrained model to a new, specific task or dataset. A model already trained to have optimal parameter weights for a general task can be saved and have its parameters further adjusted later on to better suit the characteristics of e.g. a new input dataset or a more narrow task output. The expectation of fine-tuning a pretrained model is to achieve better performance on a specific task.

2.3.3 Generalization

A common risk when training neural networks is that the network learns to memorize the specific patterns and noise of the data used for training, rather than the general underlying patterns of the data as a whole. Ideally, the network's performance on the training data should translate to unseen data. *Generalization* is a neural network's ability to estimate unseen data beyond the data used during training (Goodfellow, Bengio and Courville, 2016). When a model generalizes too much and is unable to capture any relevant relations within the data, it is underfitting to the task. This is reflected in the model estimations having a high bias and low variance, with bias being the difference between model estimations and target values, and variance being the variability of estimations. Conversely, when a model lacks generalization, it is overfitting to the task. This is reflected in the model estimations having a low bias, but also a high variance.

Regularization is the general term used for methods that combat overfitting (Goodfellow, Bengio and Courville, 2016). Regularization aims to improve the generalization of a model by

limiting the model's complexity and constraining its optimization. Such constraints involve methods like L1 and L2 regularization, early stopping, and dropout, all of which aim to improve model generalization in different ways:

- $L1$ and $L2$ regularization modify the model cost function with a penalty term (Goodfellow, Bengio and Courville, 2016). The latter, $L2$, is more commonly known as *weight decay* (WD), and is proportional to the square of the absolute values of the model weights. Including such a penalty encourages smaller weights, simplifying the model, and making it more robust to noise and irrelevant features of the data.
- Early stopping is a method used to stop model training when its performance on a validation set has not improved over a set period (Goodfellow, Bengio and Courville, 2016). The approach prevents further unnecessary training of a model as it starts to memorize the statistical noise of the training data.
- Dropout is a regularization technique specific to neural networks (Srivastava et al., 2014). During model training, randomly selected neurons can be ignored, or dropped, with a specified probability. If a network relies heavily on certain individual neurons, introducing dropout layers could force it to make more robust estimations.

To assist in estimating a model's ability to generalize, a commonly used method is to divide the dataset into two subsets, a training set and a testing set. The training set is used for training the model, whereas the test set is used for evaluating the performance of the trained model. Optionally, a validation set can also be used to evaluate the performance of a model during training.

2.3.4 Convolutional Neural Networks

The *Convolutional Neural Network* (CNN) is a subclass of deep neural networks (LeCun, Bengio and G. Hinton, 2015). They are inspired by the mechanisms of the visual cortex (Gu et al., 2018) and excel at capturing image features at different abstraction levels, making them a powerful tool for computer vision tasks such as image classification and object detection.

The core concept of CNNs is the convolution operation, which involves applying a *kernel* to the input data to extract features at various spatial levels for the output. In the context of single-channel images, such as grayscale images, the convolution operation can be mathematically represented as shown in Equation (2.3), where I is the input image, K is the kernel, (m, n) are the spatial indices, and $\star$ denotes the convolution operator[1].

[1]Technically the cross-correlation operator. Many machine learning libraries, such as PyTorch (Paszke et al., 2019), use cross-correlation interchangeably with convolution.

$$O(i,j) = (I \star K)(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n) \tag{2.3}$$

In this equation, the input image I is convolved with the kernel K, producing an output feature map O, where M and N are the dimensions of the kernel. This operation slides the kernel across the single-channel input image, computing the sum of element-wise multiplications at each position, thus extracting local features. This 2D convolution operation is visualized in Figure 2.4. For multi-channel images, such as RGB images, the convolution operation extends this process by applying a set of kernels to each input channel – also known as *filters* – resulting in a multi-channel output. Each output channel is computed by summing the convolved results from all input channels and adding a bias.

The trainable parameters of a CNN make up the convolutional filters. These filters can vary in size, and their specific values are iteratively computed and updated during model training through backpropagation. The outputs of the convolution process are feature maps, which retain important information from the original image in a typically smaller representation. These feature maps, which are progressively reduced in size from pooling and downsampling processes, serve as input for subsequent convolutional layers in the network. From this progressive feature extraction, along with nonlinear activation functions, CNNs can learn complex patterns and representations from raw image data.

Pooling is a fundamental process of CNNs to reduce the dimensions of the feature maps. This leads to fewer connections in the network resulting in fewer parameter weights to learn when training the network. Decreasing the complexity of the network in this way helps with reducing its computational complexity and the control of overfitting. The pooling operation functions similarly to convolutional filters, but differs in its computation of output, not having trainable values tied to each filter index. Instead, three types of pooling exist: average pooling, max/min pooling, and global pooling. Each pooling operation performs a different reduction operation, by outputting the average value, the largest/smallest value, or the global average of the input, respectively.

Typically, the final feature map outputs of a CNN are connected to linear, or fully connected layers that match the desired output format of the task being solved. The feature maps are flattened into a one-dimensional space, preserving the spatial structure of the features, and can then be connected to one or more of these layers.

ResNet

Designing and training novel computer vision models requires large amounts of data, which might not be available or computationally feasible. Numerous pretrained convolutional

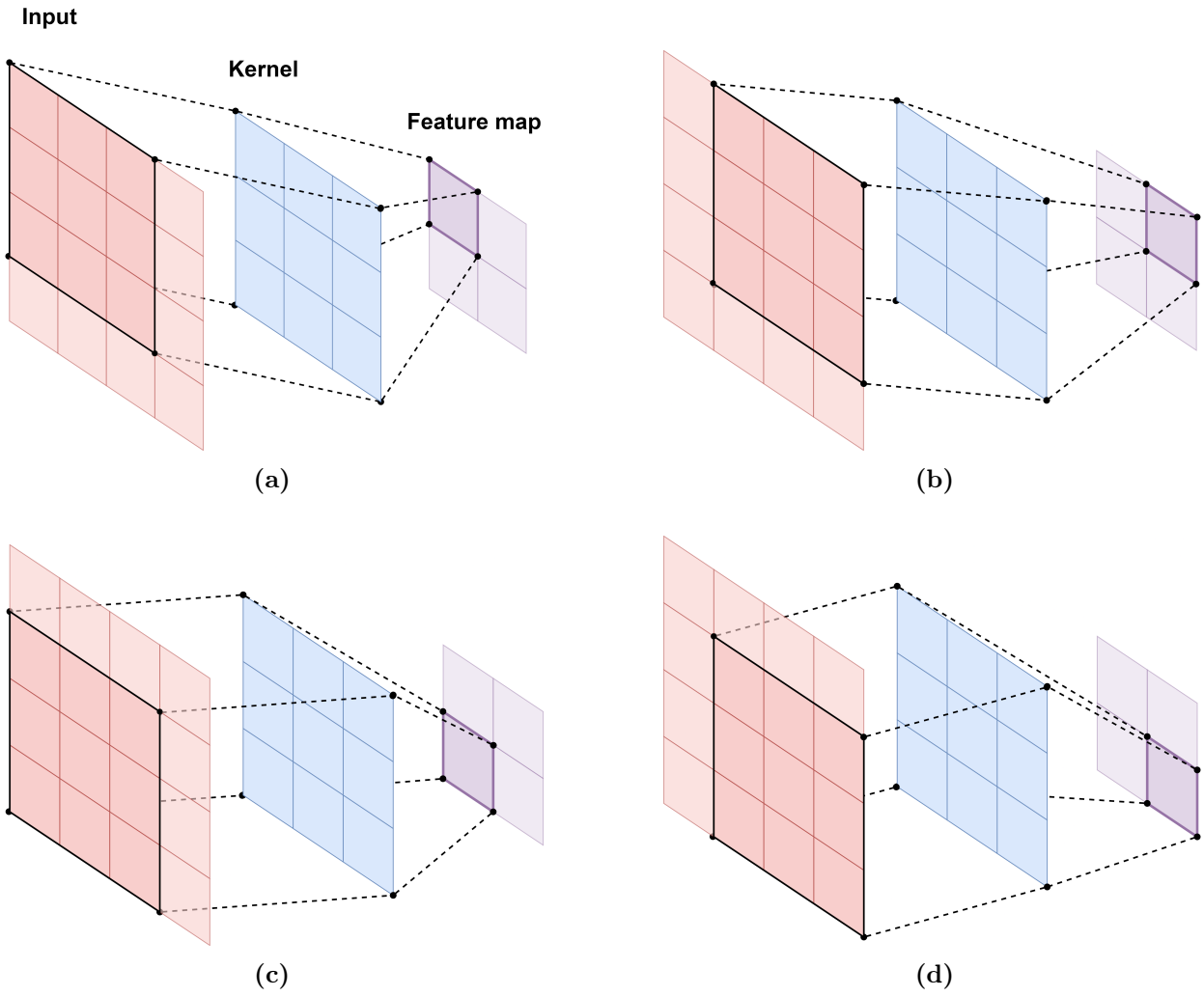


Figure 2.4: Visualization of a simple 2D convolution operation with a stride of 1. The red matrix is the input, the blue matrix is the filter, and the purple matrix is the feature map output. In the order of a), b), c), and d), a 3x3 kernel "moves" over different segments of the input matrix. The kernel applied to the highlighted input values produces the corresponding values in the resulting 2x2 output.

neural networks have been developed and proven effective for various computer vision tasks. By leveraging these pretrained models, researchers and developers can achieve high performance without the need to build a model from scratch, making it a practical and efficient approach for many applications.

ResNet (K. He et al., 2016) is one example of an existing and popular CNN architecture used for computer vision tasks. The architecture is a residual network consisting of modules made up of residual blocks with convolutional layers. These blocks aim to combat the degrading problem of deep neural networks, where training accuracy degrades with an increase in neural network depth. In traditional neural network architectures, each layer is responsible for learning a direct mapping from its input to its output. Residual blocks function differently by using skip connections or shortcut connections. Instead of directly learning an underlying mapping from the model input to the desired output, each layer learns a residual mapping, essentially finding the difference between its specific input and output. This residual mapping is often easier to optimize, allowing overall better performance for the model. For each layer, the residuals are added back to the original input, effectively creating shortcut connections throughout the architecture. The process maintains important features from the original input throughout the model training, mitigating the loss of information that may occur in deep neural networks. ResNet-18, as visualized by Figure 2.5, is one example of this architecture, consisting of an initial convolutional layer, followed by four modules each consisting of two residual blocks with convolutional layers, resulting in 17 total convolutional layers that feed into the final linear layer. More complex ResNet models exist, that follow the architecture of using residual blocks, but increase the number of blocks within each module.

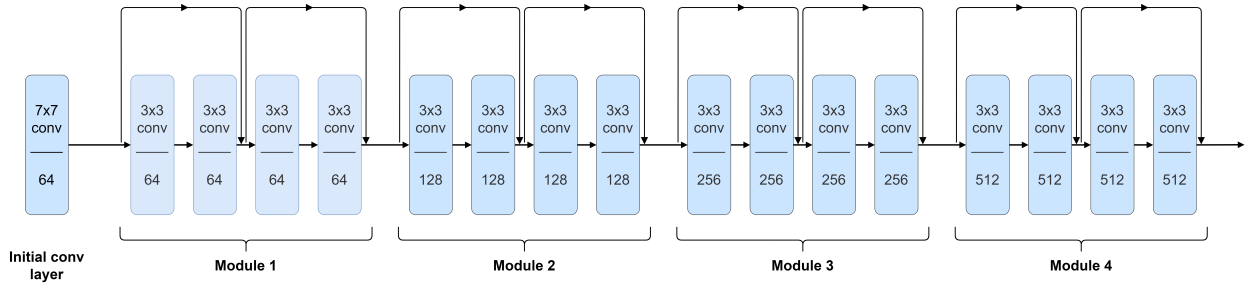


Figure 2.5: Overview of the residual block architecture of ResNet-18. Each convolutional layer is denoted with the filter size (top) and the number of feature maps output by the layer (bottom). The different colors mark the four separate residual modules, each containing four residual blocks. The vertical rounded arrows showcase the skip connections.

In the context of EEG analysis, when studying EEG responses over time, we hypothesize that specific hierarchical layers of a CNN might be better at reconstructing different time intervals of an EEG response time series. Based on knowledge and interpretations of the typical hierarchical structure of CNNs, the network's understanding of the inputs evolves

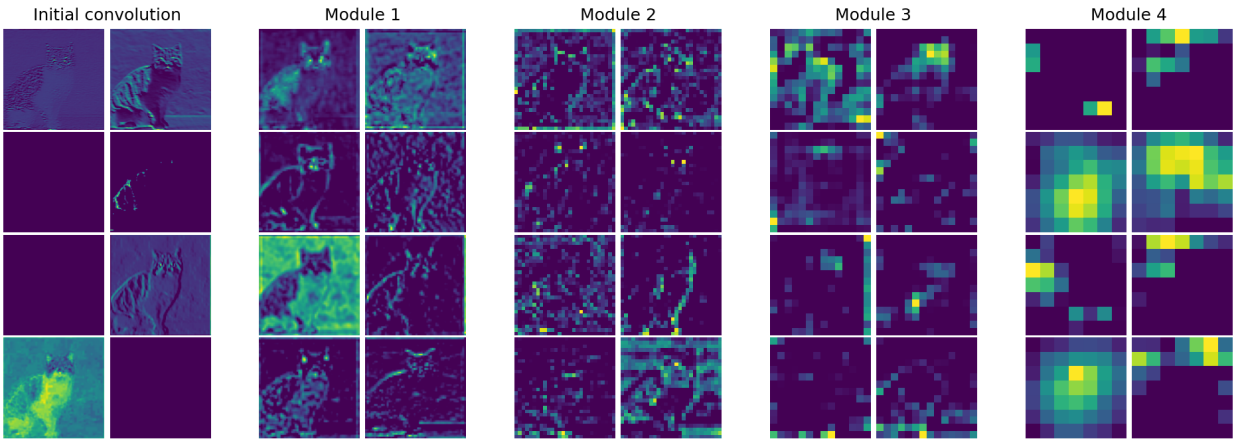


Figure 2.6: A selection of resulting feature maps from passing the `cat_03s.jpg` image from the THINGS-EEG dataset to a pretrained ResNet-18 model, fine-tuned on data of a single subject from the same dataset. From left to right, the sections of feature maps are example outputs from the initial convolutional layer and the additional four modules of the ResNet-18 architecture.

throughout the layers (Olah, Mordvintsev and Schubert, 2017; Olah, Satyanarayan et al., 2018). Earlier layers tend to focus on lower-level features like edge detection, progressively followed by layers focusing on more sophisticated shapes, textures, and patterns, ending with layers understanding high-level features like full object concepts. This hierarchical structure closely resembles how the different areas of the visual cortex function. Neurons in the early visual cortex tend to respond strongly to lower-level features (Goodfellow, Bengio and Courville, 2016). In later layers of the visual cortex, some neurons respond to very specific concepts and are invariant to whichever format, orientation, size, etc., the visual stimuli were presented in.

Figure 2.6 shows example output feature maps of a ResNet-18 model trained for image classification from its initial convolutional layer through each of its four internal modules.

2.3.5 Autoencoders

Autoencoders are a type of neural network that focuses on encoding input data into a reduced dimensional representation and then reconstructing the input from that representation (Goodfellow, Bengio and Courville, 2016). This method can learn a more compact representation of complex data, capturing the latent variables of the data. These variables, or features, are hidden within the data, not being directly observable, and shape the way the data is distributed. The hidden features collectively form the latent space representation. Simply put, the architecture of an autoencoder consists of an encoder and decoder part as visualized in Figure 2.7.

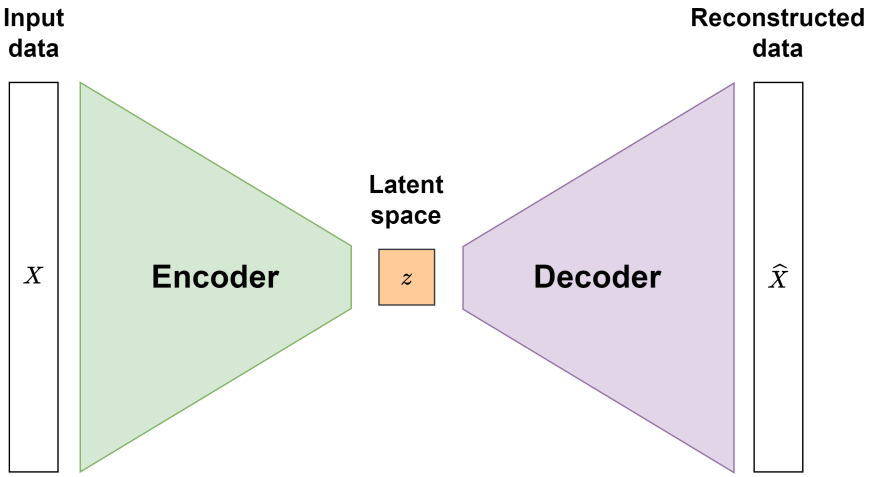


Figure 2.7: Simplified overview of the general autoencoder architecture. The encoder and decoder are both typically deep learning architectures. The input X is encoded into a smaller latent space, z, from which the decoder learns to reconstruct the input, $\hat{X}$. The performance is measured by the reconstruction loss between X and $\hat{X}$. The encoder and decoder are separate (but often similar) neural network architectures.

The encoder's purpose is to map the input data to a lower-dimensional latent space representation. This mapping comprises a series of transformations through standard fully connected or convolutional layers. The encoder aims to compress the input to a specified latent space size, which captures the most prominent features useful for reconstructing the original input.

The decoder's purpose is to reconstruct the original input data from the resulting latent space representation of the encoder. Similar to the encoder, the reconstruction is carried out through transformations in neural network layers. The decoder aims to estimate outputs that resemble the input data as accurately as possible, minimizing the reconstruction error.

2.4 Evaluating Performance

When dealing with data that has high temporal sensitivity, like EEG, it can be challenging to determine the similarity or distance between two samples. A significant challenge is the potential for time shifts in the data, which occur when external disturbances or latency affect the timing of signal recording, thus shifting the data points in time. In these situations, traditional distance metrics like Euclidean distance or the *mean squared error* (MSE) and *root mean squared error* (RMSE) are often inadequate. MSE can be mathematically expressed as shown in Equation (2.4), where N is the total number of samples, X denotes the observed values, and $\hat{X}$ is the predicted values.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2 \tag{2.4}$$

Additionally, RMSE can be mathematically expressed as shown in Equation (2.5) and extends MSE by taking the square root of this relationship to place the resulting error back in the same order of magnitude as the samples.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2} \tag{2.5}$$

Due to the nature of these metrics, they only measure distances between data points that appear temporally aligned in the recordings, and fail to account for delays or latency that may have caused these points to be misaligned at the actual time of occurrence.

2.4.1 Addressing Time Shifts in Time Series Comparisons

To ensure a sensible comparison between two different time series, we need a metric that accounts for the temporality in EEG data. This makes it possible to quantify how close the predicted values of EEG signals are to the ground truth in terms of both signal strength and temporality. An example of such a metric is the *Wasserstein distance*, based on computational optimal transport theory, which seeks to minimize the 'effort' or 'cost' of solving a task (Peyré and Cuturi, 2020). In the context of EEG comparison, it could provide a robust framework for comparing temporal structures in the data by quantifying the minimal 'cost' of transforming one distribution into another. With this approach, the Wasserstein distance inherently accounts for temporal shifts between the time series, if any.

Despite the appealing characteristics, the downside to the Wasserstein distance – and optimal transport algorithms in general – is their computational demand. This disfavors the application of Wasserstein distance in deep learning, as deep learning tasks often rely on large amounts of data. However, *Sinkhorn divergences*, outlined by Feydy et al. (2019) – referred to as *Sinkhorn* in this thesis – provide an affordable approximation of the Wasserstein distance, which allows for application in deep learning tasks. Due to the Sinkhorn divergence's mathematical complexity, we do not delve into its mathematical definition and use it only for what it represents conceptually.

2.4.2 Representational Similarity Analysis

Representational Similarity Analysis (RSA), initially presented by Kriegeskorte, Mur and Bandettini (2008), is a measure that abstracts from the raw activity patterns of brain responses, to instead focus on the dissimilarities in how the brain or a computational model represents the information. The approach involves computing representational dissimilarity matrices (RDMs), finding a shared representation for the predicted brain signals and the ground truth to assess whether an encoding model captures similar relationships between representations of stimuli, compared to the neural representations.

The process of computing RDMs – also visualized in Figure 2.8 – involves the following steps:

1. All stimuli are processed, either by the brain with the neural activity being recorded, or by a model that predicts neural activity. The outputs are the activity patterns.
2. For each pair of resulting activity patterns, the dissimilarity between them is computed. The dissimilarity is given by 1 minus the correlation. The choice of correlation metric depends on the situation - examples could be linear correlation, euclidean distance, or other distance metrics.
3. The computed dissimilarities are used to construct the RDM. As a result, the RDM is always symmetric across the diagonal, where all dissimilarity values are zero.

This process is carried out for encodings achieved by both the brain and the respective model. The two resulting RDMs exist in the shared representation between the model and brain encodings and can now be compared directly.

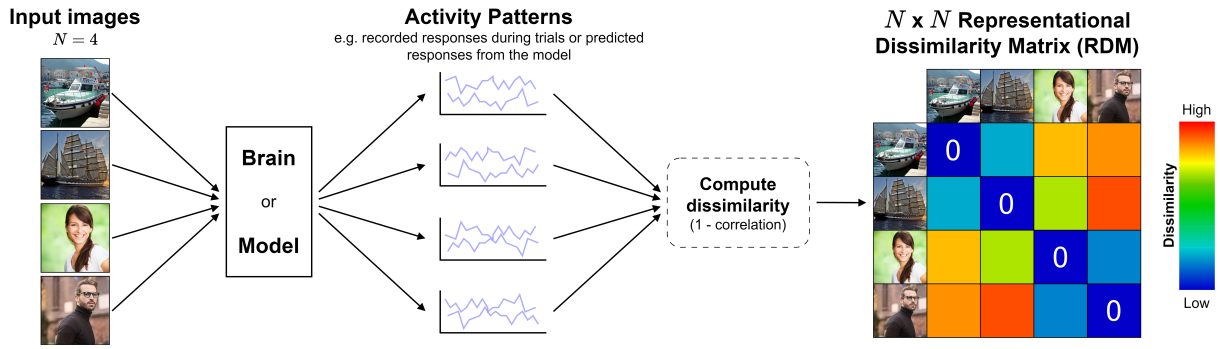


Figure 2.8: Visualization of RDM. The resulting encoding from either a brain or model is compared for each pair of stimuli. The results are assembled in a dissimilarity matrix, where each cell compares the response patterns between the two corresponding images. Here, the diagonal is all zero since a dissimilarity between an encoding and itself is always zero. This process is repeated for both the brain and model for the resulting RDMs to be compared. *NB: the RDM pictured here does not represent actual dissimilarity values.*

2.5 Deep Learning for Neuroimaging

The applications of neuroimaging techniques are many, and thanks to the rapid progress in deep learning, automatic analysis has helped make their application more practical – largely because it reduces the need for trained professionals to extract useful information from the signals. As a result, the interplay between deep learning and neuroimaging has fostered a wide range of studies that add to our understanding of human behavior. The most popular studies focus largely on feature extraction from the brain data and use the features for tasks in emotion recognition, motor imagery, mental workload, seizure detection, diagnosis of neurological disorders, and event-related potential detection (Craik, Y. He and Contreras-Vidal, 2019) – with the latter also relating to describing visual processing in the brain through image classification and -reconstruction. While some methodologies in neuroimaging are more popular than others, the studies of these domains present various approaches to their data collection, data processing, and model architectures. The approaches naturally differ depending on the neuroimaging technique, though, where applicable, we focus on EEG over other neuroimaging techniques in this section. Prevalent for many neuroimaging machine learning tasks is the use of deep learning model architectures to solve the task – with CNNs as one of the leading architectures in EEG-based tasks (Craik, Y. He and Contreras-Vidal, 2019; Roy et al., 2019).

