Deep Learning for Magnetoencephalography

By

Tim Chard
BSc Macquarie University

A thesis submitted to Macquarie University
for the degree of
Master of Research
Department of Computing
April 2021

MACQUARIE
University
SYDNEY·AUSTRALIA

Typeset in LaTeX 2_ε.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed) _____ Date: _____
Tim Chard

Abstract

Deep learning has been used in a wide range of applications, but it has only very recently been applied to Magnetoencephalography which is used to understand a variety of cognitive processes; for instance it can be used to understand how we process language or identify cognitive decline such as dementia. Work published in 2019 showed that it was possible to apply deep learning to categorise induced responses to stimuli across subjects. While trailblazing in its application of deep learning, it used relatively simple neural network (NN) models compared to other domains such as image and natural language processing.

In these other domains, there is a long history in developing complex NN models that combine spatial and temporal information in a range of ways. This thesis proposes more complex NN models that focus on modeling temporal relationships in the data, and applies them to the challenges of MEG data such as vulnerability to noise. In addition, it explores other insights from image processing to this domain, such as the unexpectedly high importance of approaches to data normalization. It applies these techniques to an extended range of MEG-based tasks, and finds that our new NN models outperform existing work on temporally-oriented tasks.

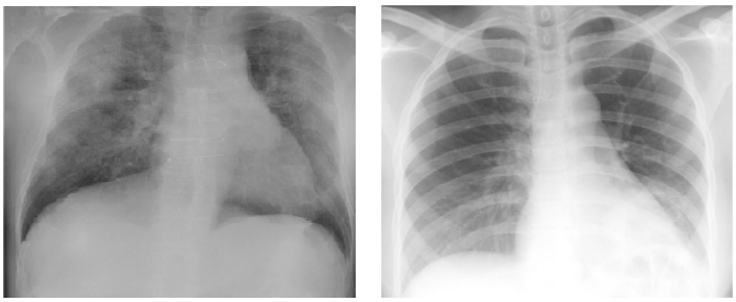
Contents

1 Introduction

Deep learning has achieved some truly amazing feats in recent years across a range of applications, from playing Go and near-autonomous self-driving cars to natural language question answering and algorithmic trading algorithms. Deep learning is a sub-field of machine learning that uses highly non-linear multi-layered networks to learn increasingly complex relationships with sufficiently high confidence that they can be employed in safety-critical applications.

One domain that was an early adopter of deep learning, and that has produced many architectures that are applicable in other domains, is image processing. One type of neural network that arose in image processing and that is now used more broadly is the Convolutional Neural Network (CNN) [1, 2], whose architecture was inspired by the human vision system [3]. CNNs are multi-layer neural networks where each layer is well suited to learning spatial relationships of the previous layers. This leads to deeper layers gradually learning more complicated patterns in the data. For instance, in a model that recognises digits, the first layers may learn to isolate simple colour gradients, middle layers combine these gradients to form simple shapes and later layers would combine these shapes to identify an individual number or letter. This type of network has been applied to a variety of different datasets, ranging in complexity from identifying the digits zero through nine (such as in MNIST [4]) to much large classification tasks with hundreds of different everyday objects (such as in CIFAR-10/100 [5] and ImageNet [6]). As the datasets became more complex, architecture developed correspondingly, starting with LeNet 5 [7] to more complex architectures such as VGGNet [8] and ResNet [9] which both perform well on more difficult tasks.

In a new domain and for a new task, initial adoption is largely driven by ease with which models can be adapted from existing domains. As an example from the medical domain that has seen easy adaptation from existing image processing models, medical imaging such as x-rays (see Figure 1.1) [10–13] are a natural extension of the CNNs that are used in the computer vision domain. When the hierarchical structure embodied in a



(a) COVID-19 infection. (b) non-COVID-19 infection.

Figure 1.1: X-rays of respiratory infections. From Wang et al. [10]

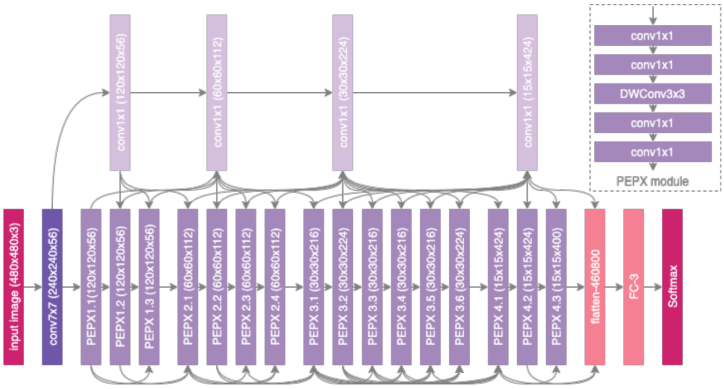


Figure 1.2: COVID-net architecture. From Wang et al. [10]

CNN is present in a domain, models can be adapted and built very quickly,such as the models that have been developed to identify COVID-19, such as the convolution-based COVID-net [10], which exploits these similarities and proposes an architecture using PEPX modules (see Figure 1.2) that build on architectures from computer vision. On the other hand, transferring to a very different domain can require very different architectures and require overcoming substantial challenges. For instance, in natural language processing, GPT-3 [14] has not only been able to generate large-high-quality paragraphs from writing prompts but has even been able to answer general knowledge questions correctly despite never being trained on these tasks. However, these achievements did not come easily, and an entirely new neural network architecture was developed. The successful adoption of deep learning in a domain depends on many factors, but there are two critical components: architectures and data.

Magnetoencephalography (MEG) is a brain imaging technique that uses magnetic fields generated in the brain to detect brain activity at a high temporal resolution [15] and there are a few characteristics that are not shared with existing domains which makes adaptation more difficult. One key difference compared to other domains is that while the data is dense along the temporal dimension (often sampled at around 1KHz), it is sparse spatially where the data is only sampled at a small number of locations around the head. Existing deep learning architectures from image processing, the domain most focussed on spatial structure, expect that adjacent values in the data are a result of adjacent values

in the source; this assumption is partially violated for MEG data where it is only valid in individual channels. This would be the equivalent to arbitrarily changing the order of the rows in an image, making the adaptation of existing architectures not straightforward.

Until recently, conventional machine learning approaches have been the most common way to analyse MEG data. Many of these depend on features extraction steps such Independent Component Analysis (ICA), Common Spatial Patterns (CSP) [16], FBCSP [17] and xDAWN [18]; and then employing a conventional classification technique such as Linear Discriminant Analysis (LDA) [19, 20], Support Vector Machines (SVM) [19, 21–23] and Hidden Markov Models (HMM) [24]. However, in 2019 Zubarev et al proposed two deep learning architectures for MEG data [21], LF-CNN and VAR-CNN (see Figure 1.3), which were motivated by the theoretical models of the processes that produce the magnetic fields in the brain, and to allow them to visualise how these networks determine their prediction. They evaluated the proposed models on three different tasks in the context of real-time brain-computer interfaces. Two of these tasks were related to classifying the type of stimulus that was presented to the subject (for example, distinguishing auditory from visual stimuli), with the third being a motor imagery task where the subject was to imagine making a physical movement. They found that their models beat both the conventional machine learning models as well a deep learning model from the computer vision domain. In this thesis, we use Zubarev et al's work as a starting point for exploring the application of deep learning to MEG analysis, examining it more deeply, and extending it in various ways.

Both of the models of Zubarev et al have two main components, a spatial de-mixing layer followed by a single temporal convolution layer. The spatial de-mixing layer applies a set of spatial filters to the raw input which separates spatial patterns into higher-level features. The next layer then identifies rudimentary temporal patterns in these spatial features using a single one-dimensional convolution. Despite the relative simplicity, they found that their models outperformed both the vector machines of traditional machine learning, as well as both domain-specific and general computer vision neural network architectures. In addition, perhaps surprisingly, they found that the much more expressive VGGNet architecture [8], a pioneering architecture from the computer vision field, performed very poorly on most datasets. However, only one of the datasets, Cam-CAN, is a large one, and it is well-known that deep learning approaches need much larger amounts of high-quality data than conventional machine learning; it was on this dataset that VGGNet performed competitively. Furthermore, the task on the Cam-CAN dataset was an easy one and does not require sophisticated models to perform well, with all approaches obtaining well over 90% accuracy, and not allowing a true determination to be made about the superiority of different models. Given this, we look at three aspects of applying deep learning to MEG analysis, following from Zubarev et al.

Investigation of Models on New Datasets As noted, deep learning requires very large amounts of high-quality data; in the past, the creation of datasets such as ImageNet [6] have been pivotal for their domains. The Cam-CAN dataset, as currently released, only includes data for a single task where subjects were either exposed to either a visual or auditory stimulus. As there is such a small difference between the best and worst models, it would be useful to apply these architectures to a more difficult task. Recently, another

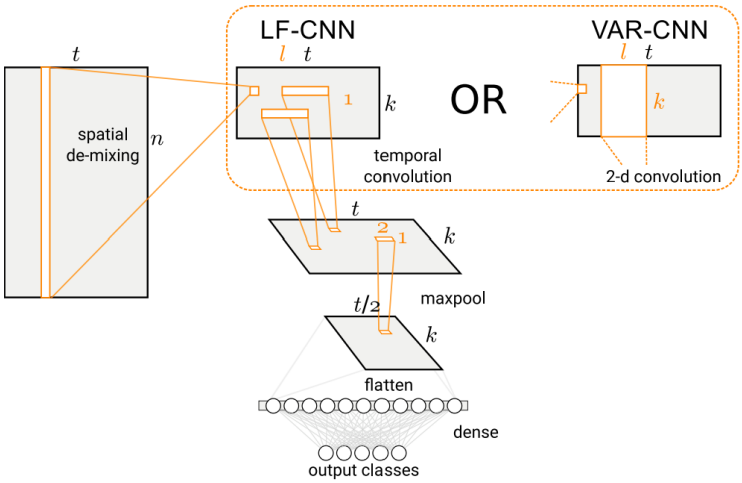


Figure 1.3: LF-CNN and VAR-CNN architecture. From Zubarev et al. [21]

large dataset has been released, the Mother Of Unification Studies (MOUS) [25] in which subjects are presented a series of words either visually or via audio. The words could form a sentence which is easy to process because of its syntactic structure, a sentence which is hard to process, or an arbitrary list of words. This leads to a variety of classification tasks with a variety of difficulties. For instance, can we distinguish audio from visual stimuli? Can we distinguish stimuli corresponding to syntactically valid sentences versus a random ordering of those words? Can we distinguish syntactically more complex sentences from syntactically simpler ones? We can perform the same task that is used on Cam-CAN, but the other classification tasks should prove to be much more challenging and leads us to ask: If we apply the same techniques to this new data, will we see similar results?

In both Cam-CAN and MOUS, the classification task is derived from a stimulus that the subject is exposed to. However, MEG has other applications, such as the diagnosis of cognitive impairment. In these applications, there is no stimulus and instead, the subject is recorded in a resting state. This is the case for the Dementia screening challenge dataset released as part of the BioMag 2021 Data Analysis Competitions.[1] In this dataset, the goal is to distinguish dementia and mild cognitive impairment from the healthy control subjects, which presents a challenge because it is a between-subject experimental design. While in the Cam-CAN dataset, each participant is exposed to both forms of stimuli (a within-subject design), in this dataset we are looking to identify something about the subjects themselves. This means that the models need to extrapolate past the inter-subject differences to perform well in this task.

> **Research Question RQ1** Our research question here is: Does the superiority of Zubarev et al's proposed deep learning architectures, LF-CNN and VAR-CNN, hold for these new large datasets and more challenging tasks?

[1] https://www.biomag2020.org/awards/data-analysis-competitions/

Development of New Models We noted above that the Zubarev et al models were relatively simple compared to some of the architectures in computer vision. For instance, comparing the complexity of the computer vision architecture (Figure 1.2) to LF-CNN and VAR-CNN (Figure 1.3) we can see that a single COVID-net PEPX module (see Figure 1.2) is made up of more layers than the entirety of the LF-CNN architecture.

As a result, these models are limited in the type of interactions that can be expressed and most focus heavily on spatial relationships in the data. However, given that the signal can only be recorded at a discrete number of locations around the head, it generates spatially sparse data. This means that it is difficult to develop a model that is able to learn a hierarchy of spatial features of the kind that are learnt by state-of-the-art computer vision architectures. This is because, as previously mentioned, an assumption that these state-of-the-art architectures make about the data is only partially valid, and it will only be able to effectively learn relationships along the temporal dimension. Therefore, we propose architectures that focus on building up a hierarchy of temporal features that are combined using residual connections, a popular method that was first used in computer vision [9].

The first idea here, of a hierarchy of temporal features, is drawn from WaveNet [26], a speech recognition and synthesis model that is capable of tracking very long temporal dependencies. These long term dependencies are fairly common in speech: for instance, in the sentence "he was sitting down because he hurt his leg", the gendered pronouns can have an arbitrary amount of distance between them. However, temporal patterns are something that existing models have not really exploited at all. With this motivation, we will develop a model that is capable of learning more complex temporal relationships.

The second idea, of residual connections, comes from image processing. While much of the computer vision research is focused on learning spatial relationships more effectively, there has also been a significant amount of more general work that can be applied to many fields. One such development was in the ResNet architecture, which introduced the residual connection [9]. The residual connections are layers that learn the best way to *alter* the input to reduce superfluous data, and these layers have allowed much larger neural networks to be trained as a result.

> **Research Question RQ2** Here, we aim to answer the question: Do our proposed architectures that specifically handle temporal characteristics outperform the existing models on our chosen tasks, both the original from Zubarev et al and our more challenging tasks?

An Additional Model Component: An Autoencoder While it is common in this domain to use different forms of dimensionality reduction, the above models, existing and proposed, all take the raw data as input to the neural network. In other domains, however, the generation of intermediate latent representations has been found to be more useful, particularly where these can be learnt from large amounts of unlabelled data. For instance, in natural language processing, it is common to develop a language representation model which is learned from unlabelled data via the task of predicting a word from its context. This is then used as the input to another model for a specific task, such as text classification or machine translation. Building a model with this representation as a starting point has been shown to greatly improve performance on downstream tasks [27, 28].

Autoencoders [29] are another possible method of achieving a similar goal. An autoencoder encodes an input into a lower-dimensional latent representation and then attempts to reconstruct the original input from this reduced representation. This causes it to learn to encode only relevant information in these representations which can then be used as input for downstream tasks. Like a language representation model, the advantage with this approach is that we train these models on data that have no labels and this will greatly increase the amount of data that is available for training.

