MSc. Computing Science

Department of Computing

Imperial College of Science, Technology and Medicine

Deep Convolutional Neural Networks for Brain Computer Interface using Motor Imagery

Author:
Ian Walker

Supervisors:
Dr. Marc Deisenroth
Dr. Aldo Faisal

Imperial College London

September 4, 2015

Submitted in partial fulfillment of the requirements for MSc. Computing Science

Abstract

This paper presents a novel application of convolutional neural networks, classifying user intent generated through motor imagery and signalled using EEG data, with the intent of using it as input to a real-time brain-computer interface (BCI). The motivation is to design a system using which a player can control a video game character in the Cybathlon 2016 challenge. To the best of the author's knowledge, it is the only paper attempting to classify more than two types of motor imagery using deep learning techniques. The paper proposes a novel method for defining covolutional filters along the scalp to disjoint groups of electrodes that measure activity in similar regions of the brain. Although initial results were found to demonstrate overfitting very late into the project, the preliminary results from a revised experimental set-up still show significant learning and opportunities to improve the results in future research.

Acknowledgments

A big thank you to my advisors, Marc Deisenroth and Aldo Faisal, for the opportunity to work on this project and for their guidance throughout.

The Tesla K40 used for this research was donated by the NVIDIA Corporation.

Contents

Chapter 1

Introduction

The aim of this project is to read a person's mind and to convert user intent into actionable signals. The particular application of the system developed here is for a pilot to control a video game character in a competition to get through a virtual obstacle course; however, there are numerous other possible practical uses. For instance, a wheelchair-bound individual could control the wheelchair using a brain-computer interface if she is unable to do so by other means, or someone fitted with a robotic limb could control it in an intuitive manner. Being able to classify brainwaves could also possibly yield further insights into the structure of the brain and to deepen our understanding of its functionality. The use of machine learning pattern recognition techniques applied here means that it is possible to discover features previously missed by researchers using techniques that rely on humans to heuristically identify relevant characteristics in the data.

A great advantage of the approach used is that it is non-invasive, fairly low-cost, and easy to adapt to different situations. All that is required is an EEG (electroencephalographic) cap for the user to wear and a computer to interpret the output. Of course, there are other methods that can be used to collect more precise data on brainwaves, such as implants surgically placed inside of an individual's skull, but if it is possible to avoid such a costly and risky procedure in favour of something one can wear externally, the latter approach is preferable. The potential disadvantage is that the data may be too noisy to be useful, but the system developed over the course of this project is able to get high accuracy in classifying user intent despite the noise in the data. Applying this approach in a practical setting thus seems to be a plausible possibility.

From the user's perspective, the means by which she can signal her intended action is straightforward but takes a bit of getting used to. The user imagines one of four actions, such as making a fist or pushing her feet into the floor, without actually performing the action. The resulting patterns in the brainwaves recorded can then be used to determine which action it is that the user imagined and to translate that signal into the intended output.

Although there was found to be overfitting to the data in the initial approach very late into the project, a revised experimental set-up still showed promising preliminary results, as well as avenues for potential improvement, which could be explored in future research. For one subject, an accuracy of around 80% was achieved in clas-

sifying three types of motor imagery using untransformed voltage data. This paper's contribution is to apply convolutional neural networks in a novel context, designing the algorithm which makes it possible. Many common applications of neural networks are to graphic images, where it is rather intuitive how one can apply a convolutional filter. It is less obvious how one can apply such a filter to a set of electrodes reading EEG data, but this paper develops a procedure by which to group electrodes, so that it becomes natural to apply one or more convolutional layers in the model.

Chapter 2

Background

2.1 Problem Setting and Application

The intent of this project is to provide the EEG signal analysis necessary for a BCI system to compete in ETH Zürich's Cybathlon 2016 [1]. The challenge is described as follows:

> *Pilots are equipped with brain-computer interfaces (BCIs) that enable them to control an avatar in a racing game played on computers. Each avatar moves forward by itself, and thus, eventually reaches the finish line of the race even with a bad or no input signal from the pilot. However, there are obstacles on the race track that the pilots should avoid. Sending an appropriate command using the BCI within the correct time frame allows pilots to avoid these obstacles (elements) and gain a bonus (e.g. accelerate). Incorrect commands or incorrectly timed commands yield a disadvantage (e.g. deceleration of the avatar). A maximum of three different commands can be sent from the BCI simultaneously. One command is mandatory, and the other two are optional, but allow the pilots to gain additional advantages.*

In this competition, the primary BCI type will be the EEG and all signals for commands must come from measured brain activity, i.e. no muscular or ocular movements may be used. Additionally, no audio or visual queues may be used to invoke steady state visually evoked potentials (SSVEPs) apart from the output of the race. Teams will be ranked based on the time taken to complete the race.

The race is implemented in the BrainRunners video game. Up to four teams will race their avatars simultaneously. The pilots will send one of three commands *JUMP*, *KICK*, or *SPEED* when their in-game avatar reaches a certain coloured *action pad*. Incorrectly timed or executed commands before or within action pads will cause the avatar to stumble and lose time. Since the avatar will always move forward, the race can be run without any inputs, but this will result in a time of 235 seconds. If an action pad is reached, but no command is sent, the pad crossing time is 12 seconds, if the correct command is sent, the time is 2 seconds. Additionally, on a KICK action pad, if the command is correctly executed, all other competitors on the pad receive a 2 second penalty. Stumbling from erroneous commands results in a

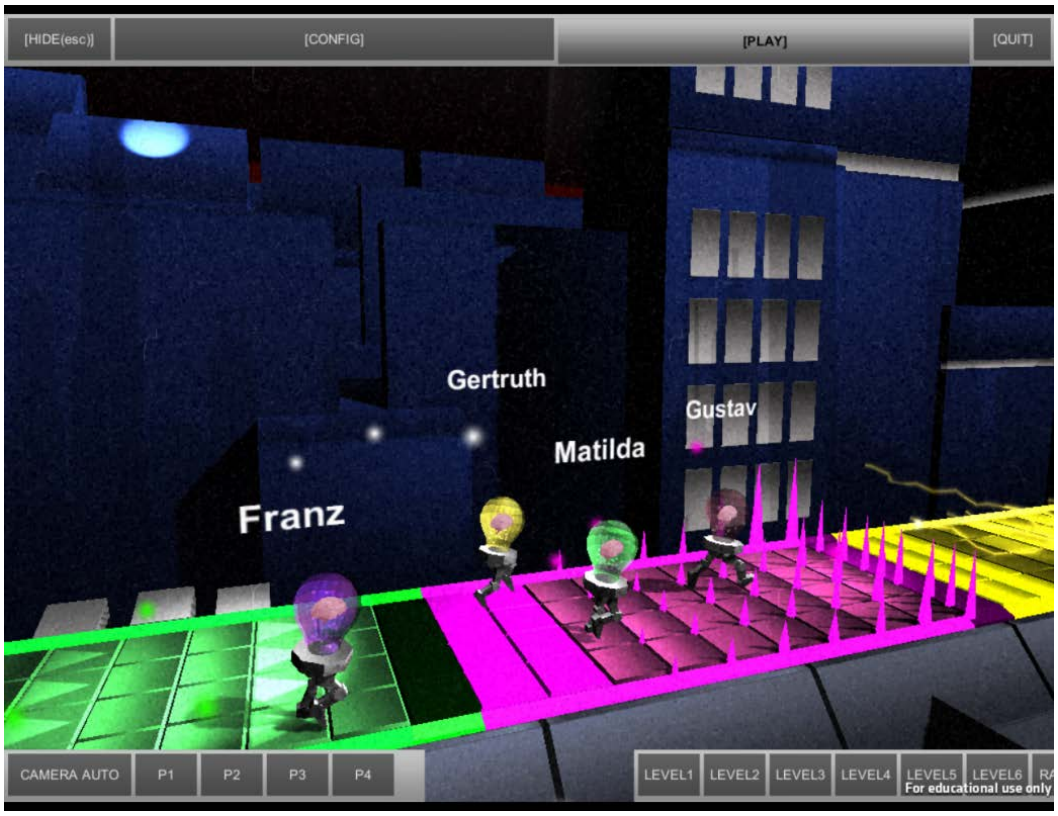


Figure 2.1: In-game screen-shot of four competitors during the race. The coloured sections of the track demonstrate each of the three action pads: green for SPEED, magenta for JUMP, and yellow for KICK.

2.5 second penalty. Figure 2.1 presents a screen-shot of the race in action, giving a sense of what pilots will see and to what they will have to react.

2.2 Literature Review

The following sections provide a background in the literatures necessary to understand the project. Section 2.2.1 describes recent achievements in BCI technology and why this field of research is of such interest and importance. Section 2.2.2 discusses what electroencephalography is, why it is selected as the basis for the BCI, and recent literature that seeks to overcome its relative disadvantage compared with other methods of analysing brain activity, its poor signal-to-noise ratio. Section 2.2.3 discusses the Deep Learning techniques that will be applied and the few recent studies which apply them to EEG data.

2.2.1 Brain Computer Interface

A Brain Computer Interface (BCI), or Brain Machine Interface (BMI), is a system which allows users to communicate or manipulate external devices using only their brain signals as opposed to the standard methods for carrying out such tasks [2].

BCIs allowing patients to accomplish tasks simply by thinking about a certain action has many diverse applications that provide real benefits to their lives. Recent research has shown that much is possible with this technology. BCIs have allowed individuals to control a robotic arm to manipulate the surrounding environment [3]. They have allowed patients to move around in a wheelchair [4]. They have even been shown to increase the neuroplasticity of recovering tetraplegic patients and improve their quality of life [5]. BCIs which allow for three dimensional cursor motion have also been constructed [6]. This suggests there are many more potential applications of the technology to entertainment and real-world manipulation more broadly.

[2] provide an excellent survey of recent work in the area of BCIs as well as the basic definitions used throughout the field. They describe a BCI as carrying out the following four procedures. First, it must have a method for signal acquisition, a way of measuring the neurophysiological state of the brain. This can be accomplished by recording electrophysiological signals, via devices like the EEG or intracortical implants discussed below. Second, it must have a process for feature extraction, whereby the useful information gleaned from the signals, removing artifacts or noise from the analysis. Third, it then uses a translation algorithm to convert the extracted features into a signal of user intent. Finally, the system executes the user's desired output. Most research into BCIs has focused on the first three steps with the relevant question of interest being how to cheaply, safely, and accurately translate a user's brain activity into real-time actions. The following section goes into more detail about which systems for measuring the neurophysiological state of the brain are selected and their drawbacks, as well as a brief overview of some of the techniques that have been used to accomplish procedures two and three.

2.2.2 Electroencephalography

Electroencephalography (EEG) is a method to analyse brain activity by measuring the electrical activity across a subject's scalp. EEG offers many advantages for construction of a BCI system, but also several disadvantages. Firstly, and most importantly, EEG is a non-invasive method for measuring brain activity. This removes the need for costly and risky surgical procedures, such as electrophysiology, in which intracortical devices such as needles or tubes may be inserted directly into the brain material, or electrocorticography, in which an array of electrodes is implanted under the skull. Both systems risk permanent and life threatening damage to a patient's brain and require costly surgical expertise to carry them out safely. Also useful for designing a BCI, EEG does not require the patient to be stationary like other non-invasive imaging systems such as functional magnetic resonance imaging (fMRI) and magnetoencephalography (MEG), both of which can only be carried out by large scale and expensive equipment. In contrast, EEG simply requires the placement of a set of electrodes along the scalp, which, although the exact placement is important for valid results, can be carried out in a straightforward manner. EEG has the ability to produce high time resolution data, which is a necessity for near real time systems. However, EEG does possess some major drawbacks.

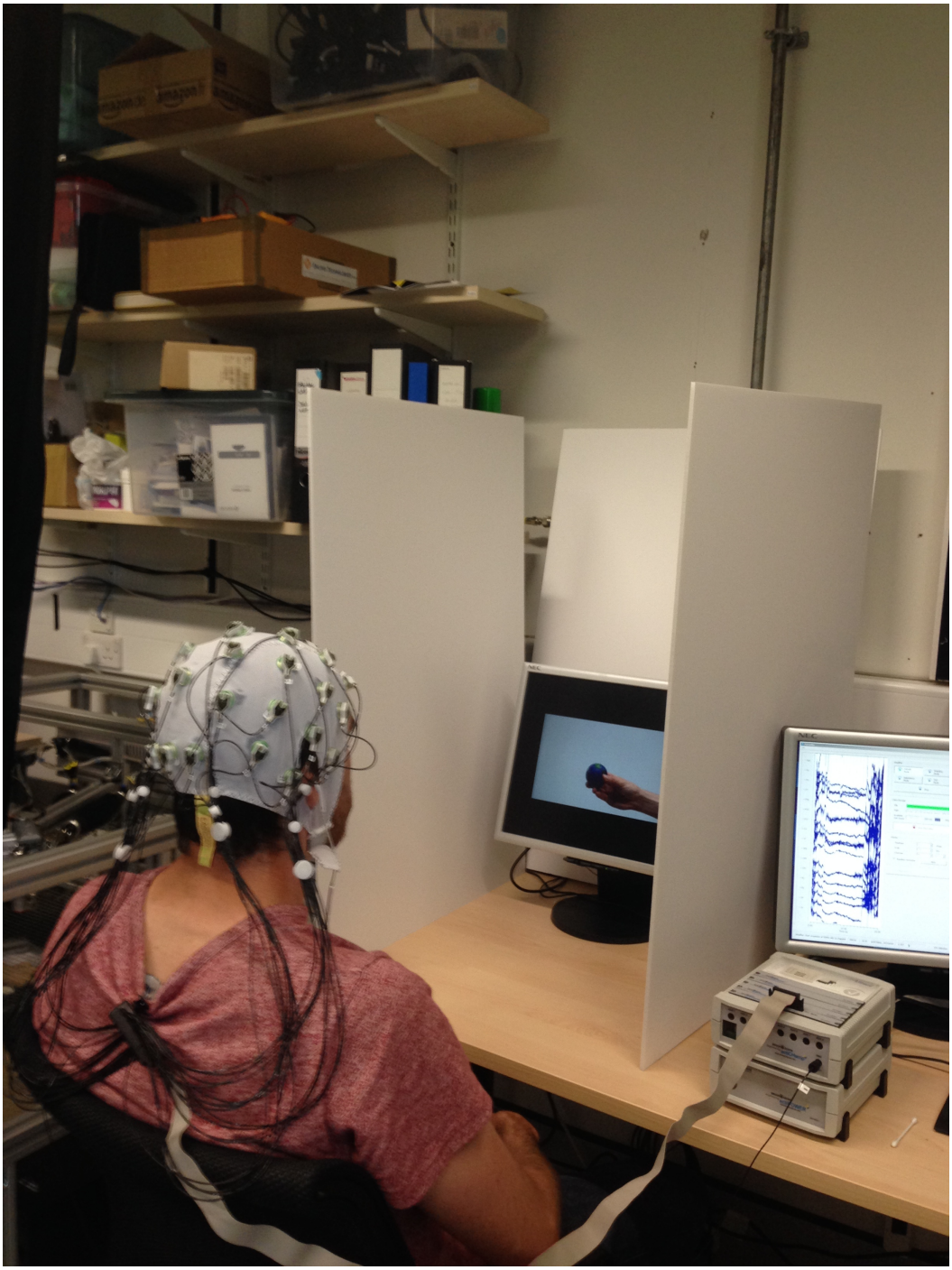


Figure 2.2: An example of a person wearing an EEG cap during an experiment. [7]

EEG's major drawback for use in a BCI is its low signal-to-noise ratio. While it does provide high resolution in time, it does not provide high spatial resolution. Invasive procedures can measure the firing of specific neurons, while MEG and fMRI can give highly accurate descriptions of which sub-regions of the brain are active at a given time step. EEG can only give the output of the typically 32, but occasionally hundreds of, electrodes, which measure the microvolts transmitted through the skull. The EEG output is a set of noisy and non-stationary time series. The indirect nature of the measurements makes the EEG inverse problem, determining which brain structures are active to produce a given output, intractable. Additionally, since it only measures activity at the scalp, it provides little to no insight as to the activity of deep structures in the brain such as the hippocampus, or even neurons in the sulci or fissures near the skull, as the signals they produce either do not reach the scalp or are lost in the much stronger signals produced by structures close to the skull such as the occipital lobe. Unfortunately, the nature of the measurement procedure also makes EEG susceptible to numerous artifacts that are not related to underlying cerebral activity. For example, electrocardiographic artifacts that arise from the heart beating, electromyographic artifacts which could be recorded if a patient moves an arm while recording, or movement of the eyes and tongue. Even the rhythmic pulse of the alternating current of the country's electrical system must be accounted for before an accurate representation of the electrical activity emanating from the patient's brain can be produced. These problems make analysis of the underlying brain activity, and thus any classification of user intent a BCI may wish to implement, very difficult. However, recent research has demonstrated that clever filtering and feature extraction with EEG signals can produce meaningful and accurate classifications of user intent.