When collecting neuroimaging data, there are many parameters to consider. The amount of data and its structure varies greatly across domains, from all-night recordings in sleep studies to recordings that last a few hours or even minutes. Further variability in data collection is introduced by the number of subjects and the choice of neuroimaging technique, sampling methodology, and sampling rate, to name a few.

W. Lin, Li and Sun (2017) introduces a multi-model data collection by combining data from 32 EEG channels with physiological data such as respiration and skin temperature for emotion classification. During recording the subject watches short video clips, carefully selected to invoke specific emotions, and reports their perceived emotion in a questionnaire after each video. For image classification or -reconstruction tasks, a recording session typically spans a few hours, with the subject being exposed to visual stimuli during this time. The most common type of stimuli is image objects/concepts, with examples being the THINGS dataset presented in Martin N. Hebart et al. (2019) and the ImageNet dataset by Deng et al. (2009).

For recording neuroimaging data, Kumar et al. (2018) collects EEG data across 14 channels from 27 subjects. The subjects imagine a shown stimulus with their eyes closed for 10 seconds followed by a 20-second break before the next stimulus is presented. The long break between

stimuli is intended as a "cortical pallet cleanser", letting the subject return to a resting state to minimize the impacts of prior stimuli on the next. On the opposite end of the spectrum, the *Rapid Series Visual Presentation* (RSVP) is another interesting approach to recording neuroimaging data. Grootswagers et al. (2022) adopt this approach, by creating sequences of 10Hz, i.e. 10 stimuli shown per second. Each stimulus is visible for 50ms followed by a 50ms blank screen, creating rapid sequences of images from which the brain responses are recorded.

The approaches to preprocessing vary widely between studies because it also depends on the task at hand. Pereira et al. (2018) apply minimal manual preprocessing by filtering the raw EEG data before turning it into discretized "images" well-suited for their CNN approach. The argument for not preprocessing more aggressively here is that CNNs are powerful feature extractors and that any relevant features should be extracted by the model, if present in the signal. Similarly, Kuanar et al. (2018) manually excludes entire EEG samples from subjects with a large presence of noise and artifacts before applying filtering to remove unwanted signals. In a more automated preprocessing approach, Yin and Zhang (2017) uses Independent Component Analysis (ICA) to identify components for artifacts related to eye movements in the EEG data and subsequently remove these artifacts. Other studies, such as Bashivan et al. (2015), report no preprocessing steps for mitigating noise and artifacts in their neuroimaging data.

As mentioned, deep learning architectures are widely adopted approaches to EEG feature extraction across many domains in neuroimaging tasks. As an example, studies such as Yanagimoto and Sugimoto (2016), Salama et al. (2018), and the previously highlighted W. Lin, Li and Sun (2017) all take a CNN approach in extracting features for their emotion recognition/classification task on the DEAP dataset (Koelstra et al., 2011). However, other deep learning architectures are also used in similar tasks, such as the MLP approach seen in Al-Nafjan et al. (2017) or the Long-Term Short Memory (LSTM) approach – a model specialized for sequential data – seen in Alhagry, Fahmy and El-Khoribi (2017).

However, with the purpose of this paper in mind, we focus on studies centered on image classification and reconstruction from neuroimaging data—a task directly related to understanding how the brain represents visual information. Since research into this problem uses similar neuroimaging data to the studies highlighted above, the approaches in data collection, preprocessing, and model architectures are closely related to other domains. Nevertheless, it is crucial to examine the differences in results arising from using different neuroimaging techniques—EEG, MEG, and fMRI—as these differences significantly impact the outcomes and insights across all domains.

With EEG specifically, Tirupattur et al. (2018) experiment with two different CNNs, one

LSTM, and a hybrid between them for their feature extraction process - with one of the CNNs being the best-performing architecture. The extracted features from the EEG are used to predict the class label or passed on to a Generative Adversarial Network (GAN); a model responsible for generating images in a process that involves convincing a discriminator that the generated image is real. After generating images, they classify the generated images using a separate image classifier, which reports roughly 97% accuracy in image classification and reconstruction. However, the limitations of the reconstructions become apparent when comparing them to the original image. Here, it becomes evident that the model mostly captures high-level features but not many low-level features, allowing it to generate an image matching the target class, not an image that looks like the target image. Singh et al. (2023) expand on these results by swapping the CNN for an LSTM, improving the feature extraction from the EEG. However, the same limitations in image reconstruction are seen here, with the generated images only reflecting high-level features.

In a similar task, but using MEG instead of EEG, Benchetrit, Banville and King (2023) attempts to reconstruct images using streams of MEG data from the THINGS-MEG dataset (Martin N Hebart et al., 2023). They adopt a CNN approach described in Défossez et al. (2023), whose results suggest that the difference in decoding performance observed between similar studies is mainly driven by the type of brain imaging technique. Benchetrit, Banville and King (2023) modify the approach by changing the decoding method to employ a diffusion model for image generation – a model that uses probabilistic processes to generate images. By reconstructing from a stream of MEG signals, they can inspect image reconstruction from different interval sizes relative to stimulus onset. Generally, larger intervals showcase better reconstruction abilities. As with EEG, the limitations of preserving low-level features in reconstruction are also evident in their results.

Conversely, Shen, Dwivedi et al. (2019) and Shen, Horikawa et al. (2019) attempt image reconstruction from fMRI data using CNNs and report a significant presence of low-level features in the reconstructed images. The training is performed on image objects, while testing is done on image objects and artificial shapes and characters. When tested on shapes and characters, the models showcase a high correlation across color and shape. However, the increase in low-level features comes at the cost of high-level features. As a result, the reconstructions of image objects are not clearly defined or distinguishable regarding their respective high-level class.

In contrast to the continuous progress of studies focused on reconstructing images from brain signals, less effort has been put into predicting brain responses to visual stimuli. This reversed process – resembling more of a regression task with continuous estimations – would encode images into predicted brain signals instead of decoding brain signals into images. Such an

approach could potentially allow for insights into and analysis of the processing happening in the brain's visual cortex and how it relates to certain image features. To the best of our knowledge, no studies have documented an approach to investigate this reverse relationship for EEG specifically. However, studies with a focus on explaining MEG and fMRI activity from images exist. *The Algonauts Project* - launched in 2019 - is on a mission to bring the fields of biological and machine intelligence together on a single platform. With annual challenges they allow researchers and enthusiasts alike to try and solve complex problems at the intersection of these fields.

The inaugural challenge (Cichy et al., 2019) constituted two separate tracks: one focused on fMRI and one focused on MEG. For both tracks, the goal was to predict responses early and late in the visual cortex relative to stimuli onset. The fMRI track focused on a spatial analysis of two regions related to early and late responses respectively. In comparison, the MEG track focused on a temporal analysis of two windows in time referring to the expected peak activity of those two regions. The performance of the various approaches is measured by mapping outputs and data to a common similarity space using RSA. Thus, this challenge is less focused on the raw reconstructed signals and more focused on finding an embedding of the outputs that mimic the brain. Challenge solutions such as those presented by Meijer and Visser (2019) and Gaziv (2019) utilize existing convolutional neural network architectures – or modifications of them – to find valuable feature representations from the images and reconstruct the brain signal from those features.

The latest edition of the challenge, *The Algonauts Project 2023: How the Human Brain Makes Sense of Natural Scenes* (Gifford et al., 2023), focused solely on predicting fMRI data from complex natural scenes in the Natural Scenes Dataset (NSD) (Allen et al., 2022). The challenge score reflects a distance between predicted brain signals and the ground truth, thus placing more focus on the actual values in the reconstructions as opposed to the first challenge's focus on representational similarity between the models and the brain. The top three ranked submissions highlight the success of reconstructing fMRI from images with scores of 70.8, 63.5, and 61.6 respectively – where a score of 100 would indicate perfect predictive capabilities on the test set.

The best-performing submission, presented by H. Yang, Gee and Shi (2023), approaches the task with transformer models and introduces memory-related information as input, which is a compressed embedding of the images shown before the current image. Interestingly, they discovered delayed brain responses correlating with the 6th or 7th prior image(s) but not the immediate past images.

The runner-up, Adeli, Minni and Kriegeskorte (2023), uses a transformer encoder-decoder architecture to map images to fMRI responses. They tailor their fMRI response predictions

to specific brain regions by adjusting their model to predict different regions of interest based on different abstraction levels in the encoder. They hypothesize that some regions of interest are more accurately predicted with visual input from different levels of abstraction in the encoder - which they highlight to be the case for early visual regions and lower abstraction levels in the encoder.

In third place, Nguyen et al. (2023) present an approach using a CNN in two stages - a pretraining stage and a fine-tuning stage. The initial pretraining stage uses data from all subjects. Here, an embedding is designed for the subject ID into subject features which is concatenated with the features extracted from the images by the CNN. The resulting features are passed to two independent blocks of fully connected and batch normalization layers, which predict the left and right hemispheres of the brain respectively. Fine-tuning is done by taking the CNN from the pretraining stage and training it on individual subjects to predict multiple segments of the brain data, e.g. full signal, regions, and individual vertices. The average response over all segments is used as the final prediction.

3 Proposed Method

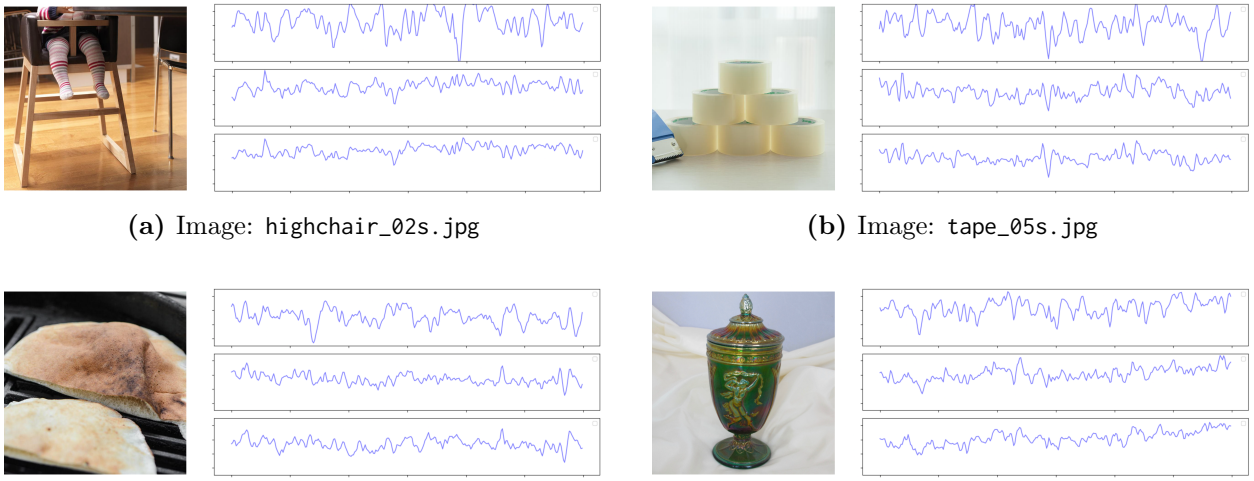
This section details the methodologies and approaches explored in this thesis. First, we describe the dataset, preprocessing pipeline, and performance metrics. We then describe our three model designs, MLPEEG, ResNetEEG, and ResNetLatentEEG, alongside the rationales behind their design. Next, we describe an approach using different CNN layer outputs for EEG reconstruction, before finally outlining the specific configurations used during experiments.

3.1 The THINGS-EEG Dataset

This study uses data from the THINGS-EEG dataset (Grootswagers et al., 2022); a high-quality dataset containing EEG responses from 50 human subjects. The responses are recorded in rapid serial visual presentation (RSVP) streams of 22,248 RGB images distributed across 1,854 object concepts from the THINGS concepts and images database (Martin N. Hebart et al., 2019), i.e. 12 images per concept. The image sequences are presented in a 50% duty cycle, where each image is shown for 50ms, followed by a blank screen for 50ms. A fixation point is shown at the center of all images to help maintain fixation. Figure 3.1 shows example images with corresponding EEG responses.

The responses are captured using 64 electrodes (channels), with electrode positions adhering to the 10-10 system (one of the modifications of the international standard 10-20 system), and with a sample rate of 1000Hz. The THINGS dataset contains data ranging from -100ms to 1000ms relative to each stimulus onset for all 50 subjects.

We reduce the amount of data used in our experiments by focusing on a single subject and selecting different subsets of channels in the brain's visual cortex between experiments - mostly focused on channels covering the occipital region and ventral pathway. By concentrating on data from one subject, we ensure consistency in individual EEG response patterns and eliminate the possibility of inter-subject variability. Experimenting with different subsets of channels allows for studying the various processing stages in the visual cortex. Additionally, we only use data points from EEG recordings that lie in the range of 0 to 300ms after stimulus onset. This decision is based on the time it takes for the visual cortex to process the stimuli after exposure (Kruger et al., 2012).



(a) Image: `highchair_02s.jpg` **(b)** Image: `tape_05s.jpg`

(c) Image: `pita_09s.jpg` **(d)** Image: `urn_02s.jpg`

Figure 3.1: Examples of image-EEG pairs from the THINGS-EEG dataset. The corresponding raw EEG responses across the three channels {`O1, Oz, O2`} are shown for each image. The EEG responses range from -50μV to 30μV in the interval of 0-300 milliseconds after stimulus onset.

3.2 Preprocessing

Preprocessing is a vital step in all machine learning tasks and can be the difference between a model that trains effectively and performs well and a model that produces no results. Reasons for preprocessing include ensuring consistency in the input, reducing noise, or normalizing the data into meaningful representations. Choosing optimal preprocessing for neuroimaging tasks is not trivial – so for each type of preprocessing used in this study, we will argue for its importance.

3.2.1 Image Preprocessing

The dimensions of the images in the THINGS-EEG dataset vary from 480x480 pixels to 6712x6712 pixels. To ensure a uniform dataset and avoid the need for interpolating pixel values during upsampling, we downsize all images by resizing them to 256x256 and cropping them to 224x224. This is the preferred dimensions specified by PyTorch (Paszke et al., 2019) when using their pretrained models. While downsampling inevitably leads to loss of image details, this approach is preferred to the uncertainty of image upsampling, which would introduce random noise into the unique image to EEG mapping. Furthermore, we normalize the images using specific means and standard deviations, which also adhere to the PyTorch standard for their pretrained models. This PyTorch preprocessing pipeline is applied for all models regardless of using pretrained models or not to ensure consistency in input between our models. A comparison between an original image and its preprocessed version is shown

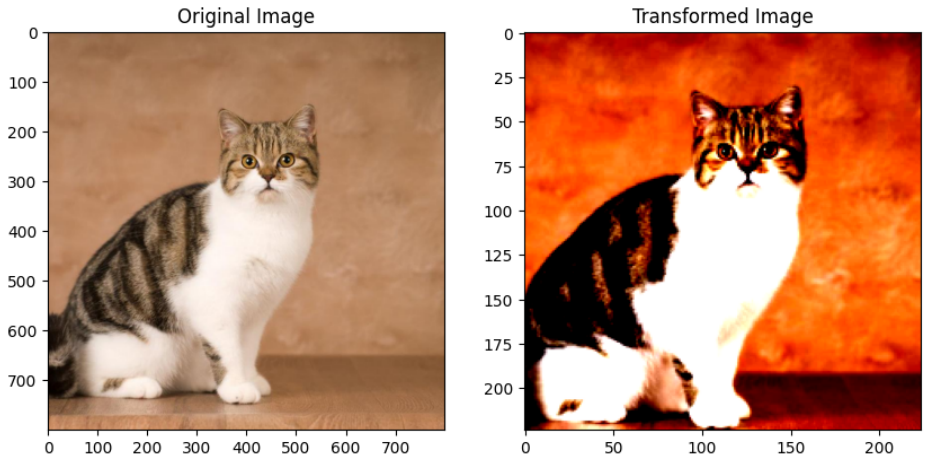


Figure 3.2: Visualization of an image before and after going through our image preprocessing pipeline. The transformed image has a lower resolution and is visually altered by the normalization. Image: `cat_03s.jpg`.

in Figure 3.2.

3.2.2 EEG Preprocessing

From the 64 available EEG channels, we focus on select subsets of channels in the most relevant areas of the visual cortex, which are known to capture information specific to our objective (Kruger et al., 2012). The first subset, {`O1, Oz, O2`}, consists of the channels in the occipital region, which is the earliest processing stage in the visual cortex. Next, the subset {`O1, Oz, O2, T7, T8`} includes `T7` and `T8`; two channels associated with later sections of the temporal region and the ventral pathway, which represents a later processing stage. Lastly, the subset {`O1, Oz, O2, T7, T8, TP7, TP8, P7, P8, PO7, PO8`} includes all channels from both the occipital region and the ventral pathway. Including channels that capture different functionalities of the visual cortex allows us to study the difference in the performance of reconstructing each channel subset representing different processing stages in the visual cortex. A spatial 2D visualization of the three subsets can be seen in Figure 3.3.

Preprocessing of the raw EEG signals involves multiple steps, as shown in Figure 3.4. The pipeline is applied to raw EEG responses epoched to the first 300ms of brain recordings after stimulus onset.

We filter the EEG signals by applying a bandpass FIR filter; a digital signal processing technique that uses lowpass and highpass filters to isolate frequencies below and above certain thresholds. This filtering helps reduce noise and avoid possible artifacts when downsampling, improving the overall signal-to-noise ratio in the EEG signal. Specifically, we use the Butterworth filtering design from the *scipy* library (Virtanen et al., 2020) to design a bandpass

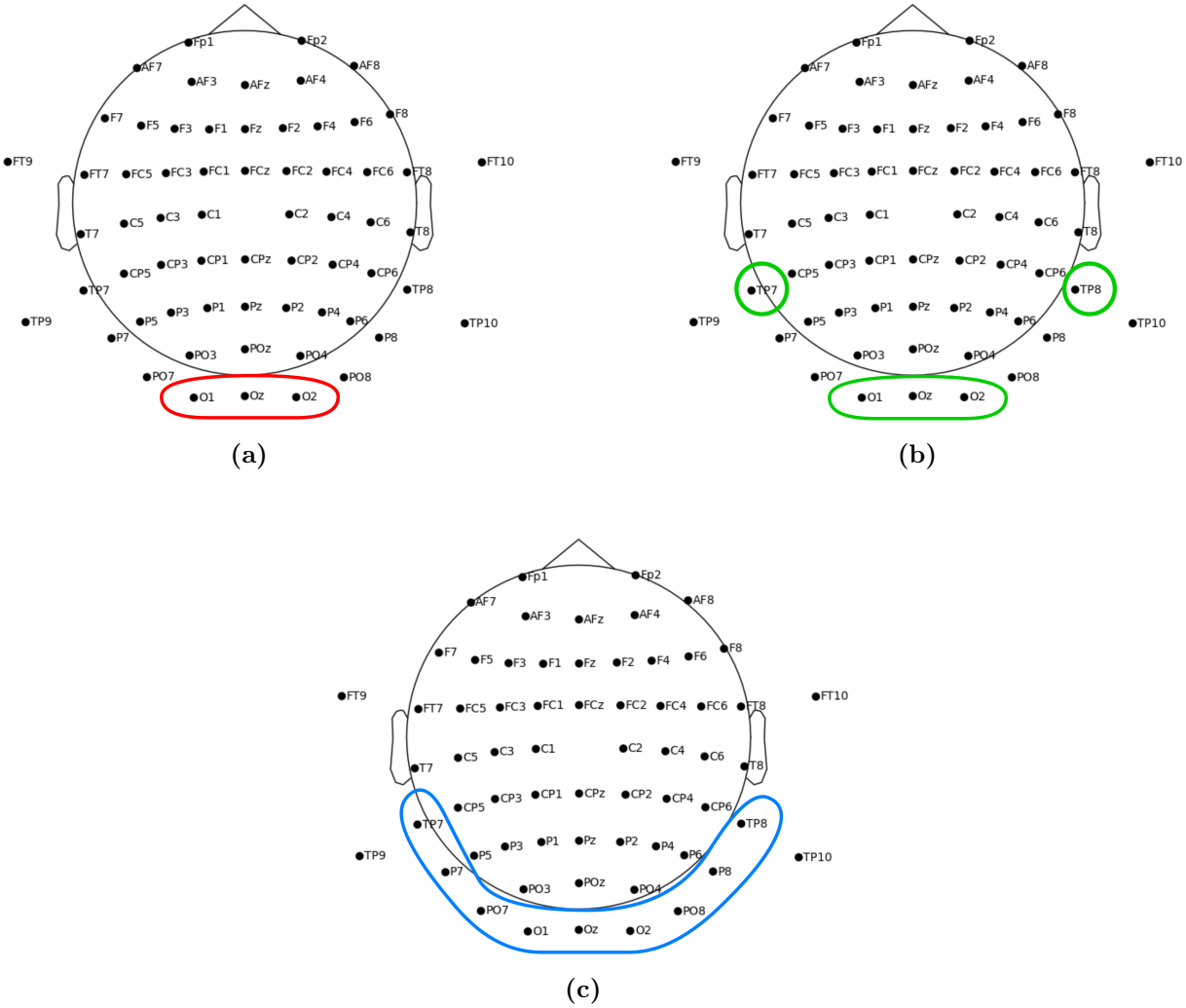


Figure 3.3: 2D representations of the 64 channels used in the THINGS-EEG data collection process, highlighting the selected subsets used in our experiments. **(a)** The visual cortex. **(b)** The visual cortex and two channels from the ventral pathway. **(c)** The visual cortex and the entire ventral pathway.

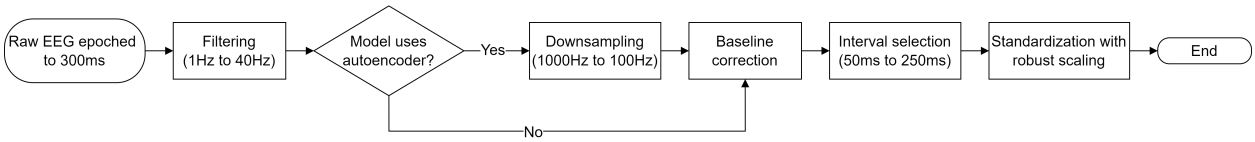


Figure 3.4: Flow chart for the EEG preprocessing pipeline.

filter, with low- and highpass values of 40Hz and 1Hz respectively. These values are inspired by the EEG preprocessing presented by Lawhern et al. (2018).