Existing models have under-utilised temporal information, and there have not been architectures that allow a complex hierarchy of features to be developed simliar to computer vision architectures. For this reason, the autoencoder that we develop is only capable of learning temporal features. This is particularly attractive because we can produce a representation that can substitute the raw data in existing architectures and this will allow us to examine our assumption of the value of these temporal features.

> **Research Question RQ3a** Can we further improve the performance of a model by using the intermediate latent representation of an autoencoder?

Data preprocessing and normalization In other domains (such as in natural language processing [30, 31] and for autonomous vehicles [32, 33]) it has been shown that neural network architectures are capable of end to end learning which has eliminated many of the feature engineering steps, but there are a number of steps that are taken for granted for this to be possible. For instance, all pixels in an image are integers between 0 and 255 even though the raw signal from the sensor of a camera may vary significantly between cameras. The generated image is calibrated in two steps; first by the camera itself and then by the operator who is able to easily identify corrections that need to be made.

While the first step is possible for MEG, in practice, there can be orders of magnitude difference between the amplitude of the signal that is produced by two different machines. This is compounded by the fact that a human operator can not instinctively identify when there is a problem with the intensity of the signal in the same way that a photographer would identify a bright scene from an over-exposed picture. Instead, these sort of adjustments are generally made on a per-trial basis, but it is not clear exactly what the best approach is for deep learning on MEG data.

In machine learning, it is generally accepted [3] that features should have a mean of zero and a standard deviation of one. This has become even more important in deep learning where it can catastrophically affect training models: it has been shown that initialization of the neural network weights can be very important [34, 35] in addressing these issues. However, the term ‘feature’ becomes less concrete when applied to more complex domains. For instance, in computer vision, each colour channel is considered a ‘feature’ and is adjusted separately because for each channel there is an affine transformation that can move an arbitrary pixel to any other arbitrary location.

Normalizing data on a per-trial level is a significant difference from computer vision where datasets are normalized with statistics calculated across the entire dataset. While there are reasonable justifications for making either choice there hasn’t yet been a direct comparison of these two paradigms. We will perform this comparison and in addition, we will evaluate a few other steps that are involved in preprocessing the dataset.

Research Question RQ3b Is it better to use subject-level normalization or a global-level normalization on datasets?

This thesis is structured into three main sections that correspond to our research questions. First, we will look at the replications and extension of the existing Zubarev et al work which will serve as a baseline. We will then define our proposed neural network models and our evaluation framework, and compare them against the existing models. Finally, we will describe our exploration of improving performance of the models in terms of preprocessing and normalization.

2 Background

In this chapter, we will give an overview of related work and describe the foundational background of this thesis. This chapter is split into two main strands. In the first, we will discuss machine learning, a method for learning predictive models from data, which is currently a major paradigm for Magnetoencephalography (MEG) analysis. We will then focus on deep learning, a subfield of machine learning where the models are neural networks that build up hierarchical non-linear representations of the data. Deep learning models have produced state-of-the-art results in many domains.

The second strand of the chapter will cover our domain of interest, the processing of brain activity, specifically as captured by MEG. In this strand, we will describe the characteristics of MEG data, and the various ways it has been processed. We then bring these two strands together by discussing recent work by [21] that has applied deep learning models to MEG data. This is the core work that we use as a starting point in this thesis, to investigate new deep learning models and methods for processing MEG data.

2.1 Machine Learning

While there are a lot of different applications in machine learning, many problems can be broken down into one of two types, regression or classification. Regression tasks are those that look at predicting a continuous variable, such as predicting a subject's age. Classification looks at predicting discrete quantities, for example predicting the gender of the subject. In this thesis we will be primarily focused on classification, which we can define generally as: given inputs $x_i \in \mathcal{X}$ and labels $y_i \in \mathcal{Y}$ we would like to find some function $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that the predicted labels $\hat{y}_i = f(x_i)$ are good approximation to the true labels y_i[3, 36].

There are many different algorithms that are used in machine learning, such as Linear Regression, Naive Bayes, Nearest-Neighbors and Decision Trees. However, we give only

a high-level overview here for reasons of space; more details of these are available in [3, 36]. The heart of many of these algorithms is gradient descent, which is an optimization algorithm that can be used to iteratively find weights that minimize a loss function which used to quantify the performance of the model [3, 36]. A key one that we use in this thesis is the Support Vector Machine (SVM).

SVMs are a fast and accurate method to solutions where features exhibit demarcation points which can be used as a class boundary [36]. In their simplest form, they do this by finding a hyperplane through an input feature space that linearly separates most classes. More formally given an input space $\mathscr{X} \in \mathbb{R}^p$ with p input features and $\mathscr{Y} \in \{-1, 1\}$, we would like to find a weight vector $w \in \mathbb{R}^p$ and a bias $b \in \mathbb{R}$ so that $\text{sign}(w^T \phi(x_i) + b) = y_i$. It may not be possible to find a solution where this holds for all x_i but enforcing this constraint is known as a hard-margin and requires that the data to be linearly separable. However, if this constraint is not enforced it is known a soft-margin SVM. In addition to the Linear SVM that we have just described there are a number of variants such as the Radial Basis Function SVM and details of these can be found in [3].

2.1.1 Deep Learning Architectures

In this section, we will give an overview of relevant background information in the field of deep learning; a more comprehensive discussion can be found in [3]. We will first briefly discuss the fundamental concepts behind neural networks. We then will look at a number of models that will be relevant to MEG data. Following that, we will look at some aspects of the training of these models that will have implications for application to MEG data.

Neural networks are multi-stage classification models that are represented as graphs. Each node (neuron) in the graph can be thought of as a separate regression problem; these are organized into layers. In the simplest case, a fully-connected layer, all of the nodes in the same layer receive the same input. The output from one layer is fed into the input of the next. Given an input to a layer x_i, a weight matrix W and a bias b, each neuron will calculate $\text{Activation}(x_i \cdot W + b)$ where Activation is a form of non-linearity. Commonly a Rectified Linear Unit (ReLU) is used.

Convolutional Neural Network (CNN) The introduction of CNNs was a breakthrough for image processing tasks because they are well suited to learn location-independent features. For instance, if we wanted to learn to detect if a picture contains a ball, a CNN could learn a *Filter* to detect a ball and apply that in multiple locations, whereas a fully-connected network would need to learn how to detect a ball separately for each location. Unlike a fully-connected layer, nodes are only connected to a small number of close-by nodes in the previous layer. The layer learns a set number of *Filters* (*Kernels*), that function as feature detectors, the shape of which dictates the receptive field, how many nodes in the previous layer are connected to each output.

Stacking these convolution layers allows the network to learn hierarchical structures. Lower layers learn low-level features such as simple gradients, and later layers gradually become capable of detecting more complicated shapes and finally to higher levels features that are used to ultimately make the classification [3]. Figure 1.3 depicts a CNN.

ResNet One of the most ubiquitous models that is used in practice today is the ResNet architecture [9] which was developed by He et al in 2015. The key insight that they had was to utilize residual or skip connections. These layers learn a residual function which alters the original input and performs the operation $h_{W,b}(x_i)+x_i$ instead of $h_{W,b}(x_i)$. This design was motivated by the observation that adding extra layers could reduce the performance of a model. With a skip connection, if a particular layer is not improving performance the model is able to reduce the residual and in doing so the layer acts as an identity function. While a neural network could learn the identify function without this connection, using skip connections makes this easier to optimize because the ReLU function will output 0 for all negative values.

WaveNet WaveNet [37] is a convolution-based network that was designed to learn temporal relationships in raw audio waveforms. WaveNet made use of convolutional dilation which is similar to stride, but spreads out the input neuron instead of the output as with stride [3]. In effect given a dilation d and a kernel size k, it is equivalent to increasing the kernel size to $k \times d$ but only taking every dth input nodes. This means that it is able to have a significantly larger receptive field while maintaining the same number of trainable parameters and maintaining the memory footprint as well. WaveNet stacked these dilated convolutional layers into blocks, where each successive layers dilations was double the previous. This allows the model to capture both short term and long term patterns in the input.

Autoencoders Autoencoders are a method of unsupervised pre-training [38, 39]. Autoencoders are conceptually very simple and transform an input into a lower-dimensional latent representation and then reconstruct the input from the latent representation. Therefore, the encoder acts as a form of compression and this compressed representation can be used by downstream tasks to improve performance. Autoencoders typically use the Mean Square Loss to minimize the difference between the reconstructed and original data.

There are a number of variations of autoencoder that are popular in different areas. The convolutional autoencoder [40] for instance incorporates convolutions and has seen success with image data. Another approach is to artificially add noise to the input as in a denoising autoencoder [41]. It is also possible to stack autoencoder so that subsequent autoencoders try to reproduce the latent representation of the parent autoencoder. This type of autoencoder is unsurprisingly called Stacked Autoencoders and has already seen some use in this domain [42, 43]

Transformers Transformers are a recent development that was first introduced by Vaswani et al. [44]. They have since been extensively adopted in state-of-the-art architectures in natural language processing [27, 28, 45], and have also been applied in other domains such as computer vision much more recently [46, 47]. The key component of the transfer architecture is a multi-headed self-attention which allows the models to dynamically focus on relevant parts of the data by using multiple separate Attention operations.

There are other types of attention that can be used but transformers typically use the Scaled Dot Product Attention which is defined in Eq 2.1 It takes three tensors as input, the query Q, the key K and the value V. A softmax operation is performed on the dot product of the query and the key which is scaled based on the d_k, size of the dimensions of embeddings

used by the query and key. This dynamically generates the weighting that is applied to the value.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.1)$$

In Figure 2.1 we can see that before the Attention operation there are linear layers and these allow the network to accept the same input for the query, key and value. This multi-headed attention is combined with a residual connection, and is followed by a position-wise linear layer which is combined with a second residual connection. These two layers are then stacked together six times to form the encoder of a transformer. A detailed description of the whole architecture can be found in [3, 44].

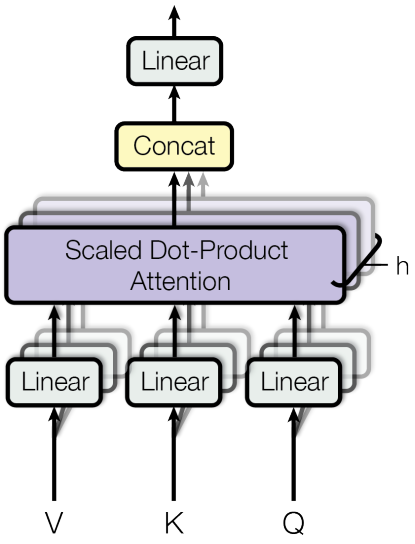


Figure 2.1: Multi-Headed Attention. From Vaswani et al. [44]

2.1.2 Training

Weight Initialization & Layer Normalization As architectures became more complex, it became apparent that both weight initialization and normalization was important to allow models to train successfully. Both of these affect the dynamics of the networks in terms of two related problems: exploding and vanishing gradients [3]. Xavier initialization was one of the first that was proposed to address these issues. [35] argued that it is important to consider the variance not just of the outputs but the gradients as well. Their initialization solution was to change the variance of the generated weight to be a function of the number of outputs and inputs in the layer. He initialization [34] is an extension of this that has been designed to work on the now more common ReLU activation, where it helps to prevent dead neurons in the network; it is considered current best practice for the ReLU activation functions and its variants [3]

In addition to weight initialization, there are a number of other techniques that look at addressing the normalization of inputs to layers at each layer, for example, Batch [48], Group [49] and Layer Normalization [50]. These layers explicitly control the mean and variance of the outputs of the previous layers through parameters learnt during training. These normalization layers have been shown to dramatically increase the speed at which models converge as well as to increase the performance of the final model.

2.2 MEG

All brain activity is the result of electrical currents that occur in the brain and therefore by recording this electrical activity we will be capturing brain activity. Two related technologies are able to capture these electrical impulses: Magnetoencephalography

(MEG) and Electroencephalography (EEG). In most cases, EEG captures the activity by recording the electrical potential on the surface of the scalp, while MEG detects the magnetic flux that is created by electric fields[51]. This work focuses on MEG, although we will also touch on work that is applied to EEG where similar spatiotemporal relationships in the data are relevant. Both MEG and EEG have a much higher temporal resolution (often sampled at around 1KHz) but are not known for their spatial resolution. This contrasts with functional magnetic resonance imaging (fMRI) which provides high spatial resolution but can not be reliably used to study rapid changes in the brain.

The magnetic flux that is captured by MEG is in the picoTesla and femtoTesla range which is more than 7 orders of magnitude smaller than Earth's ambient magnetic field [51]. This means that there is a great deal of effort to shield these machines from all other sources of magnetic fields. However, even with these precautions MEG still has a low signal-to-noise ratio [51], so methods to overcome this challenge have received a lot of attention.

There are two general types of MEG data, event-based or continuous [15]. In event-based sessions, subjects are responding to a change in the environment, typically a stimulus, such as being shown a checkerboard pattern on a screen for a small period of time. Continuous recordings, on the other hand, do not have discrete events and may not have any events at all. Instead, these sessions look at long-running activities, such as performing a physical action, but also include sessions where the subject is at rest.

2.2.1 Preprocessing and feature engineering

The nature of MEG data leads to a number of challenges when applying machine learning techniques, such as complexity in training the models and the risk of over-fitting a model to training data. There are a number of methods that are used as part of preprocessing and feature engineering that help to address these issues. For instance, band power captures the energy for a given frequency band in a small temporal window and can be calculated in a number of ways [52, 53]. Common Spatial Patterns (CSP) are another method of feature engineering that find a set of linear spatial filters that distinguish two classes [54]; these have been used in [55], for example.

Unlike band power, the spatial filters in CSP are generated through a form of supervised learning. Despite this, however, it is still necessary to select the subject-specific frequency bands that are used. Filter Bank Common Spatial Patterns (FBCSP) [56] are an extension of CSP that allow for the automated selection these frequency bands.

There are also unsupervised methods such as Primary Component Analysis (PCA) [57] and Independent Component Analysis (ICA) [57, 58]. While PCA is a method of linear dimensionality reduction, ICA separates a data into a combination of maximally independent components and these methods can be used individually [59] or together [60]. ICA has also been shown to be useful in artifact suppression [59].

End-to-end learning One of the often-cited advantages of applying deep learning is that it removes the need to perform feature engineering. This is not always easy but end-to-end learning has been demonstrated in a wide range of domains, such as autonomous vehicles [61], speech recognition [62, 63] and many more [32, 64, 65]. Even in those applications there is still some pre-processing steps that are used, such as normalization and cropping.