The main way to classify user intent in an EEG-based BCI is through motor imagery. Motor imagery is when a subject imagines performing a certain action such as opening or closing the left or right hand or moving a foot. Motor imagery has been used to achieve surprisingly accurate BCI control. For example, [8] attain 98% accuracy for one patient using a common spatial patterns (CSP) analysis of the EEG signals. The particular brain wave activity that can be used to identify motor imagery was first identified by [9]. Specifically, [10] demonstrate that the μ rhythm, which appears in the motor cortex at frequencies of 7.5 to 12.5 Hz, desynchronizes on the hemisphere of the brain contralateral to the imagined action, and increases in the ipsilateral hemisphere. Using data of this form, [11] use empirical mode decomposition (EMD) coupled with CSP to exploit the spatial relationship of the EEG signals. With this, the authors manage to obtain an average of 77.7% accuracy among 10 participants. [12] attempt to deal with the non-stationarity of the EEG signals by applying Morlet wavelets along with the CSP algorithm. The resulting BCI system achieves a maximal accuracy of 91% and can be used to control a virtual car in 3 dimensions.

More recently, [7] use a left/right hand motor imagery classifier which is based on a combination of Morlet wavelets and CSP. The system achieves an average accuracy of 88% across 8 subjects. Impressively, this accuracy can be attained with only 15 trials of each type of imagery for each patient, demonstrating a marked improve-

ment in user training speed from previous studies. This paper serves as the basis of analysis for the current project and the data used by [7] will be used for initial results of this study.

2.2.3 Deep Learning

The main difficulty faced thus far in building an effective BCI using EEG data remains effectively separating the signal from the noise and then interpreting that signal in a meaningful way. Identifying which features of the EEG output are most reliable as well as selecting the most appropriate EEG spectral components is crucial to understanding the subject's intentions. In previous EEG-based BCI studies, identification strategies have been based on the input of human experts to identify and mark particular features of the EEG signal. Deep learning methods have the potential for automatic feature detection from a wider range of the data which may have been previously overlooked from focusing solely on the output of a few electrodes or frequencies. For this reason, while the overall classification accuracy is the main goal of this study, the learned features which a trained network may utilise may be of equal significance to future research.

Some studies have in fact applied recent advances in Deep Learning techniques to EEG data. [13] implement a Time-Delay Neural Network (TDNN) to predict the propagation of epileptic seizures through the brains of two patients who suffer from the disease. They enhance the analysis by using Independent Component Analysis (ICA) to filter the data fed into the TDNN. The authors use this process to predict 1200 observations, or three seconds worth, of EEG data from the previous 800 time steps with very low cross-validation mean square error for one of the patients. In subsequent research, the authors apply a CNN to predict patients' epileptic seizures by classifying brain activity into inter-ictal and pre-ictal states [14]. The authors consider EEG recording samples of five seconds for six electrode channels. Concerned that the literature as stands relies on unnecessary reductions in the number of extracted features and overly simplistic classification methods, four different ways of bivariate feature extraction are tested: maximal cross-correlation, non-linear interdependence, difference of short-term Lyapunov exponents, and a wavelet analysis based measure of synchrony. These features are then used to classify the brain states of 21 patients from the Freiburg EEG database using logistic regressions, CNNs, and SVMs. The structure of the CNNs is based on the structure found in [15]. The CNNs produce impressive results. For 20 out of 21 patients, they manage to achieve zero false alarm seizure predictions, up to 99 minutes before the onset of a seizure. These results clearly demonstrate the sensitivity and specificity a model using EEG data can attain when using CNNs. However, both these studies are concerned with seizure detection not with classifying motor imagery, and both use intracranial EEG (iEEG), also known as ECoG, which requires an invasive medical procedure to place electrodes under the skull.

[16] apply a Deep Belief Network (DBN) to the scalp EEG data produced by four male students imagining left or right hand movement. The network shows the ability to correctly classify the signals on average 83% of the time. The authors use

a window of seven seconds for each trial, during which the subjects are shown a blank screen for two seconds, followed by an instruction for one second, and four seconds for the motor imagery. They note that classification accuracy is much higher during the first two seconds of motor imagery than the subsequent two seconds, hypothesising that the concentration of the subjects falls off rapidly. The authors do not consider classifying more than two states, leaving it to future research. They also do not consider the application of CNNs, and the method for feature extraction is very rudimentary, looking only at weak classifiers trained on single channel series.

This study seeks to expand the existing literature by applying CNNs to EEG data of motor imagery that is not binary. It also seeks to improve on existing results by incorporating some of the more successful methods of feature extraction from previous studies, while ensuring these pre-processing stages still allow the BCI to run in real-time.

2.3 Convolutional Neural Network Basics

This section intends to give the necessary background in convolutional neural networks used to generate the results of this paper. The explanations presented here lean heavily on the following resources: [17] [18] [19] [20] [21] [22]. A number of other good explanations of these topics can also be found online.

Convolutional neural networks are a type of feed-forward neural network originally proposed by [15]. Feed-forward networks pass an input through one or more layers of *neurons* or *nodes*, where each neuron represents a linear combination of its input. These linear combinations are then passed through a, typically non-linear, activation function, and passed to the next layer, culminating in an output layer. Depending on the structure of these networks, they are able to learn highly non-linear functions and have been applied with great effect, recently learning to identify sclerotic bone lesions [23] and to conduct pedestrian detection to aid drivers [24], as well as other interesting applications.

2.3.1 Fully Connected Networks

The most basic feed-forward neural network is a fully connected network in which each neuron is connected with all of the previous layer's neurons.[21] [20] An example with three layers is presented in Figure 2.3. Each edge in the diagram represents a weight connecting a node to a node in the previous layer. A three layer fully connected neural network can be summarized as a function $f : \mathbb{R}^N \to \mathbb{R}^M$, where N is the dimension of the input and M is the dimension of the outputs [25]:

$$f(\mathbf{x}) = s(W^2(s(W^1\mathbf{x} + b^1)) + b^2) \tag{2.1}$$

where $f(\mathbf{x})$ is the output of the network. $W^l \in \mathbb{R}^{N_l \times N_{l-1}}$ is the weight matrix between layer l and layer $l-1$. $w^l_{ji} \in W^l$ is the weight that connects the node j in layer l with node i in layer $l-1$; these are the edges in the diagram. $b^l \in \mathbb{R}^{N_l}$ is the bias vector of layer l where each $b^l_i \in b^l$ is the bias associated with node i of layer l,

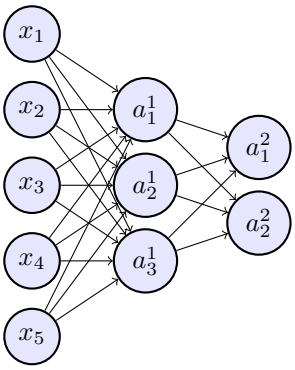


Figure 2.3: Example of a fully connected neural network with five inputs, x_i, comprising layer 0, three hidden units in layer 1, and two output units in layer 2.

which serves to shift the linear combination of inputs and weights. Each $s(\cdot)$ is an activation function, applied element-wise, described in more detail below, which can be used to produce a non-linear shift in the data. Thus, the output, or activations, $\mathbf{a}^{l+1}$ of a given layer $l+1$ of a fully connected network can be generally described as:

$$\mathbf{z}^{l+1} = W^{l+1}\mathbf{a}^{l} + b^{l+1} \tag{2.2}$$

$$\mathbf{a}^{l+1} = s\left(\mathbf{z}^{l+1}\right) \tag{2.3}$$

The goal of the learning task is then to learn the set of all weight matrices and bias vectors $\Theta = \{\mathbf{W}, \mathbf{b}\}$ in the network by minimising some loss function. In a supervised learning task, there exists a set of labelled examples $\mathbb{D} = \{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{x}_i \in \mathbf{X}$ is an example and the corresponding $\mathbf{y}_i$ is its label. Loss functions can thus be designed that compare the predicted value of the network for a given input with the actual label of that input. To see how a neural network can produce a prediction for labelled data see Section 2.3.3. Several potential loss functions exist, but this paper will focus on the categorical cross entropy:

$$\mathcal{L}(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j} p(y^i = j)\log\left(h_\Theta^j(\mathbf{x}^i)\right) \tag{2.4}$$

where $\{\mathbf{x}^i, y^i\}$ is the i^{th} labelled example and m is the number of labelled examples. $p(y^i = j)$ is the *true* probability distribution, for certain labels this means that $p(y^i = j) = 1$ if $y^i = j$ and $p(y^i = j) = 0$ otherwise. $h_\Theta^j(\mathbf{x}^i)$ is the hypothesis of the neural network when parametrized by Θ that the i^{th} example belongs to the j^{th} class, i.e. it is the probability that the neural network assigns to the event that $\mathbf{x}^i$ belongs to the j^{th} class. The closer the model is to correctly predicting the example's label, the lower the cross entropy score. If there are only two classes, this loss function can be represented as binary cross entropy [22]:

$$\mathcal{L}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right) \tag{2.5}$$

The $\mathcal{L}(\Theta)$ can be modified by adding additional terms to put constraints on the parameters. These include the commonly used L1 and L2 regularizations, as well

as other methods [22], with the intent to reduce over-fitting, a problem to which neural networks are particularly prone.

The learning task is therefore:

$$\Theta^{\star} = \arg\min_{\Theta} \mathcal{L}(\Theta) \tag{2.6}$$

2.3.2 Gradient Descent

There are many ways to solve the minimisation problem in equation 2.6, but the standard method for solving it is gradient descent. In mini-batch gradient descent, the gradient is computed over a subset of size m of the training examples, referred to as the batch or the mini-batch. Once all training examples have been used to compute the gradient, i.e. when all mini-batches have been processed, this signifies the end of a training epoch, one full pass through the data.

Stochastic gradient descent [19] is controlled by another hyperparameter to consider, the learning rate η. η determines how large of a step is taken when the parameters are updated. Large values for η mean that convergence may occur quicker, but, depending on the topology of the cost function, may cause the algorithm to miss the global minimum or oscillate. Thus, selection of the proper η is essential to optimal learning, as illustrated in Figure 2.4.

The algorithm is prone to becoming stuck at local minima, and so other algorithms and modifications have been proposed to resolve some of these issues. This paper focuses solely on using stochastic gradient descent to solve the learning task, but some problems discussed in the results section indicate future research should

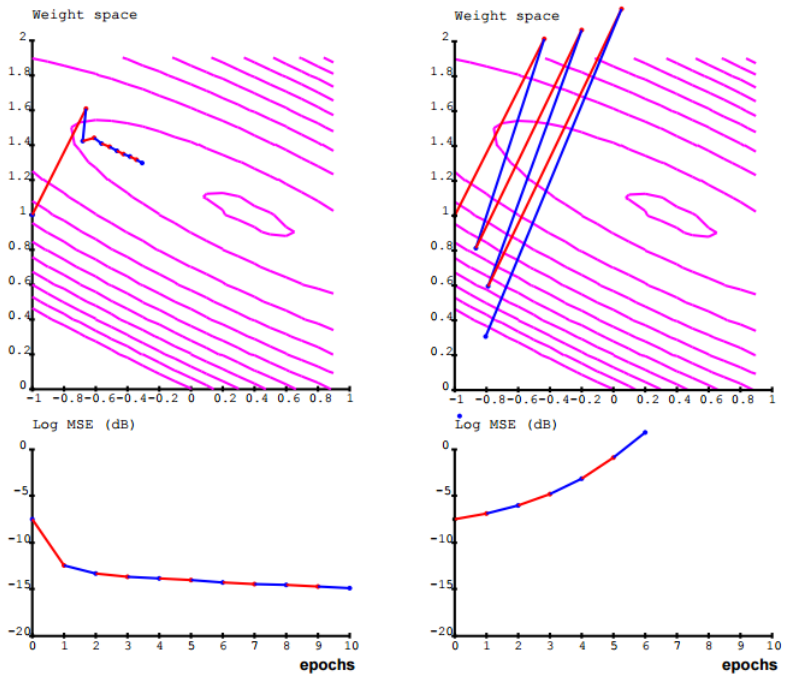


Figure 2.4: Example of stochastic gradient descent, taken from [26].

investigate more sophisticated algorithms.

2.3.3 Activation Functions

Activations are a major choice point when designing neural networks. [18] They determine how the inputs are transformed throughout the network, which is essential for the network's ability to learn complex functions. A purely linear activation function could be chosen at each layer, but then the outputs would be simply linear transformations of the inputs, thus ruling out the ability to learn more complex functional forms. Thus, non-linear activations are preferred for their increased expressibility.

Traditionally, researchers have used the sigmoidal function [20]:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

and the hyperbolic tangent [20]:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.8}$$

These functions have the advantage that they are non-linear, differentiable everywhere, and have easily computable derivatives, which is useful for efficient computation of gradient descent:

$$\frac{\delta\sigma(z)}{\delta z} = \left(\frac{1}{1 + e^{-z}}\right)\left(\frac{-e^{-z}}{1 + e^{-z}}\right) = \sigma(z)\left(1 - \sigma(z)\right) \tag{2.9}$$

$$\frac{\delta \tanh(z)}{\delta z} = 1 - \tanh^{2}(z) \tag{2.10}$$

Note that the advantage of each of these activations during gradient descent is that the derivative of each is a function of the original function. Since $\sigma(z)$ and $\tanh(z)$ are calculated during the feed-forward phase, they need not be recomputed during the back-propagation phase.

Another widely used activation function is the Rectified Linear Unit or ReLU. [17]

$$\text{ReLU}(z) = \max(0, z) \tag{2.11}$$

This has several documented advantages over the previous functions. Since the output is either 0 or the input, it can be quickly computed relative to the other presented functions. Networks using the ReLU also suffer less from the vanishing gradient problem and impose sparsity on the activations. The activation of the final layer is typically reserved for the softmax$(\cdot)$ function. [19] This function has several nice properties that make it very useful for classification and prediction:

$$P(Y = j|z) = \text{softmax}_{j}(z) = \frac{e^{z_j}}{\sum_{i} e^{z_i}} \tag{2.12}$$

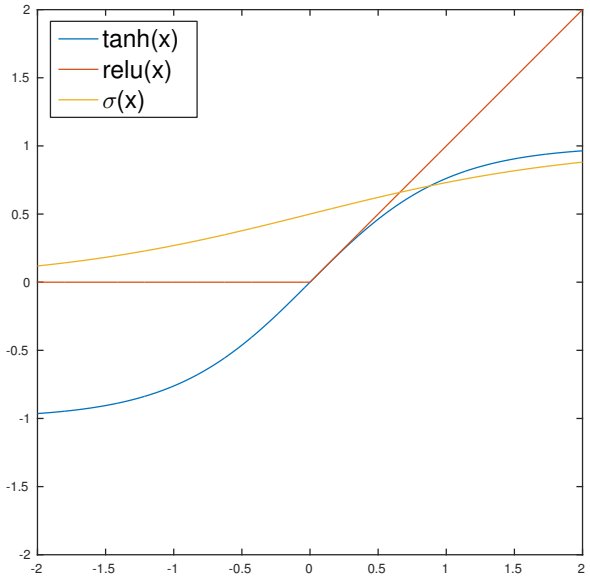


Figure 2.5: Plot of regularly used activation functions $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, $\sigma(z) = \frac{1}{1+e^{-z}}$, and $\text{ReLU}(z) = \max(0, z)$ for $-2 \leq z \leq 2$.

Equation 2.12 presents the softmax for an arbitrary class j in the classification problem. Diagrammatically, the number of classes is represented as the number of output nodes, for example, Figure 2.3 could be interpreted as having two classes. Thus, in the output layer $z = W^l \mathbf{a}^{l-1} + b$ is the linear combination of weights and inputs from the previous layer, with z_i the linear combination associated with class i in the output layer. The softmax$(\cdot)$ is calculated by dividing by the sum of the exponentiation of the linear combination associated with each possible class, and so has the nice interpretation of being a probability distribution over the classes. This leads to a simple classification rule where the hypothesis or prediction of the neural network parametrised by $\Theta = \{\mathbf{W}, \mathbf{b}\}$ for a given input $\mathbf{x}$ is given by:

$$h_\Theta(\mathbf{x}) = \hat{y} = \arg\max_j P(Y = j|\mathbf{x}, \Theta) \tag{2.13}$$

For simplicity, this paper uses $\tanh(\cdot)$ activations for intermediate layers and uses a softmax$(\cdot)$ layer for classification in the final layer. However, future research would experiment with alternative activation functions such as the ReLU$(\cdot)$ for intermediate layers.

2.3.4 Sparsely Connected Networks

The first step towards converting a fully connected neural network to a convolutional neural network is by changing the *receptive field* of the neurons. The receptive field of a neuron refers to the number of nodes in the previous layer to which the neuron is connected. [22] For example, in Figure 2.3, the neurons in the second and third layers have receptive fields which cover all the nodes of the previous layer, hence the fully connected nature of the network. The network presented in Figure 2.6 has a second layer whose neurons have a receptive field including only three nodes in

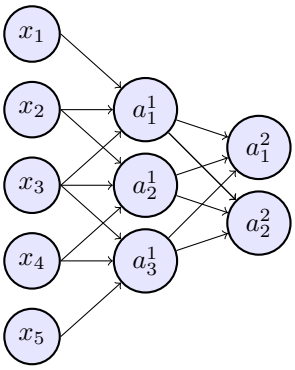
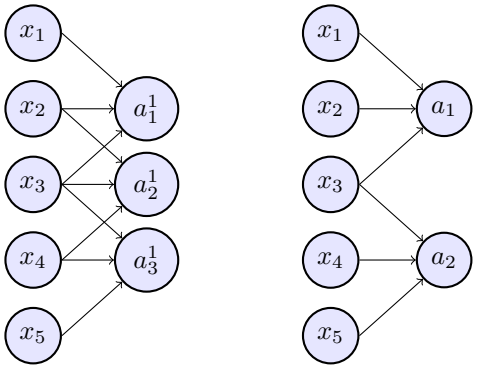


Figure 2.6: Example of a sparsely connected network, where the receptive field of the neurons in the second layer is three as opposed to five in the example network in Figure 2.3. The output layer is fully connected to the second layer.