We downsample the raw EEG from 1000Hz to 100Hz with *Torchaudio* library from PyTorch (Y.-Y. Yang et al., 2022), inspired by Benchetrit, Banville and King (2023) and Défossez et al. (2023). The downsampling helps remove high-frequency noise or artifacts that may be present in the signal. Additionally, by reducing the number of time points through downsampling, we make the reconstruction task easier, without losing significant details captured with the higher frequency. The downsampling from Torchaudio also applies minimal filtering, preserving the main response features of the data while slightly reducing noise.

Also inspired by Benchetrit, Banville and King (2023) and Défossez et al. (2023), a baseline correction is applied by calculating the mean of the signal per channel in the first 50ms relative to stimulus onset (which is not part of our time frame). For each channel, we subtract the respective mean value from the remainder of the EEG signal. By subtracting the mean signal from before the stimulus onset, the baseline correction attempts to remove any signal noise that may still be present from prior stimuli.

We specifically select the interval from 50ms to 250ms relative to stimulus onset for the final experiments. This decision is based on *1)* knowledge of generic feature processing happening in the early visual cortex as early as 30-40ms after stimulus onset (Kruger et al., 2012), *2)* our interest in only including active brain responses related to the sample image, and *3)* to ensure the signal response to the image is located within the time frame. Related to the second point, earlier responses in the 30-40ms range might correlate with previous stimuli presented during data recording. Since we do not include this additional dependency in our model input, we believe choosing a slightly later time point after stimulus onset will reduce the impact of previous stimuli on the selected response timeframe.

Finally, standardization is applied using a channel-wise robust scalar using the *scikit-learn* library (Pedregosa et al., 2011), similar to the approach seen in Benchetrit, Banville and King (2023) and Défossez et al. (2023). Robust scaling is a standardization method using statistics from the data, ensuring more robustness to outliers. Performing the operation channel-wise also helps isolate the response features seen within the individual channels. A visualization of the step-wise transformations that an EEG signal undergoes in our preprocessing pipeline is shown in Figure 3.5.

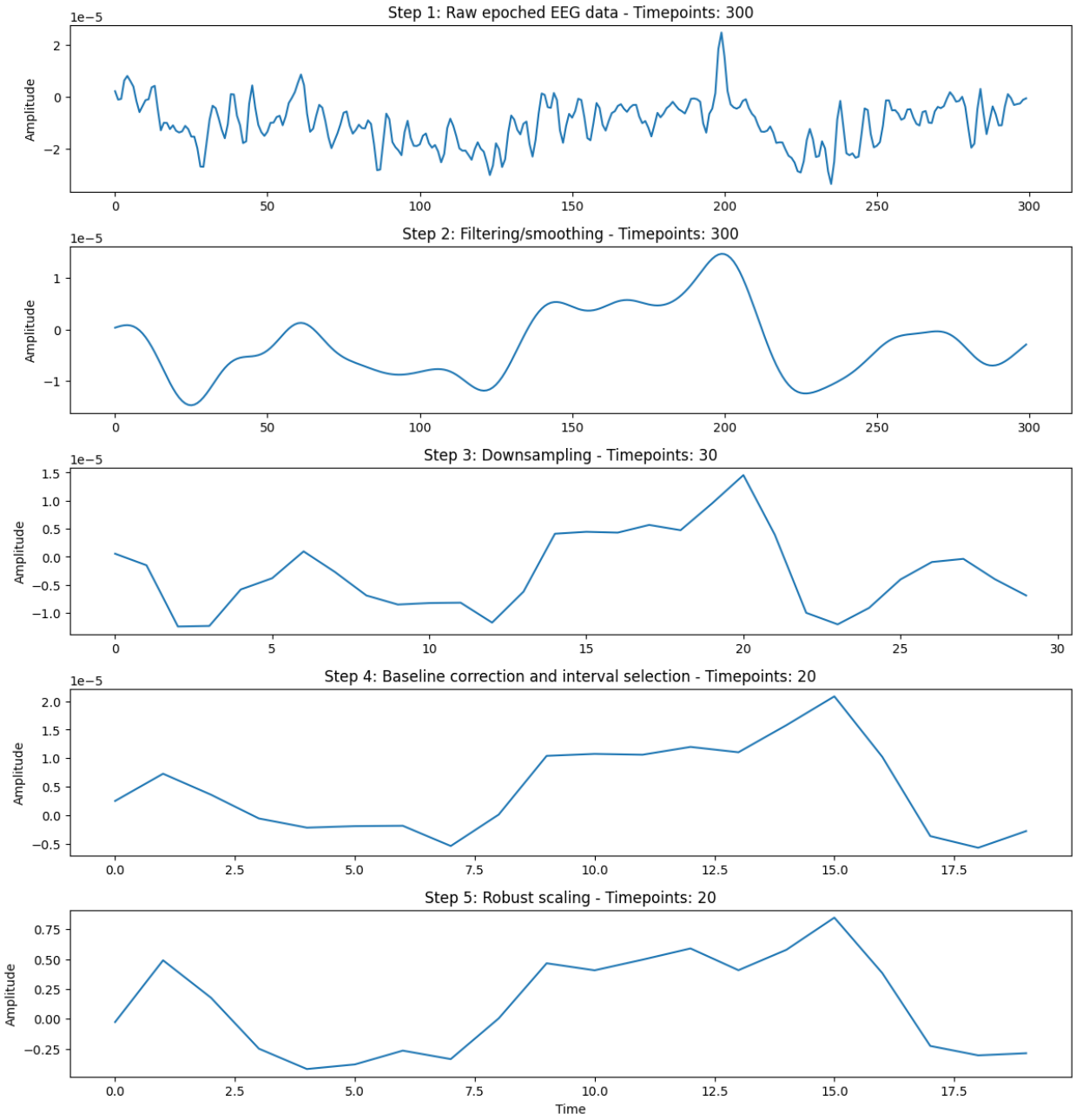


Figure 3.5: Overview of the preprocessing pipeline used for EEG data. The raw EEG data is filtered using a bandpass FIR filter with a lowpass of 40Hz and highpass of 1Hz, before being downsampled from 1000Hz to 100Hz (by a factor of 10). Then follows a baseline correction by subtracting the mean of the first 50ms from the rest per channel, and the desired interval is selected from t_{min} to t_{max}. Lastly, we standardize the data by performing robust scaling.

3.3 Loss Functions and Evaluation Metrics

When reconstructing EEG responses, the choice of the loss function and evaluation metric must be carefully considered because of the temporal nature of the data. Here, we adopt MSE and Sinkhorn as loss functions and evaluation metrics. MSE is included because of its low computational demand as a loss function. Sinkhorn is included for its ability to capture distances between signals that account for temporality in the signals, but at the cost of slightly higher computational demand. By experimenting with both, the comparison can hopefully highlight differences (if any) in choosing task-specific loss functions over general-use loss functions.

Besides MSE and Sinkhorn, which focus on the actual reconstruction error, we also employ RSA to compare RDMs between brain and model encodings.

3.3.1 Algonauts Challenge Evaluation Metric

To assist in the interpretation of our results and to make our results more comparable to existing work, we adopt the scoring system used in Algonauts 2023 Challenge (Gifford et al., 2023). It represents the model's encoding accuracy on a score between 0-100. However, this scoring system is designed for measuring performance in the reconstruction of fMRI, which doesn't translate directly to our task. We modify the scoring formula to account for our task's added dimensionality given by EEG's temporality.

Equation (3.1) shows the original challenge score metric, which closely resembles the Pearson Correlation Coefficient (Cohen et al., 2009), correlating the reconstructed and true fMRI responses. Here, v is the index of vertices (over all subjects and hemispheres), i is the index of the test stimuli images, and n is the total number of images. G and P correspond to, respectively, the ground truth and predicted fMRI test data, while $\overline{\mathrm{G}}$ and $\overline{\mathrm{P}}$ are the mean ground truth and mean predicted fMRI test data averaged across test stimuli images, R is the Pearson correlation coefficient between G and P, and NC is the noise ceiling (Gifford et al., 2023).

$$\text{metric} = Mean\left\{\frac{R_1^2}{NC_1}, ..., \frac{R_v^2}{NC_v}\right\} \cdot 100 \tag{3.1}$$

$$R_v = \text{corr}(G_v, P_v) = \frac{\sum_{i=1}^{n}(G_{v,i} - \bar{G}_v)(P_{v,i} - \bar{P}_v)}{\sqrt{\sum_{i=1}^{n}(G_{v,i} - \bar{G}_v)^2 \sum_{i=1}^{n}(P_{v,i} - \bar{P}_v)^2}}$$

Our modified version of the challenge score metric – which we will refer to as the *Modified Algonauts Challenge Score* (MACS) – is shown in Equation (3.2). We compute Pearson values per time point for each channel and take the mean of this to reduce dimensionality, which allows us to inspect average correlations for each channel across all time points, and a total correlation across all channels and time points. The variables of the equations are mostly identical. However, to account for our added dimensionality and different structure of EEG data, we replace the vertice index, v, with two new variables. The first, c, is the index of the EEG channel, where C is the total number of channels. The index of the time point is represented by t, where T is the total number of time points. Furthermore, our modified scoring system differs by simply representing a mean of the resulting correlations, disregarding the noise ceiling and squared correlations.

$$\begin{aligned} \text{metric} &= Mean\Big\{ R_1, ..., R_c \Big\} \cdot 100 \\ R_c = \text{corr}(G_c, P_c) &= \frac{1}{T} \sum_{t=1}^{T} \frac{\sum_{i=1}^{n} (G_{i,c,t} - \bar{G}_{c,t})(P_{i,c,t} - \bar{P}_{c,t})}{\sqrt{\sum_{i=1}^{n} (G_{i,c,t} - \bar{G}_{c,t})^2 \sum_{i=1}^{n} (P_{i,c,t} - \bar{P}_{c,t})^2}} \end{aligned} \tag{3.2}$$

Since computing the MACS is computationally demanding, it will be used solely for evaluation performance after training to provide another perspective on the model's encoding performance.

3.3.2 Estimating Performance of Random Guessing

To provide a reference for our evaluation metric results, we estimate the performance of reconstructing EEG responses from randomly guessing. We do this by randomly sampling from the distribution defined by the mean and standard deviation for each channel in the EEG responses. We compute the RMSE, Sinkhorn, and MACS for these randomly guessed reconstructions over several permutations to arrive at the estimates of random chance.

Another approach to this could have been to initialize models with random weights and test their reconstructive ability as evaluated by our metrics. This would require testing multiple randomly initialized modes to accurately estimate the performance of randomly guessing. We opted for the first approach, as we do not believe that the estimates of random guessing would be considerably different from these model-specific computations.

3.4 MLP as a Baseline Image-to-EEG Model

To begin our experimentation of EEG reconstruction, we first propose implementing a simple MLP, referred to as *MLPEEG*. The intention is to have the MLPEEG serve as a baseline model, providing context to the results from our more sophisticated models tasked with EEG reconstruction. It enables us to evaluate if the added complexity of our other model designs is justified by their performance.

The input images of the THINGS-EEG dataset have dimensions of $C \times H \times W$, where C is the number of channels (RGB), H is the height, and W is the width. For our specific case, the images have dimensions $3 \times 224 \times 224$ after preprocessing. This results in $3 \times 224 \times 224 = 150,528$ data points per image. When using an MLP to fully connect each input pixel to the output layer, this would lead to an immense number of parameters. Specifically, if we were to fully connect all $150,528$ input pixels to each unit in a layer, we would end up with hundreds of millions of parameters to train. This large number of parameters can make the training process computationally infeasible and lead to overfitting due to the model's excessive complexity.

To mitigate this issue, we include an initial average pooling layer in the MLPEEG model design that reduces the dimensions of the image. This results in substantially fewer neuron connections in the network, while retaining the most important information of the image data. After average pooling, we flatten the image data to create a vector that can be input to a fully connected layer. Then follows two linear (fully connected) layers with non-linear activations (ReLU), with the primary goal of feature extraction, which feeds into a final smaller linear layer that matches the expected output length. The output of the final linear layer is reshaped to fit the dimensionality of channels and time points of the target EEG. As a result, the model does not focus on the possible relations between the different channels of the EEG. A visualization of the MLPEEG model architecture is shown in Figure 3.6.

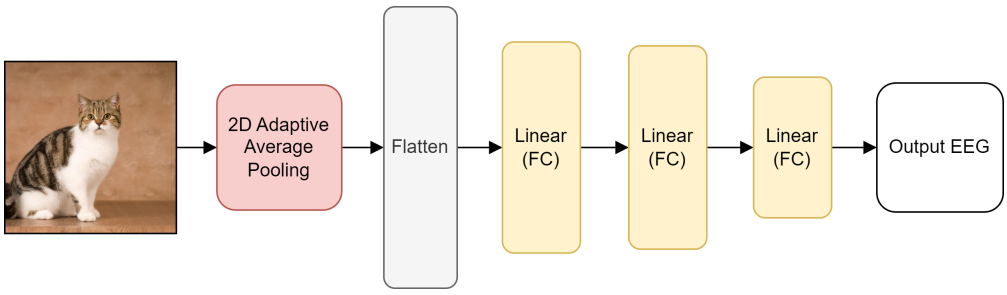


Figure 3.6: Simple visualization of the MLPEEG

3.5 ResNet as an Image-to-EEG Model

Designing a more complex second model for the reconstruction task, we propose repurposing a ResNet-18 model for EEG reconstruction, referred to as *ResNetEEG*. We specifically choose the ResNet-18 architecture for its proven performance in computer vision tasks as highlighted by K. He et al. (2016). To modify the ResNet architecture for a regression task rather than classification, we map the output of the ResNet CNN's feature maps to a single linear layer that outputs the reconstructed EEG responses directly. This type of linearizing encoding model attempts to reconstruct brain responses using the features extracted by a computational model such as a CNN. A visualization of the ResNetEEG model architecture is shown in Figure 3.7.

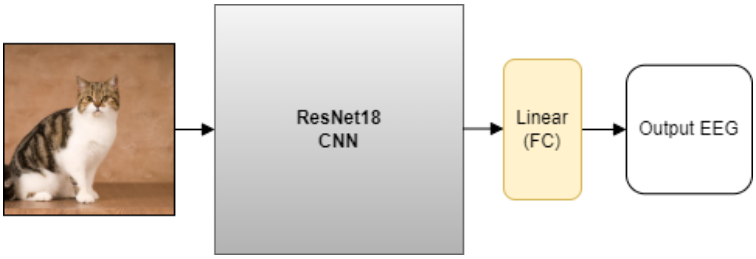


Figure 3.7: Simplified overview of the ResNetEEG model. The ResNet-18 CNN block follows the module architecture presented in Figure 2.5.

Based on our experiments with different weight initialization strategies, we consistently initialize ResNetEEG with pretrained and fine-tuned weights, as opposed to using randomly initialized weights. This approach leverages the information the model has learned from previous computer vision tasks. PyTorch provides pretrained ResNet weights that achieve 69.76% top-1 classification accuracy on the ImageNet dataset (Deng et al., 2009). These pretrained weights, optimized for image classification, can be fine-tuned to better fit our specific dataset, resulting in a top-1 classification accuracy of 57.71%. The observed decrease in accuracy is due to the complexity of our dataset, which contains 1854 image classes compared to the 1000 classes in the ImageNet dataset.

Given that CNN architectures are particularly adept at extracting lower-level features from images, the relevance of which dataset and class distribution the CNN is trained on might be negligible for the resulting extracted feature maps in the long run. However, employing pretrained and fine-tuned weights in ResNetEEG reduces model training time, because the pretrained convolutional layers are already adapted to process images from our dataset. This is the main rationale behind using pretrained weights for ResNetEEG.

3.6 Temporal Convolutional Network Autoencoder

In an attempt to reduce the complexity of the mapping between the feature maps output of our ResNetEEG model design and its final EEG output, we implement a *Temporal Convolutional Network Autoencoder* (TCNAE) as proposed by Thill et al. (2021). By creating a smaller latent space representation of the EEG responses, the goal is to simplify the target EEG to be reconstructed by ResNetEEG, compared to the original target EEG. The TCNAE functions by taking EEG responses as input and encoding them into a latent space before attempting to reconstruct (decode) the responses from the latent space again. The encoder and decoder architectures of the TCNAE are (near-identical) mirrors of one another. Notably, we do not perform a downsampling of the EEG signals from the preprocessing pipeline when using the TCNAE, as the architecture itself applies downsampling to the time points and thus requires an initial larger count of time points.

3.6.1 Encoder

The encoder architecture consists of seven convolutional blocks of convolutional layer pairs. Each block contains a dilated convolutional layer with 64 filters and a kernel size of 8, followed by a pointwise (1x1) convolution of 16 filters. The dilation in the first layer is 1 and is doubled for each subsequent dilated convolution, which means the final dilated convolution in the encoder uses a dilation of 64. The encoder implements skip connections, passing the output of all pointwise convolutions to the next dilated convolutional layer and a concatenation layer at the end of the encoder. Here, the outputs of the pointwise convolutions are concatenated, followed by one last pointwise convolution and an average pooling layer. The resulting output then represents the latent space. An overview of the encoder architecture is visualized in Figure 3.8.

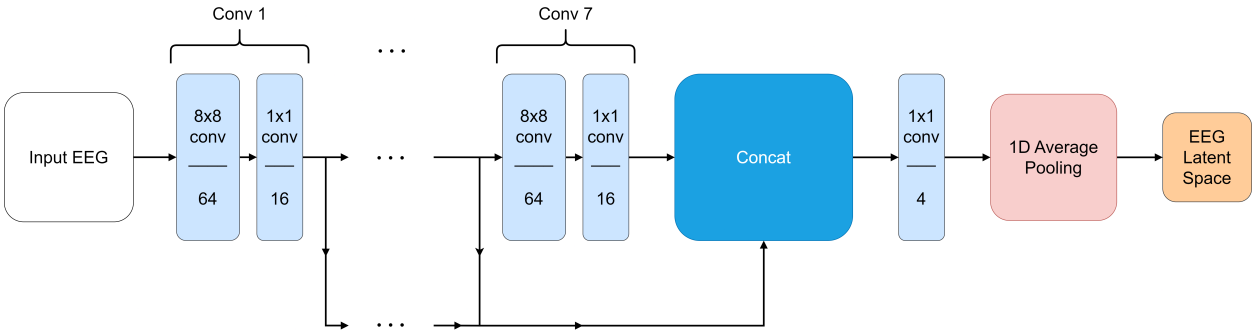


Figure 3.8: Overview of the encoder architecture in the TCNAE. It consists of 7 blocks of convolutional layer pairs with skip connections into a concatenation layer and an average pooling layer that outputs the resulting latent space. Each convolutional layer is denoted with the kernel size (top) and the number of feature maps output by the layer (bottom).

3.6.2 Decoder

The decoder architecture has an almost identical structure to the encoder, with the same seven convolutional blocks implementing skip connections for a future concatenation layer. However, for the dilated convolutions in the convolutional blocks, the dilation values are reversed. The first layer has a dilation of 64, which is then halved for every subsequent dilated convolution, resulting in a dilation of 1 in the final block. Additionally, the decoder starts by upsampling the latent space by the same factor we performed average pooling with, before passing the data to the convolutional blocks. After concatenating the results of the skip connections, we perform one last pointwise convolution to return the data to the original shape. An overview of the encoder architecture is visualized in Figure 3.9.

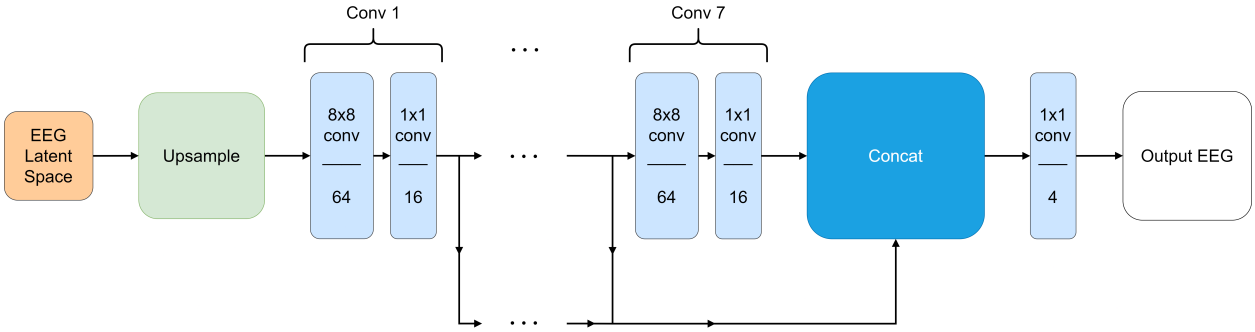


Figure 3.9: Overview of the decoder architecture in the TCNAE. The average pooling layer is swapped for an upsampling layer and the order is reversed relative to the encoder. Each convolutional layer is denoted with the kernel size (top) and the number of feature maps output by the layer (bottom).

3.6.3 Choosing Latent Space Size

Given that our input data differs from that described in the original TCNAE paper, we must adapt the model's parameters accordingly. Specifically, we have to select an appropriate average pooling size that aligns better with our data. For instance, while Thill et al. (2021) deal with time series data comprising of 130,000 data points and utilizes an average pooling filter of size 32, our dataset consists of only 600 time points. Using such a large average pooling size might result in excessive loss of information. To address this issue, we conduct a random hyperparameter search to identify the optimal parameters for our setup. This involves experimenting with different average pooling sizes alongside other hyperparameters such as batch size, learning rate, weight decay, and choice of optimizer for all channel subsets. While the TCNAE will most likely achieve the best performance when using a lower pooling size, we balance performance with the size of our smaller latent space representation. This is done to achieve the desired outcome of simplifying the prediction of EEG responses. The resulting hyperparameter configurations will be presented in Section 3.9.

3.7 ResNet as an Image-to-Latent-EEG Model

Introducing the TCNAE into the context of reconstructing EEG responses from visual stimuli, we propose a third model design specialized in reconstructing EEG responses from a latent space, referred to as *ResNetLatentEEG*. This model is a modification of our ResNetEEG model, which combines the two domain-specific models, ResNet-18 and the TCNAE, to reconstruct EEG responses.

The goal of the design is to utilize the CNN of ResNet for efficient image feature extraction, and the pretrained decoder of the TCNAE to simplify the mapping to the EEG response through the use of a latent space EEG representation. The decoder is used to reconstruct the full EEG response from the latent representation predicted by the CNN and linear layers during training, achieving a marginal loss in the decoding process, determined by the autoencoder efficiency. We seek to keep all weights of the pretrained decoder frozen to ensure consistent decoding from the latent space. A visualization of the ResNetEEG model architecture is shown in Figure 3.10.