For MEG data, there are a number of pre-processing steps that are widely used such as downsampling [21], band-pass filtering [21, 66, 67], normalization [21, 23, 68]. However, the effect of each of these has not been systematically explored in the context of deep learning.

Normalization Gradient Descent is an incredibly powerful technique, but it can be very sensitive to input data. It is generally accepted [3] that data should have a mean of zero and standard deviation of one. However, as problem domains have become more complex, simple methods of normalization can remove informative relationships. This adjustment typically involves calculating the normalized input $\hat{x}_i$ by subtracting the mean and dividing by the standard deviation.

However, there are a variety of ways to calculate the mean and standard deviation. For example, there are two types of normalization that can be applied to each item in a dataset: GLOBAL or LOCAL normalization. Global normalization calculates the mean and variance of the entire dataset and then adjusts each item with the same mean and variance [3] and calculates $\hat{x}_i = (x_i - \mu)/\sigma$. On the other hand, LOCAL normalization calculates a mean and variance for each input x_i.

LOCAL normalization is a common approach for MEG and EEG data [21, 23, 68], and is typically called BASELINE normalization, where the mean and variance are calculated on a baseline period before the stimulus. WINDOW normalization calculates the mean and variance directly from the item so that $\hat{x}_i = (x_i - \bar{x}_i)/s_{x_i}$.

2.2.2 Applications

Deep learning is a growing area of research in EEG but conventional machine learning remains the most common paradigm. We have touched on some; a comprehensive review of applications with EEG data can be found in [69] for machine learning and in [70] for deep learning. However, it is only recently that deep learning has been applied to MEG. We will now describe work by Zubarev et al which we use as a starting point in this thesis.

Zubarev et al proposed two deep learning architectures [21], LF-CNN and VAR-CNN, which were designed to model the processes that generate the signals created in the brain. They also developed methods for analyzing the models' predictions, where they looked at the spatial and temporal features that were most strongly related to each predicted class. This allowed them to visualise how these networks derived their prediction and how these prediction were associated with regions of the brain.

They performed four different experiments that looked at three different tasks, and focused on generalization to new subjects. They therefore evaluated each of their models on a held-out group of subject/s, which were selected randomly. The first two were evaluated on one subject, the third on two subjects, and the fourth on 50 subjects. These experiments were in the context of brain-computer interfaces, and so in addition to the standard Validation and Test accuracy for evaluation they also simulated a real-time brain-computer interface environment. This was similar to Test accuracy but the model parameters were updated after it had made a prediction for each trail and aimed to gauge how well the models could adapt to new subjects as they are being used.

The first experiment involved seven subjects that were exposed to five different types of sensory stimulus, which included visual, auditory and electrical stimulation. The second

and third experiment looked at predicting an imagined physical action when a visual cue was presented. While the second and third experiment involved the same task, the third experiment was conducted in a live real-time brain computer interface environment. The final experiment used 250 subjects from the Cam-CAN where two types of sensory stimulus was classified.

Cam-CAN dataset Cam-CAN is the largest MEG dataset that is available and consists of more than 600 subjects [71]. This MEG dataset is part of a much larger study that look at the effects of aging, but at the time of writing only the second stage of the study has been released. In this stage subjects were involved in three MEG sessions, a resting state recording and a passive and active sensorimotor task. The same stimulus was used in both the active and passive stage which consisted of a checkerboard pattern or an auditory tone played at one of three different frequencies (300Hz, 600Hz and 1200Hz). In the passive stage, which is used by Zubarev et al, the trial consists of a unimodal stimulus, either auditory or visual, and no action is required by the subject. In contrast, during the active stage subjects were exposed to both forms simultaneously and were required to respond when they observed either stimulus.

Models Zubarev et al introduced the LF-CNN and VAR-CNN models and evaluated them against a number of other models, both conventional machine learning classifiers as well as deep learning models from both EEG and computer vision. Specifically, they compared their models against SVMs with both linear and radial basis function kernels, EEG-Net [66], Shallow FBCSP-CNN [72] and VGG-19 [8]

The LF-CNN and VAR-CNN models use raw MEG data and are relatively simple with two main components, a spatial convolution followed by a temporal convolution and then a linear layer (see Figure 1.3). The surprising thing about these models is that they perform well despite only containing two (ReLU) non-linearities in the entire network. This suggests that there are relatively simple relationships in the data that correlate highly with the output class. Despite the fact that models are going to be limited in the relationships that they are able to model, Zubarev et al found that they were able to outperform the other models in each comparison they made. They also found that the other neural network architectures were not always able to outperform conventional machine learning baseline.

2.2.3 Summary

Deep learning is rapidly being adopted by new fields, but this is not always easy and it is only just starting to be adopted in MEG. We have seen one application that demonstrates deep learning can outperform other methods. However, this was applied to only one large dataset, where they performed a task where the baseline models were already able to perform (more than 90%) well which limits the potential improvement on the task. With this in mind, in the next chapter we will build on their initial work and extend it to new datasets with more difficult tasks.

3 Foundation

In §2.2.2 we observed that, on the only large dataset Cam-CAN, the classification task was easy, with all architectures performing well. In this chapter, we will discuss our replication of Zubarev et al and how we extended this work to a newly released large dataset and more challenging tasks. On this new dataset, we will evaluate and compare the performance of the two models LF-CNN and VAR-CNN that Zubarev proposed against their SVM baselines, to see if their superiority still holds, and to investigate their suitability as baselines for our new models in Chapter 4.

3.1 Method

In this chapter, we have aimed to deviate from the original work as little as possible, although some minor changes were unavoidable in our extension to new datasets. We will start this section with the new dataset that we will be using in this chapter, the MOUS dataset. After this discussion, we will describe the training and evaluation of models.

3.1.1 Datasets and Tasks

In addition to Cam-CAN, another large dataset has also been recently released, the Mother Of Unification Studies (MOUS) [25] with a much more interesting stimulus. The focus of the study was understanding the way that we process written and spoken language and was specifically looking at how we process individual words in a sentence. The study consisted of 204 participants and like the Cam-CAN dataset, the subjects were exposed to an auditory and a visual stimulus. In this case, however, subjects were only subjected to one stimulus, with half being shown written text and the other half hearing spoken words. In each case, the stimulus consists of linguistic utterances (in Dutch), either a valid sentence or an arbitrary list of words. In addition, there were two types of sentences: a sentence that is easy to understand with a main clause plus a simple subordinate clause,

and a sentence with a relative clause which is harder to understand. In this work, we will look at three different classification tasks; auditory vs visual stimuli, sentence vs word list and simple vs hard sentence. Examples of the stimuli can be found in Appendix A.1.

While we will perform a number of different classification tasks, we will use the same data for each. For each trial, we extracted a window of the data as the subject was presented with the target word. We epoched the data in the same way as Zubarev et al, taking 300ms before the onset and 500ms after. However, this results in a small change in the resulting input because the MOUS is sampled at a higher frequency. So while the final Cam-CAN data has 64 time samples, the final MOUS data has 85. In both cases we are normalizing by the mean and standard deviation of the first 36 samples of the epoched data which means that the MOUS data includes more prestimulus data.

The other deviation from the Cam-CAN data is the number of channels that are included. We are using 270 channels from the MOUS (compared to 204 in Cam-CAN related tasks), but we have also increased the number of time samples that are included as well. This is not all of the channels that are available in the datasets: there are a number of channels which are not present in all recordings and there were also channels from other sources, such as EEG. These channels were not considered in any of our experiments. In total ten channels were not present in all recordings (BP2, EEG061, EEG062, EEG063, EEG064, MLC11, MLF62, MLT37, MRF66, MRO52); of these channels, 5 were MEG related channels and were consequently excluded.

This data provides options for several classification tasks with a variety of difficulties, and we have constructed three new tasks for this dataset. The AUDIOVIS task attempts to distinguish audio from visual stimuli. The SENTWORDLIST task attempts to distinguish syntactically correct sentences from lists of words. The HARDEASYSENT task looks at detecting relative processing difficulty of a sentence.

The auditory vs visual stimulus task (AUDIOVIS) is similar to that for the Cam-CAN dataset in Zubarev et al: we are trying to predict if the subject was seeing a written word or if they were hearing a spoken word. However, it is important to note that unlike the Cam-CAN datasets this is a between-subject variable. A subject was shown either the words, or they heard them; none of the subjects experienced both. This is important for two reasons. First, different subjects might encode the relationships between the stimuli very differently and second, deep learning is very good at picking up on unintended features (particularly noise) that correlates highly with the output class: [73] demonstrated this for image classification systems, using an example where the presence of snow in a photo could have led a model to predict a dog as a wolf. In this case, we will have to consider the possibility that the neural network is picking up a characteristic of the session (such as background noise) instead of the stimulus; we analyse the results in light of this.

In our second classification task (SENTWORDLIST) we aim to distinguish a sentence from an arbitrary list of words: more specifically, if the stimulus target word is part of a syntactically correct sentence or part of an arbitrary list of words. Similarly, our third classification task (HARDEASYSENT) is to predict if the target word was part of a sentence with a syntactically complex structure or a simple one.

We can further break down each of these last two classification problems into more fine-grained tasks by also taking into account how the stimulus is presented. We will evaluate each task by training models on three different subsets of the data; audio, visual

Dataset	Architecture	Val. Acc.	Test Acc.	Upd. Acc.	Train Time
Cam-CAN	LF-CNN	**94.52**	**92.53**	92.41	278
	VAR-CNN	93.86	92.33	**92.80**	426
	SVM (Lin.)	92.06	89.36	89.71	**252**
	SVM (RBF)	93.73	92.10	91.27	795
MOUS	LF-CNN	82.46	**81.49**	87.01	**848**
	VAR-CNN	**91.45**	80.93	89.02	948
	SVM (Lin.)	70.82	68.63	74.39	1960
	SVM (RBF)	64.71	57.30	**95.47**	1427

Table 3.1: Results of the Audio vs Visual task on both the Cam-CAN and MOUS datasets

and both together. This means that in each of the audio and visual subset we will be restricted to 102 subjects, although this restriction may allow models to focus on more fine-tuned features for each of the modes of stimulus.

3.1.2 Training and Evaluation

We have only made superficial changes to the training and evaluation code of [21], for the most part to update to more recent versions of the required dependencies. The models were trained on a CPU using Tensorflow [74] with early stopping and these models typically took less than 30 minutes to train.

As in Zubarev et al, we separated the data into three parts, a training, validation and test set. We follow Zubarev in looking at between-subject generalization, so the data was separated by subject such that each subject was assigned to a single split. Of the total data used, 20% of the subjects were assigned to the test set, the remaining split between the training set and the validation set with a 9-10 split. The models were trained in a standard fashion and used early stopping on the validation loss with a patience of 3. We used the same metrics as Zubarev et al as described in §2.2.2, Validation and Test accuracy as well as Pseudo-real-time accuracy which we refer to as Update Accuracy.

3.2 Results

From the upper half of Table 3.1 we can see very similar results to the original work on the Cam-CAN dataset in terms of validation accuracy. However, there is a larger difference in both the Test and Update accuracy than might be expected. Since both of the models have similar accuracy on the test set, this difference is likely caused by differences in which specific subjects are in which subset of the data. Results for AUDIOVIS on MOUS are in the lower half of Table 3.1. As mentioned in §3.1.1, given the between-subject nature of the task it is not surprising that it is more challenging. Nonetheless, the models perform well on this task with both models getting between 80% and 90%, which is a substantial improvement over the SVM baseines.

In both the SENTWORDLIST and HARDEASYSENT tasks, the classes are evenly distributed and this means that the random change baseline is 50% in each case. The results for these

Subset	Architecture	Val. Acc.	Test Acc.	Upd. Acc.	Train Time
Audio	LF-CNN	**66.67**	**66.51**	**65.45**	**435**
	VAR-CNN	65.35	64.27	64.81	447
	SVM (Lin.)	58.30	58.11	59.14	1703
	SVM (RBF)	64.12	62.95	63.82	625
Both	LF-CNN	**56.36**	59.99	**60.14**	**540**
	VAR-CNN	**56.36**	**60.07**	59.72	1183
	SVM (Lin.)	53.77	53.91	55.40	2412
	SVM (RBF)	55.27	55.75	58.80	1474
Visual	LF-CNN	**58.99**	**56.05**	**56.61**	**354**
	VAR-CNN	57.89	55.28	55.70	661
	SVM (Lin.)	54.40	53.41	54.76	2040
	SVM (RBF)	50.94	52.36	55.47	817

Table 3.2: Results of the Word List vs Sentence task on the MOUS datasets

tasks are in Tables 3.2 and 3.3 respectively. These are significantly more challenging, with the accuracy ranging between 55% and 65%. We can also see that the models proposed by Zubarev et al are again better than the SVM models, particularly in SENTWORDLIST. We can see that the SVM (RBF) performs well on the Update Accuracy metric on the MOUS dataset. However, this metric may not provide a reliable measure of performance on this dataset because subjects are only exposed to a single stimulus. It is therefore possible that information from the first trial in a session influences the performance on subsequent trials in the same session.

One perhaps surprising result is that there was a sizeable difference between the different modalities of the task. It may have been expected that models trained on either of the audio or visual subsets may perform better than models trained on both together. However, we see from both Table 3.2 and Table 3.3 that the audio subset has substantially higher accuracy for SENTWORDLIST while the visual subset takes the lead in HARDEASYSENT. (One possibility is that this is the result of subvocal articulation where a subject may be more prone to articulate words in more complex sentences that are presented visually). Ultimately, however, we have seen evidence that it possible to distinguish someone who is processing sentence from someone who is processing an arbitrary list of words. There is also slightly weaker evidence that we can identify the complexity of a sentence that someone is processing when it presented visually.

3.3 Summary

In this chapter we have successfully replicated the existing work with only minor differences in the results. The differences that we have seen could have come about from minor differences in training: specifically, the subjects used in the evaluation and the version of dependencies that were used. We followed Zubarev et al as closely as possible, but the

Subset	Architecture	Val. Acc.	Test Acc.	Upd. Acc.	Train Time
Audio	LF-CNN	49.12	48.73	50.00	165
	VAR-CNN	50.22	50.79	52.00	191
	SVM (Lin.)	**51.22**	**51.10**	**52.37**	719
	SVM (RBF)	49.39	50.60	50.29	**154**
Both	LF-CNN	**55.21**	53.43	**54.24**	**392**
	VAR-CNN	54.65	52.62	52.77	578
	SVM (Lin.)	52.68	51.64	51.67	2035
	SVM (RBF)	55.16	**54.18**	54.18	821
Visual	LF-CNN	**58.77**	**59.40**	57.22	199
	VAR-CNN	57.24	56.98	58.30	233
	SVM (Lin.)	56.27	56.12	56.41	880
	SVM (RBF)	57.19	58.96	**58.83**	**220**

Table 3.3: Results of the Simple vs Complex sentence task on the MOUS datasets

specific subject splits were not recorded so can not be reproduced exactly. In addition, deep learning libraries are rapidly evolving and can lead to substantial differences in the same model, and this likely has played a role in the differences but we can't draw any definite conclusions.