(a) Stride = 1 (b) Stride = 2

Figure 2.7: Example of a sparsely connected layer with a receptive field of three and strides = 1 or 2.

the input layer. Since neurons at each layer are not connected to every neuron in the previous layer, this network can be referred to as a sparsely connected network. Reducing the receptive field of the neurons has the advantage of reducing the number of weights the model needs to learn, from 21 in the fully connected network to 15 in the sparsely connected network, which can reduce the chances of over-fitting and improve computation time. While this may reduce the expressibility of the network, with valid assumptions on the receptive field, this may not negatively impact accuracy and may actually improve convergence during learning.

Another way to reduce the number of connections in the network is to introduce the concept of stride. Stride refers to the spacing of the receptive fields in the previous layer. Focusing on the sparsely connected layer, Figure 2.7 presents a second layer with a stride of one and receptive field of three (2.7(a)) and one with a stride of two and receptive field of three (2.7(b)). This once again reduces the number parameters needed.

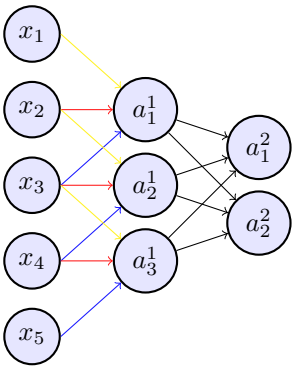


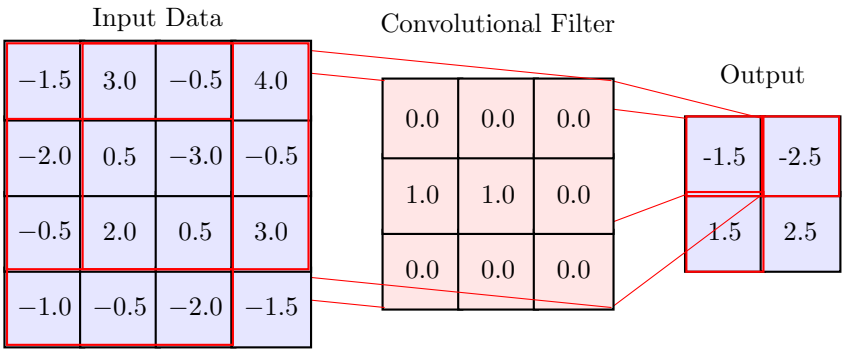
Figure 2.8: Example of a convolutional neural network. Edges of the same colour signify shared weights, thus giving the network its name.

2.3.5 Convolutional Neural Networks

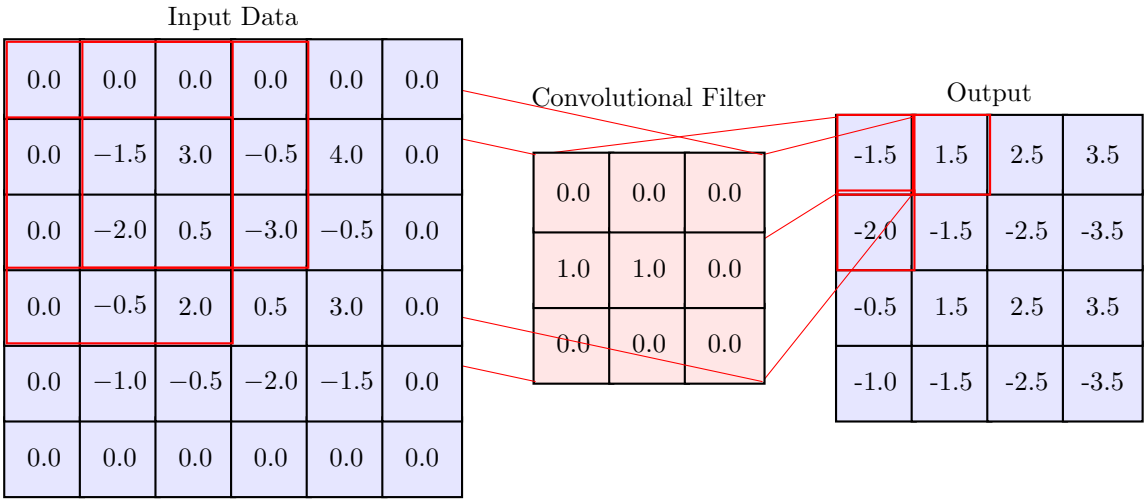
The number of parameters can be further reduced by imposing shared weights among the receptive fields of each neuron in the layer. [22] This means that between receptive fields weights in the same position are constrained to be the same. Figure 2.8 illustrates a sparsely connected layer with shared weights. Since the same weights are applied in the same configuration across the input nodes, a sparsely connected layer can therefore be thought of as a convolution of a filter of a certain shape across the layer's inputs, hence the name, convolutional layer. The convolutional layer imposes further assumptions on the structure of the input data, thus reducing the expressibility of the network, however, it can massively reduce the number of parameters the model needs to learn, reducing the chances of over-fitting and improving convergence and computation time. In the toy example in Figure 2.8, the model need only estimate 9 weights compared with 21 for the fully connected network. Each layer can also be designed to learn multiple filters, thus improving the expressibility.

A nice property of the convolutional layer is that the filters can be viewed as being applied to data in the input that is *next to* each other. [17] Indeed, in its original application the convolutional neural network was first applied to 28×28 images of hand written digits in the MNIST data set [15]. The convolutional filters learned in the first layer were essentially edge detectors, which succinctly parse the spatial information in the image and can be combined in later layers to form more complex features. The convolutions can also be applied by the same principle to other data such as time series that display similar properties. In this paper, convolutions will be defined across the time dimension of EEG signals and the spatial dimension of electrodes which are physically close to each other. Both of these will be explored in more detail in Chapter 3. Figure 2.9 demonstrates an example of applying two dimensional filters to two dimensional input data. Note that the output data is also commonly referred to as the feature maps.

The convolutional layer is typically paired with a *sub-sampling* or *pooling* layer. Typical options for this layer are either average-pooling or max-pooling. [20] In an average-pooling layer, the elements of the input within a certain window are averaged together to form the output. In a max-pooling layer, the maximal element



(a) Example of a learned 3×3 convolutional filter passing over a 4×4 input with stride $= 1$ to produce a 2×2 output.



(b) The same input with zero padding of thickness 1 on each side. Now passing the learned 3×3 filter over the input with stride one produces a 4×4 output. Zero padding is sometimes applied to the input data if the researcher wishes to have direct control over the size of the feature map.

Figure 2.9: Examples of two dimensional convolutional filters applied to a two dimensional input.

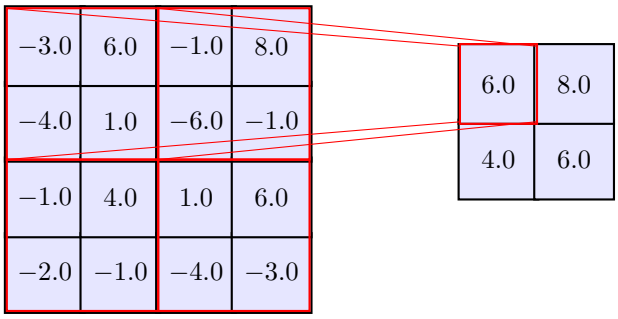


Figure 2.10: Example of 2×2 max-pooling with stride $= 2$ on a 4×4 input.

within each window of the input data is used as the output, an example of which is presented in Figure 2.10. This paper uses max-pooling layers as they can further reduce the scale of the input, which can greatly reduce the number of parameters the model needs to compute. It has another nice property in that it makes subsequent layers indifferent to slight translations in the input data. [22] For example, if the learning task is to identify an increase in neural activity within a given window, it matters less that it occurs at the fifth time slice or the sixth times slice, than whether it occurred at all. In this way, the max-pooling layer can also aid in convergence. However, max-pooling is a blunt tool which drops potentially meaningful data from consideration. Too much max-pooling, i.e. windows that are too large in one or more of the dimensions, may lose too much data for the network to distinguish between classes, thus reducing accuracy.

Convolutional neural networks can also be trained using stochastic gradient descent with only slight modifications to take into account shared weights and max-pooling, and this is the process through which learning will be achieved in this paper. The exact structure of the convolutional neural network used is presented in Section 3.6.

2.4 Hardware and Software

A major concern with deep neural networks is the amount of time it takes for the learning to converge. Deep neural networks with thousands of nodes in each layer, along with large amounts of training examples, mean that training a neural network can take hours, days, if not weeks, even on a high quality CPU. For this reason, most modern attempts at training deep learning models are carried out using parallel processes on GPUs. Most of these programs are written using CUDA [27], which was created by NVIDIA for use on their GPUs. All experiments done for this study are carried out on one of NVIDIA's most powerful GPUs, the Tesla K40c, [28]. This GPU was graciously provided by NVIDIA for use with this study. This GPU has 2880 CUDA cores, 12 GB of RAM, and is capable of 1.43 Tflops on double precision floating point numbers. The Tesla k40c has been shown to improve throughput by about 15 times over a CPU for a baseline problem. In an exploratory phase of this paper, even bigger improvements were recorded with the same logistic regression running for 580 minutes on a CPU taking only 15 minutes when run on the Tesla. This speed

up is of course very useful when searching for valid hyperparameters such as the learning rate, depth of the network, and number of nodes in each layer.

To take advantage of the GPU, the deep learning algorithms must be programmed in CUDA, which utilises C++. However, there are many possible libraries that can be used which implement the CUDA functionality with additional capabilities specifically designed for machine learning. This study uses Theano for the implementation of the machine learning algorithms. The code used to create the neural networks is based on Theano implementations provided by [25]

Chapter 3

Contribution

3.1 Data

The data are produced from motor imagery experiments carried out by [7] and [29].

3.1.1 Left and Right Hand Motor Imagery Data

The data provided by [7] is comprised of eight male subjects, aged between 23 and 29 and without maladies. Each subject was asked to perform 30 motor imagery trials during which they imagined opening and closing either the left or the right hand, 15 trials were performed for each hand for each subject. Each trial is defined as a ten second period, starting with a sound to draw the subject's attention to the screen, after one second a cross appeared in the center of the screen, and at two seconds an arrow indicating left or right appeared on the screen for the subject to imagine the corresponding action for the following eight seconds. The description of the rest of the experimental set up from [7] is reproduced below:

> The participants were seated in a comfortable chair approximately 70 cm away from a digital computer screen. Conventional headphones were used to present the auditory stimuli, which were played at a safe volume level according to the guidelines given by The National Institute for Occupational Safety and Health (NIOSH). The computer screen was surrounded by white plastic panels in order to reduce undesired visual information from the environment and subjects predisposition to briefly look away from the screen once fatigued, causing ocular artefacts. EEG signals were acquired with 32 Ag-AgCl referential active electrodes placed on a Easy Cap recording cap (EASYCAP GmbH, Herrsching, Germany), and arranged according to the international 10-20 system and then amplified by BrainVision actiChamp (Brain Products GmbH. Gilching, Germany) and recorded with PyCorder software (Brain Vision LLC, Morrisville, USA). The latter also offers a graphical user interface to adjust the impedance between subjects scalp and electrodes: before each recording, the impedance was adjusted via common procedure to be below 15 k for every electrode. The channel placed in the right mastoid

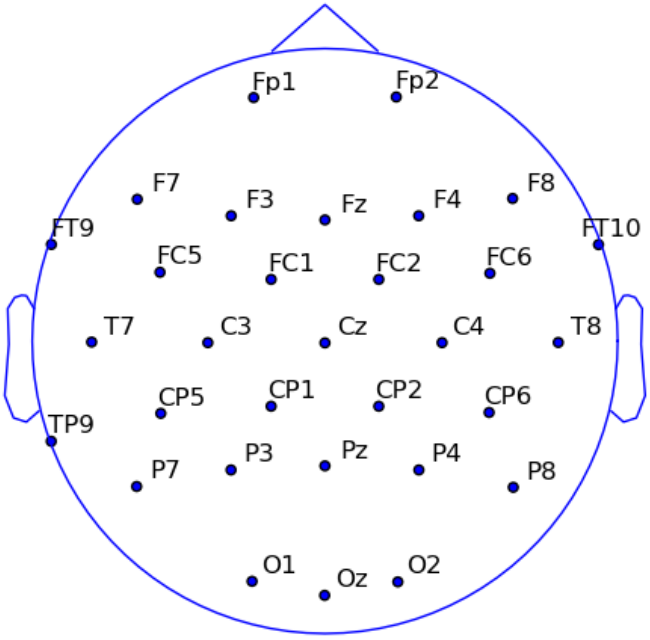


Figure 3.1: International 10-20 system layout of the 31 electrode channels used in the left and right hand motor imagery data provided by [7]. TP10 on the right mastoid would be the 32^{nd} electrode, but has been removed as it is used as the reference electrode.

> (TP10) was chosen as reference. A notch filter at 50 Hz was also applied by the actiChamp in order to remove noise from the standard AC electrical line current.

The electrode layout used is displayed in Figure 3.1. As described above the the channel placed at the right mastoid (TP10) was used as a reference, meaning the data for each trial consists of 31 electrode channels recording electric activity in μV at 500Hz. Thus, each trial $r \in \mathbb{R}^{31\times5000}$.

3.1.2 Cybathlon Motor Imagery Data

The second dataset is provided by [29] and seeks to mimic the format of the Cybalthon racing video game described in Section 2.1. The experimental set up is very similar to that of [7], but instead of trials with an arrow indicating left or right, the subjects are now presented a random video of game-play in which the in-game avatar runs across one of the four possible patches, green, yellow, or purple pads, and the grey non-pad areas. The subjects are then asked to imagine one of four actions at the start of the respective area: opening and closing the left hand (purple), opening and closing the right hand (green), pushing down with both feet (yellow), and squeezing the abdominal muscles (grey). The experimental setting again uses

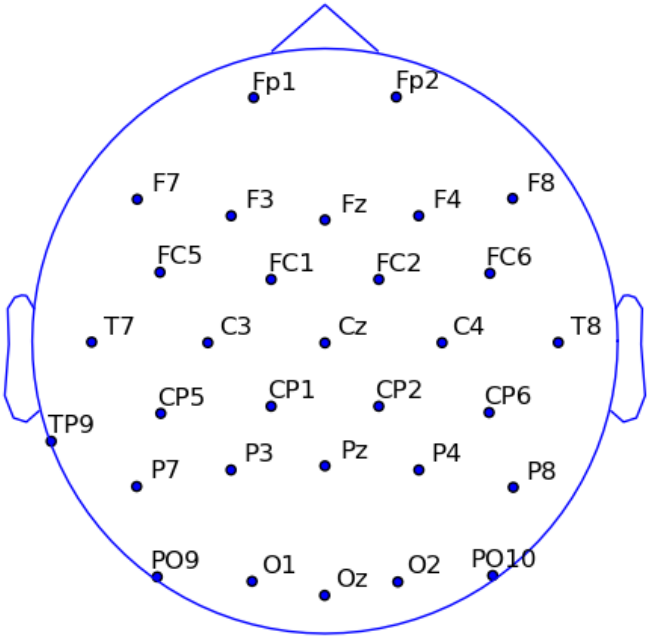


Figure 3.2: International 10-20 system layout of the 31 electrode channels used in the grey, green, yellow, and purple motor imagery data provided by [29]. TP10 on the right mastoid would be the 32^{nd} electrode, but has been removed as it is used as the reference electrode.

the right mastoid channel (TP10) as the reference channel, but uses a slightly different electrode layout presented in Figure 3.2. In addition, electrodes were placed on the left and right legs, left and right arms, and the stomach, as well as recording an EOG sensor for verifying the individuals are not actually performing the actions and to aid in the rejection of muscular-skeletal artefacts. This leaves 31 electrode channels on the scalp recording electrical activity in μV at 200Hz.

3.1.3 Data Preprocessing and Artefact Rejection

For both datasets, artefacts were rejected by z-transforming each channel and applying a threshold of 25 to the cumulative z-score. Any trials that exceeded this were discarded, as in [7]. For a given trial and a given channel, the z-score is the following:

$$z_t = \frac{x_t - \mu}{\sigma} \tag{3.1}$$

Here, μ is the mean of the observations across time and σ is similarly the standard deviation of those observations. These z-scores were then summed across channels (still for a given trial) and those trials in which these cumulative z-scores were found to be too high were discarded. The number of trials kept and discarded for each

Table 3.1: Number of trials for each subject, left-right imagery

Subject	Trials kept		Trials rejected	
	L	**R**	**L**	**R**
MM	4	4	11	11
KP	15	13	0	2
AS	14	15	1	0
BM	13	14	2	1
AM	13	15	2	0
MX	13	13	2	2
GZ	22	19	3	6
AF	26	25	4	5

Table 3.2: Number of trials for each subject, 4-class imagery

Subject	Trials kept				Trials rejected			
	Grey	**Green**	**Yellow**	**Purple**	**Grey**	**Green**	**Yellow**	**Purple**
EG	30	28	21	18	4	3	3	1
LG	89	93	77	66	21	5	5	4

subject can be found in Tables 3.1 and 3.2. Subsequently, the raw channel data had linear trends removed and a 6^{th} order Butterworth bandpass filter between 1 Hz and 30 Hz was applied.