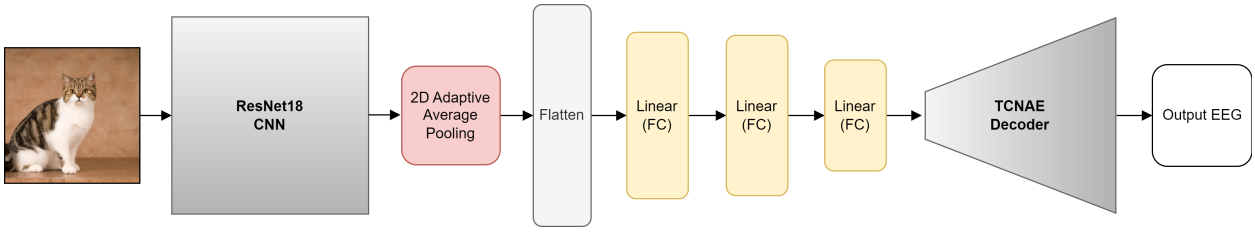


Figure 3.10: Overview of the ResNetLatentEEG architecture. The ResNet-18 CNN block follows the module architecture presented in Figure 2.5, and the TCNAE decoder block follows the decoder architecture from Figure 3.9. The intermediary linear layers (except for the last one) also contain a dropout layer with a dropout rate of 0.5, a batch normalization layer, and a ReLU activation function. The predicted EEG latent space is output by the last linear layer.

3.8 Using Different ResNet Module Outputs to Reconstruct EEG

To investigate the effectiveness of using a pretrained ResNet-18 model in our model designs, and herein its different modules, we explore the possibility of steering our two models', ResNetEEG and ResNetLatentEEG, reconstruction abilities toward specific time intervals of the EEG responses, represented through our chosen EEG channel subsets. Leveraging the modular structure of the ResNet architecture allows us to separate the model at a specific module, using its output at that particular point for the EEG reconstruction task. Referring

back to Figure 2.5, the idea is to divide the ResNet-18 architecture between its modules and map the output from that particular section to the values of an EEG response.

As briefly introduced in Section 2.3.4, we hypothesize that outputs of shallower modules in the CNN, resembling the initial processing stages in the visual cortex, could potentially be better at learning the reconstruction of earlier intervals of EEG responses. Conversely, outputs of deeper modules in the CNN could be better at learning the reconstruction of later EEG responses. To study this, we utilize the different EEG channel subsets to represent different response intervals. We might be able to observe varying response magnitudes across the different EEG channels depending on the time relative to the onset of the currently observed stimulus. Relating this to our models, the hypothesis suggests that certain model configurations in terms of module depth might better resemble certain processing stages in the visual cortex, potentially resulting in better reconstructions of EEG for the channels in the respective area of the brain.

We seek to experiment with training our two EEG reconstruction models, ResNetEEG and ResNetLatentEEG, each using a different section of modules from the ResNet-18 architecture in their respective designs. Specifically, we experiment with mapping the outputs of the first 2, 3, or 4 modules of ResNet-18 to EEG. We do not look further into using the output of only the initial convolutional layer or the first module, as these did not show any positive difference in performance patterns from tests during development.

3.9 Model and Training Setup

3.9.1 Hyperparameter Configurations

A random search of hyperparameters for each of our three image-to-EEG models forms the basis for our hyperparameter configurations. The random searches were conducted for model training over 400 epochs, using ∼18000 training samples and MSE as loss function, with the task of reconstructing the first subset of EEG channels, {`O1, Oz, O2`}, and using 4 ResNet modules for the applicable models. From the results of the random searches, we hand-pick the overall best-performing configurations. The respective model hyperparameter configurations are then generalized across all our models' reconstruction of all EEG channel subsets and the use of any number of ResNet modules. We made this choice due to the random search process being very time-consuming. The resulting configurations are shown in Table 3.1.

Model Name	Optimizer	LR	WD	Batch size	Dropout rate
MLPEEG	Adam	0.0001	0.0001	128	-
ResNetEEG	SGD	0.01	0.0005	128	-
ResNetLatentEEG	Adam	0.0001	0.0001	256	0.5

Table 3.1: Training configurations for the proposed Image-to(-Latent)-EEG models.

Furthermore, we perform a random search of hyperparameters for the TCNAE across the different channel subsets to find the best balance between dimensionality reduction degree, based on the pooling size, and model performance. From the random search, we hand-pick the hyperparameter configurations shown in Table 3.2. We freeze the weights of the TCNAE decoder when training ResNetLatentEEG to avoid worsening its performance.

Channels subset	Pooling size	Optimizer	LR	WD	Batch size
{O1, Oz, O2}	25	Adam	0.001	0.0001	128
{O1, Oz, O2, T7, T8}	10	Adam	0.001	0.0001	128
{O1, Oz, O2, T7, T8, TP7, TP8, P7, P8, PO7, PO8}	8	Adam	0.001	0.0001	128

Table 3.2: Training configurations for the TCNAEs across different subsets.

3.9.2 Training Configurations

When training our models, we split the dataset into training, validation, and testing sets, with respective sizes proportional to roughly 80%, 10%, and 10%. All models regardless of hyperparameter configuration, channel subset, and number of ResNet modules are trained for 1000 epochs to ensure convergence. We train all configurations of models twice: once with MSE as loss function and once with Sinkhorn as the loss function. This allows us to compare the difference in training performance when using a loss function that attempts to capture temporal deviations against a loss function that doesn't.

We save the best model state throughout training, with the lowest validation loss defining the best state. This is done to maximize the possibility that the saved model is the one that performs best when presented with unseen data. We opt to only save model states beginning from epoch 50 to ensure that we save model states when they have had time to learn from the data, as in some cases, poor generalization might result in overfitting, with continuously increasing validation loss from the beginning of training. This saved model state is then used for performance evaluation.

For the models using a CNN as part of their architectures, we utilize a pretrained and fine-tuned ResNet-18 model. The weights of the pretrained ResNet-18 model are not frozen during training to further optimize its weights for the EEG reconstruction task.

4 Results

In this section, we outline the outcomes of our experiments. We begin by detailing the performance of our Temporal Convolutional Network Autoencoder. Next, we present the performance of our proposed models – MLPEEG, ResNetEEG, and ResNetLatentEEG – in reconstructing EEG responses across the three subsets of EEG channels, using MSE as the loss function. For all models, smaller values of the RMSE and Sinkhorn metrics indicate a better performance, while larger values of the MACS metric indicate a better performance. We also include the results from the Representational Similarity Analysis (RSA) for each model. Finally, we highlight additional noteworthy observations from the results.

4.1 TCNAE

Our implementation of the TCNAE achieves desirable performance across all three subsets of channels, which is reflected in all three performance measures listed in Table 4.1. Both the low RMSE and Sinkhorn values indicate a marginal error in the reconstructions from the latent space EEG, and the impressively high MACS indicate how the decoded response aligns closely with the EEG. Identical for RMSE, Sinkhorn, and our MACS metrics, is the slight gradual decrease in performance for larger channel subsets. This is expected behavior from the more demanding task of predicting more data points.

Channels	**Pooling size**	**RMSE**	**Sinkhorn**	**MACS**
3	25	0.06	0.59	99.65
5	10	0.12	1.45	99.41
11	8	0.15	1.83	98.12

Table 4.1: Results from testing trained TCNAE models with each channel subset.

The reconstructive capabilities of the TCNAE are also evident when visually comparing original and reconstructed EEG signals as seen in Figure 4.1. The reconstructed signal follows the same pattern as the original signal, however, despite being trained using filtered EEG signals, the model seems to capture high-frequency signals as indicated by the small fluctuations in the reconstructed signal.

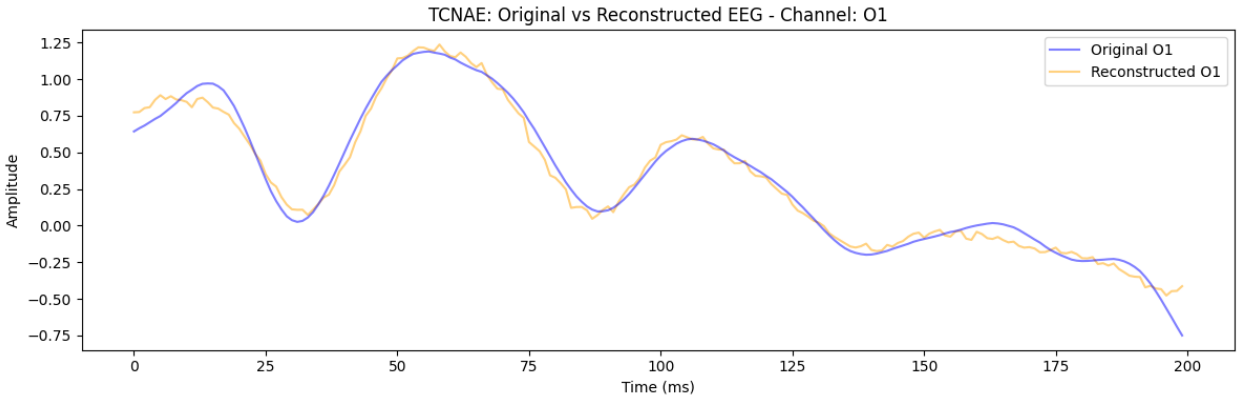


Figure 4.1: Example of the TCNAE's reconstruction ability for the O1 channel. The blue line represents the original EEG response and the orange line represents the reconstructed EEG response by the TCNAE.

We confirm this observation by plotting a spectral density plot over the difference between our original and reconstructed signals in Figure 4.2. We see a clear difference between the original and reconstructed signals in higher frequencies, between 40Hz and 150Hz approximately. This also makes sense from the perspective of our preprocessing pipeline, which includes a filter from 1 Hz to 40Hz on the EEG signals. In summary, the autoencoder effectively captures the EEG signals below 40Hz – the signals we train it on – but introduces variation in higher frequencies, likely stemming from not explicitly defining the frequency range in the model's configuration.

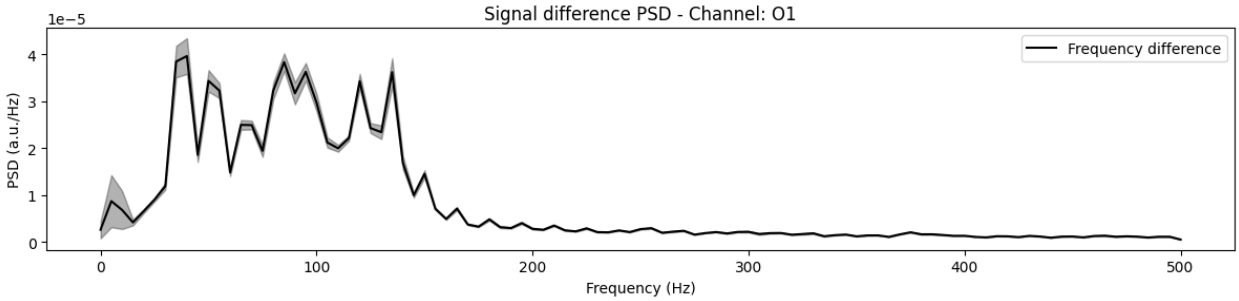


Figure 4.2: Spectral density plot for the O1 channel, over the difference between the original and target signals. The shaded area denotes the standard deviation. There is a clear difference in the signal frequencies between 40Hz and 150Hz approximately.

4.2 Image-to-EEG models

Here we present the main results of our three image-to-EEG model designs – MLPEEG, ResNetEEG, and ResNetLatentEEG – in the task of reconstructing EEG responses from visual stimuli. This includes their encoding fidelity performance through our chosen evaluation

metrics, their encoding similarity through RSA, and the impact of using different ResNet module outputs.

4.2.1 Encoding Fidelity Performance

The performance of our three image-to-EEG models, trained using MSE as the loss function and evaluated with three different metrics, is summarized in Table 4.2. Overall, none of our models excel in the task. This is particularly evident when comparing the results to the metric values for randomly guessing, where the RMSE is at 0.77 and Sinkhorn at 4.14 – both benchmarks that our models fail to surpass across all configurations. The best RMSE values seen between MLPEEG, ResNetEEG, and ResNetLatentEEG are 0.91, 0.78, and 0.77 respectively, indicating their performances on this metric all being as poor or worse than random guessing. A similar trend is seen for the best Sinkhorn values between the models, respectively being 5.68, 4.17, and 36.33. Interestingly, ResNetLatentEEG achieves a significantly higher evaluation on this metric than the other models. The consistent underperformance compared to random guessing on the test set suggests, that while the models learn from the training data, they struggle to generalize to unseen data. Regarding the MACS results, the models perform slightly better than random guessing, reaching up towards 9.58 for ResNetEEG reconstructing 3 channels and using 4 ResNet modules. Since this metric might reflect a model's encoding in comparison to the brain, these results indicate that our models' encoding aligns somewhat more closely with brain activity than random guessing does.

As expected, our MLPEEG model shows the least promising performance. The increase in performance between the MLPEEG and the best-performing ResNetEEG models is notable and is most likely attributed to the improved image feature extraction connected with CNN architectures. This doesn't translate well into ResNetLatentEEG, where the added complexity and the smaller latent space do not improve performance compared to ResNetEEG. The results of the ResNetLatentEEG will also be limited by the performance of the TCNAE model when decoding the latent space EEG into the full response. Interestingly, RMSE and Sinkhorn change very little across the different configurations of channel subsets and ResNet modules for ResNetLatentEEG, although its MACS vary more. Its RMSE and MACS are also fairly close to the best-performing ResNetEEG models, but its Sinkhorn values are closer to the worst-performing ResNetEEG models.

Similar for all three models is the decrease in most performance metrics when predicting EEG responses for larger subsets of channels. This is an expected result since the task naturally becomes more difficult when the model needs to predict more samples. Furthermore, we see an overarching similarity between all ResNet-based models in their ability to solve the

Model	Channels	ResNet modules	RMSE	Sinkhorn	MACS
Random Guessing	-	-	0.77	4.14	0
MLPEEG	3	-	0.91	11.55	3.44
MLPEEG	5	-	0.97	5.68	2.35
MLPEEG	11	-	0.97	6.69	1.40
ResNetEEG	3	2	2.25	49.99	7.09
ResNetEEG	3	3	1.01	8.68	7.66
ResNetEEG	3	4	0.78	4.17	9.58
ResNetEEG	5	2	2.50	48.88	4.95
ResNetEEG	5	3	0.95	5.70	5.50
ResNetEEG	5	4	0.79	4.31	7.67
ResNetEEG	11	2	1.55	16.03	4.88
ResNetEEG	11	3	0.84	4.27	5.38
ResNetEEG	11	4	0.80	4.75	7.23
ResNetLatentEEG	3	2	0.82	52.92	8.81
ResNetLatentEEG	3	3	0.83	70.59	8.39
ResNetLatentEEG	3	4	0.80	49.11	8.30
ResNetLatentEEG	5	2	0.80	41.13	5.15
ResNetLatentEEG	5	3	0.80	38.15	5.51
ResNetLatentEEG	5	4	0.78	43.41	6.92
ResNetLatentEEG	11	2	0.80	36.33	4.55
ResNetLatentEEG	11	3	0.80	38.45	4.50
ResNetLatentEEG	11	4	0.77	40.06	6.61

Table 4.2: Evaluation metric results for RMSE, Sinkhorn, and MACS from testing our three trained models, MLPEEG, ResNetEEG, and ResNetLatentEEG with different configurations. This includes which channel subset is being reconstructed (3, 5, or 11 channels), and how many ResNet modules are being used (2, 3, or 4) for the two ResNet-based models. For each model, the best and worst evaluation metric results are highlighted in green and red, respectively. All results shown in this table are from our models trained with MSE as loss function.

task, albeit limited initially, getting worse when using fewer modules of the model - with a significant decrease in performance from the third to the second module in ResNetEEG models. Despite features in earlier modules better resembling the feature extraction of the human visual cortex, the effectiveness of deeper feature extraction proves superior. This effect is visible in all three performance measures.

The three metrics often show a correlation between their increases and decreases in performance, but this is not always the case. Sinkhorn displays a non-linear correlation with RMSE and MACS, leading to ambiguity in its interpretation. Additionally, we can observe cases where Sinkhorn displays a negative correlation to the other two, e.g. the MLPEEG's change in performance from 3 to 5 channels.

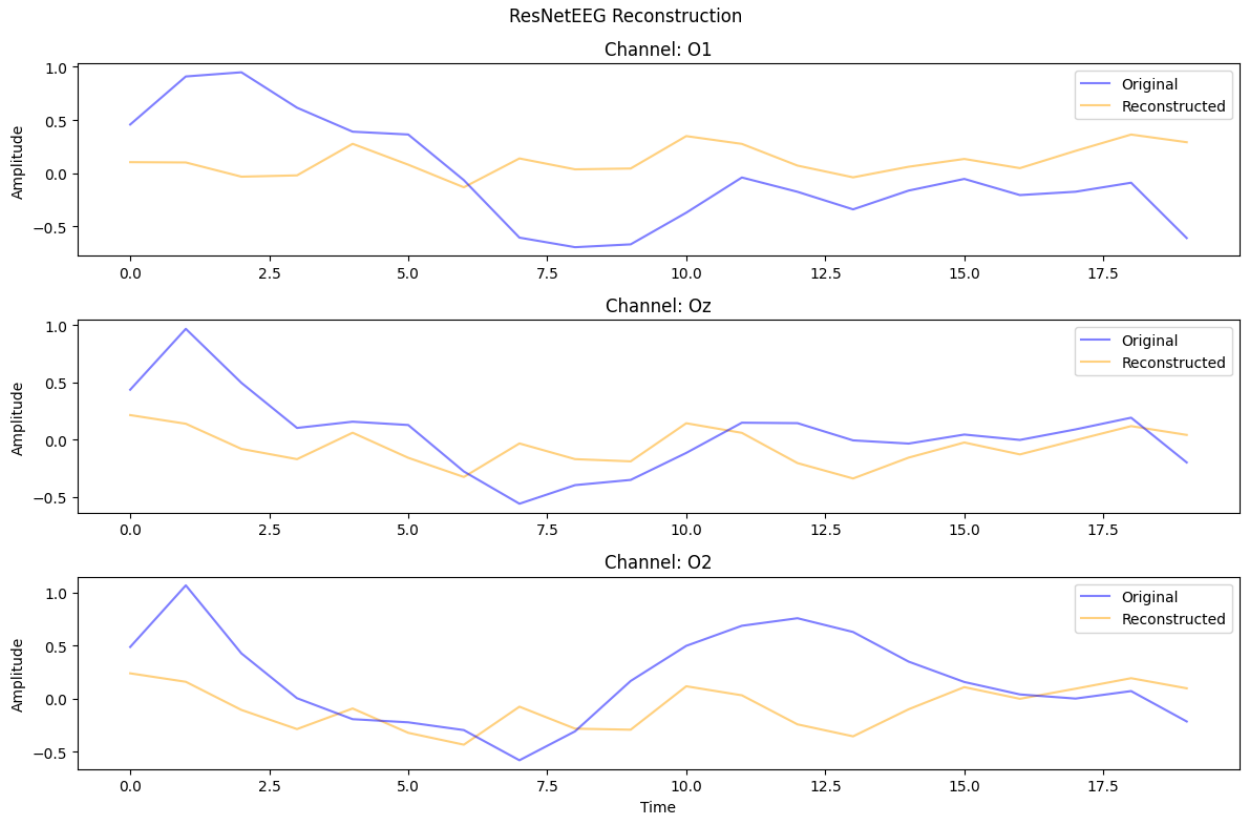


Figure 4.3: Example of the ResNetEEG model's reconstruction ability on the subset of three channels, {`O1, Oz, O2`}, shown from top to bottom respectively. The blue line represents the original EEG response and the orange line represents the reconstructed EEG response of the model. The model was trained with MSE as the loss function, reconstructing 3 EEG channels and using 4 ResNet modules.

The reconstructed EEG responses reflect the models' poor RMSE and Sinkhorn results, exemplified in Figure 4.3. Even for some of the better-evaluated reconstructions, challenges can be seen in capturing both the general response pattern over time as well as the change in amplitude. Most models tend to have large difficulties reconstructing EEG responses with varying change in amplitude. In light of the models' poor generalization, it makes

sense for them to have learned to reconstruct a relatively flat EEG response, as that would result in the lowest average error over unseen data. The slight resemblance in the curvature of the response amplitude of the reconstructed response could be indicative of the better-than-random guessing performance seen in all evaluated MACS. However, the failure to reconstruct most of the more noticeable changes in amplitude demonstrates the model's inability to encode information similarly to the brain.

We also experimented with Sinkhorn as the loss function. Naturally, the model demonstrates better performance in the Sinkhorn metric, as it specifically optimizes its solution to this metric. However, both the RMSE and MACS metrics generally indicate worse performance. These models also display noticeably more volatile training in general. Due to the ambiguity of Sinkhorn and the generally better performance on RMSE and MACS from models trained with MSE, we only include those models in this section. For further information on the results from models trained with Sinkhorn, see Appendix E.1.

4.2.2 Encoding Similarity through RSA

We use RSA to inspect the representational similarity between the brain and the models' encoding from image to EEG signals for more insights into the model's performance, however, Sinkhorn is not included in this analysis due to excessive computation time. We compute the results as Pearson correlations between RDMs of the original EEG signals and the reconstructed EEG signals corresponding to the samples from the test set. Our models do not capture any representational similarity, as the virtually zero Pearson values indicate no correlation between our models' and the brain's encodings of the images to EEG. This pattern is identical across all models. For the actual correlation values, refer to Appendix D.

An example visualization of a resulting pair of RDMs can be seen in figure Figure 4.4. The size of the RDMs (2225×2225) make them uninterpretable to the human eye, however, at a glance, there seems to be generally lower dissimilarity between the reconstructed EEG signals compared to the original EEG signals. This could indicate that the model is often guessing close to the same values, resulting in similar representations for different samples in the dataset - a theory that also aligns with the correlations between RDMs.