We have also shown that all of the new tasks are more difficult than those for the Cam-CAN dataset. Two of these are significantly more difficult with both the LF-CNN and VAR-CNN doing about 10% better than chance on the HARDEASYSENT task. We have also seen that there is a difference in performance between the modalities of the stimulus and the task, with audio stimulus being easier to identify than visual stimulus for the SENTWORDLIST while the opposite relationship holds for the HARDEASYSENT.

We have seen that the models proposed by Zubarev et al have generally outperformed the SVM baselines. As a result the LF-CNN and VAR-CNN models will be suitable baselines for the new models that we propose in next chapter.

4 New Models

In this chapter we will introduce a number of new neural network architectures that are designed to address some of the weaknesses of the models proposed by Zubarev et al which we will use as a baseline. In addition, we will also implement two architectures from the computer vision domain.

We apply these new models to the classification tasks of Chapter 3, and compare them experimentally against the existing models and the computer vision architectures. In addition, we introduce the new Hokuto dataset, which differs from the previous ones in that it does not have a stimulus, but rather records the resting state of individuals with varying degrees of cognitive impairment. We explore how to extend all the models to work with this dataset, and compare them on a classification task derived from it.

4.1 Models

In this section, we will propose a number of different network architectures that are capable of modelling more complex relationships than the models used in Chapter 3. Our first architecture, TimeConv, is our simplest one that explores the value of temporal relationships in the data. These ideas are then refined into a spatially invariant autoencoder architecture (TimeAutoencoder) that is built with a number of temporal residual blocks which exploit these temporal relationships with residual connections [9]. Our third model (SERes) combines the same temporal residual blocks with the spatial relationships that have already been found to be effective in the models of Zubarev et al. Our final architecture (SETra) uses the Transformer architecture [44] which follows from the way that we have thought about the MEG image as a sequence of spatial embeddings.

TimeConv The input to these modesl consists of a 2-dimensional matrix, one representing time and the other representing the spatial aspect of the data. This architecture is designed to help us gauge the importance of temporal relationships in the data.

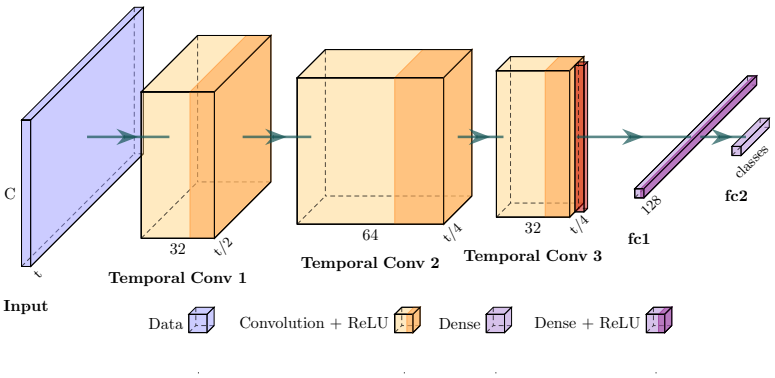


Figure 4.1: The TimeConv architecture.

As such, a separation between temporal and spatial relationships is enforced throughout the body of the network, such that the network learns spatial features only at the final fully-connected layers. In Figure 4.1 we can see the high-level structure of the network.

Even though the main body is composed of just three temporal convolution layers, this is already deeper than previous work. Each temporal convolution layer consists of a 2d convolution, limited to act only on a single channel, and this encourages the network to focus on temporal aspects of the data and effectively disallows it from learning spatial features. More specifically, they have a kernel size of $(1, 5)$, dilation of $(1, 3)$ and a stride of $(1, 3)$ and use "same" padding. These convolutions are followed by a standard ReLU activation function. There are 32 filters in the first layer, 64 in the second, and 32 in the last and were chosen to allow for more expressiveness than the models of Zubarev et al while also keeping the number of elements in the output of the last layer relatively small. These layers are followed by an adaptive max-pooling layer, which means that ultimately the network is learning a non-linear function that maps the activity in each channel into a 32 dimension temporal embedding (the number of filters in the last layer). These embeddings are then concatenated and fed into a dense layer with 128 output features, a ReLU and finally an output layer with the number of classes as its dimension.

Temporal Residual Block The Temporal Residual Block is a self-contained logical group of layers; we will use these in different configurations of these blocks in SERes and the TimeAutoencoder. The structure of this block follows from the ideas in the previous model, but unlike TimeConv, the Temporal Residual Block is completely prevented from learning any spatial features. This means that this block is completely independent of the size of the spatial embeddings (or the number of MEG channels) in the input.

We realise this by using a 1-dimensional kernel in a 2-dimensional convolution. This has two consequences. First, because the kernel is 1-dimensional, it is incapable of learning any spatial relationships. Second, because it is a 2-dimensional convolution, the kernel is applied to all channels and this encourages the network to learn low-level relationships that exist in all channels.

This convolution may seem similar to the LF-CNN because they both use a 1-dimensional kernel, but they are significantly different in the type of patterns they are able to learn. Let us consider how these convolutions will act on a black and white image. If we were to use the same number of kernels and use the same kernel size then both will have the same

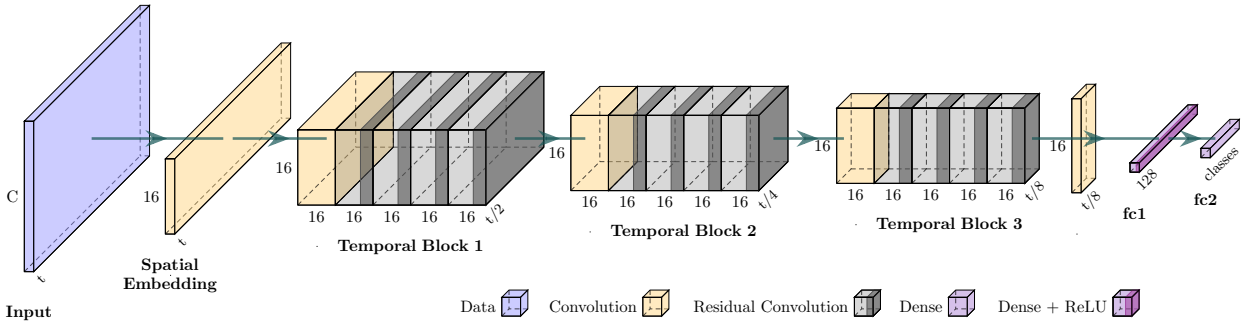


Figure 4.2: The Spatial Embedding Residual (SERes) network architecture.

number of trainable parameters. Both architectures will apply a 1-dimensional kernel to each row of the image. However, the LF-CNN will apply a different kernel to each row, whereas our convolution will apply each kernel to all rows.

Each block consists of four Temporal Residual Convolution layers which are similar to residual layers used by He et al [9]. However, we reduce the kernel to 1-dimension which means each convolution uses a kernel size of $(1, 3)$, dilation of $(1, 3)$ and padding of $(1, 3)$. We use the same number of filters in each layer which we leave as a hyper-parameter.

Our implementation of the residual connection is very similar to the PyTorch implementation [75] and, ignoring the convolution that is being used, only differs in how batch normalization is applied. While we use a single batch normalization layer before either convolution, the PyTorch implementation uses two layers directly after each convolution.

SERes While TimeConv learnt temporal relationships in the raw data, the temporal layers of both the LF-CNN and VAR-CNN have operated on a spatial embedding. The SERes architecture (Figure 4.2) combines both of these ideas, by first learning a spatial embedding which effectively reduces the number of "channels" to 16. So given an input $(1, C, t)$ with C channels and t time samples, the spatial embedding layer will produce an embedding of size $(1, 16, t)$. This spatial embedding is then processed by three temporal residual blocks. Each of the three blocks has 16 filters and reduces the number of time samples by a factor of eight, leading to an output of $(16, 16, t/8)$. To reduce the number of parameters and avoid problems like overfitting, we apply a dimensionality reducing (1×1) convolution which outputs $(1, 16, t/8)$ and is then flattened to a $16 \times t/8$ feature vector. The head of the network is very simple and consists of two layers. The first has 128 output features and uses the ReLU activation. The last predicts the weighting of the classes which are used as part of the cross-entropy loss, which combines the softmax activation and the negative log-likelihood loss.

SETra Transformer architectures have received a lot of attention recently, although most of this has remained in the natural language processing domain. One of the strengths of a transformer comes from its ability to use information present across the entire input. In comparison, a convolutional network can only exploit information inside the receptive field of the neuron and this means that details that are separated in the input can only be combined as the receptive field gradually grows in later layers. However, very recently there have been a few architectures that use the Transformer on computer vision problems [76, 77]. Successfully implementing a transformer architecture in this domain would mean that the network is capable of combining details regardless of where they occur in

the image. For example, the presence of a baseball at the edge of the image is helpful in identifying that the subject in the photo is playing baseball, but this type of information can only be used later in the network. A transformer on the other hand would be able to use this information immediately. For an image, it quickly becomes computationally infeasible to use attention on each pixel, because the number of pixels itself grows quadratically with image size. The existing designs have avoided this by using a downsampling mechanism or restricting the scope of the attention. For instance, Image Generative Pre-trained Transformer (I-GPT)[76], an architecture that directly adapts an existing architecture from NLP, resizes the image until the operation is feasible, while the even more recent Vision Transformer (ViT) [77] takes a more sophisticated approach and splits the images into patches that are then attended to.

In our domain, however, this problem is already addressed from the way that we are thinking about the problems: as a sequence of spatial embeddings. This means that while other approaches grow quadratically with the size of the image, our approach grows linearly. To compare our approach directly with the ViT, given an input with size (224×224), ViT would split this into 196 patches of 16×16 that need to be attended to, whereas our method would attend to all 224 rows (or columns). This means that the operation should still be feasible, but does not require any form of downsampling.

One of the biggest differences between this architecture and other transformers is the way the model is trained. Most transformers are pre-trained using a next token prediction task. However, the SETra is trained exactly the same way as the other classifiers, which means that there is no need for masking or positional embeddings.

Like the previous model, we use a spatial embedding of size 16 — this can be considered as the equivalent of a "word embedding" in the NLP domain. We use four transformer layers with an embedding size of 16 and a feedforward dimension of 64 which are much smaller than is typical, but necessary to allow the model to train.

4.2 Experimental Setup

For this chapter, we have reimplemented the models of Zubarev et al: the original implementation used a proprietary data format that was not suitable for our reimplementation.[1] This reimplementation allows us to implement both existing and new models in the same framework for consistent evaluation.

4.2.1 Datasets and Tasks

While this chapter uses the datasets of Chapter 3, there were slight differences because of the issue of the proprietary data format. Compared to Zubarev and the previous chapter, the data was split in a slightly different manner. While we still used a training, validation and test sets, the size of each subset was slightly different, to allow us to run repeated experiments while still holding out a test set for use once only at the end, to genuinely measure model generalization. Specifically, we assigned 60% of the subjects to the training

[1] Specifically, the `mne` toolbox introduced limitations on the development of new models and introduction of other datasets and tasks. We reimplemented its core functionality, and the existing models themselves, for this chapter. Results under this reimplementation are consistent with those of Chapter 3.

set, 20% was allocated for validation and the remaining 20% were allocated to the test set. In addition to the training, validation and test set, we also used a development set which was created by partitioning half the data (instead of subjects) from the validation set. We used the validation set for early stopping and hyperparameter selection, and the development set as a stand-in for the test set for producing results over the course of experiments. (Note that unlike the other splits, the validation-development split is not looking to evaluate inter-subject performance and instead is used to evaluate intra-subject performance. This means that while the validation and test sets consist of different subjects, the validation and development have the same subjects but different trials.)

In addition, there are subtle differences in our preparation of both the Cam-CAN and the MOUS datasets. In each case, we epoched the data in the same way as Zubarev et al, by taking a 800ms window that starts 300ms before the stimulus onset. However, after the data was epoched Zubarev et al downsampled and then normalized it, whereas we do these in the reverse order. In §3.1.1 the data is normalized using the first 280ms of the downsampled data, and the rest is processed by the neural networks. This means that this data includes a small amount of prestimulus data. In contrast, for strict correctness, we normalize based on the whole prestimulus period, and the input to the networks starts precisely at the stimulus onset.

Cam-CAN In addition to the AUDIOVIS task that was used on the Cam-CAN dataset in the previous chapter, our re-implementation allowed us to easily add an additional task. The Cam-CAN uses an auditory stimulus that consists of three distinct tones and so in addition to the AUDIOVIS task described in §2.2.2, we also perform a TONE task where we aim to predict the specific tone that is being played.

Unlike the previous chapter, we did not limit the number of subjects to 250 because performance generally increases when neural networks are trained on larger amounts of data. Instead, we only excluded five subjects (CC120208, CC510220, CC610462, CC620193, CC620685) where there were issues with the data (for example, sessions with no data), leaving 644 subjects. The training set consists of 388 subjects, the validation and development set share 128 subjects and the test set has the remaining 128 subjects.

MOUS Our re-implementation of this dataset only deviates from the previous chapter in two ways that have already been mentioned: the normalization of the data and the number of subjects that are being used. In this chapter, our test set contains 40 subjects, our validation and development set shares 40 subjects and the remaining 124 subjects belong to the training set.

Hokuto This dataset is new to this chapter. It was released as part of the Biomag 2021 Dementia screening challenge,[2] where the goal was to develop systems for the diagnosis of dementia and mild cognitive impairment. Unlike the other datasets, there is no stimulus and instead, this data consists of a resting state recording.[3] This dataset consists of 100 control subjects, 16 subjects with mild cognitive impairment and 29 subjects suffering from dementia. To address the class imbalance of this task, we will pool the subjects into two classes: healthy and some cognitive impairment. The latter group combines subjects with mild cognitive impairment and dementia.

[2] https://www.biomag2020.org/awards/data-analysis-competitions/

[3] It could not be used in the experiments of Chapter 3 as `mne` does not support this.