3.1.4 Voltage and Transforms

Even after this preprocessing, the voltage data remains very noisy, as presented in Figure 3.3. Deciphering any underlying structure in these plots is of course a well documented problem in the literature and the main thrust of this project. Although extracting meaningful structure directly from the raw voltage data may be preferable for computation time, during the real time runs of the BCI and for the generality of the solutions learned, it may be difficult for machine learning algorithms to handle such data. For this reason, it may be necessary to transform the data into a form in which structure can be more readily identified.

Signal processing has been used to great effect in classifying EEG signals. The two main algorithms are the Fourier Transform and the Morlet Wavelet. However, since these are simple linear transforms applied to each channel, it is reasonable to assume that a convolutional layer may be able to learn a useful approximation of each. Thus, it may not be necessary to apply these transformations, with the preferable property of being able to rely on the voltage data. Therefore, the results include the application of the neural networks to both transformed and not transformed voltage data.

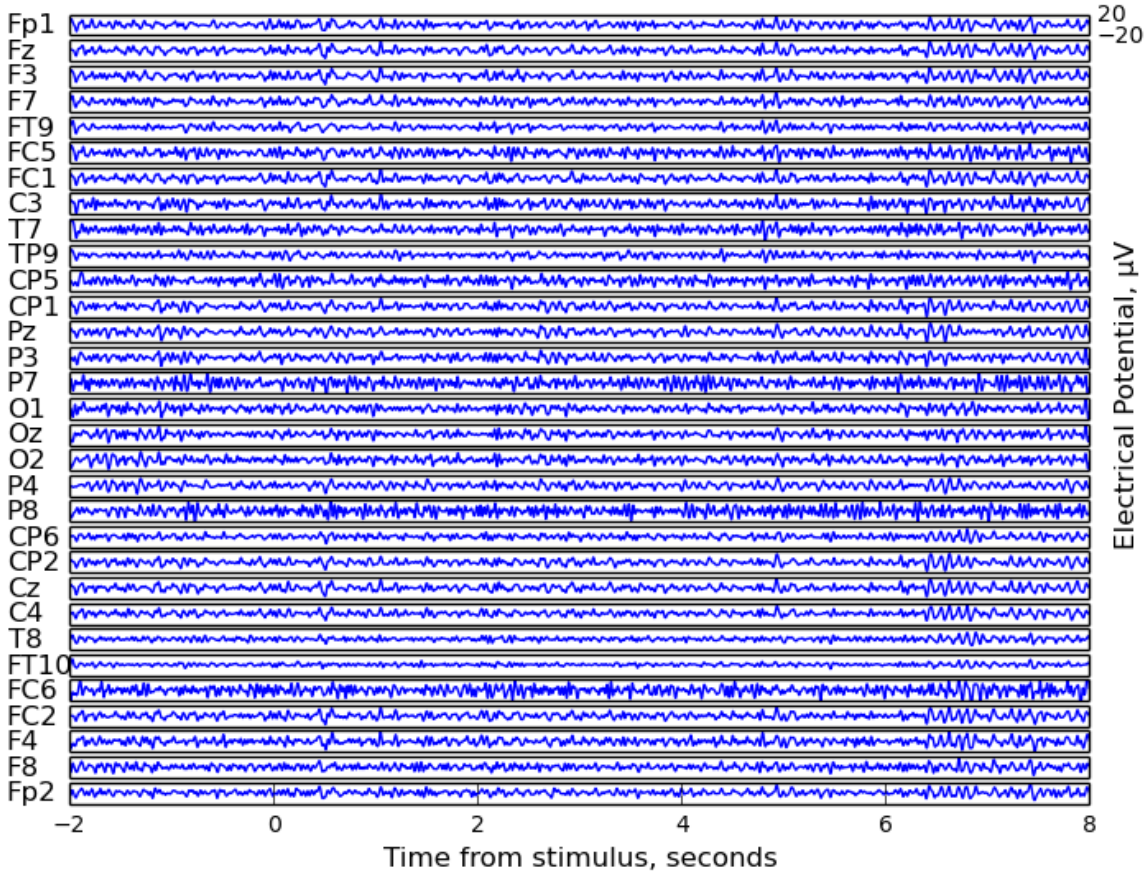


Figure 3.3: Electrical activity in μV for the 31 electrodes used in the left and right hand motor imagery experiment. These data come from the first trial of the left hand motor imagery task for the subject AF in the study by [7]. Each electrode channel has been preprocessed by linear detrending and application of a 6^{th} order Butterworth bandpass filter between 1 Hz and 30 Hz. Each subplot has a range of -20 to 20 μV.

3.2 Approximating Real-time Analysis

As the intention of the neural network is to be used to classify user intent in a real-time BCI, the labelled examples they are trained on must be constructed to reflect this fact. For a given subject, after preprocessing, there exists a data matrix $\mathbf{V} \in \mathbb{R}^{n \times C \times T}$, where n is the number of trials, C is the number of electrode channels, and T is the number of voltage readings. T is not necessarily the full 10 seconds of EEG measurements for each trial, but only those selected for analysis. It is possible that individuals tire rapidly during the motor imagery task, as has been documented in some cases such as [16], as their minds begin to wander after the initial stimulus and focus. If this is the case, then observations towards the end of the trial may not accurately represent a motor imagery task, introducing another degree of freedom which may impact the accuracy of the model. Indeed, how much of the trial data in the time dimension is used becomes important for achieving meaningful learning discussed in the results. A potential counter argument to cutting off the data before

the 10 second mark is that data that is towards the end of the trial could still represent legitimate motor imagery by the subject, and so data is being thrown away, which may lead to poor out of sample performance. However, even if the subject is still imagining the motor task at the end of the 10 second trial, the signal may be a lot weaker, and so the label for the individual's intent may be implicitly thought of as imagining motor task m at 8 seconds after stimulus. Given the problem setting in which a correct command must be sent as the avatar reaches a pad in the Cybathlon race, correctly predicting a motor imagery task 8 seconds after stimulus is also not needed nor desirable.

There will be a trade off between how accurate a classification can be and how fast the classification can be obtained. Feeding more data into the classifiers should allow them to pick up on more distinguishing characteristics, but larger input examples will increase computation time online, and may reduce sensitivity to changing user intent. For this reason, the choice of the time dimension of the training examples may distinguish an excellent BCI classifier from a poor one. 50 and 100 time slices were chosen as a good starting point for this analysis, as at 500 Hz this represents 0.10 or 0.20 seconds of data, respectively, and at 200 Hz, this represents about 0.25 or 0.50 seconds of incoming data, respectively.

3.2.1 Voltage Data

Extracting the labelled training examples is slightly different for the Morlet transformed data and the voltage. For the voltage data, to extract the maximal amount of training examples from the experimental data, the examples were constructed by taking each 100 time observations with maximal overlap. For example, the first training example would include the time observations $0, \ldots, 99$, the second would include the time observations $1, \ldots, 100$, etc. By reshaping in this manner, the subject data matrix $\mathbf{V}$ becomes $X \in \mathbb{R}^{N \times C \times \tau}$, where N is the number of examples, given by $N = n(T - \tau + 1)$, and τ is the length of the time window used. For the voltage data, $\mathbf{X}$ is now the matrix of examples to be used in the training.

To generate the labels for the examples, the first examples from a trial are given the label 0 for *no thought*, the first example to be labelled as a motor imagery example is the first example that includes a time slice after the motor imagery task begins. This is the first example to include a time slice after the two second mark for each trial. This and all subsequent examples are labelled as the motor imagery task m of the given trial. The *no thought* examples may be dropped depending on the desired comparisons. This results in a vector of labels $\mathbf{y} \in \mathbb{R}^N$. This process is illustrated in Figure 3.4.

3.2.2 Morlet Wavelet Data

Applying a Morlet wavelet transform allows for some preprocessing of the voltage data to extract relevant time and frequency information. It helps to pick out activity in the voltage data signal at and around the wavelet's central frequency. The application of the Morlet wavelet transformation is different, but can also approximate

a real-time setting. Since the Morlet wavelet can be calculated before the run-time, using a support of four seconds, it can be convolved with the incoming data rapidly. This means that the system would have to wait for 4 seconds before beginning to classify incoming data with a fully supported transformation when the system is first turned on, but this is not a limiting factor for the problem setting.

The Morlet function $\mathbf{\Psi}(t, f_0)$ is defined as:

$$\mathbf{\Psi}(t, f_0) = \frac{1}{\sqrt{\sigma_t \sqrt{\pi}}} e^{-\frac{t^2}{\sigma_t^2}} e^{j2\pi f_0 t} \tag{3.2}$$

For each trial r and for each channel c in $\mathbf{V}$, the Morlet wavelet is applied as follows:

$$W^{r,c}(a, t) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \mathbf{V}^{r,c}(t) \mathbf{\Psi}\Big(\frac{\tau - t}{a}, f_0\Big) d\tau \tag{3.3}$$

From past research, we know that the rhythm of interest for motor imagery tasks, the μ-rhythm, ranges from 7-12 Hz. [30] [31] Following [7], the central frequency of the wavelet, f_0, is then defined to be 10 Hz to pick up on features in the relevant frequency range. Similarly following [7] and [32], m is set to 14, where $m = \frac{f_0}{\sigma_f}$ and $\sigma_f = \frac{1}{2\pi\sigma_t}$. Doing so sets the time and frequency resolutions, which depend on σ_t and σ_f, respectively. Once we compute $W^{r,c}$, we transform it element-wise to get the power, which is the square of the amplitude. For an element w in $W^{r,c}$, the transformation is $|w|^2$. The power is what is used from here on out when referring to the Morlet-transformed data.

Only those sections of the data which have enough voltage observations to span the wavelet convolution are used. This leaves the transformed data matrix $\mathbf{W} \in \mathbb{R}^{n \times C \times T_W}$ for each subject, where $T_W = T - 4F_s + 1$, where $4F_s$ is the length of the Morlet wavelet with -2 to 2 second support with sampling frequency F_s. $\mathbf{W}$ is then reshaped to form the training examples as with the voltage data above, to produce a matrix of examples of a given time slice length of Morlet wavelet transformed voltage data. Again, maximal overlap of the time slices is used to obtain the greatest possible number of training examples. This produces the matrix of examples $X_W \in \mathbb{R}^{N_W \times C \times \tau}$, where τ is the length of the time slice and $N_W = T_W - \tau + 1$ is the number of examples.

If the label for each example is defined as above, with the first example to be labelled as the trial's motor imagery task to be the first which includes data after the stimulus, then the first example in X_W has label m for the motor imagery task in the trial for which the example is derived. Unfortunately, this means that there are no *no thought* labelled examples in X_W as there are not enough observations at the beginning of the trials to apply the Morlet wavelet to solely *no thought* data. Thus, the label for each example is simply m, the motor imagery task for the trial from which the example is derived. The labels are stored in $\mathbf{y_w} \in \mathbb{R}^{N_W}$. This process is illustrated in Figure 3.5.

3.3 Defining Convolutions in Electrode Space

A major question for applying convolutional neural networks to the scalp EEG data is how to define the convolutional filters to extract information from both the time

Figure 3.4: Example of how the trial data is converted into a matrix of labelled examples, $\mathbf{X}$, when using voltages. In this example, the last two seconds of EEG readings are dropped, signified by the grey area. The overlapping red and yellow frames demonstrate how the trial is chunked using time slice length τ. Note that the first example to be labelled as the motor imagery task is the first

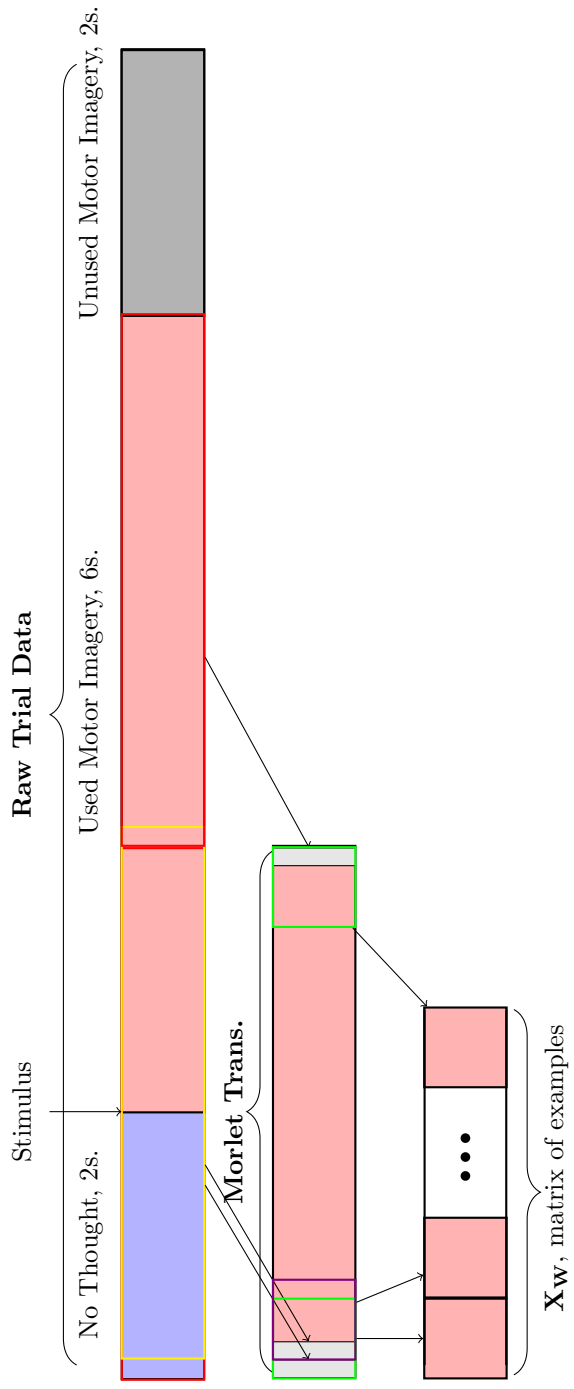


Figure 3.5: Example of how the trial data is converted into a matrix of labelled examples, $\mathbf{X_W}$, when first transforming the data with a Morlet wavelet with 4s support. In this example, the last two seconds of EEG readings are dropped, signified by the grey area. The overlapping red and yellow 4s frames on the raw trial data demonstrate how the Morlet wavelet transforms 4s of voltage trial data into a single Morlet transformed observation, the light grey segments. The overlapping green and purple frames on the transformed data demonstrate how the transformed data is chunked using time slice length τ. Note that all examples in $\mathbf{X_W}$ are labelled as the motor imagery task.

and spatial dimensions. In its original application, the convolutional neural network was applied to classifying handwritten images, with the implicit assumption that adjacent pixels in the image represent regions of the real life object which are a similar distance apart. One can think of each example for the matrix X as a $C \times \tau$ *image*, with an electrode channel axis of size C and a time axis of size τ. Convolutional filters that have an electrode channel dimension greater than one will convolve the *neighbouring* electrodes, thus incorporating the spatial information between the electrodes. It seems intuitive that the spatial information may be of great use as an individual electrode may miss the activity of an underlying brain structure that a convolution of electrodes in the same vicinity on the scalp — and thus covering similar underlying brain structures — may identify.

The problem is how to represent the neighbouring electrodes so that the convolutions can be applied in a way that hews to the anatomical reality. One could think of treating each electrode as a *pixel* placed in relation to the other electrodes along the scalp. A convolution could be achieved by padding the the *pixels* between the electrodes with zeros. However, it is not clear how much to pad the space between the electrodes to achieve an approximation of equidistance. Padding with zeros between electrodes on the scalp would also imply that these constructed electrodes were reading no electrical activity, a false assumption given the biological reality. Instead, one could think of interpolating the voltages or transformed voltages in the areas between the electrodes and using the interpolated image of the electrical activity across the scalp to convolve over. An example of this approach can be seen in Figure 3.6 below.