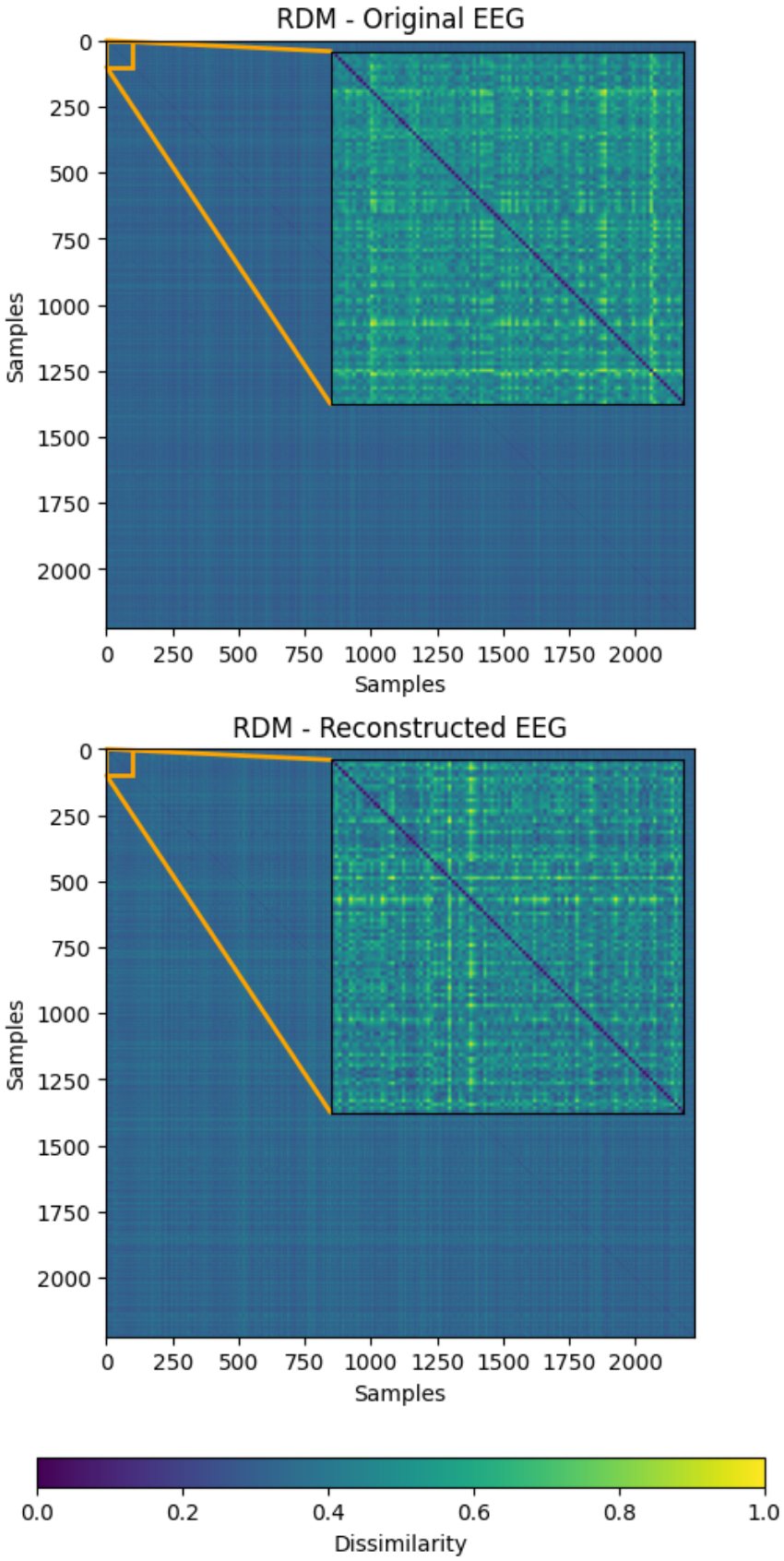
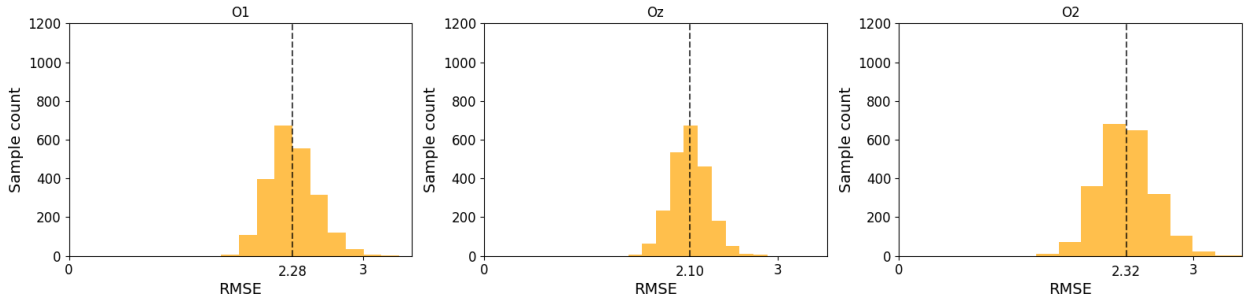


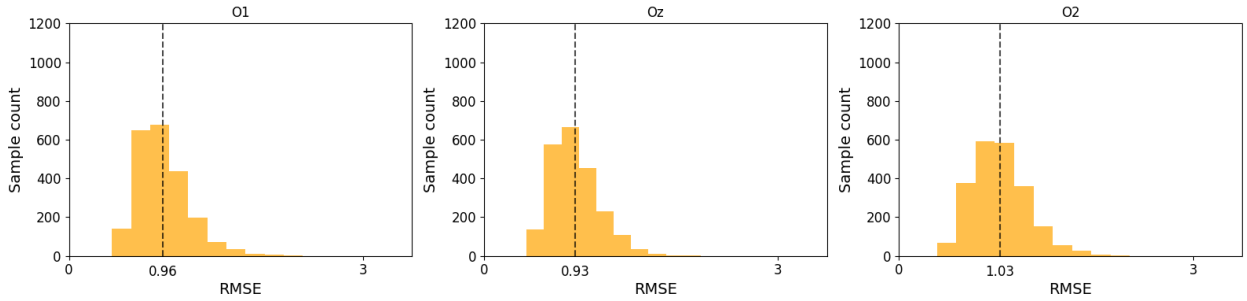
Figure 4.4: RDMs for both original and reconstructed EEG signals. The reconstructed signals are the resulting output from ResNetEEG configured to reconstruct 3 channels with 4 ResNet modules. The dissimilarity is based on RMSE normalized to a range of $[0, 1]$.

4.2.3 Analysis of using Different ResNet Module Outputs

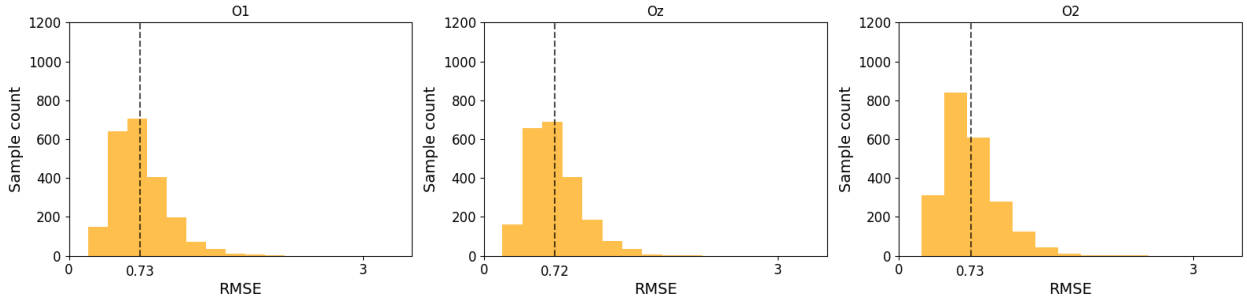
To learn more about the validity of the proposed hypothesis in Section 3.8, further analysis of the results is necessary. Due to time constraints, we picked only one of our model designs on which to base this analysis. Keeping in mind that all models generally demonstrated poor performance, ResNetEEG is chosen for this purpose since it shows the most promise across RMSE, Sinkhorn, and MACS, compared to ResNetLatentEEG. Additionally, we focus the presented analysis on the three-channel subset, {`O1, Oz, O2`}, since they correspond to locations in the early visual cortex.



(a) ResNetEEG with 2 modules



(b) ResNetEEG with 3 modules



(c) ResNetEEG with 4 modules

Figure 4.5: Distribution ResNetEEG's RMSE performance for each reconstructed EEG channel (subset: {`O1, Oz, O2`}) across all samples in the test set. The dashed, vertical line indicates the mean RMSE value for that channel. **(a)** shows the error distribution when trained with 2 modules. **(b)** shows the error distribution when trained with 3 modules. **(c)** shows the error distribution when trained with 4 modules.

The histograms of Figure 4.5 show the distributions of RMSE between the reconstructed and original EEG across all samples in the test set for ResNetEEG. Here, we compare configurations of 2, 3, or 4 ResNet modules. The most obvious takeaway is that the RMSE decreases when predicting EEG with deeper ResNetEEG architectures, corresponding to an increase in performance. From 2 to 3 modules, we see a significant improvement in terms of the mean RMSE with the distribution shifting closer to the random chance performance observed in Table 4.2. The improvement from 3 to 4 modules is minor, but the distribution now aligns with random chance. The same patterns are evident for ResNetEEG on the 5 and 11-channel subsets - for visualizations of these, refer to Appendix H. While there is no indication of shallower modules providing better reconstructive performance on channels representing the early visual cortex, a couple of limitations hinder us from saying with certainty that the observed results indicate a relationship captured by our models. Firstly, the models demonstrate generally poor reconstructive ability, not leaving much to conclude from. Additionally, the models are not configured or trained to distinguish between the different channels.

	2 modules		**3 modules**		**4 modules**	
Time Interval	**RMSE**	**MACS**	**RMSE**	**MACS**	**RMSE**	**MACS**
50-150ms	1.97	7.35	0.95	5.61	0.78	9.09
150-250ms	2.50	6.82	1.06	9.71	0.77	10.07
50-100ms	2.57	2.55	1.07	0.88	0.79	2.03
100-150ms	1.08	12.16	0.81	10.34	0.77	16.16
150-200ms	1.33	8.19	1.09	10.03	0.78	9.76
200-250ms	3.27	5.44	1.04	9.40	0.77	10.37

Table 4.3: RMSE and MACS for ResNetEEG reconstructing the three-channel subset, {`O1, Oz, O2`}, and using 2-4 ResNet modules on separate intervals relative to stimulus onset.

To provide another perspective on this matter, Table 4.3 shows RMSE and MACS performances for various time intervals of ResNetEEG's reconstructed EEG responses when using 2, 3, or 4 ResNet modules. We do not include Sinkhorn in this analysis, as its magnitude depends on the time points evaluated, resulting in a different scale than the other presented Sinkhorn results. While the use of any number of modules displays relatively poor performance across all the selected time intervals, a pattern may be deduced in consistently worse MACS results from the early intervals. Specifically, the 50-100ms interval returns a significantly worse MACS across all module depths. Furthermore, the 100-150ms interval

consistently showcases the best RMSE and MACS, with 4 modules achieving the best evaluations for both metrics. Mostly similar patterns can be seen on the 5 and 11-channel subsets - refer to Appendix G for these results. Overall, the few observed patterns do not suggest different ResNet modules resulting in better EEG reconstructions of different time intervals. For the same reasons as before, however, we are unable to conclude this with certainty.

5 Discussion

In this section, we examine the results generated by our models, exploring the underlying causes and potential explanations for these outcomes. Following this, we generally discuss the strengths and limitations of our methodology, the intricacies of the reconstruction task, the specific challenges encountered during the research process, and potential options for future work.

5.1 Discussion of Results

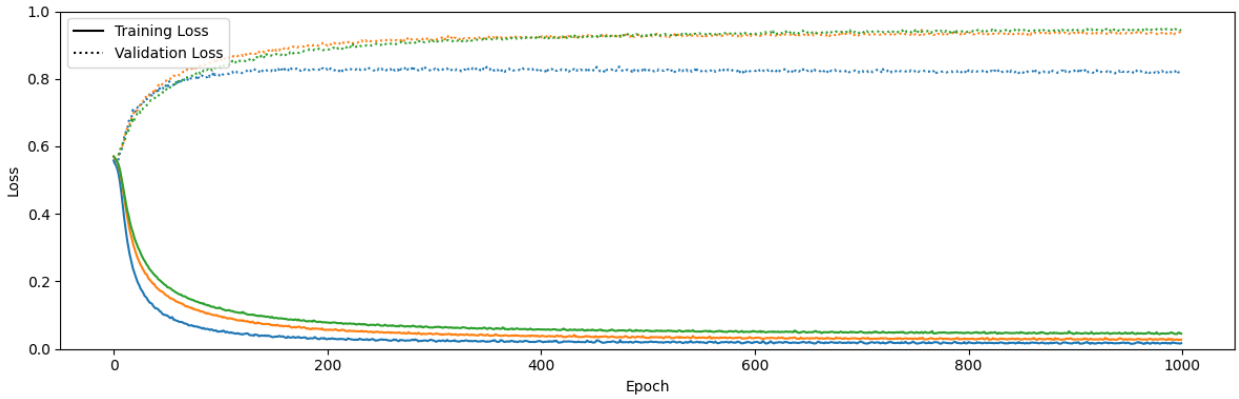
The results from our experimentation strongly indicate that our models neither solve the reconstruction task well nor correspond much to how the brain encodes images. Even though we do observe a few interesting results, like the better-than-random guessing MACS results across all models, we believe it is essential to approach these findings with a high degree of skepticism. Given the overall poor performance observed from our models, the few results that indicate otherwise may be a result of underlying issues in our methodology.

5.1.1 Model Overfitting

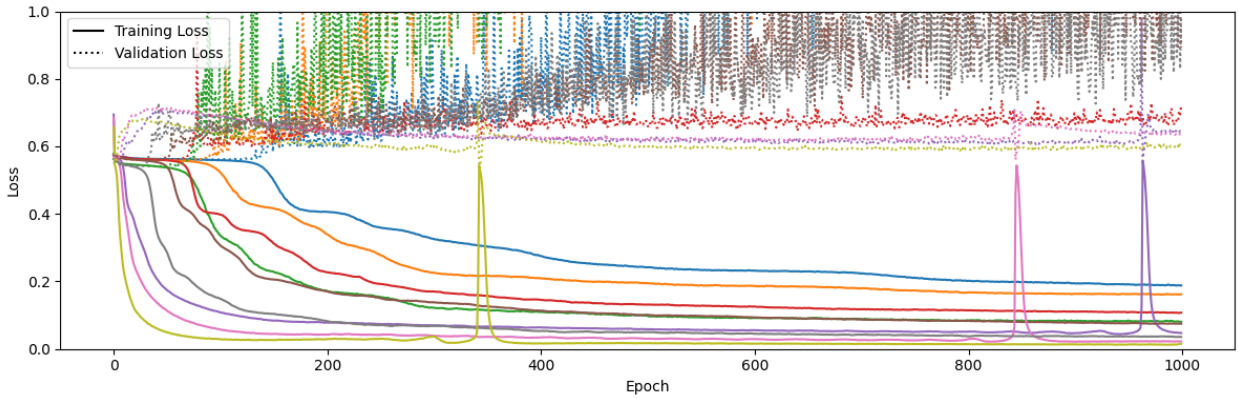
The performance of the models employed in this study can be analyzed through their training and validation loss curves, as visualized in Figure 5.1.

Figure 5.1a illustrates the training and validation loss curves for the MLPEEG model across different channel subsets. The three pairs of curves are nearly identical, indicating consistent behavior across subsets. The distinct pattern of decreasing training loss contrasted with increasing validation loss suggests a clear case of overfitting, where the model learns the noise of the training data to perform well during training, and as a result, performs poorly on unseen data.

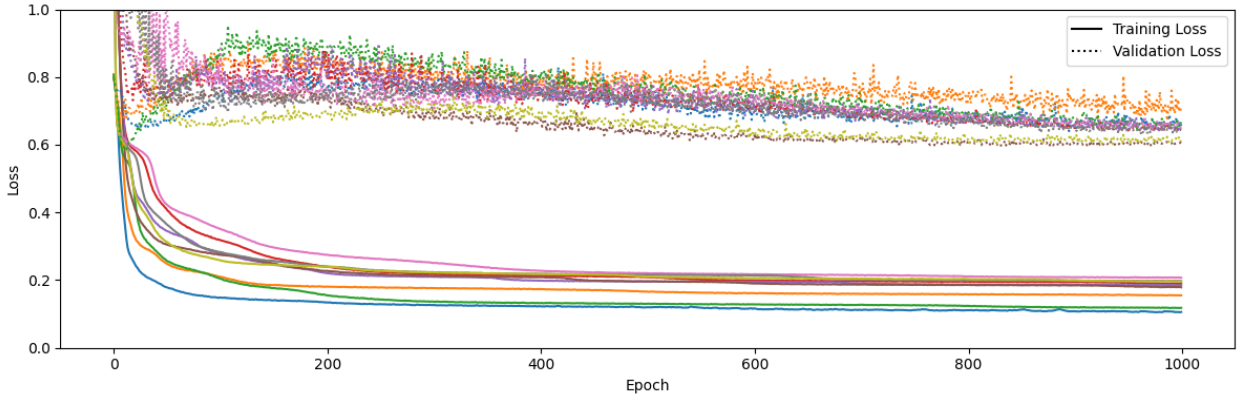
Figure 5.1b presents the loss curves for the ResNetEEG model. The variability in these curves is attributed to the different numbers of ResNet modules used in training. Models with fewer modules (ranging from 2 to 4) exhibit higher volatility in validation loss and converge at higher training loss values, reflecting instability and poor reconstruction performance. As the number of modules increases, these effects diminish, resulting in more stable validation



(a) Training and validation loss for MLPEEG.



(b) Training and validation loss for ResNetEEG.



(c) Training and validation loss for ResNetLatentEEG.

Figure 5.1: Training and validation loss curves for all configurations of our models, grouped by model. The solid lines represent training loss and the dashed lines represent training loss.

loss curves and lower training loss. This pattern indicates issues with both overfitting and regularization, as the model struggles to generalize to unseen samples. The large peaks observed in the training loss can be a result of the relatively high learning rate of 0.01 used during training.

Figure 5.1c shows the loss curves for the ResNetLatentEEG model, where results are similar regardless of the configurations of ResNet modules and channel subsets. While there is slightly more volatility in validation loss with fewer ResNet modules, the curves ultimately converge to similar values. Like the ResNetEEG model, the primary issues seem to be related to regularization rather than overfitting alone. Notably, the training loss converges at a significantly higher value compared to the MLPEEG and ResNetEEG models. This is likely attributed to a combination of decoding loss from the TCNAE and especially our design of the layers between the ResNet model and the TCNAE decoder.

5.1.2 Impact of Regularization

Throughout the study, we experimented with three regularization techniques: dropout layers, weight decay, and early stopping. While we do not have a direct comparison of their individual impact on the model performances, we can still discuss our view on their usefulness.

Dropout

The use of dropout layers was by far the most impactful regularization in terms of model learning and generalization. Aside from our main results, we also experimented with dropout layers and different dropout rates for all three of our model designs. Our attempts to include dropout layers for our MLPEEG baseline model during development provide an example of regularization mitigating overfitting, but drastically reducing how well the model learns from the training data. Thus, the challenge becomes to find the balance between the models not overfitting and not learning anything at all. This pattern repeats itself when introducing dropout for our other models. Refer to Appendix B.2 for a visualization of the MLPEEG example.

As a general opinion on this matter, we believe it is more interesting to have the models learn patterns from the training data and generalize poorly as a result, rather than the models not being able to learn much from the data at all, as it typically leaves more room for further analysis.

Weight Decay

The magnitude of the weight decay was determined by our hyperparameter random search results. As expected, a common trend across all random searches was a higher weight decay almost always correlating with better model performances on the test set. However, in most cases, this also correlated with the training loss being significantly higher than the models using low weight decay. The regularization helps with generalizing performance on the unseen data, but as with our use of dropout layers, the training loss reveals the same underfitting issues. The improvement in performance on the test set is only a result of the model's essentially random guessing instead of overfitting to the training set.

Early Stopping

Initially, we implemented early stopping as a measure to stop models that overfit to improve efficiency in our experiments. However, early stopping became less of a factor in our pipeline than expected, as we ultimately disregarded the use of it. The technique is useful when you want to stop training due to convergence or continuous signs of overfitting on the validation set. However, in our case, because we consistently experienced either poor training or poor generalization with our models, we believed it to be more interesting to see the full training of a thousand epochs for each model and hyperparameter setting. This would provide a better overview of the generalization issue and ensure equal frames for comparison between all model configurations.

Further Regularization: Data Augmentation

We considered improving upon model generalization through data augmentation, however, did not implement it in practice. Traditionally, data augmentation involves increasing the size of the dataset by applying various transformations to the existing data. Besides a larger dataset, this also increases the diversity of samples for model training.

Transformations typically applied in the augmentation of images include rotating, flipping, resizing and/or rescaling. However, cutouts (DeVries and Taylor, 2017) is another interesting augmentation method that can be applied to images. This method involves cutting out parts of the input during model training, which can improve the robustness and overall performance of CNNs. The main concern of adopting this approach in our EEG reconstruction task is that we are unable to control whether the cutouts remove parts of images that are essential to the activations in the brain and thus, for the resulting EEG response. For this reason, we decided not to implement data augmentation.

5.1.3 Using Different ResNet Module Outputs

To begin with, we acknowledge the severe limitations in our approach to investigating the intermediary hypothesis regarding different module outputs of ResNet potentially being better at learning the reconstruction of different time intervals of the EEG responses. The results presented in Section 4.2.3 intend to highlight this topic, but fail to properly encapsulate the impact of using different ResNet module outputs. In an ideal setting, the distributions between channels seen in Figure 4.5, and the encoding fidelity performance for smaller time intervals seen in Table 4.3, could support arguments both for and against the hypothesis. However, because our models are not designed or trained explicitly to treat channels or certain time intervals differently from each other, we cannot with certainty conclude anything from the presented results that would accurately represent the proposed hypothesis. For this reason, it also makes sense that we do not see patterns in our encoding fidelity performance indicative of this.

There is also a limitation in the hypothesis and analysis, namely that the brain is highly non-linear. As such, while certain channels may correspond to what we perceive as early or late processing in the brain, we cannot expect these regions to exclusively activate at these points in time. Furthermore, a limitation comes from the use of the current time interval in our EEG signals. Because we use larger EEG time intervals for each sample, the collective error computed for a single reconstructed sample ranges over a larger time interval. If an inspected model, independent of performance on the different channel subsets, were reconstructing the first 50ms accurately, but inaccurately for the remainder, the overall error computed would still reflect poor performance.

While we may not be able to conclude anything on patterns regarding the intermediary hypothesis, we can still see from our results, that more modules of the ResNet-18 architecture result in better EEG reconstructions of our models – albeit still poor. This can likely be attributed to extracted features being more robust in the deeper modules. This leads to another possible limitation of this experimental approach, namely that it might not be beneficial to divide the ResNet-18 architecture due to its residual blocks and skip connections. This type of architecture progressively builds up output, and the partial outputs inside the model may not be of as much use as the resulting output of the fully processed input.

Ultimately, to be able to say more about the hypothesis, future work on this topic could explore models reconstructing different, smaller intervals of the EEG response after stimulus onset separately. With this approach, each response reconstruction error would be more isolated for the specific time interval, and patterns suggested by the hypothesis might be easier to identify through the reconstruction error.

5.2 Discussion of Models

5.2.1 Choosing Computer Vision Model in Neuroimaging Tasks

We had several options to choose from when selecting a computer vision model for our task, as PyTorch (Paszke et al., 2019) offers pretrained weights for a wide range of existing models. The challenge was to find a balance between model performance and training time. We initially implemented smaller CNN models in the form of AlexNet (Krizhevsky, Sutskever and G. E. Hinton, 2017) and SqueezeNet (Iandola et al., 2016). On the ImageNet dataset, AlexNet provides 56.522% top-1 classification accuracy, while SqueezeNet provides 58.092% top-1 classification accuracy. Through fine-tuning, we achieved approximately 44% and 47% top-1 accuracy on the THINGS dataset, respectively. The decrease in performance can be attributed to the larger number of classes in our dataset.