The subjects were recorded in two separate locations, with 93 being recorded at location A and 51 at location B. There was a large class imbalance at location A where 73 subjects were in the control, 19 had dementia and only one had mild cognitive impairment. Location B on the other hand was more balanced with 27 subjects in the control class, 14 with mild cognitive impairment and 10 with dementia.

We will perform two different tasks on this data. The first (IMPAIRMENT) is to directly predict if the subject has any cognitive impairment. The second is to predict which location the recording took place in. This second task is to allow us to explore what kind of spurious features a machine learner might pick up: in ideal circumstances, this second task should not be possible. However, if there are strong location-based signals, it may be the case that a learner for IMPAIRMENT is influenced by this in its predictions.

To look at the effect of the recording site we will also break down the IMPAIRMENT task by the recording site in the same way that we broke down the SENTWORDLIST and HARDEASYSENT tasks by the mode of the stimulus. Here we will train models on just the data from location A, then train new models on location B and again on all of the data.

The data was recorded with different machines at each location, and while both used a 160-channel gradiometer they were sampled at different frequencies (1000Hz and location A and 2000Hz at location B). Unlike the other datasets, we used WINDOW normalization because there is no stimulus in this datasets.

Apart from these considerations, we use a similar approach to the other datasets. The data was first band-pass filtered to 1-45 Hz and the data from location B is downsampled by a factor of two. We then epoched the data into 1000ms windows at which point we rescaled the window to have a mean of zero and a standard deviation of one. The data is then downsampled by a factor of 8 in the same fashion as the other datasets.

4.2.2 Training and Evaluation

Evaluation The main metric is classification accuracy, as in Zubarev and §3.1.2. In addition, we look at the variability of results in two ways. In the first, we calculate standard deviation over subject accuracies on the test set (indicated by ± in the subject accuracy columns of results). In addition, we calculate the 95% confidence Wilson score interval for the classifier [78], which is calculated on a binomial assumption; the lower and upper bounds are indicated by LB, UB respectively.

Training In each case, we trained three different models and then applied the model with the best validation accuracy to the test set. During all development we use the development set as a proxy for the test set.

We trained each model on a GPU with a batch size of 128, using the Adam gradient descent optimization algorithm [79] with a learning rate of 10^{-3} which optimized the cross-entropy loss of each model. We used early stopping on the validation loss with a patience of 3.

4.2.3 Baselines

Our core baselines are LF-CNN and VAR-CNN from [21]. Like [21], we also include high-performing computer vision models: GoogLeNet [80]and ResNet18 [9], described in

§2.1.1.[4] These models are designed to process images with three colour channels, so we added a 1×1 convolution with three filters to form three "colour" channels.

4.3 Results

AUDIOVIS In Table 4.1 we have the results for the AUDIOVIS task for both the Cam-CAN and MOUS datasets. Compared to our replication in §3.2 (Table 3.1), we can see that accuracies here are somewhat higher than in the previous chapter, which is likely due to the differences in training. Specifically, we trained three models and evaluated the model with the best validation score, and we used all of the available data in the Cam-CAN dataset. In terms of consistency, we can see that there are almost identical patterns between the models on the datasets. Our replication saw very little difference among the models on Cam-CAN, while the only difference on MOUS was that VAR-CNN outperformed the LF-CNN on all evaluation sets (validation, development and final test), not just initial validation.

Overall, the differences in results between the two datasets are substantial, despite conceptually being a similar task. While our TimeConv and SERes models outperform VAR-CNN and LF-CNN on Cam-CAN, the results are the other way around for MOUS. On the Cam-CAN dataset, the lower bound of the Subject Accuracy for SERes is higher than the test accuracy of any of the other architectures. This is likely due to the much larger variation in the results for MOUS than Cam-CAN. We can see that the standard deviation is around 5 on Cam-CAN but is between two and four times larger for MOUS. This may well be due to that fact that this is a between-subject task and it is possible that the models are picking up on characteristics of the session.

ResNet18 approaches the level of performance of the SERes and this may because they are capable of capturing similar types of temporal relationships. However, both ResNet18 and GoogLeNet are much larger, which means that they tend to overfit more easily and that do not reach performance levels seen elsewhere.

We also note that there is a substantial difference between our models that have a larger component of dense connections. The TimeConv has a much larger fully-connected layer than the SERes and the SETra is not constrained by convolutions at all. This may indicate that these models are over-fitting more easily.

TONE The results for the TONE task are in Table 4.2 and we can see that almost all of the models are able to significantly beat the random chance baseline of 33.3%. However, there is much less variability between the models on this task compared to AUDIOVIS on the MOUS, with all but two models doing better than 46%.

Notably, GoogLeNet performs the worst across all metrics. This is not the only time that we have seen similar results from a computer vision architecture, over the course of our experiments on the development set. We have noticed that the larger a model is, the less likely it is to successfully converge.[5]

[4] [21] use the older VGGNet, modified to include batch normalization. When we implemented it, we were unable to train a single model successfully. Instead of altering the architecture, we implemented

Dataset	Architecture	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
Cam-CAN	SERes	**94.99**	**94.92**	**95.87**	**95.87±4.0**	(**95.16** - **96.58**)
	TimeConv	94.28	94.30	94.80	94.80±4.9	(93.95 - 95.66)
	SETra	93.29	93.02	94.38	94.38±5.5	(93.42 - 95.34)
	LF-CNN	93.24	93.09	94.38	94.38±5.5	(93.41 - 95.34)
	VAR-CNN	93.29	92.73	94.33	94.33±5.5	(93.36 - 95.30)
	GoogLeNet	93.22	92.89	93.86	93.86±5.4	(92.91 - 94.81)
	ResNet18	94.24	94.09	95.03	95.03±4.8	(94.19 - 95.87)
MOUS	SERes	78.03	78.97	79.34	79.29±14.0	(74.77 - 83.82)
	TimeConv	72.20	71.91	74.36	74.49±11.4	(70.79 - 78.19)
	SETra	72.13	72.97	74.11	73.80±8.2	(71.14 - 76.46)
	LF-CNN	79.22	80.20	82.21	81.72±15.2	(76.78 - 86.66)
	VAR-CNN	**82.06**	**83.32**	**83.55**	**82.82±16.7**	(**77.42** - **88.23**)
	GoogLeNet	78.07	79.21	81.68	81.30±19.2	(75.07 - 87.53)
	ResNet18	77.45	77.96	76.86	75.91±21.5	(68.95 - 82.88)

Table 4.1: Results of the AUDIOVIS task on both the Cam-CAN and MOUS datasets. Included is the Validation Accuracy, Development Accuracy, Test Accuracy and the Lower and Upper Bound of the Subject Accuracy

Architecture	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
SERes	45.16	46.72	46.09	46.09±8.5	(44.61 - 47.58)
TimeConv	44.19	46.17	45.70	45.70±8.3	(44.25 - 47.16)
SETra	**45.89**	**46.90**	**46.50**	**46.50±8.7**	(**44.97** - **48.02**)
LF-CNN	45.81	46.88	46.34	46.34±8.9	(44.77 - 47.91)
VAR-CNN	45.65	45.42	45.31	45.31±9.4	(43.67 - 46.95)
GoogLeNet	33.15	33.80	33.45	33.45±2.8	(32.95 - 33.95)
ResNet18	44.40	46.12	44.00	44.00±9.5	(42.33 - 45.66)

Table 4.2: Results of the TONE task on the Cam-CAN dataset

SENTWORDLIST In Table 4.3 we have the results of the SENTWORDLIST task on the MOUS dataset. We can see that SERes outperformed our baselines (LF-CNN and VAR-CNN) in every case for the Visual and Both subsets. The computer vision models do not do well on this task and like the TONE, the GoogLeNet in particular falls behind. As in §3.2, we see that there is a substantial difference between the different subsets. This may be due to the way that the information can be processed. When a word is presented visually the subject can immediately focus on any part of the word. For audio, on the other hand, the subjects attentions will always initially be at the start of the word which may result in more informative temporal relationships. Despite having half the available data, the

architectures that already incorporated batch normalization.

[5]This is consistent with our attempted training of VGGNet.

Audio subset achieves the best results.

Training on all of the data does not seem to improve compared to the results of each subset. This may be because the low-level features are not easily transferred from one modality to the other. This makes sense because these models are limited to learning a spatial embedding which only allows focusing on the brain activity in 16 distinct ways. When trained together, the models will need to compromise to focus on activity that applies on both modalities. Increasing the size of the spatial embedding could improve the results, although it may also make it easier for the network to overfit.

Subset	Architecture	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
Audio	SERes	**65.91**	**65.61**	**65.79**	**65.93±4.1**	(**64.01** - **67.86**)
	TimeConv	63.44	64.46	64.95	64.90±4.8	(62.68 - 67.13)
	SETra	63.63	64.19	64.78	64.79±4.8	(62.54 - 67.03)
	LF-CNN	63.40	64.64	63.78	63.86±4.9	(61.59 - 66.13)
	VAR-CNN	64.34	64.27	65.40	65.46±4.8	(63.23 - 67.69)
	GoogLeNet	61.15	62.54	60.39	60.49±5.1	(58.09 - 62.89)
	ResNet18	63.08	64.07	65.36	65.41±4.2	(63.47 - 67.34)
Both	SERes	**60.22**	**60.54**	**60.26**	**60.56±7.2**	(**58.23** - **62.89**)
	TimeConv	57.92	58.37	58.96	59.11±4.4	(57.68 - 60.55)
	SETra	58.40	59.82	59.29	59.52±6.0	(57.58 - 61.46)
	LF-CNN	58.55	58.75	58.73	59.00±5.9	(57.10 - 60.91)
	VAR-CNN	57.30	58.62	57.99	58.15±5.9	(56.23 - 60.06)
	GoogLeNet	52.50	53.63	52.87	52.82±3.4	(51.74 - 53.91)
	ResNet18	55.83	58.47	58.08	58.22±4.8	(56.67 - 59.78)
Visual	SERes	54.64	56.08	56.55	56.57±3.1	(55.03 - 58.11)
	TimeConv	55.88	**56.77**	56.12	56.09±3.1	(54.54 - 57.65)
	SETra	54.83	55.70	**57.05**	**57.00±3.2**	(**55.43** - **58.56**)
	LF-CNN	**54.99**	55.41	56.22	56.18±2.8	(54.79 - 57.56)
	VAR-CNN	54.54	55.20	55.53	55.51±3.2	(53.90 - 57.11)
	GoogLeNet	53.86	55.35	55.06	55.04±0.7	(54.70 - 55.38)
	ResNet18	53.28	55.43	56.22	56.19±3.3	(54.54 - 57.84)

Table 4.3: Results of the SENTWORDLIST task on the MOUS dataset

HARDEASYSENT In Table 4.4 we have the results of the HARDEASYSENT task on the MOUS dataset for both the Audio and Visual subsets. Results for the Both subset can be found in Appendix A.2. Like the SENTWORDLIST task, we can see a difference between the subsets, but in this case, the classes are more easily distinguishable with a Visual stimulus compared to the Audio equivalent. We also note that none of the models trained on the Audio subset did better than chance (50%), whereas in both of the other subsets each model lower bound of the 95% confidence interval was better than chance.

This also seems to be a task that does not see much benefit of the temporal features that our proposed models focus on. While the LF-CNN outperforms the other models on

the Visual subset, as well as the Both subset (See Appendix A.2), there does not seem to be a substantial difference between the other models. In any case, this is a challenging task but all of the models are able to distinguish the difference when the stimulus is presented visually.

Subset	Architecture	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
Audio	SERes	50.64	49.25	49.77	49.80±0.6	(49.52 - 50.08)
	TimeConv	50.49	48.97	49.97	50.00±0.5	(**49.77** - 50.22)
	SETra	50.41	50.53	50.63	50.50±5.2	(48.10 - 52.90)
	LF-CNN	**54.24**	49.83	51.54	51.48±4.9	(49.22 - **53.74**)
	VAR-CNN	50.13	**51.48**	**51.88**	**51.63±4.5**	(49.52 - 53.73)
	GoogLeNet	53.06	48.44	50.91	50.81±4.6	(48.66 - 52.95)
	ResNet18	52.54	48.78	49.72	49.49±3.9	(47.67 - 51.31)
Visual	SERes	57.73	57.25	59.34	59.27±3.9	(57.32 - 61.22)
	TimeConv	55.34	53.79	58.09	58.05±3.9	(56.14 - 59.96)
	SETra	56.87	57.56	60.09	60.06±3.1	(58.53 - 61.59)
	LF-CNN	58.82	59.40	**60.76**	**60.66±3.0**	(**59.18** - **62.15**)
	VAR-CNN	**59.24**	59.06	59.08	59.00±3.3	(57.35 - 60.64)
	GoogLeNet	57.84	59.05	59.55	59.16±1.1	(58.60 - 59.72)
	ResNet18	56.76	55.96	58.72	58.43±3.0	(56.96 - 59.89)

Table 4.4: Results of the HARDEASYSENT task on the MOUS dataset

IMPAIRMENT In Table 4.5 we have the results of the IMPAIRMENT task on the Hokuto dataset. This dataset was one of the most challenging, for three reasons: it does not involve an evoked response, it is the smallest in terms of the number subjects which are recorded at two separate locations, and it is tracking a between-subject variable. Despite this, we were still able to train four different models that outperformed the random chance baselines. This is the smallest dataset: there are only 14 subjects in the test set and only five at location *B*, with large confidence intervals as a consequence.

For Location A, GoogLeNet, ResNet18, LF-CNN and VAR-CNN all perform very well on the validation set. The same models also perform well on the validation set for Both locations. This may be because the models have been able to consistently learn features that work well for that majority of subjects from Location *A* in the validation set.

More generally, the models trained at Location *A* performed much better than Location *B*. However, at least some of this is due to the class imbalance at each location. The random chance baseline at Location *A* was 66.66% while it was 60% at Location *B*. The models at Location *B* were much closer to random chance than at Location *A*: the mean test accuracy at Location *A* is 78.88%, and at location *B* it is just 59.88%

Ultimately, there does not seem to be enough data to draw any strong conclusions, although the results at Location *A* seem promising. However, we also looked at predicting which location the subject was from, and all architectures achieve near 100% accuracy. The mean test accuracy across all the models is 99.51%, with the lowest score 97.53%. We are forced to conclude that the models are easily able to identify the individual machine

that was used to record the data. Consequently, this need to be carefully considered in experimental design when looking at between-subject variables.