However, this again has the undesirable quality of essentially assuming the valid construction of interpolated *electrodes*, as well as the difficulty of achieving the cor-

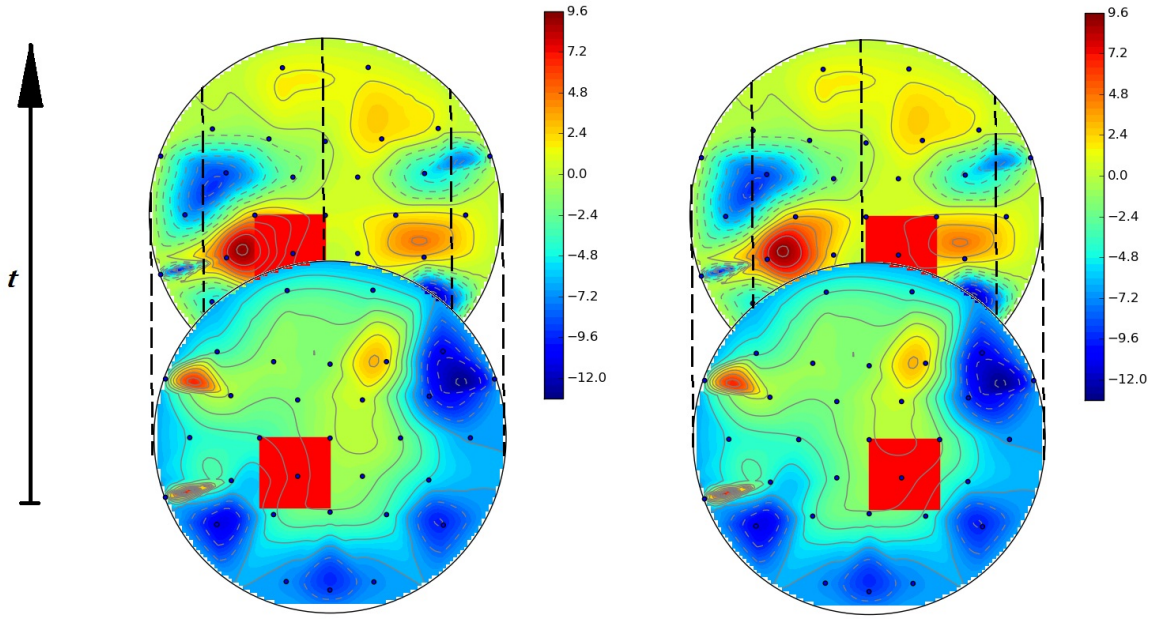


Figure 3.6: A possible example of how one might apply a spatial convolution to the electrodes after interpolating brain activity between them. This approach was not used.

rect interpolation of the points on the scalp, a non-trivial task even with the assumption of a hemispherical skull.

Instead, a more elegant solution, the one used in this paper, is to find the disjoint nearest neighbours for each electrode and to organise the channel dimension of X by the groupings. Disjoint nearest neighbours means that from the set of electrodes, groups of n electrodes are found such that no electrode is in more than one group and the sum of the distances between electrodes in a group is minimised. In this way, a convolutional layer may be defined that incorporates spatial information from electrodes on the scalp above similar regions of the brain and reduces the dimensionality of the input data in the spatial direction. (An example of groups formed in such a manner can be seen in Figures 3.9 and 3.10.) In other words, we find groups g_1, g_2, ..., g_k, where $g_i = \{node_1^i, ..., node_n^i\}$, such that $g_1 \cap (g_2 \cup ... \cup g_k) = \emptyset$ and $g_2 \cap (g_1 \cup g_3 \cup ... \cup g_k) = \emptyset$ and so on for each group. Within each group — indeed, it is the basis on which groups are selected — the total distance between every pair of electrodes is minimized:

$$\min \sum_j \sum_{k>j} dist(node_j, node_k) \tag{3.4}$$

This is explained more rigorously in Algorithm 1.

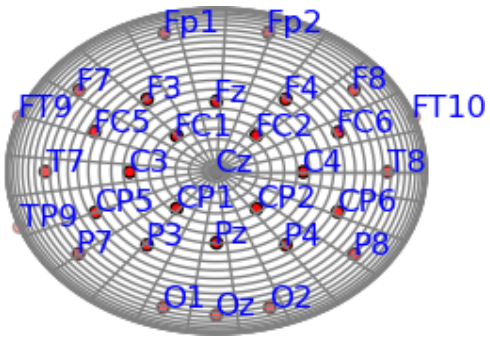
In order to separate electrodes into such groups, we must define some measure of the distance between the electrodes. An obvious candidate is the Euclidean distances between electrodes along the scalp. These distances vary between individuals, but, the international 10–20 system is designed so that the relative area covered by the electrode array is consistent between individuals. The Easy Cap recording cap [33] thus comes with a layout of relative distances between the electrodes projected onto two dimensional space. The problem with defining Euclidean distances between the electrodes in the two dimensional space is that when electrodes near the base of the skull, such as those on the mastoids TP9 and TP10 (see Figures 3.1 and 3.2), are projected onto the two dimensions, they appear much closer to the other electrodes than along the scalp distance would suggest, thus erroneously implying that they cover similar regions of the brain. Thus, along the scalp Euclidean distances would be more indicative of electrodes covering similar regions.

To quickly compute the along-the-scalp distances, the skull is assumed to be a hemisphere. The 2D schematic of electrode placements is assumed to be a vertical projection of the surface of the hemisphere onto the plane at the bottom of the hemisphere. From this assumption, the surface coordinates are reconstructed. The spherical projection of the electrode locations is then given by:

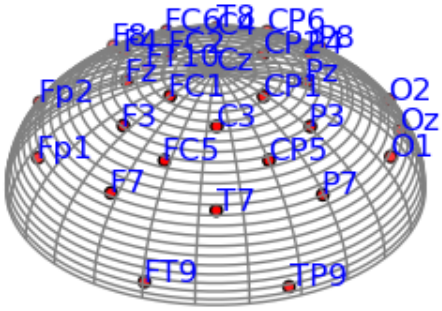
$$z = \sqrt{r^2 - x^2 - y^2} \tag{3.5}$$

from the equation for a sphere centered on the origin, with radius r and the relative locations x and y of the electrodes provided by the Easy Cap using an international 10–20 layout. Figures 3.7 and 3.8 present how the electrode layouts from each of the motor imagery experiments appear when projected onto a hemispherical skull.

The Euclidean distance along the scalp between any two electrodes is then sim-

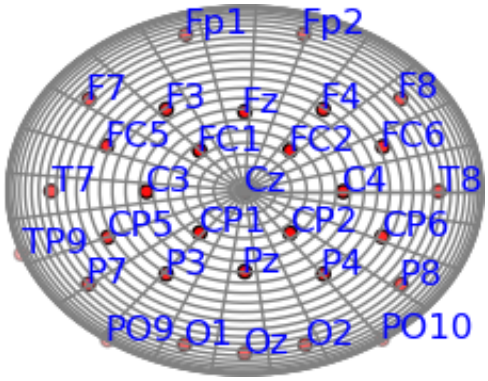


(a) Spherical projection viewed from above, front of head at top.

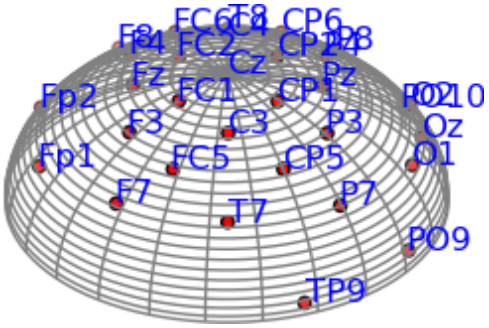


(b) Spherical projection viewed from side, front of head at left.

Figure 3.7: Spherical projection of electrodes used in left and right hand motor imagery experiment, from 2D layout in Figure 3.1.



(a) Spherical projection viewed from above, front of head at top.



(b) Spherical projection viewed from side, front of head at left.

Figure 3.8: Spherical projection of electrodes used in grey, green, yellow, and purple motor imagery experiment, from 2D layout in Figure 3.2.

ply the arc length between the two, given by:

$$s = r \arccos\left(\frac{\mathbf{v_1}^T \cdot \mathbf{v_2}}{\|\mathbf{v_1}\|\|\mathbf{v_2}\|}\right) \tag{3.6}$$

where $\mathbf{v}$ is the vector of Cartesian coordinates of an electrode, and $\|\mathbf{v}\|$ is the magnitude of the vector. Given a matrix of three dimensional Cartesian coordinates of electrode locations $\mathbf{L} \in \mathbb{R}^{C\times 3}$, a distance matrix $\mathbf{D} \in \mathbb{R}^{C\times C}$ containing the distance along the scalp between each electrode can be computed as:

$$\mathbf{D} = r \arccos\left(\frac{1}{r^2}\mathbf{L}\mathbf{L}^T\right) \in \mathbb{R}^{C\times C} \tag{3.7}$$

where $\arccos$ is applied element-wise. Since all electrode locations fall on the surface of the hemisphere centered at the origin, it holds that $\forall \mathbf{v} \in \mathbf{L}$, $\|\mathbf{v}\| = r$.

A K-Disjoint grouping of N electrodes is defined as:

$$\mathbf{G_K} = \left\{ \mathbf{g_0}, \mathbf{g_1}, \ldots, \mathbf{g}_{\lfloor \frac{\mathbf{N}}{\mathbf{K}} \rfloor - \mathbf{1}} \mid \forall \mathbf{g_i}, |\mathbf{g_i}| = K; \cap_{i=0}^{\lfloor \frac{N}{K} \rfloor - 1} \mathbf{g_i} = \emptyset \right\} \tag{3.8}$$

i.e. it is a set of $\lfloor \frac{N}{K} \rfloor$ groups, $\mathbf{g_i}$, of electrodes each of which contains K electrodes, and no electrode falls in more than one $\mathbf{g_i}$. Let $\mathbb{G}_\mathbf{K}$ be the collection of all possible K-Disjoint groupings for a set of N electrodes. With a distance matrix $\mathbf{D}$, one can then find the K-Disjoint Nearest Neighbours grouping, defined as:

$$\underline{\mathbf{G_K}} = \underset{\mathbf{G_K} \in \mathbb{G}_\mathbf{K}}{\arg\min} \sum_{\mathbf{g_i} \in \mathbf{G_K}} \mathrm{d}(\mathbf{g_i}) \tag{3.9}$$

where $\mathrm{d}(\cdot)$ is a function that uses $\mathbf{D}$ to calculate the sum of the distances between each electrode pair in a group of electrodes. This function is presented in Algorithm 2. Thus, the K-Disjoint Nearest Neighbours grouping $\underline{\mathbf{G_K}}$ is the K-Disjoint grouping for which the sum of the total distance within the electrode groups is minimised.

Algorithm 1 K Disjoint Nearest Neighbours

Let $\mathbf{C}$ = the set of possible combinations $\binom{N}{K}$ from N possible electrodes
procedure GETDISJOINTNEARESTNEIGHBOURS(K, $\mathbf{D}$, $\mathbf{C}$)
    if $|\{c_i | c_i \in \mathbf{c}, \mathbf{c} \in \mathbf{C}\}| < K$ **then**
        Return $\emptyset$, 0 ▷ If not enough electrodes to form group, exit algorithm
    end if
    $mD = \infty$ ▷ Initialize total min distance
    $G = \emptyset$ ▷ Initialize set of disjoint groups
    for each combination $\mathbf{c} \in \mathbf{C}$ **do**
        $\tilde{\mathbf{C}} = \mathbf{C} / \{\mathbf{x} | \mathbf{x} \in \mathbf{C} \text{ and } \mathbf{x} \cap \mathbf{c} \neq \emptyset\}$ ▷ Remove groups overlapping with $\mathbf{c}$
        $minDist = \infty$ ▷ Initialize min distance for subset
        $grouping = \emptyset$ ▷ Initialize set of groups for subset
        for each $\tilde{\mathbf{c}} \in \tilde{\mathbf{C}}$ **do** ▷ Optimize recursively over the remaining set
            $groups, distance$ = GETDISJOINTNEARESTNEIGHBOURS (K, $\mathbf{D}$, $\tilde{\mathbf{C}}$)
            if $distance < minDist$ **then**
                $minDist = distance$
                $grouping = groups$
            end if
        end for
        if $minDist+$ groupDist($\mathbf{c}$) $< mD$ **then** ▷ Check new total dist $<$ abs min
            $G = grouping \cup \mathbf{c}$ ▷ Add $\mathbf{c}$ to set of disjoint groups
            $mD = minDist+$ groupDist($\mathbf{c}$) ▷ Update total min dist
        end if
    end for
    Return G, mD ▷ Return disjoint groups, total min dist
end procedure

The main algorithm for finding these K-Disjoint Nearest Neighbour groupings is Algorithm 1. It is defined recursively, using the distance matrix $\mathbf{D}$, the size of each

Algorithm 2 Group Distance

```
1: procedure GROUPDIST(c, D)
2:     groupDistance = 0                          ▷ Initialize total dist in group
3:     if |c| = 0 then                            ▷ If c is empty, exit algorithm
4:         Return groupDistance
5:     end if
6:     for i ∈ {0, 1, ..., |c| − 1} do    ▷ Sum up dist between all pairs of electrodes
7:         for j ∈ {i + 1, i + 2, ..., |c|} do
8:             groupDistance = groupDistance + D_{i,j}
9:         end for
10:    end for
11:    Return groupDistance                       ▷ Return total dist in group
12: end procedure
```

group K, and the set of all possible K-sized combinations of electrodes $\mathbf{C}$ as inputs. K can in principle be whatever size the user chooses, smaller or equal to the total number of electrodes N, of course. For this project, time constraints meant that K was only chosen once, although in future work it would be good to consider how the choice of K affects the accuracy in estimating the full model. K was chosen to be 3 for two reasons. First, it seemed inadvisable to group together electrodes which may be reading activity in entirely different parts of the brain, which may happen if groups size is too large. Second, using groups of 3 in the first convolutional layer makes for a natural progression to the second convolutional layer, which would then convolve over pairs of groups. Additionally, since $N = 31$, using groups of 2 or 4 would result in dropping some data in either the first or second convolutional layer.

The algorithm works by trying to group different combinations of electrodes and keeping track of the set of groups and the total minimum distance achieved with a particular combination. The total minimum distance is the sum of pairwise distances between electrodes in each group, with the resulting distances in each group summed together across groups. For each possible K-sized electrode combination, the algorithm separates the group from the remaining electrodes and recursively reruns the algorithm on the now smaller set. As improvements are made on the minimum total distance measure, the groups are added to the set of disjoint groups, which is outputted once the algorithm finishes running together with the measure of total minimum distance achieved. Algorithm 2 is a supporting function called by Algorithm 1. It finds the pairwise distances between each distinct pair of electrodes in a group and sums these distances together to return the so-called group distance.

Unfortunately, the method for calculating $\underline{\mathbf{G_K}}$ requires a massive search space, which scales very poorly with the number of electrodes – on the order of O(n^n). To avoid this difficulty, the electrodes are initially separated into three general bins: front of head electrodes, top of head electrodes, and the rest. These bins are presented in Table 3.3. A modified version of Algorithm 1, which improves efficiency by keeping track of the absolute minimum distance and the distance for a given grouping so far, is then applied to each bin and the results appended to form $\underline{\mathbf{G_K}}$.

The resulting $\underline{\mathbf{G_3}}$, the 3-Disjoint Nearest Neighbours grouping, for the left and

(a) Initial general bins for left and right hand motor imagery experiment layout.

Grouping	Electrodes
Front	Fp1, Fp2, F7, F3, Fz, F4, F8, FC1, FC2
Top	C3, CP5, CP1, Pz, P3, P7, P4, P8, CP6, CP2, Cz, C4
Rest	FC5, FC6, FT10, FT9, O1, O2, ,Oz, T7, T8, TP9

(b) Initial general bins for grey, green, yellow, and purple motor imagery experiment layout

Grouping	Electrodes
Front	Fp1, Fp2, F7, F3, Fz, F4, F8, FC5, FC1, FC2, FC6, Cz
Top	T7, C3, C4, T8, CP5, CP1, CP2, CP6, Pz
Rest	O1, O2, Oz, P3, P4, P7, P8, PO10, PO9, TP9

Table 3.3: General initial groupings of electrodes to find the K-Disjoint Nearest Neighbours grouping.

right hand motor imagery experiment is presented in Figure 3.9(a) and the resulting $\underline{\mathbf{G_3}}$ for the grey, green, yellow, and purple motor imagery experiment is presented in Figure 3.10(a). Note that for each layout, TP9, the electrode on the left mastoid, is left without a group. For simplicity, the TP9 channel is dropped from all subsequent analysis. These are the electrode groupings that are used for all subsequent analysis, though future work could analyse the robustness of the results to choosing different K values.