With room for increased training times in our pipeline, and in search of better performance, we later opted to implement ResNet-18 – the smallest of the ResNet family. One reason was that we could fine-tune it to achieve 57.71% top-1 accuracy on the THINGS dataset, which is a significant improvement over AlexNet and SqueezeNet. Another reason came from Schrimpf et al. (2018), who proposes a *Brain-Score* to measure how brain-like certain neural network architectures are in object recognition tasks. Of the 25 models they test, ResNet-18 ranks as the 11^{th} most brain-like model, with AlexNet and SqueezeNet ranking at the bottom. Interestingly, ResNet-18 outperforms ResNet-34 on the Brain-Score metric, however, ResNet-50, ResNet-101, and ResNet-152 all rank higher. We briefly experimented with ResNet-152, which was fine-tuned to achieve 74.79% top-1 accuracy on the THINGS dataset, but we deemed it infeasible for our thesis because of the increase in training time. The most brain-like model is found to be DenseNet-169 (Huang et al., 2018), which also achieves similar performance in classification accuracy compared to the larger ResNet models.

We acknowledge that there are models that can improve the image feature extraction in our setup over ResNet-18. However, the significance of an improved feature extraction for EEG reconstruction remains to be answered.

5.2.2 Model Architectures

All our model architectures rely on linear layers to map previous neural network outputs to the time point values of an EEG response. Because of the experimental nature of the study, we have mostly relied on our knowledge about such mappings and roughly similar tasks when designing the architectures and use of different types of layers for the task. Because

such layers typically have a substantial impact on model performance, this is an area with room for further experimentation and analysis.

Overall, we recognize that our approach to designing the architectures for our three distinct models, which linearly encode to EEG, could have benefited from a more empirical strategy. The extensive variety of model configurations we analyzed – including different model designs, hyperparameter settings, channel subsets for reconstruction, the number of ResNet modules, and various loss functions – limited the number of model combinations we could feasibly train within the given time frame. Depending on the complexity of the models and the resources of the machine we trained on, each full training session of 1000 epochs required between 4 and 20 hours, impacting the level of systematicity we could implement in our development process. In terms of model architecture experimentation, we explored varying the number of linear layers, adjusting the output dimensions at each layer, applying different regularization magnitudes through dropout layers, and selecting appropriate activation functions. Despite these efforts, we believe our design approach could have been more systematic. The lack of a systematic approach also makes it difficult to argue for the effect that any architectural changes to our model may have.

Specific to our ResNetLatentEEG model, we also identify some contradictions related to the overall goal of the model design. By introducing a smaller latent space representation of the EEG, we aimed to make the mapping from feature maps to EEG responses simpler. However, we simultaneously increased the complexity of the linear layers significantly for ResNetLatentEEG. As a result, the model architecture might be too complex, considering the initial goal of the model design. We believe this might be reflected in the performance of the model, as it tends to generalize worse than other models earlier in the training, possibly focusing more on the noise of the EEG. To get an idea of the impact this could have on our results, we trained a single ResNetLatentEEG model with reduced linear layers that matches the architecture of ResNetEEG, but still with the purpose of predicting the latent representation of the EEG. It was trained for only 300 epochs due to time constraints with the following configuration: all 4 ResNet modules, 3-channel subset, and MSELoss as the loss function. This model showcased identical performance in RMSE, Sinkhorn, and MACS compared to the more complex ResNetLatentEEG of the same configuration listed in Table 4.2. The simple ResNetLatentEEG converged quicker, faster, and at a lower value in training loss, and showed less sign of overfitting compared to the complex model, but has similar issues of generalization to unseen data. Combined, these could hint at the current ResNetLatentEEG architecture being overly complex. However, we lack the systematic testing to conclude anything from it. Refer to Appendix B.1 for a comparison of loss plots between the simple and complex ResNetLatentEEG.

5.2.3 Adaptation of Model Architectures to New Domains

While our thesis specifically has been about attempting to reconstruct EEG responses from perceived stimuli, moreover, it has been an experimental showcase of domain adaptation in deep learning. Different deep learning architectures have in many cases been proven particularly efficient for certain types of tasks, such as CNNs for image analysis and classification. As both knowledge within the field of deep learning and other areas influenced by the development of deep learning expands, the possibilities of how existing architectures can be utilized expand alongside it.

Through our experimentation of adapting known architectures such as MLPs, CNNs, and autoencoders to an EEG encoder model task, one of the primary challenges we encountered was finding an accurate representation of EEG responses within the frameworks of our chosen model architectures. With EEG data inheriting both temporal and spatial dimensions, analysis of such data requires thoughtful considerations of various factors. We have attempted to be considerate of these factors, for example through our choice of evaluation metrics and models. However, our results reflect how it has not been adequate to achieve a similar EEG encoding from images to that of the human brain.

Our ResNetLatentEEG model serves as an excellent example of the challenge in accurately representing EEG responses. The model mainly consists of three parts: the ResNet CNN, intermediary linear layers, and the TCNAE decoder. The ResNet CNN and TCNAE decoder both effectively solve their respective tasks. ResNet-18 solves its specific task very well, as outlined in Section 5.2.1. Similarly, our autoencoder is very effective in its decoding of a latent representation of EEG. While it encounters some challenges with reconstruction frequencies, it manages to capture the most important features of the signal. Other than striving for even smaller latent representations, there is not many aspects which can drastically improve its performance further. It could be interesting to experiment further with this model, as it achieves promising results.

For better domain adaptation of existing models, the remaining challenge is to bridge their domains. This must happen through complex enough architectures that encapsulate the brain's encoding of images to EEG, while avoiding overfitting the inherent noise present in the task. However, the complexity does not necessarily need to come from added layers, but may just as well come from more sophisticated connections and/or inputs.

5.2.4 Probabilistic Approach

The combination of ResNet-18 and the TCNAE decoder is the most complex model design experimented with in this study. However, we briefly considered other approaches, one

of them being a lean into the generative aspect of reconstruction by using probabilistic generative models.

Throughout our study, a persistent concern has been the numerous factors that can cause EEG responses for a certain perceived stimulus to differ significantly. These factors include the individual subject's response, the preceding image sequence, and various small details in the recording environment. Each of these elements, among others, contributes to the variability of the response pattern, indicating that the exact image being processed by the visual cortex does not solely determine the EEG response. Consequently, the distribution of EEG responses for a given target image can vary greatly. Adapting our use of deep learning model architectures for the large variance in the response patterns, a possible change in approach could be to change the reconstruction task from a regression approach to a generative one. By using a probabilistic generative model instead of a deterministic one, the model could maybe account better for the variance by sampling from a learned distribution. The challenging part would then be to determine, how such a distribution could be learned.

5.3 Evaluation Metrics

A large part of our study has focused on the choice of evaluation metrics for the reconstruction task. From early on, we identified the challenges in determining similarities in time series data and the need for a metric, which can account for the temporality in EEG data. However, we do not believe we have found a particularly efficient evaluation metric for our specific setting.

5.3.1 Challenges of Current Metrics

MSE and RMSE

MSE and RMSE particularly fail to encapsulate the potential time shifts in the EEG. Time series which appear to be temporally aligned may have experienced delays or latency causing a misalignment of points from their actual time of occurrence. There may also be cases, where the general shape of two response patterns are similar, but differ in amplitude. In such cases, MSE/RMSE would not capture the underlying similarities very well, but instead only focus on the, maybe less important, smaller differences in amplitude. Besides the flaws of MSE/RMSE, it remains the easiest to interpret and also proved to be the most stable loss function, with the least amount of irregularities across configurations.

Sinkhorn

The Sinkhorn metric inherits a different set of flaws that reduce its usefulness as an evaluation metric and loss function. Its most significant shortcoming is its interpretability. Compared to MSE/RMSE, it is rather unintuitive and difficult to deduce the impact of the difference in magnitude. We realize that with a better initial understanding of its mathematical definition, we could have found it easier to interpret. Because of its limitation in interpretability for this study, our evaluation of Sinkhorn relies mostly on the correctness ceiling observed in the TCNAE results, where we know the task has been efficiently solved. Even so, we had cases where a relatively low Sinkhorn was evaluated, despite other metrics indicating worse performance. Sinkhorn also proved undesirable as a loss function for model training, because of inconsistencies in the weight optimization, such as exploding values or volatile spiking between epochs. Additionally, ResNetEEG experienced configurations where computing Sinkhorn resulted in `Infinite` values, consequently ending the training process. This effect only occurred when training ResNetEEG with all four modules on the two smallest EEG subsets.

Algonauts Challenge Score

For the Algonauts Challenge Score, the challenge lies within the adaptation of the metric to be used in the context of EEG data. The inclusion of channels and specific time points in place of voxel indices changes the nature of the metric enough for it to not be directly comparable to the original score. With the original metric, it is quite intuitive to compute the Pearson correlation coefficient for a specific voxel over all stimuli images. In our context of using EEG data, the response values belong to separated channels and are highly dependent on other temporally close response values given the nature of time series data. Our modifications to the metric ended up inheriting the same flaw as with MSE/RMSE, namely that the individual time point correlations are susceptible to misalignment due to time shifts within the data. We also did not include noise ceilings of the channel data in our modifications, as we experienced similar challenges in how a noise ceiling that is related to each channel can be included. In short, the inclusion of our computed noise ceilings did not make sense numbers-wise, which resulted in us omitting it entirely. These factors all contribute to our skepticism of the presented MACS results.

5.3.2 RSA being intended for fMRI

With RSA originally designed for the analysis of fMRI encodings (Kriegeskorte, Mur and Bandettini, 2008), limitations are unavoidable when adapting for EEG data because of the added temporal dimensionality, which adds complexity to the analysis.

When computing Representational Dissimilarity Matrices (RDMs), we rely on a metric that accurately captures differences between signals, where temporal deviation may be a factor. In our case, we opt for RMSE because of its computational efficiency. However, this choice comes with the risk of calculating misleading distances. Specifically, RMSE may not fully account for the temporal structure and non-linear relationships present in the data. Sinkhorn presents itself as a potential solution, although, the ambiguity of this metric makes it difficult to interpret and compare results properly. Additionally, Sinkhorn was not included in the RSA because of its computation expensiveness relative to RMSE.

5.3.3 Estimating Model Performance through Noise Ceiling

A noise ceiling is an estimate of the upper limit of predictive performance that any model can achieve on a dataset given the variability and noise present in its data. The noise ceiling makes interpretation of model performance much easier because it provides the context of where the limit lies. This could be particularly useful when working with EEG, notorious for its low signal-to-noise ratio.

The noise ceiling also allows for the normalization of performance based on the upper limit, e.g. as seen in the Algonauts Challenge (Gifford et al., 2023). To mimic this, we initially attempted to compute noise ceilings representative of our data, however, several limitations came to light during this process, which ultimately led to disregarding the use of a noise ceiling.

Since we use data from only a single subject, we need a noise ceiling matching this. This would require repeated recordings of responses to the same image to assess the consistency in signals for the same image. Since each subject only sees each image once, finding the intra-subject variability from the data is impossible as-is. From here, the natural thing would be to compute the inter-subject variability instead, which gives a noise ceiling across subjects. This would likely be representative of the task, but not entirely accurate as EEG responses are very subjective. However, setting up the pipeline to handle this amount of data and compute noise ceiling was deemed infeasible due to time constraints.

As a final resort, we computed the noise ceiling and -floor between EEG signals, instead of image to EEG signals. The noise ceilings were computed by splitting the samples into two halves containing equal amounts of samples from each class. For each sample from a given class in one half of the split, we compute their correlation with a sample from the same class in the other half. We do this with several random permutations and take the mean of the resulting correlations. Finally, we take the channel-wise mean of correlations to arrive at noise ceilings of just below 0.20 for all 11 channels, which indicates that even in the best-case scenario, a model can only explain or predict 20% of the variance in the EEG signals from

EEG signals. The noise floor is computed similarly, with the exception that we split the data randomly instead of ensuring an even class distribution between the splits. With noise floors of approximately 0.003 for all 11 channels, it indicates that a model should *at least* be able to predict 0.3% of the EEG signals from the EEG signals. While these numbers are not unreasonable given the low signal-to-noise ratio of EEG, we ultimately decided not to introduce them into our performance pipeline, as they represent a task much different from ours. Refer to Appendix J for the computed noise estimates.

5.3.4 Fluctuation Feature Analysis

In search of a proper evaluation metric, we also implemented the *time series similarity measure*, as defined in H. Chen and Gao (2020), which is based on fluctuation feature analysis. This approach identifies the most significant fluctuations within two series and establishes mappings between them. Notably with this approach, the most significant fluctuations may not align temporally across both time series, forcing temporal shifts in the fluctuation mappings. Consequently, we select a mapping that minimizes the distance between matched fluctuation pairs from the two time series. With this approach, the objective was to enable more rational comparisons between our reconstructed EEG responses and the targets.

However, the shortcomings of this time series similarity measure quickly became evident:

1. The complexity of the approach results in a very high computational cost, which makes it infeasible to employ as a loss function during model training.
2. The metric is heavily dependent on the structure of the two series it draws comparisons between. Through the filtering and downsampling of our preprocessing pipeline, we greatly reduce the amount and amplitude of fluctuations in the signals. This also results in inconsistency, since the difference in the amount of fluctuations becomes too large. Since the metric relies on these fluctuations, it becomes ineffective.
3. A concern with the metric was the uncertainty in how to map fluctuation points between the two series as this wasn't explicitly stated in its definition. We briefly experimented with different greedy and a dynamic programming approach, as visualized in Figure 5.2. The visualization also highlights the shortcomings of having few fluctuations.

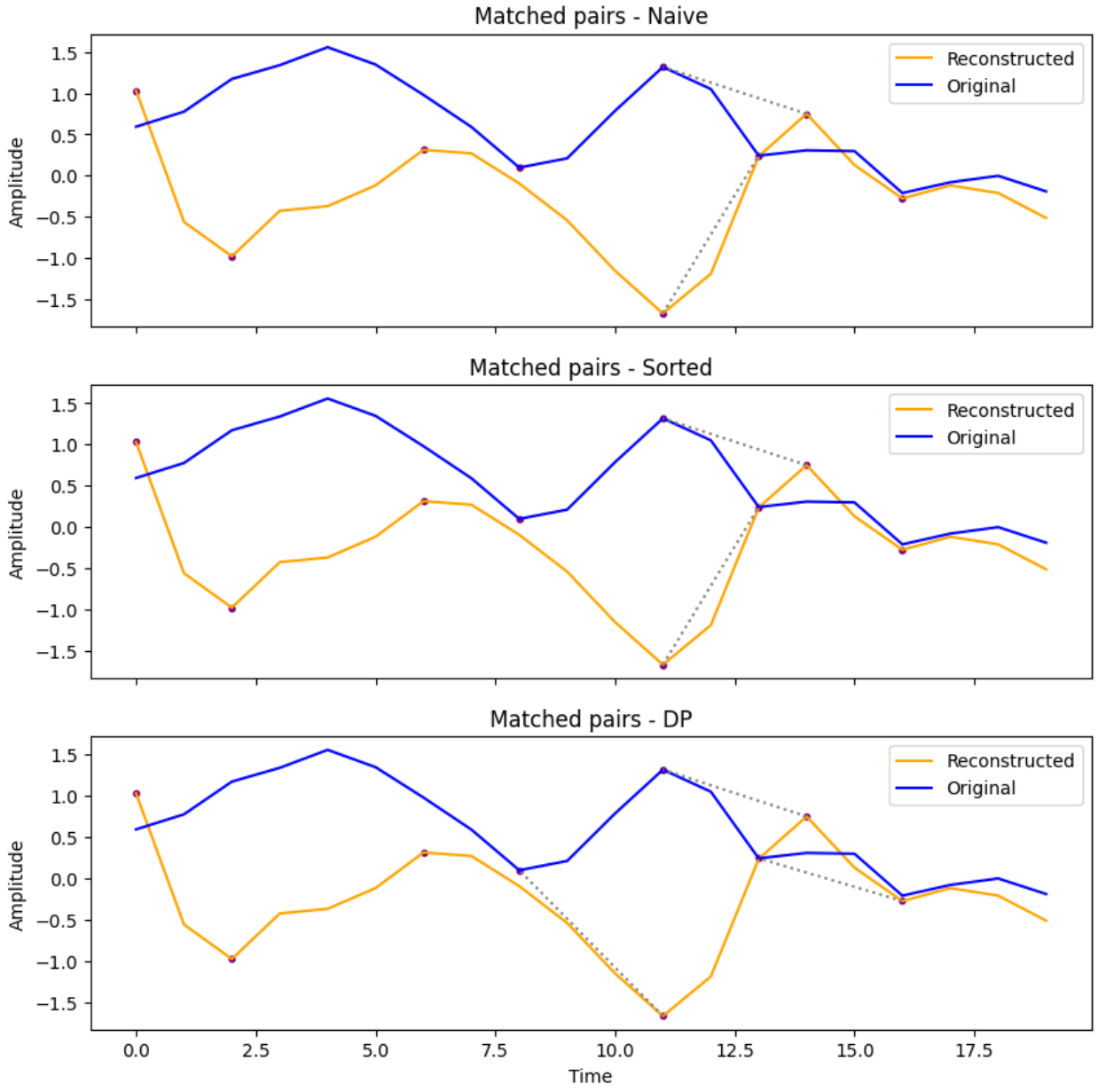


Figure 5.2: Demonstration of the computed mappings between a reconstructed and target preprocessed EEG response using the time series similarity measure method. The first plot shows a naive, or greedy, approach with the algorithm always choosing the closest time point across the time dimension as the matching point. The second plot shows a similar approach, but the mapping is determined by sorting all points in terms of their distance in both the time and amplitude dimensions. The third plot shows an approach using dynamic programming, where the goal of the algorithm is to minimize the total distance between all mappings. As can be seen from all plots, the algorithm has difficulties computing fluctuation points (marked by the points having a mapping to another point) when there exists either a limited or skewed amount of extreme points between the time series. The three approaches thus do not result in different pairings for most comparisons.

5.3.5 CLIP Loss

For future work, an interesting loss function to include is the CLIP loss used by Benchetrit, Banville and King (2023). They use a combination of CLIP and MSE losses to train models that benefit from both training objectives of the metrics. While their task is to reconstruct images from MEG signals, they specifically mention applying the loss function in both brain-to-image and image-to-brain directions. This supports the possibility of the metric being useful for the task of reconstructing EEG responses from images.

5.4 Initial Inspection of the Dataset

Due to the highly experimental approach of this study, our final results could potentially have benefited from a deeper initial analysis and inspection of the data. Initial inspection of the data is typically done in machine learning for reasons including, but not limited to:

1. **Feature Selection**: Initial data inspection can help identify which features are essential for solving the task at hand. By examining the data closely, we can get a better understanding of the most relevant aspects of both image and EEG data. Ultimately, this makes it easier to select features that the model should focus on. As an example, a technique such as Principal Component Analysis (PCA), could help identify the most significant features that capture the maximum variance in the data.

2. **Task Feasibility Estimate**: Thorough data analysis can help with estimating how well the task can be solved with the given data. It helps in identifying potential patterns, correlations, and anomalies that could influence the model's performance. This step is crucial in understanding the limitations and strengths of the dataset and can provide a better understanding of the final results.

3. **Model Development and Architecture**: Observations made during initial data inspection can make choices regarding the development of model architectures and processes more informed. For example, specific patterns or structures in the data might suggest the use of particular types of models or guide the design of model layers such as convolutional filter parameters.

In our case, a more thorough initial data inspection might have helped us arrive at more tailored preprocessing steps to improve the signal-to-noise ratio or remove inherent noise or bias, provide better reasoning for interval and channel subset selection, or highlight if certain visual features of images correlate with specific EEG patterns. We highlight this as a point of improvement to consider for future work.

5.5 Limitations of the THINGS-EEG Dataset

During the study, we also experienced some limitations of using the THINGS-EEG dataset for the reconstruction task.

5.5.1 Compatibility of Analysis Approach and Dataset Recording Format

One of the underlying restrictions we see with our current approach is the compatibility between the recording format of the dataset and our methodology of analyzing the possible connection of the visual cortex and deep neural networks. With the dataset containing EEG responses recorded in RSVP format, each image is only seen by the subject for 50ms. Furthermore, based on our previous mention of generic feature processing happening in the initial visual cortex 30-40ms after stimulus onset, we made the choice of using data samples focused on the time interval of 50ms after stimulus onset to 250ms. By comparing the chosen time interval with the recording format, it is clear to see that our chosen time interval spans the EEG response to multiple stimulus cycles. As a result, the intervaled EEG response corresponds to the brain activity of both the focused stimuli and subsequent stimuli, visualized in Figure 5.3. Thus, one can question if it even makes sense to analyze responses in the later part of the interval (from 100ms after stimulus onset and onwards), as new stimuli is already being presented at this point and processed by the visual cortex.

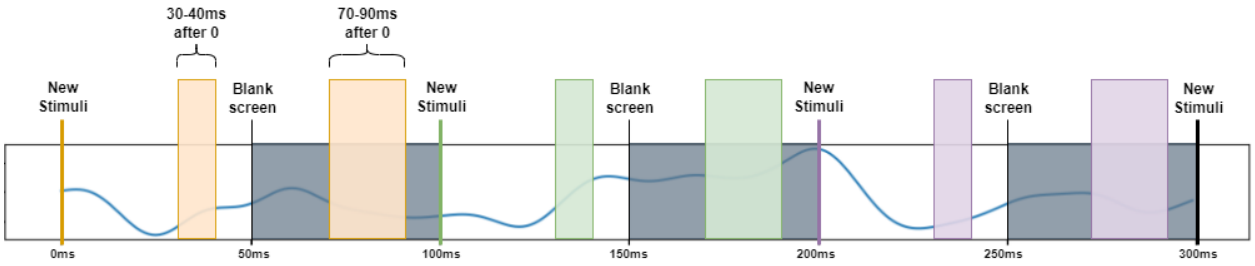


Figure 5.3: Demonstration of the recording format of the THINGS-EEG data, shown with an example EEG response in a time interval between start of stimulus onset and the proceeding 300ms. Each different color represents a new stimuli being presented. The colored columns showcase the estimation of when the visual cortex starts the processing of features (roughly 30-40ms after stimulus onset) and object recognition (roughly 70-90ms after stimulus onset).

Our current analysis involves finding a possible relation between the processing of stimuli through different subsets of CNN layers and the resulting EEG reconstruction abilities. Due to the discussed uncertainty surrounding the analysis of the latter parts of the defined interval, we have reservations about the ability to identify reliable patterns related to the presented stimuli in the 100-250ms range of the EEG response. We might be unable to

expect, that the information seen in the channel subsets recording the activity from the ventral and dorsal pathways of the visual cortex are completely reflected in which stimulus was shown. Given that we can see responses happening in the `T7` and `T8` channels, we know the brain is processing information at this level. However, with new stimuli being presented every 100ms, new information is already being passed to the occipital region by the time object recognition and categorization are processed by the later sections. This indicates that, due to the rapid presentation cycle of stimuli, the brain is processing multiple stimuli at different stages of the visual cortex simultaneously. The recorded EEG responses from a subject watching a target image, specifically the later channel subsets, can be expected to be a result of the brain processing the target image, but also previous images seen. This proposal is also supported by H. Yang, Gee and Shi (2023), where brain responses correlate with 6-7 images before the targeted image. Based on this, the order of the presented stimuli seems to be an important factor in how a response pattern looks for target stimuli. A question then arises of how much influence a target stimuli has on the recorded EEG response in the `T7` and `T8` channels, compared to the order and in which previous stimuli were presented.