Location	Architecture	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
A	SERes	90.64	90.50	**86.41**	**86.30±30.6**	(**70.65** - **101.95**)
	TimeConv	85.64	86.76	77.69	77.52±31.0	(61.66 - 93.39)*
	SETra	90.99	92.17	79.71	79.53±31.6	(63.35 - 95.71)*
	LF-CNN	95.47	95.76	84.91	84.77±33.4	(67.67 - 101.87)
	VAR-CNN	94.07	94.61	83.22	83.08±31.8	(66.79 - 99.37)
	GoogLeNet	**96.16**	**96.25**	83.75	83.71±32.8	(66.92 - 100.50)
	ResNet18	94.20	94.78	81.17	81.12±33.0	(64.21 - 98.03)*
B	SERes	50.62	51.15	60.43	59.78±34.3	(33.88 - 85.68)*
	TimeConv	51.17	50.19	51.31	50.14±34.4	(24.19 - 76.09)*
	SETra	48.12	48.79	60.14	59.78±21.3	(**43.73** - 75.83)
	LF-CNN	58.13	58.44	61.31	60.68±33.0	(35.80 - 85.56)*
	VAR-CNN	72.75	72.42	65.49	64.66±30.6	(41.57 - 87.75)*
	GoogLeNet	**80.27**	**77.82**	67.32	67.16±36.0	(40.00 - **94.32**)*
	ResNet18	75.99	74.77	**67.40**	**67.24±35.5**	(40.44 - 94.04)*
Both	SERes	81.22	81.31	75.63	75.69±34.9	(61.92 - 89.46)*
	TimeConv	76.87	77.03	71.86	71.71±32.4	(58.92 - 84.51)*
	SETra	82.45	82.61	75.94	75.80±36.9	(61.23 - 90.37)*
	LF-CNN	82.60	82.24	73.95	73.81±33.5	(60.59 - 87.03)*
	VAR-CNN	85.58	85.65	76.82	76.69±35.5	(62.67 - 90.70)*
	GoogLeNet	85.24	85.00	74.50	74.47±34.0	(61.04 - 87.90)*
	ResNet18	**88.89**	**88.47**	**77.77**	**77.74±33.0**	(**64.70** - **90.78**)

Table 4.5: Results of the IMPAIRMENT task on the Hokuto datasets.

4.4 Summary

In this chapter, we have introduced three novel architectures and they have demonstrated that by learning temporal relationships in the data we are able to improve the performance on some tasks. In particular, the SERes was able to beat all our baseline models on the SENTWORDLIST task where it achieved the highest overall accuracy in two of three subsets (Audio and Both) and was only outperformed by our SETra in the third (Visual) subset. We point out that these temporal patterns were easiest to identify on the audio subset of the data. This difference may be due to the way that the word is processed because the subject's attention will always initially be at the start of the word for the audio stimulus whereas this is not the case for the visual stimulus.

In the tasks where the models can make use of these temporal features, SERes generally outperforms all other models including much large models like ResNet18 which has more

than 150× more trainable parameters than SERes. In contrast, other tasks such as the HARDEASYSENT task seem to be more conducive to spatial aspects and models such as SETra and the baselines seem to perform better. It would be interesting in future work to understand what key cognitive science properties distinguish HARDEASYSENT from SENTWORDLIST in terms of their temporal behaviour.

Training models successfully remains the biggest challenge in the domain, with the best performing models being the ones that are able to constrain the problem in a way that maintains important relationships but reduces overfitting. The work by Zubarev et al has shown to do this remarkably well at this, but its simplicity limits the types of interactions that it can model. On the other hand, our SERes model is far more expressive and has the potential to significantly improve on the LF-CNN and VAR-CNN if the challenge in training models is addressed. In spite of these challenges, the SERes outperforms all other models on several tasks which demonstrate the utility of these features.

Given the success of transformers in other fields, the SETra does not perform as well as one might hope. This may be in part due to a transformer's abilty to attend to information anywhere in the input, which in this case may allow it to more easily overfit to any noise that is present. This result highlights the importance of pretraining, which we will explore in the next chapter through the use of an autoencoder.

5 Model Preprocessing

5.1 Introduction

In this chapter we will explore two possible ways to improve the performance of the models that were used in Chapters 3 and 4. First, we will first look at a method of pre-training which has been shown to be effective in other domains such as natural language processing. Using an autoencoder, we will encode temporal relationships into a dimensionally reduced representation ('embedding') which can then be taken as input by other models. Not only will this pre-training potentially improve results, but it will also allow us to investigate the importance of the temporal features in the data which our new models have focused on.

Second, we will investigate the effects of normalization of the data from two different points of view: local and global. As noted in §2.1.2, good normalization has been important for building high-performing deep learning models in other domains. It is common to normalize each trial from an MEG session individually using a baseline period. However, in computer vision all of the data is normalized using the same set of statistics which are calculated across the entire dataset. We will explore both of these approaches, both independently and in combination, to determine the most appropriate method for normalization.

5.2 TimeAutoencoder

In this section, we will explore the effects of pre-training via an architecture we refer to as TimeAutoencoder. While transformers are potentially one possibility for pre-training, they are generally trained as part of a next (or missing) token classification task. In this domain, we do not have discrete tokens and therefore we can not directly adapt these

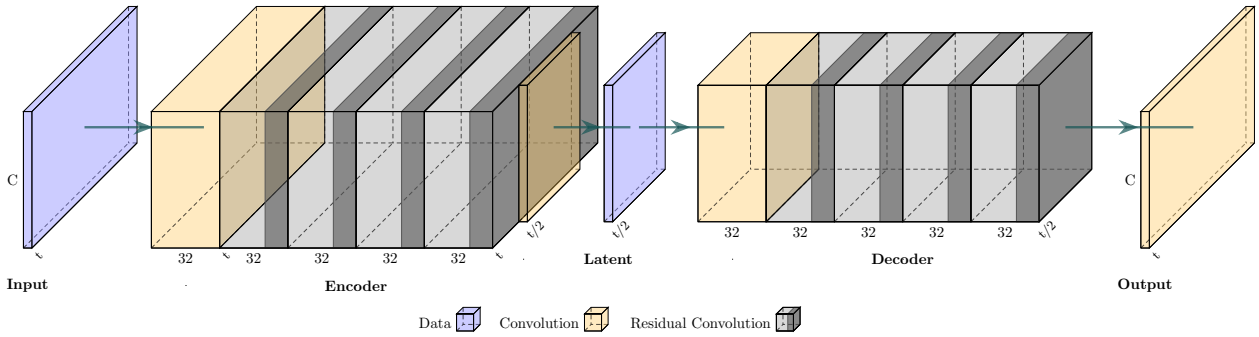


Figure 5.1: The TimeAutoencoder architecture.

training methods. However, this line of inquiry seems to be a promising one for future work.

Instead of using a next-token prediction, we focus on the traditional autoencoder training task where we train our model to reproduce an input from a compressed latent embedding. Specifically, we train our model to minimise the mean square error between the output to the decoder and the original input. This architecture builds on the ideas from our other models and relies on the Temporal Residual Block that we described in §4.1. However, unlike our other models, this architecture has no method of learning any spatial relationships at all. The intention here is that by limiting the types of interactions that it can model, it will be forced to learn low-level features that are more easily generalized in downstream tasks.

As the TimeAutoencoder only operates on temporal relationships, the latent embedding that it generates can be thought of as data that has been downsampled. We can also train the TimeAutoencoder on a larger input window to generate a latent representation that has the same dimensions as the inputs that we used in §4.2.1, allowing more temporal data to be incorporated. As training the autoencoder is significantly more computationally intensive than the other models, we will only evaluate the architecture by training the TimeAutoencoder on MOUS with 128 time steps which will then halve the temporal dimension. We will train models in five different configurations: two baselines which use the raw data, and three configurations which incorporate the encoder and differ in how the encoder is trained.

Model As with other convolutional autoencoders described in §2.1.1, the TimeAutoencoder has two main components, the encoder and the decoder. In Figure 5.1 we see these components separated by the latent representation. Given an input with size $(1, C, t)$ where C is the number of channels and t is the number of time steps, this will be compressed down to $(1, C, t/2)$ before the original input is reproduced.

The first layer in the encoder increases the number of filters to $(32, C, t)$, which is then passed to Temporal Residual Block with 32 filter layers, maintaining the same dimensions. The last layer in the encoder is a strided temporal convolution and is the bottleneck in the network. This layer uses the same settings as the other layers in the Temporal Residual Block, but in addition uses a stride of 2. It outputs the latent representation which has half the number of time steps of the input but is otherwise the same $(1, C, t/2)$.

The decoder is similar to the encoder but replaces the first and last layer convolutions with transpose-convolution equivalents. The first layer will increase the number of filters

to 32 so that the data has a shape of $(32, C, t/2)$ and like the encoder, this is input to the Temporal Residual Block. The last layer in the decoder is responsible for recreating the missing time steps and reproducing the input.

5.2.1 Experimental Setup

We will compare five different experimental configurations. The first two (RAW64, RAW128) will not use the TimeAutoencoder at all and instead use the raw data, with either 64 or 128-time steps. These will allow us to evaluate whether any performance increase is simply due to extra information that is present in the larger input. Of those that use the autoencoder, the FROZEN configuration will use the latent representation of the encoder as part of a pre-processing step, and the parameters of the encoder will not be updated. In contrast, the UNFROZEN will also use the latent representation but will be fine-tuned as part of training the classifier. The UNINITIALIZED will use an encoder with randomly initialized parameters, and like UNFROZEN the parameters will be updated while training the classifier. For each of these configurations, we will train four different model architectures (SERes, SETra, LF-CNN and VAR-CNN). We will apply these to the HARDEASYSENT and SENTWORDLIST tasks on MOUS, to assess the benefit of temporal information there.

Despite the simplicity of the TimeAutoencoder, there is a substantial computational expense and training these models took more than 8 hours. As a result, unlike in §4.2.2, we will only train one instance of each model, and because of the increased memory requirements, we will use a batch size of 32. However, we will also accumulate gradients across 4 batches to maintain an effective batch size of 128 and optimise it as described in §2.1.1. In addition, we will not develop stimulus-specific models and will use the Both subset to train our models.

5.2.2 Results

Table 5.1 (SENTWORDLIST) contains a number of interesting results. The TimeAutoencoder substantially increased the results of both the LF-CNN and VAR-CNN, where the best configuration (UNFROZEN) produced Test Accuracies of 59.58% and 59.09%, respectively. This reduced the difference in subject accuracy between SERes and LF-CNN to just 0.66% from 1.61%. For both the LF-CNN and VAR-CNN the UNFROZEN configuration outperforms all other configuration in every evaluation set.

Interestingly, the FROZEN in contrast decreases performance in most cases. We can also see the benefit of pre-training by comparing the results to UNINITIALIZED, which are significantly lower on the LF-CNN and VAR-CNN models. This might suggest that the TimeAutoencoder is overfitting and is learning how to noise as well as useful information.

Looking at SERes and SETra, we can see that there is only a minor difference between Disabled (64) and Disabled (128). We also see that these models perform no better when the encoder is added. Taken together this suggests that the extra data is not capturing useful relationships; as a result, the encoder is making better use of the information that would already be present in the 64 time samples that we used in the previous chapters.

Moving on to Table 5.2 (HARDEASYSENT) we can see no clear optimal configuration. This is not too surprising because in Table 4.4 and §4.3 we saw that the models that incorporated temporal features performed no better than those without. This result more

directly supports the idea that temporal features are more important in some tasks than others.

Architecture	Encoder	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
SERes	Disabled (128)	**59.63**	**60.67**	**60.24**	**60.46±6.3**	(**58.41** - **62.52**)
	Disabled (64)	58.21	59.02	59.42	59.62±5.7	(57.77 - 61.47)
	Frozen	59.03	59.96	59.85	60.12±6.0	(58.18 - 62.06)
	Unfrozen	55.04	56.25	56.42	56.57±4.3	(55.18 - 57.97)
	Uninitialized	55.26	55.87	55.50	55.59±3.7	(54.40 - 56.78)
SETra	Disabled (128)	59.00	59.93	**60.18**	**60.42±6.5**	(**58.29** - **62.54**)
	Disabled (64)	**59.19**	59.65	59.63	59.94±6.3	(57.91 - 61.97)
	Frozen	57.69	58.91	58.76	58.97±5.4	(57.21 - 60.74)
	Unfrozen	58.81	**60.12**	59.04	59.25±5.7	(57.39 - 61.10)
	Uninitialized	58.35	59.84	59.20	59.43±6.1	(57.46 - 61.41)
LF-CNN	Disabled (128)	57.71	58.76	58.73	58.85±4.9	(57.28 - 60.43)
	Disabled (64)	57.06	57.53	58.50	58.63±5.9	(56.73 - 60.54)
	Frozen	56.13	58.39	57.08	57.20±5.5	(55.41 - 58.99)
	Unfrozen	**58.63**	**60.18**	**59.58**	**59.80±5.4**	(**58.06** - **61.55**)
	Uninitialized	57.36	58.83	57.61	57.78±5.4	(56.03 - 59.53)
VAR-CNN	Disabled (128)	56.98	58.95	58.30	58.27±5.4	(56.54 - 60.01)
	Disabled (64)	57.96	58.88	58.28	58.61±5.9	(56.69 - 60.53)
	Frozen	56.77	57.19	57.94	58.19±5.8	(56.33 - 60.06)
	Unfrozen	**58.58**	**59.01**	**59.09**	**59.34±5.8**	(**57.47** - **61.21**)
	Uninitialized	58.06	58.93	58.58	58.84±5.9	(56.92 - 60.77)

Table 5.1: Results of the applying the TimeAutoencoder to the SENTWORDLIST task on the MOUS dataset

5.3 The Role of Normalization

We have noted in §2.2.1 that there is a difference in how normalization is processed in MEG compared to other domains. In this section we will explore the effect of normalization methods that are common in MEG and machine learning more broadly. We will look at the effects of a number of different LOCAL normalization methods and compare then together with GLOBAL normalization.