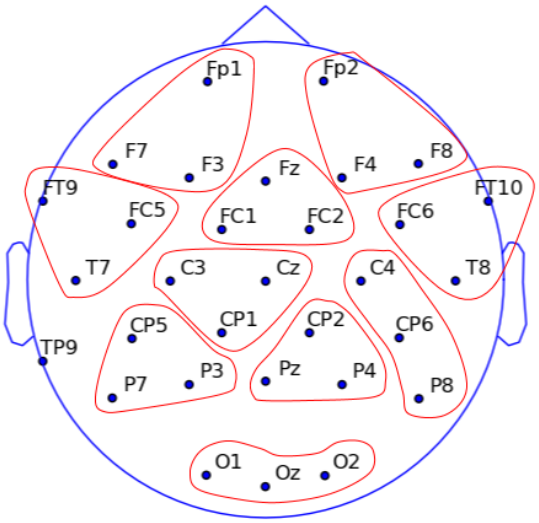
3.4 Verification of Groups Using Correlations

Since the intention of the K-Disjoint Nearest Neighbours grouping using Euclidean distances along the scalp is to group electrodes which contain similar statistical information from being above similar brain structures, an easy way to verify that this grouping is accomplishing this is to check the correlations between the electrodes. In each experiment, each subject has the preprocessed data matrix $V \in \mathbb{R}^{n \times C \times T}$, where n is the number of trails, C is the number of electrode channels, and T is the number of voltage readings. For a given experiment, subject, trial r, and channel c, $\mathbf{e^c} = \mathbf{V_{r,c}} \in \mathbb{R}^{1 \times T}$, is a time series of voltage recordings for the electrode channel c. The correlation between two electrode channels in a given trial is thus:

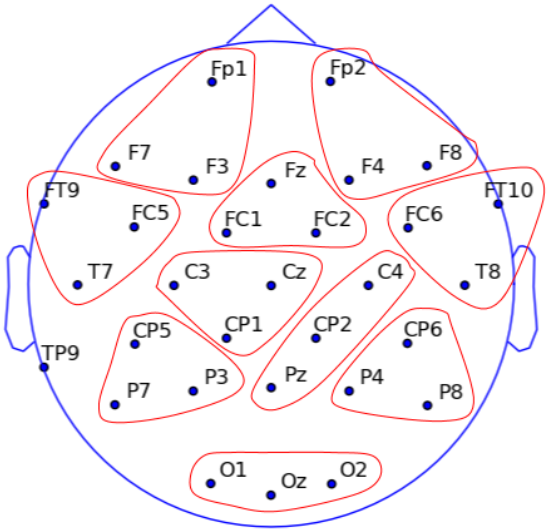
$$\rho_{i,j} = \frac{\text{cov}(\mathbf{e^i}, \mathbf{e^j})}{\sigma_{\mathbf{e^i}} \sigma_{\mathbf{e^j}}} = \frac{\sum_{k=1}^{T} (\mathbf{e^i}_k - \bar{\mathbf{e^i}})(\mathbf{e^j}_k - \bar{\mathbf{e^j}})}{T^2 \sigma_{\mathbf{e^i}} \sigma_{\mathbf{e^j}}} \tag{3.10}$$

where $\bar{\mathbf{e^j}}$ is the mean and $\sigma_{\mathbf{e^j}}$ is the standard deviation of $\mathbf{e^j}$. For subject s and trial r, the correlation matrix is:

$$\rho^{s,r} = [\rho_{i,j}] \in \mathbb{R}^{C \times C}, i, j \in \{1, 2, \ldots, C\} \tag{3.11}$$



(a) Disjoint groups produced by Euclidean distance along scalp assuming a hemispherical skull.

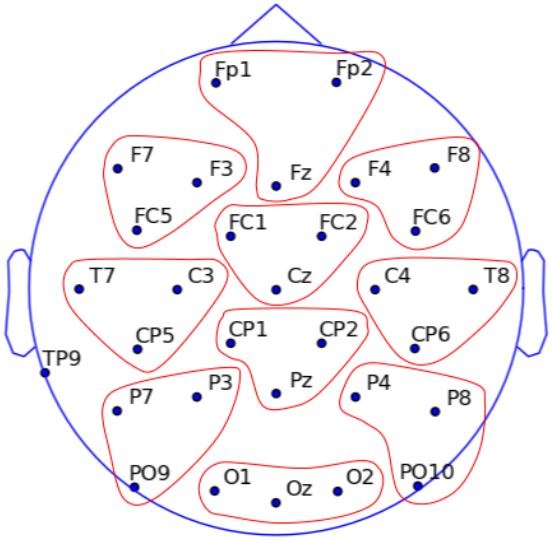


(b) Disjoint groups produced by correlation distance, calculated as the average correlation between electrodes over all trials and subjects.

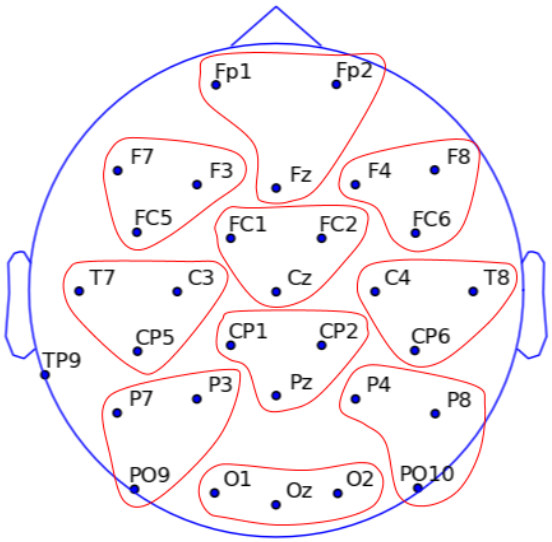
Figure 3.9: Disjoint nearest neighbour groups identified for the left and right hand motor imagery experiment, using both euclidean distance along the scalp and correlations.

For each experiment, calculate the average correlation matrix across all subjects and trials:

$$\bar{\rho} = \frac{1}{n + |S|} \sum_{s \in S} \sum_{r=1}^{n} \rho^{s,r} \tag{3.12}$$



(a) Disjoint groups produced by Euclidean distance along scalp assuming a hemispherical skull.



(b) Disjoint groups produced by correlation distance, calculated as the average correlation between electrodes over all trials and subjects.

Figure 3.10: Disjoint nearest neighbour groups identified for the grey, green, yellow, and purple motor imagery experiment, using both euclidean distance along the scalp and correlations.

The average correlation matrix gives an indication of how related is the information from each electrode channel, and one would expect the correlation between electrodes above the same regions of the brain to have high positive correlation, while those above different regions may have low or negative correlation.

Tables 3.4 and 3.5 present the top five most correlated electrode channels for

Electrode	Top 5 Highest Correlation				
Fp1	Fp2	F3	Fz	F7	F4
Fz	FC1	FC2	F4	F3	Cz
F3	Fz	FC1	FC5	Fp1	FC2
F7	FT9	FC5	Fp1	F3	Fz
FT9	F7	T7	FC5	TP9	CP5
FC5	F3	C3	FC1	F7	CP5
FC1	Fz	FC2	Cz	F3	C3
C3	CP1	FC1	Cz	CP5	F3
T7	FT9	CP5	FC5	F7	C3
TP9	FT9	CP5	T7	P7	FC5
CP5	P3	C3	CP1	FC5	T7
CP1	Cz	C3	Pz	CP2	P3
Pz	CP2	CP1	P3	P4	Cz
P3	Pz	CP1	CP5	C3	CP2
P7	O1	P3	TP9	CP5	Oz
O1	Oz	P3	O2	Pz	P7
Oz	O1	O2	P3	Pz	P4
O2	Oz	P4	O1	P8	Pz
P4	Pz	CP2	CP6	O2	C4
P8	O2	P4	Oz	CP6	Pz
CP6	P4	C4	CP2	Pz	Cz
CP2	Pz	Cz	C4	CP1	P4
Cz	FC2	FC1	CP1	CP2	C4
C4	CP2	FC2	Cz	CP6	F4
T8	FC6	CP6	C4	F8	FT10
FT10	F8	FC6	Fp2	F4	T8
FC6	F4	F8	C4	FC2	Fz
FC2	Fz	FC1	Cz	F4	C4
F4	Fz	FC2	FC1	FC6	F3
F8	FC6	Fp2	FT10	F4	Fz
Fp2	Fp1	F4	Fz	F8	F3

Table 3.4: Electrodes used in the left and right hand motor imagery experiments by [7] with the five electrodes with the highest average correlation coefficient across all trials and subjects, ordered from left to right in descending correlation. For each electrode, the highlighted electrodes are the group members determined by the disjoint groups of three algorithm using euclidean distance between electrodes across the scalp.

each electrode channel in the left and right hand motor imagery and the grey, green, yellow, and purple motor imagery experiments, respectively. The top five for each electrode are found from $\bar{\rho}$. The entries highlighted in blue are those electrodes the electrode is grouped with by the 3-Disjoint Nearest Neighbours algorithm using euclidean distance along the scalp as described in Section 3.3. From visual inspection, it is possible to see that the algorithm has done well at grouping electrodes which are highly correlated, with nearly all electrodes paired with two electrodes in the

Electrode	Top 5 Highest Correlation				
Fp1	Fp2	F3	F7	Fz	F4
Fp2	Fp1	F4	Fz	F8	F3
F7	FC5	F3	Fp1	C3	FC1
F3	Fz	FC1	Fp1	FC5	F7
Fz	FC2	F3	F4	FC1	Fp2
F4	Fz	FC2	Fp2	FC6	F8
F8	FC6	Fp2	F4	Fp1	Fz
FC5	F7	C3	F3	FC1	CP5
FC1	Cz	F3	Fz	C3	FC2
FC2	Fz	F4	Cz	FC1	C4
FC6	F4	C4	F8	FC2	Fp2
T7	CP5	P7	TP9	PO9	C3
C3	CP1	CP5	FC5	FC1	Cz
Cz	FC1	CP1	CP2	FC2	C3
C4	CP2	FC2	FC6	CP6	Cz
T8	CP6	P8	P4	CP2	FC2
TP9	PO9	P7	CP5	O1	F7
CP5	C3	P3	CP1	P7	FC5
CP1	Pz	Cz	C3	CP2	P3
CP2	Cz	CP1	Pz	P4	C4
CP6	P4	C4	CP2	P8	Cz
P7	PO9	CP5	P3	O1	Oz
P3	CP1	Pz	CP5	O1	Oz
Pz	CP1	CP2	P3	P4	Oz
P4	CP2	CP6	Pz	O2	P8
P8	P4	CP6	O2	PO10	CP2
PO9	O1	P7	Oz	PO10	P3
O1	Oz	PO9	P3	O2	P7
Oz	O1	O2	P3	PO9	Pz
O2	Oz	O1	PO10	P4	Pz
PO10	O2	Oz	PO9	O1	P8

Table 3.5: Electrodes used in the grey, green, yellow, and purple motor imagery experiments by [29] with the five electrodes with the highest average correlation coefficient across all trials and subjects, ordered from left to right in descending correlation. For each electrode, the highlighted electrodes are the group members determined by the disjoint groups of three algorithm using euclidean distance between electrodes across the scalp.

top five. Note that TP9 is the only electrode which does not appear in the top five of at least two other electrodes, lending more confidence to the algorithm leaving it as the odd one out.

As one final test, the 3-Disjoint Nearest Neighbours algorithm was rerun on the electrode layouts for each experiment using the respective $\mathbf{1} - \bar{\rho}$ as the distance matrix. The resulting groupings are presented in Figures 3.9(b) and 3.10(b) above. In

the left and right hand motor imagery experiment, the grouping is almost identical to the scalp distance layout, with only C4 and P4 flipping between groups. The groupings are identical for the grey, green, yellow, and purple motor imagery experiment. This gives more confidence that the groupings using the scalp distances are indeed grouping electrodes which correspond to similar regions of the brain. For the rest of the presented analysis, the 3-Disjoint Nearest Neighbours grouping using euclidean distance along the scalp is used.

3.5 Second Layer Convolutions

A second convolutional layer may be desirable to identify higher-level features than a single layer can extract. For this reason, the data are also arranged for the possibility of a second convolutional layer. This is done in the same manner as in Section 3.3. First, take the grouping found by the 3-Disjoint Nearest Neighbours algorithm. Then, find the centroid of each group on the scalp, where $(\bar{x}, \bar{y}, \bar{z})$ are the average coordinates of each electrode's coordinates in the group, defined as:

$$x_c = r\cos\left((\arctan\left(\frac{\bar{y}}{\bar{x}}\right)\right)\sin\left(\arccos\left(\frac{\bar{z}}{\sqrt{\bar{x}^2+\bar{y}^2+\bar{z}^2}}\right)\right) \tag{3.13}$$

$$y_c = r\sin\left((\arctan\left(\frac{\bar{y}}{\bar{x}}\right)\right)\sin\left(\arccos\left(\frac{\bar{z}}{\sqrt{\bar{x}^2+\bar{y}^2+\bar{z}^2}}\right)\right) \tag{3.14}$$

$$z_c = r\cos\left(\arccos\left(\frac{\bar{z}}{\sqrt{\bar{x}^2+\bar{y}^2+\bar{z}^2}}\right)\right) \tag{3.15}$$

These are the coordinates found by taking the average of the group members and projecting the mean vector onto the surface of the sphere. For each centroid, use (x_c, y_c, z_c) to find the along scalp distances between the centroids. Finally, use Algorithm 1 to find the 2 Disjoint Nearest Neighbours grouping of the centroids. The centroid groupings are presented in Figure 3.11.

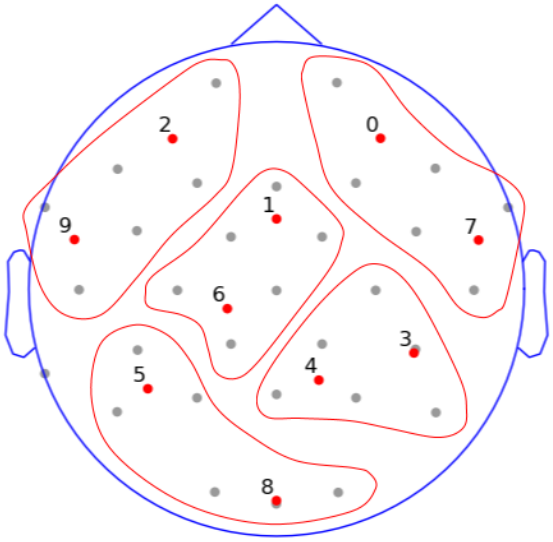
Finally, the matrix of examples $X \in \mathbb{R}^{N\times C\times\tau}$ has the electrode channel axis reordered by centroid group, by group, and by electrode by relative position from the back to the front of the head. This produces a final ordering of the channel axis for the left and right hand motor imagery layout of:

> O1, O2, Oz, CP5, P3, P7, C4, CP2, Pz, CP6, P4, P8, FT9, FC5, T7, Fp1, F7, F3, C3, Cz, CP1, Fz, FC2, FC1, FT10, FC6, T8, Fp2, F8, F4

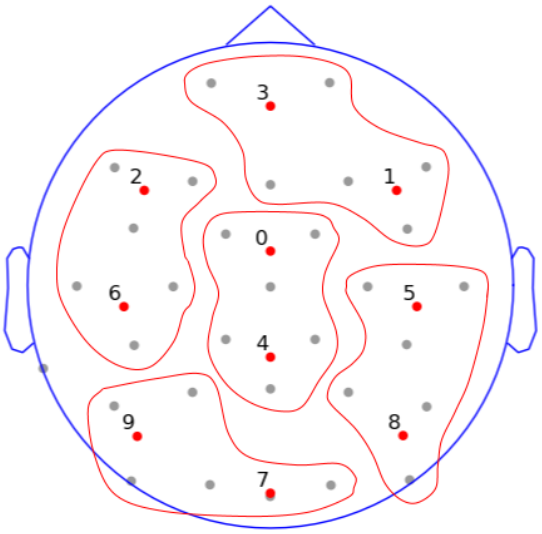
and a final ordering for the grey, green, yellow, and purple motor imagery layout of:

> P3, P7, PO9, O1, O2, Oz, P4, P8, PO10, T8, C4, CP6, CP2, CP1, Pz, FC2, FC1, Cz, T7, C3, CP5, F7, F3, FC5, Fp2, Fp1, Fz, F8, F4, FC6

Since both drop the TP9 channel, each layout reduces the electrode channel dimension by one. This is then the final example matrix $X \in \mathbb{R}^{N\times 30\times\tau}$, which is used to generate the results.



(a) Left and Right Hand Motor Imagery Experiment.



(b) Grey, Green, Yellow, and Purple Motor Imagery Experiment.

Figure 3.11: Disjoint centroid groups produced by Euclidean distance along scalp assuming a hemispherical skull.

3.6 Convolutional Neural Network Structure

In most of the results discussed in Chapter 4, the model of choice is the convolutional neural network. Specifically, a similar structure is used for each test: one convolutional layer, followed by two fully interconnected layers and a logistic regression for classification. As described in Section 2.3, the convolutional layer is defined to have

20 3×5 filters. This means it applies the filters across three electrodes and five EEG observations for each. The convolutions are not padded. This is followed by a max-pooling with a vector of 2 across the time axis. No max-pooling occurs across electrodes. For example, using a time slice $\tau = 100$, an input example is $\mathbf{x} \in \mathbb{R}^{30\times100}$. A given filter in the convolutional layer, therefore, reshapes $\mathbf{x}$ to $\mathbf{x} \in \mathbb{R}^{28\times48}$, where the time dimension is given by $(100-5+1)/2 = 48$ and the spatial dimension is given by $30 - 3 + 1 = 28$. An illustration of how the input passes through a single filter of the convolutional layer can be found in Figure 3.12. Note that this causes some members of the output to be linear combinations of electrodes in different groups, as defined above, which is not desirable, but is used because of time constraints. A better solution is to implement a stride of three as well for the convolutions, but this significantly increases the computation time of the Theano implementation used and so is dropped in the results. Using a stride of three is also preferable as it reduces the spatial dimension to 10.