With our current approach, we do not have the means to answer this question. However, our results from attempting to reconstruct EEG responses do point towards the reconstruction task being inherently difficult. It could partly be due to insufficient models for the task, but our results highly suggest that the inherent noise in the EEG data and lack of correlation within has a large impact as well. This opens up for exploring other approaches for the reconstruction task, or other signal analysis tasks in general. One specific idea, related to the influence of previous stimuli on the targeted response pattern, could be about expanding the input space to increase its informational capacity. Instead of only relying on the targeted image as input for the reconstruction models, the input could include a set of previous images in addition to the target image. Further information and constraints could also be included and experimented upon, such as the class of the images or even the next images to be shown. This approach would give more context to the model, which might make the reconstruction task, or similar signal analysis tasks, more feasible, given the brain response correlations to previously seen images.

Slightly expanding upon this idea, another approach is to focus on the EEG responses instead. Experimenting with using previous EEG response states as an addition to the input could provide useful information for the model task. From general knowledge of EEG response patterns and the rapid nature of the recording format, EEG responses are intricately intertwined rather than existing in isolation. Each response pattern is not independent in the sense that it is unaffected by how a brain response looks before or after. Instead, there's a dynamic interaction between the end of one response influencing the start of another. A single response window is dependent on other temporally close responses, forming a continuous

pattern rather than being disjoint. These interconnections emphasize the importance of considering the temporal aspects in EEG analysis, supporting the effectiveness of recursively using previous EEG responses as part of the model input.

Apart from these reflections on the recording format of the THINGS-EEG dataset, it could be interesting to compare the results of our current approach on a dataset where the stimuli presentations lasted much longer and had a longer resting period between stimuli. This would give the subjects more time to inspect and process an image and could help isolate a response pattern to a specific image, better suiting our approach of studying specific responses for individual images. The idea is that previous response patterns could have time to 'diminish' before presenting a new image, indicating a "reset" of the visual cortex back to a "default" state determined by the blank image in the resting period.

5.5.2 Dataset Size and Cross-Subject Training

Our decision to create even class distributions within the THINGS-EEG dataset and to focus model training on data from a single subject out of the available 50 resulted in a limited training data size of approximately 18,000 samples after splitting into training, validation, and test sets. Training complex deep learning models on such a limited dataset poses a risk of generalization issues, as the lack of variability might not adequately capture the complexity of the underlying patterns. To improve generalization, it would have been crucial to increase the variance within the image classes associated with EEG responses. Given the vast potential variance in real-life images of any concept, a model requires extensive coverage of this variance in the training data to generalize effectively. As the dataset only included 12 unique images of each class, the coverage of possible variance could definitely be better.

While increasing the data size by incorporating data from more subjects is an option, we chose not to do so due to the issue of inter-subject variability. Differences in brain activity across subjects for identical stimuli are inevitable and introduce much new noise across subjects that the models need to distinguish between. To combat this, we would need to introduce more sophisticated preprocessing techniques to achieve a more consistent performance, for example standardizing data across different subjects. With the lack of time to explore this in-depth, we opted to only use data from a single subject.

6 Conclusion

In this thesis, we have explored a domain adaptation of deep learning model architectures, specifically convolutional neural networks (CNNs), to the domain of EEG signal reconstruction from visual stimuli.

Our experiments employed the use of three distinct image-to-EEG models: MLPEEG, a simple MLP acting as a baseline; ResNetEEG, the ResNet architecture modified to predict EEG directly; and ResNetLatentEEG, a novel approach of combining the image feature extraction of ResNet with the decoding capabilities of a temporal convolutional network autoencoder (TCNAE). The models were evaluated in their encoding performance based on three different metrics providing three distinct perspectives: RMSE, Sinkhorn, and the modified Algonauts Challenge Score. Additionally, we analyzed the encoding similarity compared to the brain for each model through representational similarity analysis (RSA).

Our results underscore significant challenges in reconstructing EEG from visual stimuli using our approach. Across all our image-to-EEG models, we observe notably poor performance on the reconstruction task. The performance of our ResNetLatentEEG model hinges on the effectiveness of the TCNAE, which displays remarkable efficiency in encoding and decoding of EEG signals. The poor performances of all image-to-EEG models are largely attributed to overfitting and insufficient regularization. While the models leveraging the CNN architecture of ResNet-18 demonstrate an ability to learn from the training data, their reconstructive capabilities are notably limited when presented with unseen data. This indicates that the challenges in EEG reconstruction are more related to the characteristics of EEG data, as well as issues in data collection and preprocessing, rather than the specific use of CNNs for feature extraction. Experimentation with different modules of ResNet-18 also highlights the effectiveness of the image feature extraction process in CNNs, as shallower modules with more low-level features perform significantly worse than random chance on the EEG reconstruction task. Additionally, we see no notable difference in performance between reconstruction tasks on different subsets of EEG channels. The performance is identical across the three models, and naturally worse on larger subsets because the task becomes more complex. In terms of the encoding similarity as measured by RSA, the encodings from our models show virtually no correlation with those of the brain, emphasizing the ineffectiveness of our approach.

Using EEG as a neuroimaging technique for signal reconstruction presents several significant challenges. The low signal-to-noise ratio of EEG, combined with its variability, necessitates the need for sophisticated preprocessing techniques to extract sufficient meaningful information from the signal. Weak and potentially temporally shifted correlations between observed EEG responses and visual stimuli limit the types of data that can be effectively used for EEG reconstruction. The inherent time series nature of EEG data also complicates the use of effective evaluation metrics, further challenging the training of deep learning models for accurate signal reconstruction.

One of the most significant obstacles of this thesis introduced itself as the lack of comprehensible evaluation metrics to make meaningful comparisons between EEG signals. Commonly used metrics such as RMSE/MSE fail to capture any possible temporal complexities in EEG data. The Sinkhorn distance (an approximate of the Wasserstein distance) may capture temporality through comparison of EEG distributions, however, it leaves a lot to be desired in terms of interpretability. The modified Algonauts Challenge Score gives us a reference to similar reconstruction tasks of fMRI, however, when modifying it for use in EEG reconstruction, it inherits the same flaw of potential temporal misalignment in its comparisons as seen with RMSE/MSE. The implementation of a fluctuation-based similarity measure showed potential in EEG comparison but was ultimately disregarded for its computational cost and incompatibility with our preprocessing pipeline.

In conclusion, this thesis underscores the significant difficulties in reconstructing EEG signals from visual stimuli. The poor performance across all models highlights the need for further research into model architecture experimentation, EEG preprocessing, and identification (or development) of evaluation metrics that better encapsulate the complexity and noise of EEG data. By addressing these challenges, future investigations can advance our understanding of the interplay between visual perception and brain activity, ultimately leading to more effective and accurate methods for EEG signal reconstruction.

Bibliography

Adeli, Hossein, Sun Minni and Nikolaus Kriegeskorte (2023). 'Predicting brain activity using Transformers'. In: *bioRxiv*. DOI: 10.1101/2023.08.02.551743. eprint: https://www.biorxiv.org/content/early/2023/08/05/2023.08.02.551743.full.pdf. URL: https://www.biorxiv.org/content/early/2023/08/05/2023.08.02.551743.

Alhagry, Salma, Aly Aly Fahmy and Reda A El-Khoribi (2017). 'Emotion recognition based on EEG using LSTM recurrent neural network'. In: *International Journal of Advanced Computer Science and Applications* 8.10.

Allen, Emily J et al. (2022). 'A massive 7T fMRI dataset to bridge cognitive neuroscience and artificial intelligence'. In: *Nature neuroscience* 25.1, pp. 116–126.

Bashivan, Pouya et al. (2015). 'Learning representations from EEG with deep recurrent-convolutional neural networks'. In: *arXiv preprint arXiv:1511.06448*.

Benchetrit, Yohann, Hubert Banville and Jean-Rémi King (2023). 'Brain decoding: toward real-time reconstruction of visual perception'. In: *arXiv preprint arXiv:2310.19812*.

Chen, Hailan and Xuedong Gao (2020). 'A New Time Series Similarity Measurement Method Based on Fluctuation Features'. In: *Tehnički vjesnik* 27.4, pp. 1134–1141.

Cichy, Radoslaw Martin et al. (2019). 'The algonauts project: A platform for communication between the sciences of biological and artificial intelligence'. In: *arXiv preprint arXiv:1905.05675*.

Cohen, Israel et al. (2009). 'Pearson correlation coefficient'. In: *Noise reduction in speech processing*, pp. 1–4.

Craik, Alexander, Yongtian He and Jose L Contreras-Vidal (2019). 'Deep learning for electroencephalogram (EEG) classification tasks: a review'. In: *Journal of neural engineering* 16.3, p. 031001.

Défossez, Alexandre et al. (2023). 'Decoding speech perception from non-invasive brain recordings'. In: *Nature Machine Intelligence* 5.10, pp. 1097–1107.

Deng, Jia et al. (2009). 'Imagenet: A large-scale hierarchical image database'. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

DeVries, Terrance and Graham W Taylor (2017). 'Improved regularization of convolutional neural networks with cutout'. In: *arXiv preprint arXiv:1708.04552*.

Farnsworth, Bryn (2019). 'EEG (electroencephalography): The Complete Pocket Guide'. In: *IMotions, Global HQ: Copenhagen, Denmark*.

Feydy, Jean et al. (2019). 'Interpolating between Optimal Transport and MMD using Sinkhorn Divergences'. In: *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690.

Gable, Philip, Matthew Miller and Edward Bernat (2022). *The Oxford handbook of EEG frequency*. Oxford University Press.

Gaziv, Guy (2019). 'Learning to aggregate feature representations'. In: *arXiv preprint arXiv:1907.01034*.

Gifford, Alessandro T et al. (2023). 'The algonauts project 2023 challenge: How the human brain makes sense of natural scenes'. In: *arXiv preprint arXiv:2301.03198*.

Goodfellow, Ian, Yoshua Bengio and Aaron Courville (2016). *Deep learning*. MIT press.

Grootswagers, Tijl et al. (2022). 'Human EEG recordings for 1,854 concepts presented in rapid serial visual presentation streams'. In: *Scientific Data* 9.1, p. 3. DOI: 10.1038/s41597-021-01102-7.

Gu, Jiuxiang et al. (2018). 'Recent advances in convolutional neural networks'. In: *Pattern Recognition* 77, pp. 354–377. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.10.013. URL: https://www.sciencedirect.com/science/article/pii/S0031320317304120.

Haas, Lindsay F (2003). 'Hans Berger (1873–1941), Richard Caton (1842–1926), and electroencephalography'. In: *Journal of Neurology, Neurosurgery & Psychiatry* 74.1, pp. 9–9.

He, Kaiming et al. (2016). 'Deep residual learning for image recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hebart, Martin N et al. (2023). 'THINGS-data, a multimodal collection of large-scale datasets for investigating object representations in human brain and behavior'. In: *Elife* 12, e82580.

Hebart, Martin N. et al. (2019). 'THINGS: A database of 1,854 object concepts and more than 26,000 naturalistic object images'. In: *PLoS ONE* 14.10, e0223792. DOI: 10.1371/journal.pone.0223792. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0223792.

Huang, Gao et al. (2018). *Densely Connected Convolutional Networks*. arXiv: 1608.06993 [cs.CV].

Iandola, Forrest N. et al. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. arXiv: 1602.07360 [cs.CV].

Jasper, Herbert H (1958). 'Ten-twenty electrode system of the international federation'. In: *Electroencephalogr Clin Neurophysiol* 10, pp. 371–375.

Kingma, Diederik P and Jimmy Ba (2014). 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980*.

Koelstra, Sander et al. (2011). 'Deap: A database for emotion analysis; using physiological signals'. In: *IEEE transactions on affective computing* 3.1, pp. 18–31.

Kriegeskorte, Nikolaus, Marieke Mur and Peter A Bandettini (2008). 'Representational similarity analysis-connecting the branches of systems neuroscience'. In: *Frontiers in systems neuroscience* 2, p. 249.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E Hinton (2017). 'ImageNet classification with deep convolutional neural networks'. In: *Communications of the ACM* 60.6, pp. 84–90.

Kruger, Norbert et al. (2012). 'Deep hierarchies in the primate visual cortex: What can we learn for computer vision?' In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1847–1871.

Kuanar, Shiba et al. (2018). 'Cognitive analysis of working memory load from EEG, by a deep recurrent neural network'. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 2576–2580.

Kumar, Pradeep et al. (2018). 'Envisioned speech recognition using EEG sensors'. In: *Personal and Ubiquitous Computing* 22, pp. 185–199.

Lawhern, Vernon J et al. (2018). 'EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces'. In: *Journal of neural engineering* 15.5, p. 056013.

LeCun, Yann, Yoshua Bengio and Geoffrey Hinton (2015). 'Deep learning'. In: *nature* 521.7553, pp. 436–444.

Lin, Wenqian, Chao Li and Shouqian Sun (2017). 'Deep convolutional neural network for emotion recognition using EEG and peripheral physiological signal'. In: *Image and Graphics: 9th International Conference, ICIG 2017, Shanghai, China, September 13-15, 2017, Revised Selected Papers, Part II 9*. Springer, pp. 385–394.

Meijer, Anne-Ruth José and Arnoud Visser (2019). 'A shallow residual neural network to predict the visual cortex response'. In: *arXiv preprint arXiv:1906.11578*.

Al-Nafjan, Abeer et al. (2017). 'Classification of human emotions from electroencephalogram (EEG) signal using deep neural network'. In: *Int. J. Adv. Comput. Sci. Appl* 8.9, pp. 419–425.

Nguyen, Xuan-Bac et al. (2023). 'The Algonauts Project 2023 Challenge: UARK-UAlbany Team Solution'. In: *arXiv preprint arXiv:2308.00262*.

Olah, Chris, Alexander Mordvintsev and Ludwig Schubert (2017). 'Feature Visualization'. In: *Distill*. https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

Olah, Chris, Arvind Satyanarayan et al. (2018). 'The Building Blocks of Interpretability'. In: *Distill*. https://distill.pub/2018/building-blocks. DOI: 10.23915/distill.00010.

Oostenveld, Robert and Peter Praamstra (2001). 'The five percent electrode system for high-resolution EEG and ERP measurements'. In: *Clinical neurophysiology* 112.4, pp. 713–719.

Paszke, Adam et al. (2019). 'Pytorch: An imperative style, high-performance deep learning library'. In: *Advances in neural information processing systems* 32.

Pedregosa, Fabian et al. (2011). 'Scikit-learn: Machine learning in Python'. In: *the Journal of machine Learning research* 12, pp. 2825–2830.

Pereira, Arnaldo E et al. (2018). 'Cross-subject EEG event-related potential classification for brain-computer interfaces using residual networks'. In.

Peyré, Gabriel and Marco Cuturi (2020). *Computational Optimal Transport*. arXiv: 1803.00567 [stat.ML].

Robbins, Herbert and Sutton Monro (1951). 'A stochastic approximation method'. In: *The annals of mathematical statistics*, pp. 400–407.

Rosenblatt, Frank (1958). 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological review* 65.6, p. 386.

Roy, Yannick et al. (2019). 'Deep learning-based electroencephalography analysis: a systematic review'. In: *Journal of neural engineering* 16.5, p. 051001.

Rumelhart, David E, Geoffrey E Hinton and Ronald J Williams (1986). 'Learning representations by back-propagating errors'. In: *nature* 323.6088, pp. 533–536.

Salama, Elham S et al. (2018). 'EEG-based emotion recognition using 3D convolutional neural networks'. In: *International Journal of Advanced Computer Science and Applications* 9.8.

Schrimpf, Martin et al. (2018). 'Brain-score: Which artificial neural network for object recognition is most brain-like?' In: *BioRxiv*, p. 407007.

Shen, Guohua, Kshitij Dwivedi et al. (2019). 'End-to-end deep image reconstruction from human brain activity'. In: *Frontiers in computational neuroscience* 13, p. 432276.

Shen, Guohua, Tomoyasu Horikawa et al. (2019). 'Deep image reconstruction from human brain activity'. In: *PLoS computational biology* 15.1, e1006633.

Singh, Prajwal et al. (2023). 'EEG2IMAGE: image reconstruction from EEG brain signals'. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1–5.

Srivastava, Nitish et al. (2014). 'Dropout: a simple way to prevent neural networks from overfitting'. In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Thill, Markus et al. (2021). 'Temporal convolutional autoencoder for unsupervised anomaly detection in time series'. In: *Applied Soft Computing* 112, p. 107751.

Tirupattur, Praveen et al. (2018). 'Thoughtviz: Visualizing human thoughts using generative adversarial network'. In: *Proceedings of the 26th ACM international conference on Multimedia*, pp. 950–958.

Virtanen, Pauli et al. (2020). 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python'. In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Yanagimoto, Miku and Chika Sugimoto (2016). 'Recognition of persisting emotional valence from EEG using convolutional neural networks'. In: *2016 IEEE 9th International Workshop on Computational Intelligence and Applications (IWCIA)*. IEEE, pp. 27–32.

Yang, Huzheng, James Gee and Jianbo Shi (2023). 'Memory Encoding Model'. In: *arXiv preprint arXiv:2308.01175*.

Yang, Yao-Yuan et al. (2022). 'Torchaudio: Building blocks for audio and speech processing'. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6982–6986.

Yin, Zhong and Jianhua Zhang (2017). 'Cross-session classification of mental workload levels using EEG and an adaptive deep learning model'. In: *Biomedical Signal Processing and Control* 33, pp. 30–47.

A. GitHub Repository

The GitHub repository for this project can be accessed here:

https://github.itu.dk/sibp/Thesis.

B. Loss Visalizations

B.1 Simple vs Complex ResNetLatentEEG

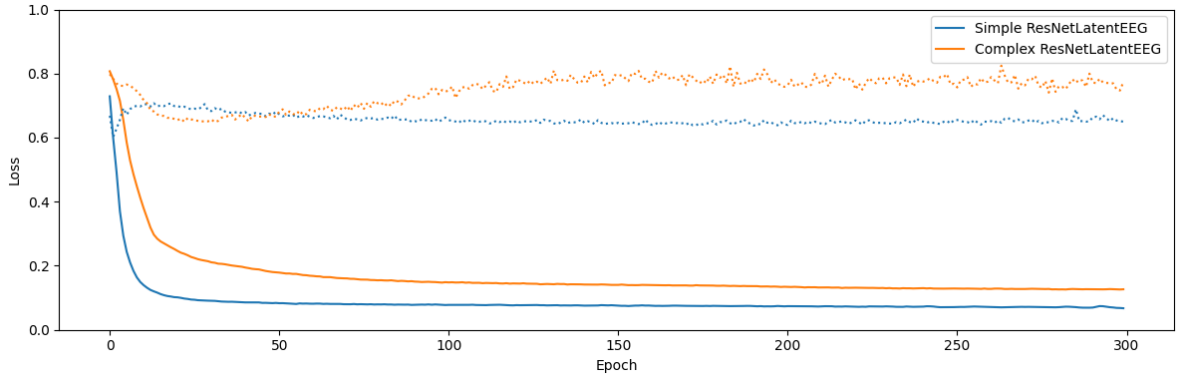


Figure B.1: Visualization of the simple ResNetLatentEEG's loss curves compared to the original, more complex ResNetLatentEEG. The models are trained with the same configuration: 4 ResNet modules and on the 3-channel subset. The solid lines represent training loss and the dotted lines represent validation loss. The simple model showcases fewer signs of overfitting on validation loss, however, still generalizes poorly to unseen data. The simple model also converges quicker and to a lower value than the complex model.

B.2 MLPEEG with and without Dropout

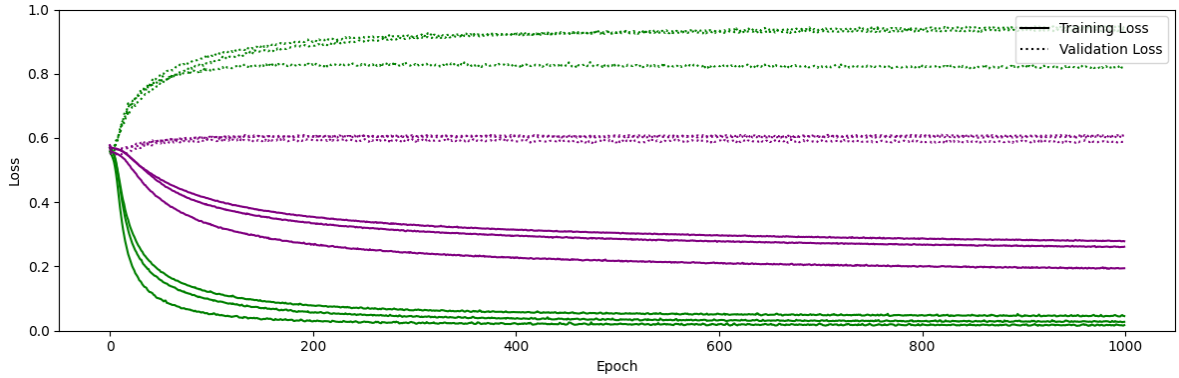


Figure B.2: Comparison of MLPEEG loss with and without dropout layers in the architecture. Green lines denote models trained *without* dropout, while purple lines are models trained *with* dropout. Training time increases with the use of dropout, however, the plot also indicates a clear decrease in performance when dropout layers are introduced.