5.3.1 Experimental Setup

Again to limit the computational cost, we will focus on the Cam-CAN dataset, and we will train each normalization strategy on three different architectures (SERes, LF-CNN and VAR-CNN) on both inputs with 64 and 128 time steps. The two different normalization strategies are BASELINE and WINDOW that have been described in §2.2.1. We

Architecture	Encoder	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
SERes	Disabled (128)	54.58	**56.10**	55.44	54.89±5.3	(53.18 - **56.60**)
	Disabled (64)	53.22	55.05	55.45	54.78±5.0	(53.16 - 56.39)
	Frozen	53.91	54.70	54.91	54.34±4.6	(52.86 - 55.82)
	Unfrozen	54.02	54.89	54.92	54.35±4.7	(52.84 - 55.86)
	Uninitialized	**54.90**	55.09	**55.50**	**55.18±4.3**	(**53.79** - 56.57)
SETra	Disabled (128)	54.96	55.22	**56.57**	**56.11±4.6**	(**54.63** - **57.58**)
	Disabled (64)	53.25	54.69	55.60	54.93±5.4	(53.20 - 56.67)
	Frozen	54.21	54.92	54.91	54.34±4.5	(52.87 - 55.81)
	Unfrozen	52.79	54.26	55.22	54.56±7.3	(52.19 - 56.92)
	Uninitialized	**55.39**	**55.92**	55.77	55.32±5.2	(53.63 - 57.00)
LF-CNN	Disabled (128)	54.61	55.04	55.14	54.35±5.2	(52.66 - 56.03)
	Disabled (64)	53.98	55.12	55.97	55.21±5.9	(53.31 - 57.10)
	Frozen	**55.02**	53.31	54.79	54.30±5.7	(52.45 - 56.15)
	Unfrozen	54.11	54.02	**56.28**	**55.67±5.8**	(**53.80** - **57.55**)
	Uninitialized	53.72	**55.26**	55.47	54.91±4.8	(53.36 - 56.46)
VAR-CNN	Disabled (128)	**56.11**	54.00	**56.12**	**55.51±4.6**	(**54.01** - 57.01)
	Disabled (64)	53.06	54.61	55.80	55.26±4.3	(53.87 - 56.65)
	Frozen	55.26	**55.08**	55.09	54.56±5.0	(52.94 - 56.17)
	Unfrozen	55.47	53.27	56.01	55.46±6.1	(53.50 - **57.42**)
	Uninitialized	54.36	55.04	55.46	55.06±5.4	(53.30 - 56.82)

Table 5.2: Results of the applying the TimeAutoencoder to the HARDEASYSENT task on the MOUS dataset

also implement SCALE, which is a LOCAL normalization that is similar to the ROBUSTSCALER used in Scikit-Learn [81], as well as the effect of not applying any normalization which we refer to as NONE normalization. In addition to being applied locally, each of these methods has a GLOBAL variant, where the LOCAL method is applied first and followed by GLOBAL normalization.

The BASELINE is the normalized method that we used for most of our previous experiments and follows the procedure used by Zubarev et al where the baseline period is 300ms prior to stimulus onset. In contrast, SCALE fits the data to the interval $[-1, 1]$ to reduce the effect of outliers in the data. More precisely, it first clips all the values to fall between the 1st and 99th percentile, subtracts the 1st percentile and divides 99th so that the data is between 0 and 1, then further rescales to the interval $[-1, 1]$.

We will follow the same methodology we used in §4.2.2 and will evaluate the best of three models. This means that we will measure the performance of each strategy on six different models from three different architectures on two different input sizes. As usual, we will evaluate the performance of these models by the accuracy metric, noting mean and standard deviation.

5.3.2 Results

Normalization	Mean Val. Acc.	Mean Dev. Acc.	Mean Test Acc.	Mean Acc. (LB - UB)
Baseline	93.80±1.0	93.59±0.8	94.82±0.8	(93.95±0.9 - 95.70±0.6)
Baseline + Global	93.66±0.9	93.49±0.8	94.61±0.8	(93.69±0.9 - 95.53±0.7)
None	50.64±0.0	49.36±0.0	50.00±0.0	(50.00±0.0 - 50.00±0.0)
None + Global	93.26±0.6	92.94±0.6	94.26±0.7	(93.33±0.8 - 95.20±0.6)
Scale	**93.99±0.6**	**93.65±0.9**	**94.97±0.6**	(**94.10±0.6** - **95.84±0.5**)
Scale + Global	93.59±0.9	93.40±0.8	94.77±0.6	(93.86±0.7 - 95.68±0.5)
Window	93.69±0.9	93.48±0.9	94.81±0.7	(93.92±0.8 - 95.70±0.6)
Window + Global	93.60±0.7	93.42±0.8	94.71±0.7	(93.80±0.8 - 95.61±0.6)

Table 5.3: Effects of normalization on the AUDIOVIS task on the Cam-CAN dataset

From Table 5.3 we can clearly see that not applying any form of normalization completely prevents training, with NONE getting exactly 50% in most cases. At first look, this is not terribly surprising; however, as there is no deviation in the models (standard deviation 0), this means that none of the 18 models that were trained were able to learn any useful features.

This is unexpected because our SERes model makes use of batch normalization, the first of which takes place before any form of non-linearity. This may have been due to an undesirable interaction between the weights of the first layer and the subsequent batch normalization, where a small update to the weights and the bias in particular would result in chaotic data distribution between batches and disrupt the normalization process.

In the AUDIOVIS task, the SCALE has a small lead in each metric, but the inter-model variations are larger than the differences between most of the normalization strategies. Looking at the mean test accuracy, BASELINE, WINDOW and SCALE score 94.82%, 94.81% and 94.97% respectively which amounts to a difference of just 0.16%.

There is also a small difference in the models that used GLOBAL normalization where the GLOBAL version performed a fraction worse than the version that just used the local normalization. However, we can see from Table 5.4 that this difference is more pronounced in the AUDIOVIS than the TONE.

We had assumed that we would see an effect of the normalization in all models, but it is possible that these effects only become significant on larger models. We leave these experiments to future work and while running such an experiment would be substantially more computationally intensive, understanding why these larger models fail to train may allow much more complex models to be trained

5.4 Summary

In this chapter, we explored two potential ways to improve performance, the TimeAutoencoder and normalization methods. In §4.3 we had found that the architectures that focused on temporal features outperformed others on the SENTWORDLIST task. The

Normalization	Mean Val. Acc.	Mean Dev. Acc.	Mean Test Acc.	Mean Acc. (LB - UB)
Baseline	**45.87±0.4**	46.22±0.6	45.91±0.4	(44.35±0.4 - 47.46±0.4)
Baseline + Global	45.34±0.7	46.29±0.9	45.71±0.8	(44.22±0.7 - 47.21±0.8)
None	33.76±0.1	32.91±0.1	33.33±0.0	(33.33±0.0 - 33.33±0.0)
None + Global	44.62±1.2	45.71±0.8	45.13±1.1	(43.68±1.0 - 46.58±1.1)
Scale	45.85±0.3	46.68±0.9	**46.25±0.9**	(**44.72±0.9** - **47.77±0.8**)
Scale + Global	45.56±0.3	46.71±0.6	46.09±0.9	(44.60±0.9 - 47.57±1.0)
Window	45.41±0.6	**46.74±0.9**	46.20±0.5	(44.69±0.5 - 47.70±0.5)
Window + Global	45.58±0.4	46.39±0.9	45.55±0.7	(44.02±0.8 - 47.08±0.7)

Table 5.4: Effects of normalization on the TONE task on the Cam-CAN dataset

results with the TimeAutoencoder— where applying it to LF-CNN and VAR-CNN, which do not incorporate temporal features, was able to increase the performance on the task — reinforce the finding that this task depends strongly on temporal features. We established using various comparison models that this performance increase was also not solely due to the increased complexity of the model.

The TimeAutoencoder did not improve the performance of all models. This may be because it encodes temporal information in a redundant way; this is an open question. It is possible that both of these problems, as well as the issue of high computational cost, may be overcome by incorporating spatial relationships and further reducing the latent embedding. This is because as it stands the computational costs are directly related to the number of channels or the size spatial embedding. While there are hundreds of channels, the spatial embeddings is only of size 16 which massively reduces the computational cost.

We have seen that there is only minor difference between local and global normalization methods but some form of normalization is critical. However, we did not investigate the effects on larger models such as ResNet18 and GoogLeNet and the effects may be more pronounced on these types of models and more research is required in this area.

6
Conclusion

In this thesis we have successfully replicated a very recent piece of work by Zubarev et al and extended it to answer several research questions.

RQ1: Does the superiority of Zubarev et al's proposed deep learning architectures, LF-CNN and VAR-CNN, hold for new large datasets and more challenging tasks? We closely replicated the original work, and as they did found that both the LF-CNN and VAR-CNN outperformed the chosen baseline machine learning classifiers. The new tasks that we introduced showed a significant range in difficulty, with the AUDIOVIS task being slightly more difficult and the HARDEASYSENT being very difficult. In addition, we also found that the modality of the stimulus affected the difficulty of the task. For instance, the SENTWORDLIST was easier when performed with an auditory stimulus as opposed to a visual one.

RQ2: Do our proposed architectures outperform the existing models on our chosen tasks, both the original from Zubarev et al and our more challenging tasks? We proposed three novel architectures, the TimeConv, SERes and SETra, which have all concentrated on learning temporal relationships in the data. We found that the SERes and SETra architectures outperformed both of the LF-CNN and VAR-CNN in several tasks, particularly those where a temporal aspect was clear.

This is similar to the pattern that we found when applying computer vision architectures. We found that both ResNet18 and GoogLeNet generally performed worse than our baselines but it was not always the case.

RQ3a: Can we further improve the performance of a model by using the intermediate latent representation of an autoencoder? We proposed the TimeAutoencoder architecture which learns a temporally enriched embeddings that can replace the input data used in other models. We found that using these embeddings significantly improved the performance of both the LF-CNN and the VAR-CNN on the same tasks where our new models had a significant lead. These embeddings were able to reduce the lead but were

not able to improve the performance pasts models that were designed to take advantage of these features. They nonetheless demonstrate the importance of both pre-training and capturing temporal relationships in the data.

RQ3b: Is it better to use subject level normalization or a global level normalization on datasets? We explored three different strategies of subject-level normalization and found that there is only a small difference among them. There was also only a small difference when comparing subject-level normalization to global-level normalization. However, not applying any normalization completely prevented models from learning any useful features and even models that incorporated batch normalization failed to train. Ultimately, this is an area that requires further exploration with a focus on larger and more complex models.

Future Work This is a new domain for deep learning, and there is a great deal of work to be done on the best way to train models. We have noted that models which contain more than one million parameters (GoogLeNet and ResNet18) would not always train successfully. This suggests that there is some unknown, intermittent and undesirable interaction taking place while training these models. Understanding the cause of this is likely to increase the performance of all architectures significantly.

In the mean time there are two other areas of research that are promising; pre-training and data augmentation. In this work we have shown the benefits of pre-training using an autoencoder; however, we constrained the encoder to only be able to learn temporal features. Developing a method of pre-training that builds on spatial features as well is likely to further increase performance.

Our SERes architecture has achieved state-of-the-art results on temporally-oriented tasks but it is not clear what makes the distinction apparent. It would be interesting to understand the underlying cognitive processes that cause the difference in temporal behavior between the HARDEASYSENT and SENTWORDLIST tasks. This understanding would also likely lead to further performance improvements in neural network architectures.

A
Appendix

	Sentence	Word List
Difficult	The nice lady gave Henk, who had bought a colourful **parrot**, a bag of seeds *Het aardige vrouwtje gaf Henk die een kleurige **papegaai** gekocht had een zak pitjes*	Bag a colourful nice a had who lady **parrot** gave the bought seeds Henk *Zak een kleurige aardige een had die vrouwtje **papegaai** gaf het gekocht pitjes Henk*
Easy	These are no regional **problems** such as those on the Antilles *Dit zijn geen regionale **problemen** zoals die op de Antillen.*	such as no those Antilles **problems** regional are the these on *zoals geen die Antillen **problemen** regionale zijn de dit op*

Table A.1: Literal translations of the Dutch sentences that were used as stimulus. Target words (highlighted in bold) are in the same ordinal position in both the Sentence and the Word List (in the original Dutch sentences)

Subset	Architecture	Val. Acc.	Dev. Acc.	Test Acc.	Sub. Acc.	Mean Acc. (LB - UB)
Audio	SERes	50.64	49.25	49.77	49.80±0.6	(49.52 - 50.08)
	TimeConv	50.49	48.97	49.97	50.00±0.5	(**49.77** - 50.22)
	SETra	50.41	50.53	50.63	50.50±5.2	(48.10 - 52.90)
	LF-CNN	**54.24**	49.83	51.54	51.48±4.9	(49.22 - **53.74**)
	VAR-CNN	50.13	**51.48**	**51.88**	**51.63±4.5**	(49.52 - 53.73)
	GoogLeNet	53.06	48.44	50.91	50.81±4.6	(48.66 - 52.95)
	ResNet18	52.54	48.78	49.72	49.49±3.9	(47.67 - 51.31)
Both	SERes	**55.16**	54.33	55.35	54.95±4.0	(53.64 - 56.25)
	TimeConv	53.74	54.15	54.19	53.57±5.3	(51.87 - 55.27)
	SETra	54.76	53.11	55.11	54.58±4.5	(53.12 - 56.03)
	LF-CNN	53.19	**55.64**	**56.61**	**55.83±5.9**	(**53.94** - **57.73**)
	VAR-CNN	55.02	54.90	55.27	54.70±5.3	(52.99 - 56.40)
	GoogLeNet	54.02	54.89	54.92	54.35±4.7	(52.84 - 55.86)
	ResNet18	54.56	53.98	55.21	54.66±5.1	(53.00 - 56.33)
Visual	SERes	57.73	57.25	59.34	59.27±3.9	(57.32 - 61.22)
	TimeConv	55.34	53.79	58.09	58.05±3.9	(56.14 - 59.96)
	SETra	56.87	57.56	60.09	60.06±3.1	(58.53 - 61.59)
	LF-CNN	58.82	59.40	**60.76**	**60.66±3.0**	(**59.18** - **62.15**)
	VAR-CNN	**59.24**	59.06	59.08	59.00±3.3	(57.35 - 60.64)
	GoogLeNet	57.84	59.05	59.55	59.16±1.1	(58.60 - 59.72)
	ResNet18	56.76	55.96	58.72	58.43±3.0	(56.96 - 59.89)

Table A.2: Results of the HARDEASYSENT task on the MOUS dataset

References

[1] Y. LeCun, Y. Bengio, *et al. Convolutional networks for images, speech, and time series*. The handbook of brain theory and neural networks **3361**(10), 1995 (1995). 1

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks*. In *Advances in neural information processing systems*, pp. 1097–1105 (2012). 1

[3] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O'Reilly Media, 2017). 1, 6, 8, 9, 10, 11, 13

[4] LeCun Yann, Cortes Corinna, and Burges Christopher. *THE MNIST DATABASE of handwritten digits*. The Courant Institute of Mathematical Sciences pp. 1–10 (1998). URL http://yann.lecun.com/exdb/mnist/http://yann.lecun.com/exdb/mnist/{%}5Cnhttp://yann.lecun.com/exdb/publis/index.html{#}lecun-98. 1

[5] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. (2009). URL https://www.cs.toronto.edu/{~}kriz/learning-features-2009-TR.pdf. 1

[6] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. *ImageNet: A large-scale hierarchical image database*. pp. 248–255 (IEEE, 2009). 1, 3

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition*. Neural Computation **1**(4), 541 (1989). 1

[8] K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015). 1409.1556, URL http://arxiv.org/abs/1409.1556. 1, 3, 14

[9] K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (2016). 1, 5, 10, 20, 22, 25

[10] L. Wang and A. Wong. *Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images*. arXiv preprint arXiv:2003.09871 (2020). 1, 2

[11] P. Lakhani and B. Sundaram. *Deep learning at chest radiography: Automated classification of pulmonary tuberculosis by using convolutional neural networks*. Radiology **284**(2), 574 (2017). URL https://doi.org/10.1148/radiol.2017162326.