The convolutional layer is followed by a fully connected layer of 500 nodes, which is followed by a fully connected layer of 50 nodes. Finally, the 50 output units from the second fully connected layer are used in a logistic regression to classify the example. The number of output nodes depends on the classification task, ranging from two to four classes. Each layer uses $\tanh(\cdot)$ activation functions:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{3.16}$$

The loss function used for all the results is defined as the standard cross entropy loss function as discussed in Section 2.3.1. The structure of the convolutional neural network is depicted in Figure 3.13.

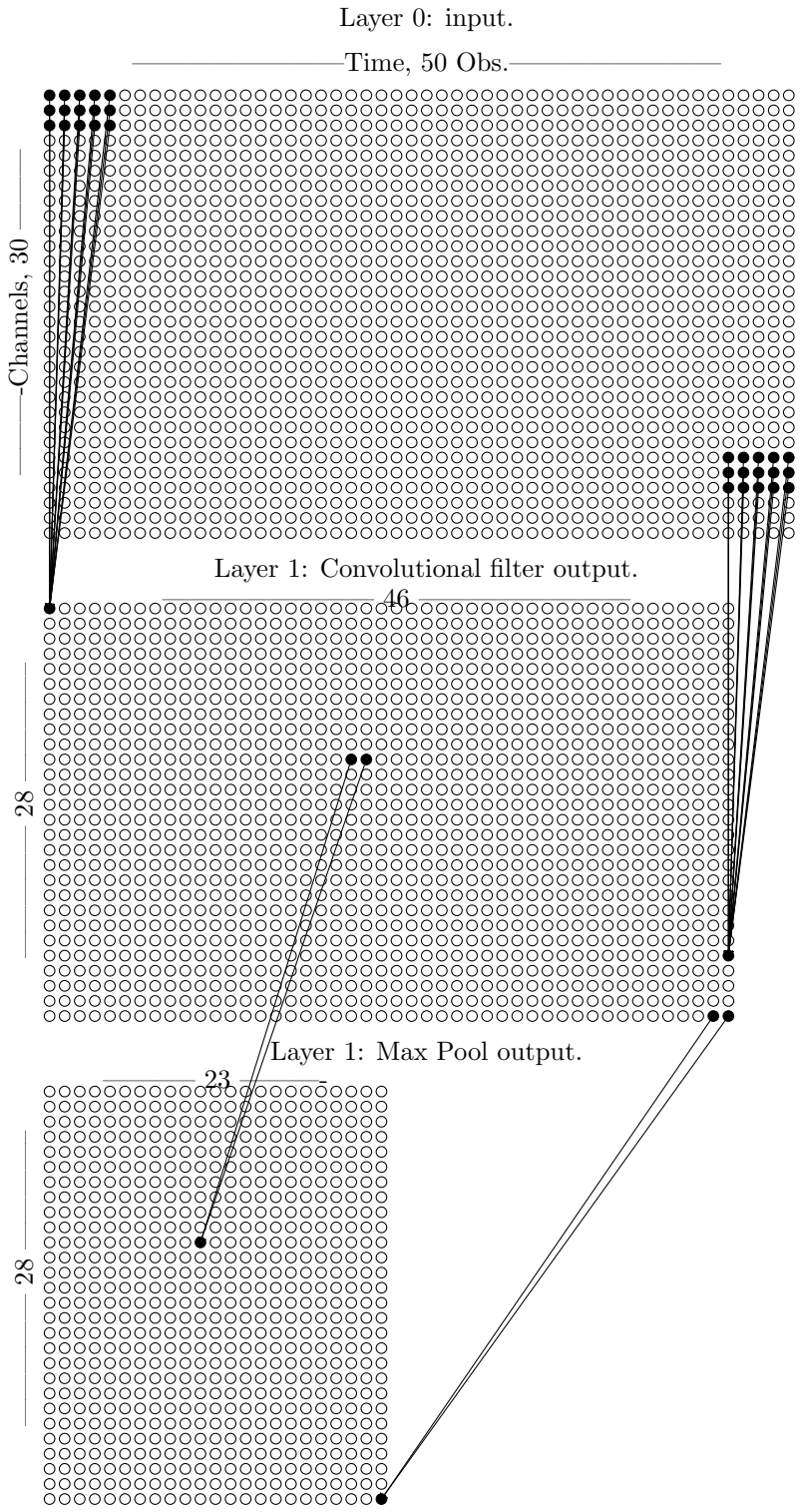


Figure 3.12: Example of how input with $\tau = 50$ moves through one filter of the convolutional layer.

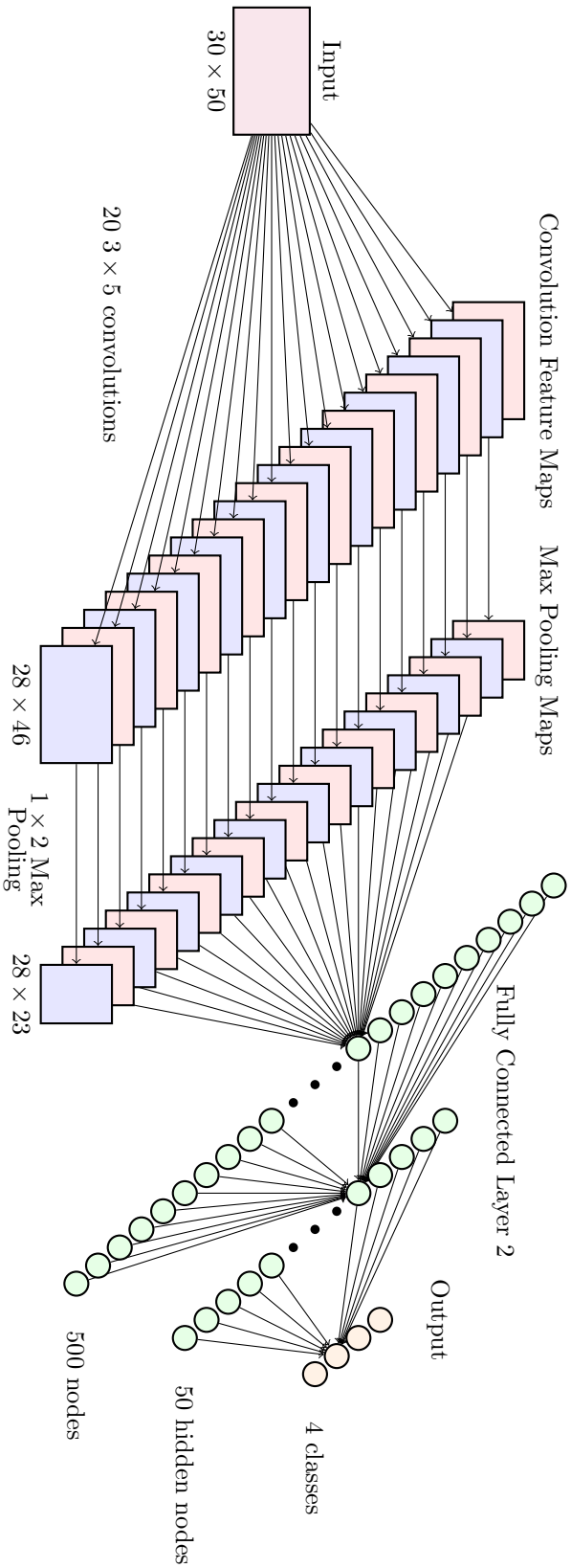


Figure 3.13: The structure of the convolutional neural network used with $\tau = 50$ and four motor imagery classes for example.

Chapter 4

Experimental Results

The results discussed in this chapter are produced by application of a convolutional neural network, the structure of which is presented in Section 3.6. In each experiment, the set of labelled examples is split into training, validation, and testing sets, which respectively account for 60%, 20%, and 20% of the labelled examples. At the beginning of each experiment, these sets are formed by randomly selecting the examples for membership into the groups without replacement within each label type. This ensures that there are about an equal proportion of each type of motor imagery task in each of the sets. Figure 4.1 shows how the data set is broken down for the

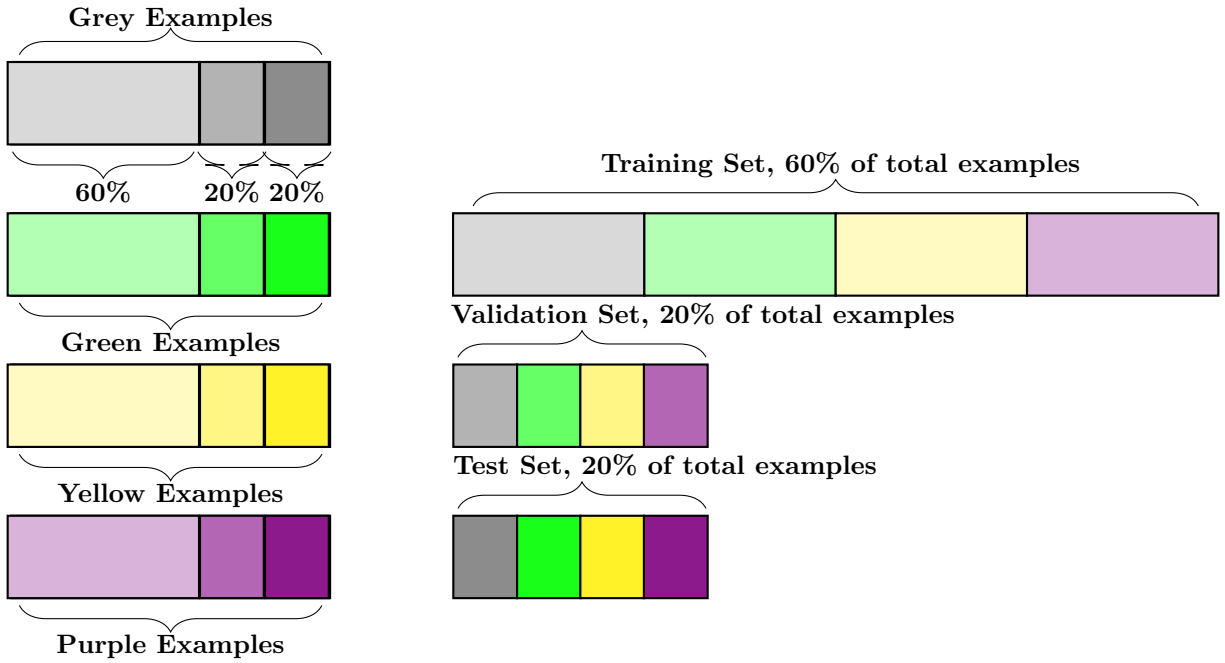


Figure 4.1: Training schedule for the Grey, Green, Yellow, and Purple motor imagery models: every run of the model, a random 60% of each imagery task's examples is taken, these are appended to form the training set. Of the remaining 40% of examples, a random 50% of each imagery task's examples is taken, these are appended to form the validation set. The remaining 20% forms the test set. This process ensures an equal proportion of classes among the three sets.

Table 4.1: Number of obs. for each subject — L/R Morlet-transformed data

	$\tau = 50$			$\tau = 100$		
Subject	Total	Left	Right	Total	Left	Right
MM	15616	7808	7808	15216	7608	7608
KP	54656	29280	25376	53256	28530	24726
AS	56608	27328	29280	55158	26628	28530
BM	52704	25376	27328	51354	24726	26628
AM	54656	25376	29280	53256	24726	28530
MX	50752	25376	25376	49452	24726	24726
GZ	80032	42944	37088	77982	41844	36138
AF	99552	50752	48800	97002	49452	47550

four colour motor imagery task. All experiments are carried out for a single subject at a time. The results are mostly generated with an input of $\tau = 50$, though some results use $\tau = 100$ or $\tau = 200$. Time constraints meant that each experiment could only be carried out once.

4.1 Left and Right Hand Motor Imagery

The first set of experiments were carried out on the data provided by [7]. This included eight subjects each of whom completed a set of motor imagery tasks in which they imagined opening and closing either the right or left hand. For each experiment, a batch size of 500 and a learning rate of $\eta = 0.01$ is used.

4.1.1 Morlet Wavelet Transformed

The first results generated use the Morlet Wavelet transformed voltage data as described in the previous chapter. They are generated based on the two-class learning task, with the subject imagining either left or right, as previously described. The number of examples for each subject can be seen in Table 4.1. Table 4.2 presents the results, which were generated using time slices $\tau = 50$, $\tau = 100$, and $\tau = 200$ with 8 seconds for each trial. The first column presents the subject (a total of eight individuals). The second column shows the number of epochs through which the model has gone by the time it reaches its minimum. The third and fourth columns show the accuracy in the validation and test sets, respectively, measured in percentage terms. This is the number predicted correctly out of the total in the set. As can be seen in the table, the error is under 1% for all entries.

Due to time constraints, it was not possible to carry out all of the experiments one would like to see. Thus the results using $\tau = 100$ and $\tau = 200$ were generated only for some of the subjects. These results are in the bottom part of the table. Once again, we see the error rate is very low. The model is able to quite accurately decipher whether each subject was indicating right or left and to predict this within the test set.

Table 4.2: Accuracy rate in converged model

Subject	Epoch	Validation	Test
$\tau = \mathbf{50}$			
AF	5000	99.303	99.292
AM	3883	100.000	100.000
AS	1865	99.360	99.991
BM	1571	99.990	99.990
GZ	1403	100.000	100.000
KP	2668	100.000	99.981
MM	1883	100.000	99.967
MX	1415	99.980	99.990
$\tau = \mathbf{100}$			
AF	4269	100.000	99.984
AS	1691	99.991	99.982
GZ	981	99.961	99.994
KP	1916	99.990	99.981
$\tau = \mathbf{200}$			
AM	1836	99.980	99.990
AS	1050	100.000	99.980

Increasing the time slice size brings down the number of epochs necessary to reach the minimum, additionally improving accuracy. There is a trade-off, however, if we think of using the model in real time, as larger time slices require more computations and are slower at responding to user intent. With a smaller time slice, there is less of a delay, since it is more sensitive to fluctuations in user motor imagery. At 500 Hz, the length of the time slice is either 0.10 seconds for 50 observations or 0.20 seconds for 100 observations. At 200 Hz, this translates to 0.25 or 0.50 seconds, respectively.

These results are also presented in Figures 4.2 through 4.9. The axes are all kept at the same scale for ease of comparison across subjects. These figures show the model's behavior as it learns. The initial guess has around 50% error, as one would expect with 2-class imagery, and gradually the error decreases towards 0 in both the test and validation sets for each subject. These figures show more clearly that feeding more data to the model through larger time slices ($\tau = 100$ or $\tau = 200$ as compared to $\tau = 50$) leads to quicker, more accurate convergence.

4.1.2 Voltage

Table 4.3 and Figure 4.10 show the results for fitting the model to left and right hand imagery voltage data. Recall that in this case, there are three classes: no thought, left, or right. The number of examples for each subject is presented in Table 4.4. As before, we see that the model learns and classifies user intent with very low error for subjects AM and AS, but it struggles to converge for subject BM.