C. Visualizations of Best and Worst EEG Reconstructions

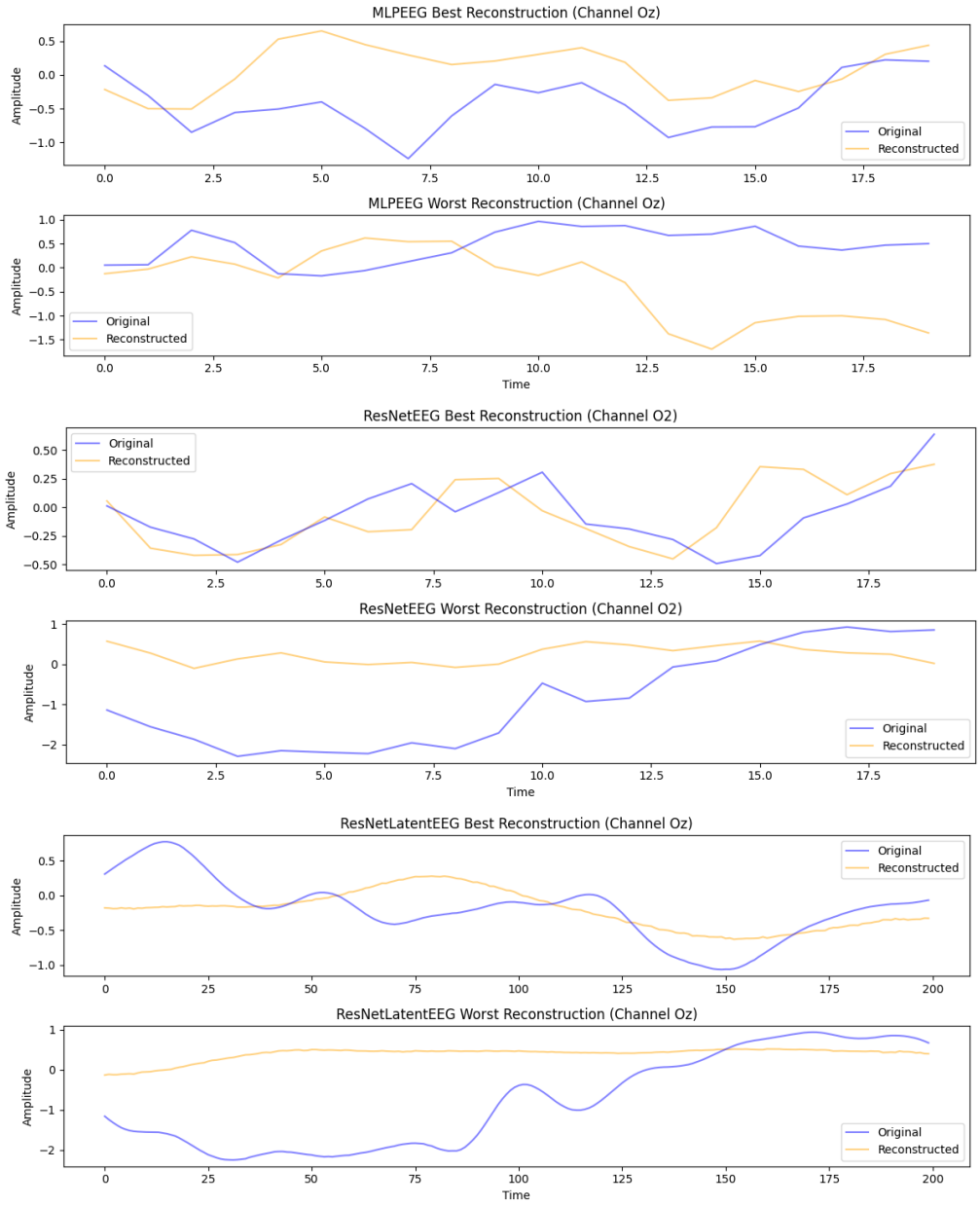


Figure C.1: For the three different models trained on reconstructing 3 EEG channels and using 4 ResNet modules where applicable, the best and worst EEG response reconstruction of the most influential channel. RMSE was used to determine the performance of the reconstructions.

D. MSE Encoding Similarity

Model	Channels	ResNet modules	Pearson
MLPEEG	3	-	-0.018
MLPEEG	5	-	-0.032
MLPEEG	11	-	-0.029
ResNetEEG	3	2	0.034
ResNetEEG	3	3	0.016
ResNetEEG	3	4	0.034
ResNetEEG	5	2	0.004
ResNetEEG	5	3	0.019
ResNetEEG	5	4	0.037
ResNetEEG	11	2	0.015
ResNetEEG	11	3	0.018
ResNetEEG	11	4	0.037
ResNetLatentEEG	3	2	0.008
ResNetLatentEEG	3	3	0.015
ResNetLatentEEG	3	4	0.012
ResNetLatentEEG	5	2	-0.009
ResNetLatentEEG	5	3	-0.001
ResNetLatentEEG	5	4	0.018
ResNetLatentEEG	11	2	0.015
ResNetLatentEEG	11	3	0.011
ResNetLatentEEG	11	4	0.019

Table D.1: Results from performing RSA on our three trained models, MLPEEG, ResNetEEG, and ResNetLatentEEG with different configurations. This includes which channel subset is being reconstructed (3, 5, or 11 channels), and how many ResNet modules are being used (2, 3, or 4) for the two ResNet-based models. All models in this table are trained with MSELoss. The Pearson values between RDMs are virtually 0 for all configurations.

E. Sinkhorn Fidelity Performance

E.1 Sinkhorn Encoding Fidelity

Model	Channels	ResNet modules	RMSE	Sinkhorn	ACS
Random Guessing	-	-	0.77	4.14	0
MLPEEG	3	-	0.85	6.50	3.76
MLPEEG	5	-	0.90	4.07	1.57
MLPEEG	11	-	0.92	4.49	1.91
ResNetEEG	3	2	0.76	3.74	12.09
ResNetEEG	3	3	0.76	3.82	11.68
ResNetEEG	3	4	Inf	Inf	Inf
ResNetEEG	5	2	0.85	4.12	4.43
ResNetEEG	5	3	0.86	4.12	5.94
ResNetEEG	5	4	Inf	Inf	Inf
ResNetEEG	11	2	0.88	3.64	1.13
ResNetEEG	11	3	0.88	5.58	3.04
ResNetEEG	11	4	0.88	3.29	1.98
ResNetLatentEEG	3	2	0.84	47.67	9.03
ResNetLatentEEG	3	3	0.82	50.87	7.06
ResNetLatentEEG	3	4	0.85	36.66	7.45
ResNetLatentEEG	5	2	0.97	62.68	2.32
ResNetLatentEEG	5	3	0.92	53.72	1.97
ResNetLatentEEG	5	4	0.91	55.19	2.66
ResNetLatentEEG	11	2	0.91	38.06	3.09
ResNetLatentEEG	11	3	0.91	55.92	3.27
ResNetLatentEEG	11	4	0.91	47.99	2.26

Table E.1: Evaluation metric results for all models across different configurations. This includes which channel subset is being reconstructed (3, 5, or 11 channels), and how many ResNet modules are being used (2, 3, or 4) for the two models using the ResNet architecture. For each model, the best and worst evaluation metric results are highlighted in green and red. All results shown in this table are from models trained with Sinkhorn as the loss function.

E.2 Sinkhorn Encoding Similarity

Model	Channels	ResNet modules	Pearson
MLPEEG	3	-	0.002
MLPEEG	5	-	0.001
MLPEEG	11	-	0.001
ResNetEEG	3	2	0.01
ResNetEEG	3	3	0.009
ResNetEEG	3	4	N/A
ResNetEEG	5	2	0.009
ResNetEEG	5	3	0.009
ResNetEEG	5	4	N/A
ResNetEEG	11	2	0.0003
ResNetEEG	11	3	0.002
ResNetEEG	11	4	0.0009
ResNetLatentEEG	3	2	0.003
ResNetLatentEEG	3	3	0.004
ResNetLatentEEG	3	4	0.001
ResNetLatentEEG	5	2	0.003
ResNetLatentEEG	5	3	0.003
ResNetLatentEEG	5	4	0.006
ResNetLatentEEG	11	2	0.001
ResNetLatentEEG	11	3	0.002
ResNetLatentEEG	11	4	0.001

Table E.2: Results from performing RSA on our three trained models, MLPEEG, ResNetEEG, and ResNetLatentEEG with different configurations. This includes which channel subset is being reconstructed (3, 5, or 11 channels), and how many ResNet modules are being used (2, 3, or 4) for the two ResNet-based models. All models in this table are trained with Sinkhorn. The Pearson values between RDMs are virtually 0 for all configurations.

F. Time Series Similarity Metric

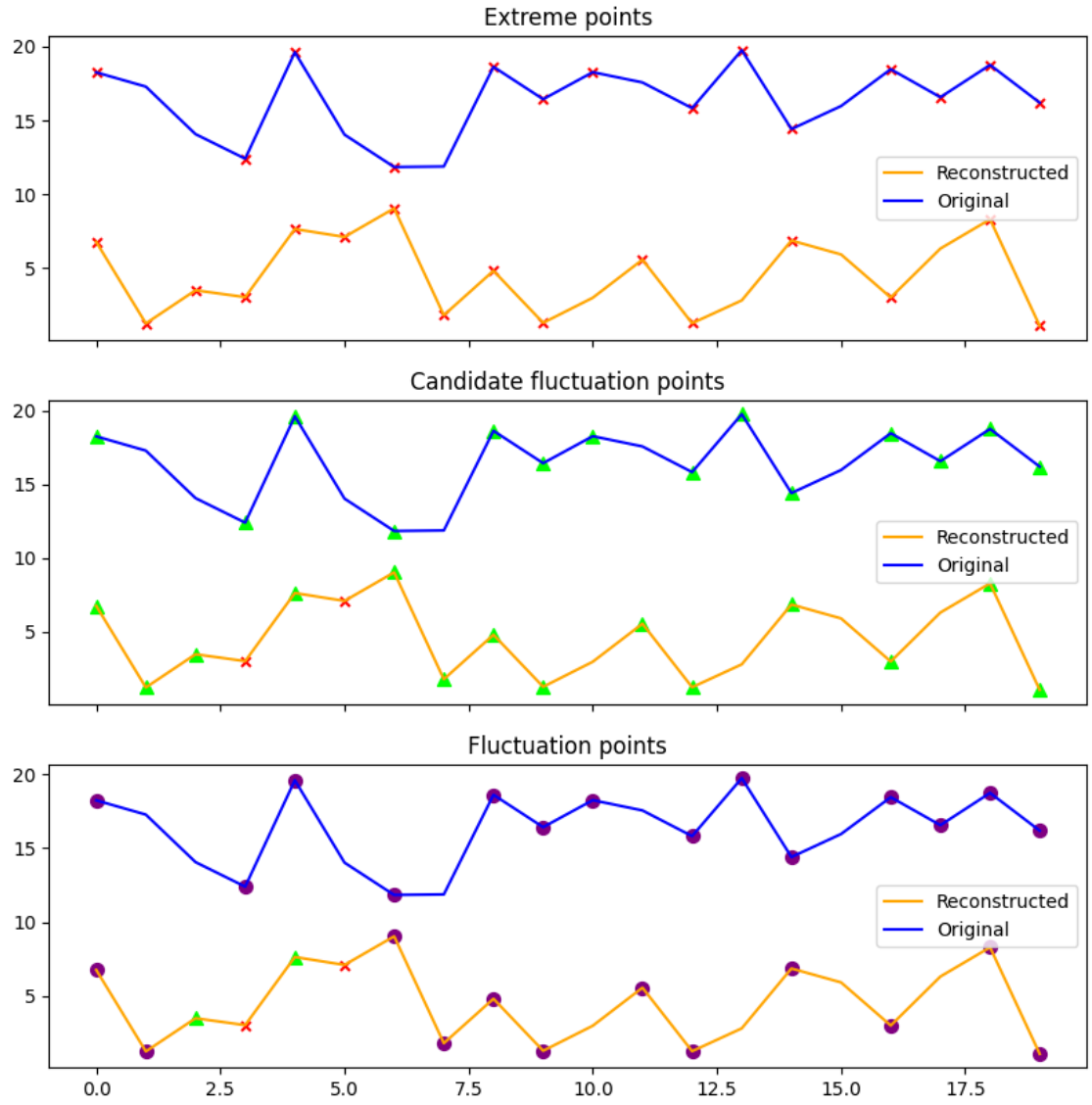


Figure F.1: Visualization of calculating the time series similarity metric that was adapted but later disregarded for this study. Firstly, the extreme points are located (valleys or peaks) which are denoted by red crosses. Then we determine candidate fluctuation points - marked with green triangles - by only including points that differ in amplitude by a certain amount from the previous extreme point. Lastly, the fluctuation points are determined such that no two valleys or peaks follow each other out of the candidates. The final fluctuation points are marked with purple circles.

G. Impact of ResNetEEG Module Configurations on Time Intervals

G.1 5 channels

	2 modules		3 modules		4 modules	
Time Interval	**RMSE**	**ACS**	**RMSE**	**ACS**	**RMSE**	**ACS**
50-150ms	2.55	3.44	0.90	4.34	0.79	6.23
150-250ms	2.46	6.46	0.99	6.67	0.79	9.11
50-100ms	3.10	0.09	0.87	0.82	0.80	1.75
100-150ms	1.83	6.97	0.92	7.85	0.79	10.72
150-200ms	2.51	6.23	0.90	7.93	0.79	9.32
200-250ms	2.41	6.69	1.08	5.41	0.80	8.90

Table G.1: For the ResNetEEG model reconstructing the five-channel subset, {O1, Oz, O2, T7, T8}, and using 2-4 ResNet modules, the RMSE and ACS for separate time windows after stimulus onset of the reconstructed EEG signals on all test set samples. Sinkhorn is excluded, as its magnitude depends on the time points evaluated and would be at a different scale than the other presented Sinkhorn results.

G.2 11 channels

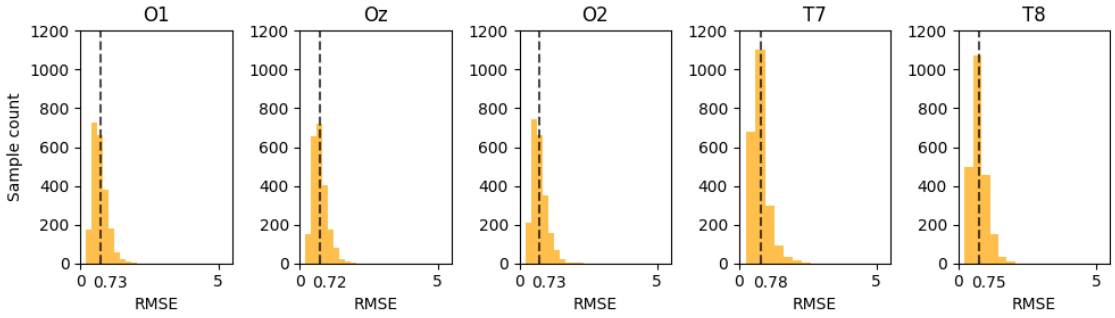
	2 modules		3 modules		4 modules	
Time Interval	**RMSE**	**ACS**	**RMSE**	**ACS**	**RMSE**	**ACS**
50-150ms	1.71	4.41	0.83	4.66	0.80	6.71
150-250ms	1.39	5.34	0.85	6.10	0.79	7.74
50-100ms	1.62	1.49	0.83	1.83	0.81	2.66
100-150ms	1.78	7.34	0.84	7.49	0.79	10.77
150-200ms	1.43	5.30	0.84	7.15	0.79	8.86
200-250ms	1.34	5.38	0.86	5.05	0.80	6.62

Table G.2: For the ResNetEEG model reconstructing the eleven-channel subset, {O1, Oz, O2, T7, T8, TP7, TP8, P7, P8, PO7, PO8}, and using 2-4 ResNet modules, the RMSE and ACS for separate time windows after stimulus onset of the reconstructed EEG signals on all test set samples. Sinkhorn is excluded, as its magnitude depends on the time points evaluated and would be at a different scale than the other presented Sinkhorn results.

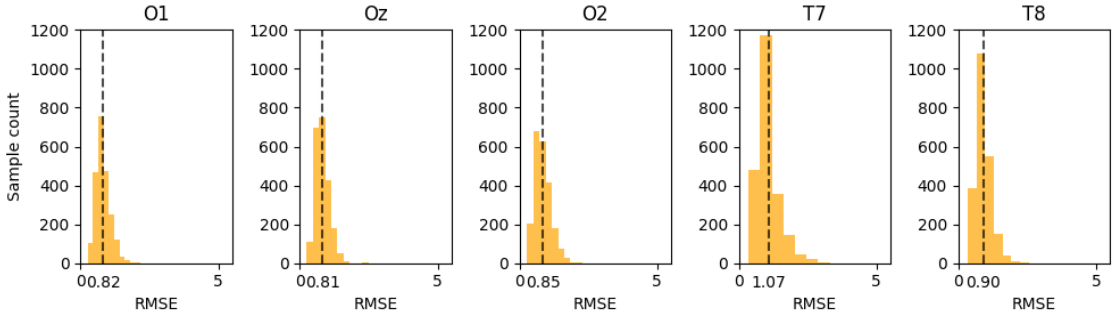
H. Channel-Wise RMSE Distributions for ResNetEEG

H.1 5 channels

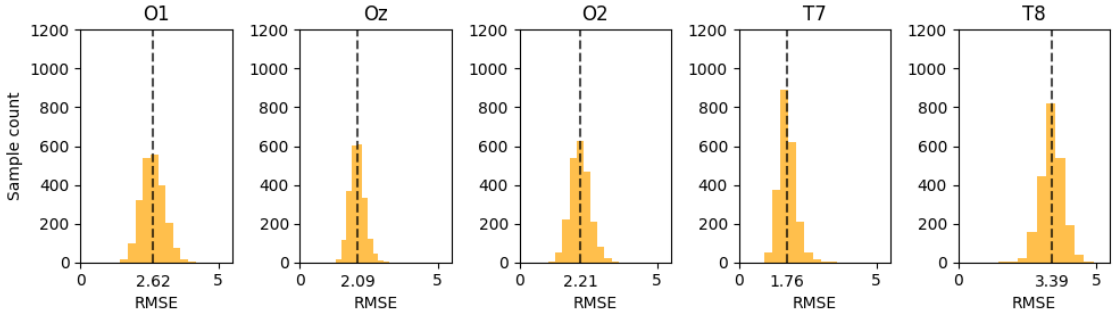
H.2 11 channels



(a) ResNetEEG with 4 modules

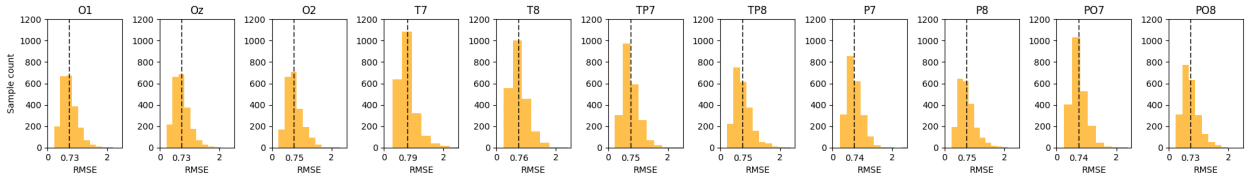


(b) ResNetEEG with 3 modules

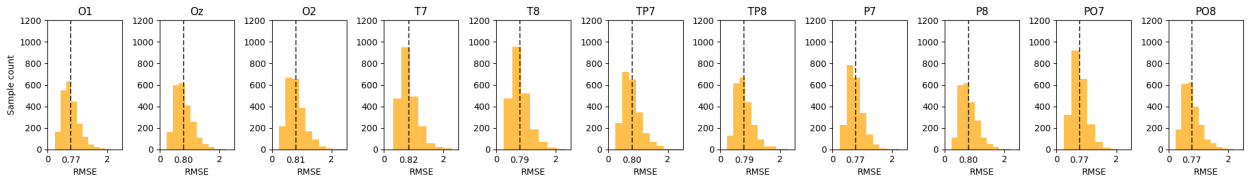


(c) ResNetEEG with 2 modules

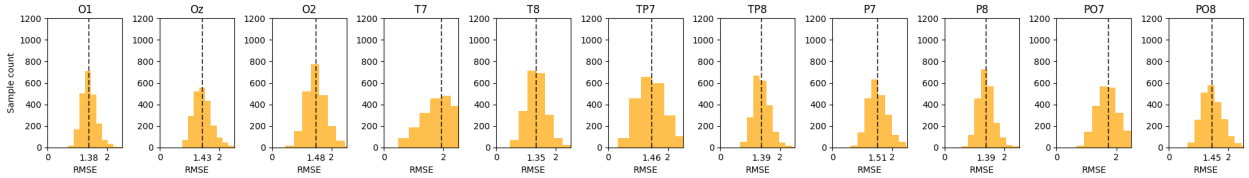
Figure H.1: Distribution of RMSE for each reconstructed EEG channel response (subset: {`O1, Oz, O2, T7, T8`}) across all samples in the test set for models reconstructing three EEG channels and using four ResNet modules where applicable. **(a)** shows the error distribution when trained with 4 modules. **(b)** shows the error distribution when trained with 3 modules. **(c)** shows the error distribution when trained with 2 modules.



(a) ResNetEEG with 4 modules



(b) ResNetEEG with 3 modules



(c) ResNetEEG with 2 modules

Figure H.2: Distribution of RMSE for each reconstructed EEG channel response (subset: {`O1`, `Oz`, `O2`, `T7`, `T8`, `TP7`, `TP8`, `P7`, `P8`, `PO7`, `PO8`}) across all samples in the test set for models reconstructing three EEG channels and using four ResNet modules where applicable. **(a)** shows the error distribution when trained with 4 modules. **(b)** shows the error distribution when trained with 3 modules. **(c)** shows the error distribution when trained with 2 modules.

I. Related Classes to Reconstructions

Model setting	Best reconstructions' related classes (Top 1-10)	Worst reconstructions' related classes (Bottom 1854-1845)
MLPEEG, 3 channels	*skin(1456), cardboard(272), snorkel(1478), rifle(1311),* ***bike(123)****, file2(584), revolving_door(1305), yoke(1849), honeycomb(789), smoothie(1474)*	***birthday_cake(132)****, costume(419),* ***potpourri(1235)****, tool(1676),* ***lettuce(920)****, stir_fry(1557), easel(542),* ***straw2(1569)****,* ***treasure(1707)****, leaf(906)*
ResNetEEG, 3 channels, 4 ResNet modules	*airplane(10), top_hat(1682),* ***dish(489)****, polisher(1215), missile(1006),* ***calamari(239)****, axe(45), clothes(361), pantsuit(1109), cement_mixer(294)*	***birthday_cake(132)****,* ***treasure(1707)****,* ***potpourri(1235)****,* ***sea_urchin(1393)****, touchpad(1691), eyedropper(566), whip(1809),* ***lettuce(920)****,* ***straw2(1569)****, laser_pointer(899)*
ResNetLatentEEG, 3 channels, 4 ResNet modules	*meatball(989), card(271), switch(1600),* ***bike(123)****,* ***dish(489)****, van(1752), chicken_wire(322), strainer(1566), pendulum(1144), calculator(240)*	***treasure(1707)****,* ***birthday_cake(132)****, cloud(364), goat(684), perfume(1154), easel(542),* ***calamari(239)****,* ***sea_urchin(1393)****, board(152), shield(1420)*

Table I.1: Overview of which stimuli classes the best and worst reconstructions of each model on the test set represents when reconstructing the first channel subset of 3 EEG channels and using 4 ResNet modules where applicable. RMSE is the metric used to define the best and worst reconstructions for the overview. Each class is represented by the name and its ID from the THINGS-EEG dataset in parenthesis. Repeatedly seen classes across the different models or best/worst reconstructions are marked with bold.

J. Noise Estimates

J.1 Noise Ceiling

The noise ceilings for each of the 11 channels, {O1, Oz, O2, T7, T8, TP7, TP8, P7, P8, PO7, PO8}, respectively:

```
[0.18993991  0.19199651  0.18955568  0.17837238  0.1927015  0.19385306
0.18987349  0.18721873  0.18881787  0.18423851  0.19425983]
```

J.2 Noise Floor

The noise floors for each of the 11 channels, {O1, Oz, O2, T7, T8, TP7, TP8, P7, P8, PO7, PO8}, respectively:

```
[0.00311633  0.00245046  0.00028904  0.00563317  0.00751754  0.00696968
0.00107624  0.00191302  0.00149518  0.00670159  0.00270326]
```