[12] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. Van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sanchez, and B. Van Ginneken. *Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks*. IEEE Transactions on Medical Imaging **35**(5), 1160 (2016).

[13] F. Milletari, N. Navab, and S. A. Ahmadi. *V-Net: Fully convolutional neural networks for volumetric medical image segmentation*. In *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, pp. 565–571 (Institute of Electrical and Electronics Engineers Inc., 2016). 1606.04797. 1

[14] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krüger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. *Language models are few-shot learners*. ArXiv **abs/2005.14165** (2020). 2

[15] J. Gross. *Magnetoencephalography in Cognitive Neuroscience: A Primer* (2019). 2, 12

[16] Z. J. Koles, M. S. Lazar, and S. Z. Zhou. *Spatial patterns underlying population differences in the background EEG*. Brain Topography **2**(4), 275 (1990). 3

[17] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan. *Filter Bank Common Spatial Pattern (FBCSP) in brain-computer interface*. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 2390–2397 (2008). 3

[18] B. P. Welford. *Note on a Method for Calculating Corrected Sums of Squares and Products*. Technometrics **4**(3), 419 (1962). 3

[19] D. Garrett, D. A. Peterson, C. W. Anderson, and M. H. Thaut. *Comparison of linear, nonlinear, and feature selection methods for EEG signal classification*. IEEE Transactions on Neural Systems and Rehabilitation Engineering **11**(2), 141 (2003). 3

[20] R. Scherer, G. R. Müller, C. Neuper, B. Graimann, and G. Pfurtscheller. *An asynchronously controlled EEG-based virtual keyboard: Improvement of the spelling rate*. IEEE Transactions on Biomedical Engineering **51**(6), 979 (2004). 3

[21] I. Zubarev, R. Zetter, H. L. Halme, and L. Parkkonen. *Adaptive neural network classifier for decoding MEG signals*. NeuroImage **197**, 425 (2019). 1805.10981. 3, 4, 8, 13, 17, 25, 26

[22] F. Wang, S. H. Zhong, J. Peng, J. Jiang, and Y. Liu. *Data augmentation for eeg-based emotion recognition with deep convolutional neural networks*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10705 LNCS, pp. 82–93 (Springer Verlag, 2018).

[23] R. M. Cichy, D. Pantazis, and A. Oliva. *Resolving human object recognition in space and time*. Nature neuroscience **17**, 455 (2014). 3, 13

[24] B. Obermaier, C. Guger, C. Neuper, and G. Pfurtscheller. *Hidden markov models for online classification of single trial eeg data*. Pattern Recognit. Lett. **22**, 1299 (2001). 3

[25] J. M. Schoffelen, R. Oostenveld, N. H. Lam, J. Uddén, A. Hultén, and P. Hagoort. *A 204-subject multimodal neuroimaging dataset to study language processing*. Scientific data **6**(1), 17 (2019). 4, 15

[26] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. *Wavenet: A generative model for raw audio*. ArXiv **abs/1609.03499** (2016). 5

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805 (2018). 5, 10

[28] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. *Deep contextualized word representations* (2018). 1802.05365, URL http://arxiv.org/abs/1802.05365. 5, 10

[29] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. *Stacked convolutional auto-encoders for hierarchical feature extraction*. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, pp. 52–59 (2011). 6

[30] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou. *Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network*. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) pp. 5200–5204 (2016). 6

[31] P. Tzirakis, G. Trigeorgis, M. A. Nicolaou, B. Schuller, and S. Zafeiriou. *End-to-end multimodal emotion recognition using deep neural networks*. IEEE Journal of Selected Topics in Signal Processing **11**, 1301 (2017). 6

[32] Y. Zhou and O. Tuzel. *Voxelnet: End-to-end learning for point cloud based 3d object detection*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 4490–4499 (2018). 6, 12

[33] A. Kendall, H. Martirosyan, S. Dasgupta, and P. Henry. *End-to-end learning of geometry and context for deep stereo regression*. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 66–75 (2017). 6

[34] K. He, X. Zhang, S. Ren, and J. Sun. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. In *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1026–1034 (2015). 1502.01852, URL https://arxiv.org/abs/1502.01852. 6, 11

[35] X. Glorot and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. In *Journal of Machine Learning Research*, vol. 9, pp. 249–256 (2010). URL http://www.iro.umontreal. 6, 11

[36] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics (Springer New York, New York, NY, 2009). URL http://link.springer.com/10.1007/978-0-387-84858-7. 8, 9

[37] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. *WAVENET: A GENERATIVE MODEL FOR RAW AUDIO*. Tech. rep. 1609.03499v2. 10

[38] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. *Greedy layer-wise training of deep networks*. In *Advances in Neural Information Processing Systems*, pp. 153–160 (2007). 10

[39] X. Chai, Q. Wang, Y. Zhao, X. Liu, O. Bai, and Y. Li. *Unsupervised domain adaptation techniques based on auto-encoder for non-stationary EEG-based emotion recognition*. Computers in Biology and Medicine **79**, 205 (2016). 10

[40] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. *Stacked convolutional auto-encoders for hierarchical feature extraction*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6791 LNCS, pp. 52–59 (2011). 10

[41] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. *Extracting and composing robust features with denoising autoencoders*. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103 (2008). 10

[42] Q. Lin, S. Q. Ye, X. M. Huang, S. Y. Li, M. Z. Zhang, Y. Xue, and W. S. Chen. *Classification of epileptic EEG signals with stacked sparse autoencoder based on deep learning*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9773, pp. 802–810 (Springer Verlag, 2016). 10

[43] S. Narejo, E. Pasero, and F. Kulsoom. *EEG based eye state classification using deep belief network and stacked autoencoder*. International Journal of Electrical and Computer Engineering **6**(6), 3131 (2016). 10

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. *Attention is all you need*. In *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 5999–6009 (2017). 1706.03762, URL http://arxiv.org/abs/1706.03762. 10, 11, 20

[45] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al. Language models are few-shot learners*. arXiv preprint arXiv:2005.14165 (2020). 10

[46] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. *Generative Pretraining from Pixels*. Tech. rep. (2020). 10

[47] Anonymous. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. ICLR pp. 1–19 (2021). 10

[48] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* (2015). 1502.03167, URL http://arxiv.org/abs/1502.03167. 11

[49] Y. Wu and K. He. *Group Normalization*. International Journal of Computer Vision **128**(3), 742 (2020). 1803.08494, URL http://arxiv.org/abs/1803.08494. 11

[50] J. L. Ba, J. R. Kiros, and G. E. Hinton. *Layer Normalization* (2016). 1607.06450, URL http://arxiv.org/abs/1607.06450. 11

[51] R. Hari and A. Puce. *MEG-EEG Primer* (Oxford University Press, 2017). 12

[52] N. Brodu, F. Lotte, and A. Lécuyer. *Comparative study of band-power extraction techniques for motor imagery classification*. 2011 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB) pp. 1–6 (2011). 12

[53] P. Herman, G. Prasad, T. McGinnity, and D. Coyle. *Comparative analysis of spectral approaches to feature extraction for eeg-based motor imagery classification*. IEEE Transactions on Neural Systems and Rehabilitation Engineering **16**, 317 (2008). 12

[54] Z. Koles, M. S. Lazar, and S. Z. Zhou. *Spatial patterns underlying population differences in the background eeg*. Brain Topography **2**, 275 (2005). 12

[55] Y. Zhang, G. Zhou, J. Jin, X. Wang, and A. Cichocki. *Optimizing spatial patterns with sparse filter bands for motor-imagery based brain–computer interface*. Journal of Neuroscience Methods **255**, 85 (2015). 12

[56] K. K. Ang, Z. Y. Chin, C. Wang, C. Guan, and H. Zhang. *Filter bank common spatial pattern algorithm on bci competition iv datasets 2a and 2b*. Frontiers in Neuroscience **6** (2012). 12

[57] T. Hoya, G. Hori, H. Bakardjian, T. Nishimura, Y. Miyawaki, A. Funase, and J. Cao. *Classification of single trial eeg signals by a combined principal + independent component analysis and probabilistic neural network approach* (2003). 12

[58] N. Castellanos and V. Makarov. *Recovering eeg brain signals: Artifact suppression with wavelet enhanced independent component analysis*. Journal of Neuroscience Methods **158**, 300 (2006). 12

[59] J. Bayliss and D. Ballard. *Single trial p300 recognition in a virtual environment* (1998). 12

[60] N. Xu, X. Gao, B. Hong, X. Miao, S. Gao, and F. Yang. *Bci competition 2003-data set iib: enhancing p300 wave detection using ica-based subspace projections for bci applications*. IEEE Transactions on Biomedical Engineering **51**, 1067 (2004). 12

[61] M. Bojarski, D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. *End to end learning for self-driving cars*. ArXiv **abs/1604.07316** (2016). 12

[62] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, A. Y. Hannun, B. Jun, T. Han, P. LeGresley, X. Li, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, S. Qian, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, A. Sriram, C. Wang, Y. Wang, Z. Wang, B. Xiao, Y. Xie, D. Yogatama, J. Zhan, and Z. Zhu. *Deep speech 2 : End-to-end speech recognition in english and mandarin*. ArXiv **abs/1512.02595** (2016). 12

[63] A. Graves and N. Jaitly. *Towards end-to-end speech recognition with recurrent neural networks*. In *ICML* (2014). 12

[64] D. Ardila, A. Kiraly, S. Bharadwaj, B. Choi, J. Reicher, L. Peng, D. Tse, M. Etemadi, W. Ye, G. Corrado, D. Naidich, and S. Shetty. *End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography*. Nature Medicine **25**, 954 (2019). 12

[65] I. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. *Building end-to-end dialogue systems using generative hierarchical neural network models*. In *AAAI* (2016). 12

[66] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance. *Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces*. Journal of Neural Engineering **15**(5), 056013 (2018). URL http://dx.doi.org/10.1088/1741-2552/aace8c. 13, 14

[67] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. *Deep learning with convolutional neural networks for EEG decoding and visualization*. Human Brain Mapping **38**(11), 5391 (2017). 1703.05051, URL http://doi.wiley.com/10.1002/hbm.23730. 13

[68] A. Laakso and G. Cottrell. *Content and cluster analysis: Assessing representational similarity in neural systems*. Philosophical Psychology **13**(1), 47 (2000). 13

[69] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger. *A review of classification algorithms for EEG-based brain-computer interfaces: A 10 year update* **15**(3) (2018). 13

[70] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert. *Deep learning-based electroencephalography analysis: a systematic review*. Journal of

Neural Engineering **16**(5), 051001 (2019). URL https://iopscience.iop.org/article/10.1088/1741-2552/ab260c. 13

[71] M. A. Shafto, L. K. Tyler, M. Dixon, J. R. Taylor, J. B. Rowe, R. Cusack, A. J. Calder, W. D. Marslen-Wilson, J. Duncan, T. Dalgleish, R. N. Henson, C. Brayne, E. Bullmore, K. Campbell, T. Cheung, S. Davis, L. Geerligs, R. Kievit, A. McCarrey, D. Price, D. Samu, M. Treder, K. Tsvetanov, N. Williams, L. Bates, T. Emery, S. Erzinçlioglu, A. Gadie, S. Gerbase, S. Georgieva, C. Hanley, B. Parkin, D. Troy, J. Allen, G. Amery, L. Amunts, A. Barcroft, A. Castle, C. Dias, J. Dowrick, M. Fair, H. Fisher, A. Goulding, A. Grewal, G. Hale, A. Hilton, F. Johnson, P. Johnston, T. Kavanagh-Williamson, M. Kwasniewska, A. McMinn, K. Norman, J. Penrose, F. Roby, D. Rowland, J. Sargeant, M. Squire, B. Stevens, A. Stoddart, C. Stone, T. Thompson, O. Yazlik, D. Barnes, J. Hillman, J. Mitchell, L. Villis, and F. E. Matthews. *The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) study protocol: A cross-sectional, lifespan, multidisciplinary examination of healthy cognitive ageing*. BMC Neurology **14**(1) (2014). 14

[72] R. T. Schirrmeister, J. T. Springenberg, L. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. *Deep learning with convolutional neural networks for eeg decoding and visualization*. Human Brain Mapping **38**, 5391 (2017). 14

[73] M. T. Ribeiro, S. Singh, and C. Guestrin. "*Why should i trust you?*" *Explaining the predictions of any classifier*. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-Augu, pp. 1135–1144 (2016). 1602.04938, URL https://arxiv.org/abs/1602.04938. 16

[74] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015). Software available from tensorflow.org, URL https://www.tensorflow.org/. 17

[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (2019). 1912.01703, URL http://arxiv.org/abs/1912.01703. 22

[76] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, P. Dhariwal, D. Luan, and I. Sutskever. *Generative pretraining from pixels*. In *Proceedings of the 37th International Conference on Machine Learning* (2020). 22, 23

[77] Anonymous. *An image is worth 16x16 words: Transformers for image recognition at scale*. In *Submitted to International Conference on Learning Representations* (2021). Under review, URL https://openreview.net/forum?id=YicbFdNTTy. 22, 23

[78] L. D. Brown, T. T. Cai, and A. Das Gupta. *Interval estimation for a binomial proportion*. Statistical Science **16**(2), 101 (2001). 25

[79] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. CoRR **abs/1412.6980** (2015). 25

[80] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1–9 (IEEE Computer Society, 2015). 1409.4842, URL https://arxiv.org/abs/1409.4842v1. 25

[81] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. *Scikit-learn: Machine learning in python*. ArXiv **abs/1201.0490** (2011). 36