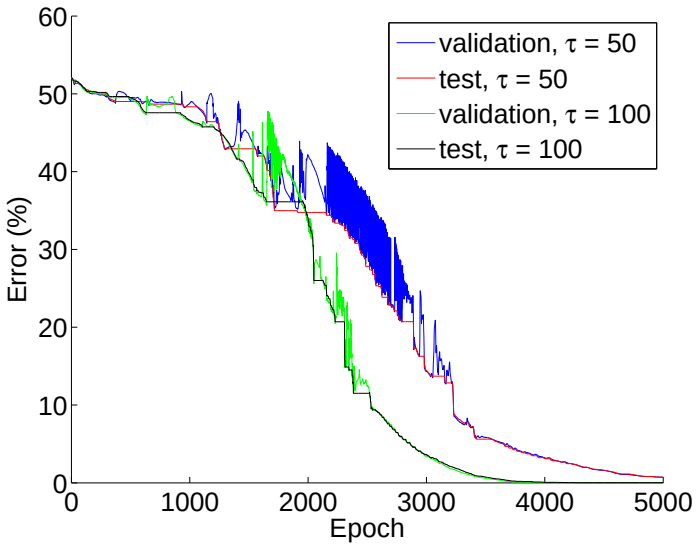


Figure 4.2: Validation and test set error for subject AF, for $\tau = 50$ and $\tau = 100$. The test errors are the red and black lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

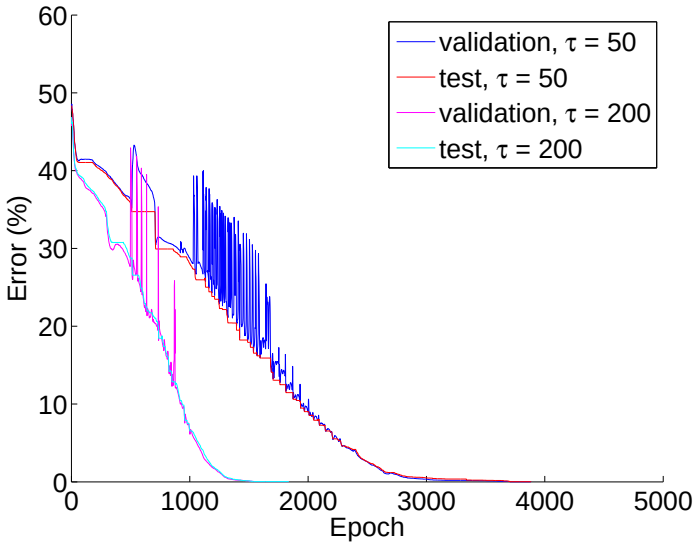


Figure 4.3: Validation and test set error for subject AM, for $\tau = 50$ and $\tau = 200$. The test errors are the red and cyan lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

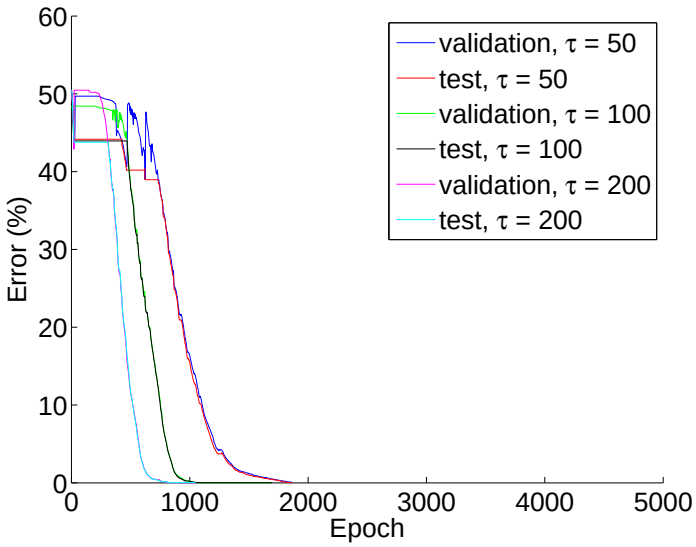


Figure 4.4: Validation and test set error for subject AS, for $\tau = 50$, $\tau = 100$, and $\tau = 200$. The test errors are the red, black, and cyan lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

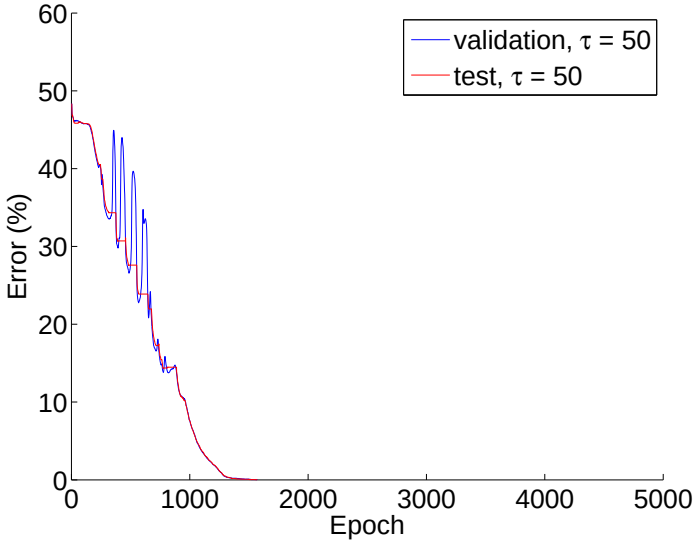


Figure 4.5: Validation and test set error for subject BM, for $\tau = 50$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

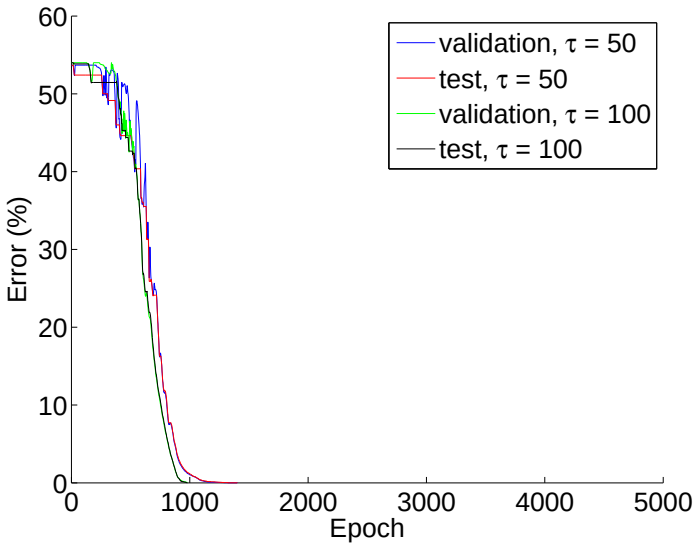


Figure 4.6: Validation and test set error for subject GZ, for $\tau = 50$ and $\tau = 100$. The test errors are the red and black lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

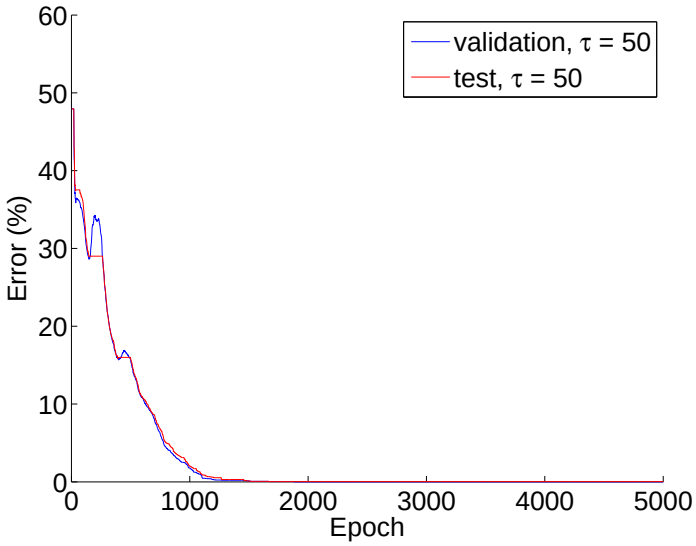


Figure 4.7: Validation and test set error for subject MM, for $\tau = 50$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

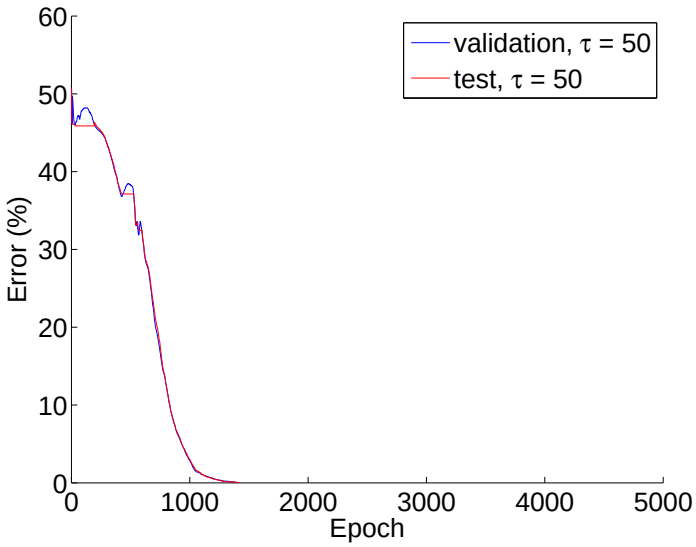


Figure 4.8: Validation and test set error for subject MX, for $\tau = 50$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

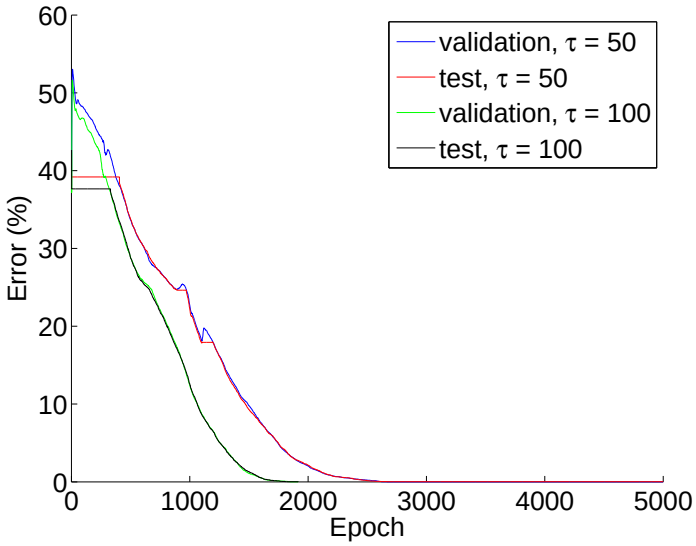


Figure 4.9: Validation and test set error for subject KP, for $\tau = 50$ and $\tau = 100$. The test errors are the red and black lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

Table 4.3: Accuracy rate at minimum validation error, L/R imagery, raw data

Subject	Epoch	Validation	Test
AM	4994	99.096	99.023
AS	4266	99.027	98.987
BM	2214	45.400	52.361

The table shows the epoch and accuracy rates in the validation and test sets for each subject at the point where the validation set accuracy is maximized. The model is given a maximum of 5000 training epochs with $\tau = 50$.

Table 4.4: Number of examples for each subject — L/R voltage data

	$\tau = 50$				$\tau = 100$			
Subject	Total	No Thought	Left	Right	Total	No Thought	Left	Right
MM	31608	7608	12000	12000	31208	7208	12000	12000
KP	110628	26628	45000	39000	109228	25228	45000	39000
AS	114579	27579	42000	45000	113129	26129	42000	45000
BM	106677	25677	39000	42000	105327	24327	39000	42000
AM	110628	26628	39000	45000	109228	25228	39000	45000
MX	102726	24726	39000	39000	101426	23426	39000	39000
GZ	161991	38991	66000	57000	159941	36941	66000	57000
AF	201501	48501	78000	75000	198951	45951	78000	75000

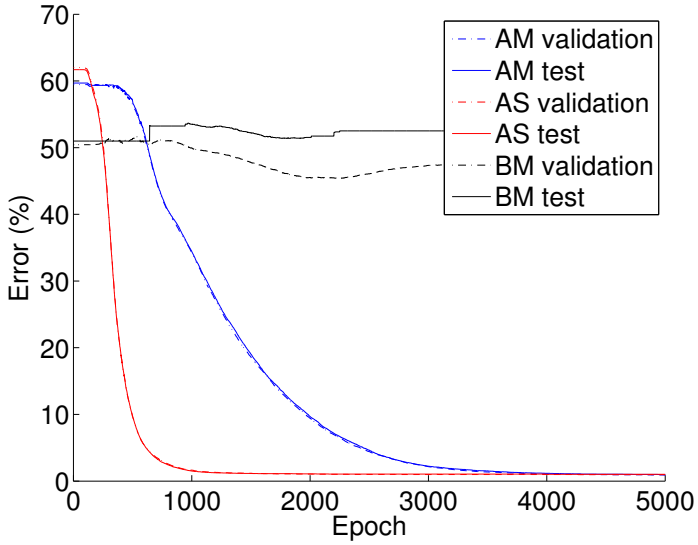


Figure 4.10: Validation and test set error for three subjects for $\tau = 50$. The test set errors are the solid lines and the validation set errors are the dot-dashed lines. The test error is only calculated when a new minimum of the validation error is reached.

These results and those in the previous section seem intuitive. Increasing the time slice increases the amount of information available to the model, thus improving convergence time and accuracy. Similarly, it is easier for the model to fit the Morlet transformed data than it is to do so with the noisier voltage data.

4.2 Grey, Green, Yellow, and Purple Motor Imagery

The second set of experiments were carried out on the data provided by [29]. This included two subjects each of whom completed a set of motor imagery tasks in which they imagined opening and closing either the right or left hand, contracting the abdominal muscles, or pushing down with both feet. For each experiment, a batch size of 500 and a learning rate $\eta = 0.001$ is used with 6 seconds of data per trial.

Table 4.5: Number of examples for each subject — 4-class raw data

			$\tau = 50$			
Subject	Total	No Thought	Grey	Green	Yellow	Purple
EG	150447	34047	36000	33600	25200	21600
LG	504075	114075	106800	111600	92400	79200
			$\tau = 100$			
EG	145597	29197	36000	33600	25200	21600
LG	487825	97825	106800	111600	92400	79200

Table 4.6: Number of examples for each subject — 4-class Morlet-transformed data

			$\tau = 50$		
Subject	Total	Grey	Green	Yellow	Purple
EG	72944	22560	21056	15792	13536
LG	244400	66928	69936	57904	49632
			$\tau = 100$		
EG	68094	21060	19656	14742	12636
LG	228150	62478	65286	54054	46332

Table 4.7: Accuracy rate in minimal model

Subject	Epoch	Validation	Test
$\tau = \mathbf{50}$			
EG	5000	65.750	64.467
LG	5000	42.236	42.518
$\tau = \mathbf{100}$			
EG	600	100.000	99.867
LG	1691	100.000	99.939

4.2.1 Morlet Wavelet Transformed

Data is available on two subjects for the experiment with four different classes of imagery. The number of examples for each individual is presented in Table 4.6. Experiments were conducted both with time slice size $\tau = 50$ and $\tau = 100$, and are presented in Table 4.7. In the first case, the model was set to run for 5000 epochs, but as can be seen in the table, it appears that was not enough to reach convergence for either subject. Running the model with $\tau = 100$ produces better results — the model is able to converge much quicker, again attaining a similar level of accuracy as previously observed in subjects with two types of imagery.

The results are again presented additionally as figures, Figures 4.11 and 4.12. These make it easier to see the behavior of the model as it learns. With $\tau = 50$ one can see that there is some learning as the error rate diminishes in both the test and validation set for each subject, but 5000 epochs prove insufficient to achieve convergence. Meanwhile, the error rate goes to zero much quicker when the model is given a larger time slice ($\tau = 100$), and the model achieves nearly perfect accuracy.

4.3 Revised Results

Clearly, the previous results would be unprecedented in the literature and form a very strong basis for an EEG-based BCI. Unfortunately, although the results in the previous section were very impressive, further tests of their robustness revealed that there may be a significant amount of over-fitting with the previous methods. The original intent was to test the ability of the learned models to classify user intent from

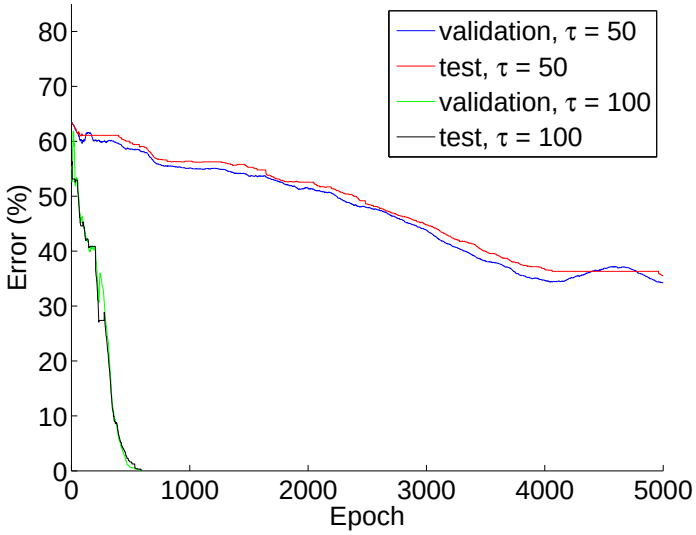


Figure 4.11: Validation and test set error for subject EG (4-class imagery), for $\tau = 50$ and $\tau = 100$. The test errors are the red and black lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

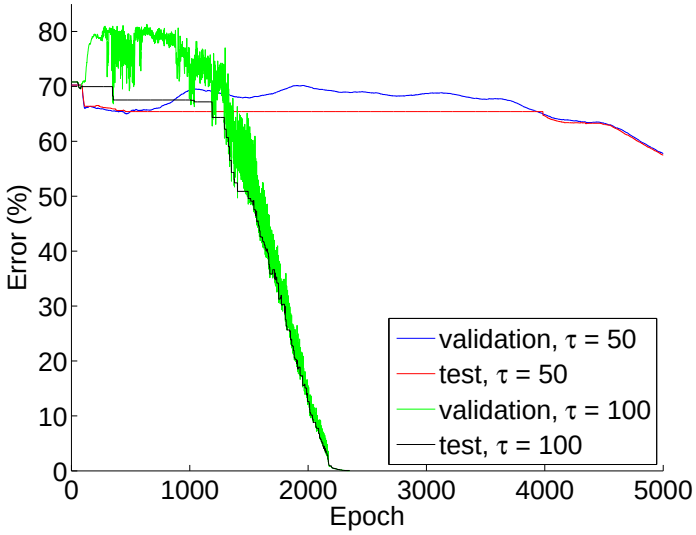


Figure 4.12: Validation and test set error for subject LG (4-class imagery), for $\tau = 50$ and $\tau = 100$. The test errors are the red and black lines, respectively. The test error is only calculated when a new minimum of the validation error is reached.

Left and Right Hand		
Subject	Hold Out Test Set Accuracy	Accuracy on Hold Out Trial Examples
AF	100.00	52.83
AM	99.96	53.51
AS	99.48	49.23
BM	99.85	51.89
GZ	100.00	50.82
KP	100.00	52.87
MM	100.00	46.36
MX	99.95	51.92
Four Type Motor Imagery		
Subject	Hold Out Test Set Accuracy	Accuracy on Hold Out Trial Examples
LG	99.95	27.11
EG	99.83	20.37

Table 4.8: Results for the hold one trial out test for each individual in each experiment. Presented results are only for the Morlet transformed data, meaning the experiments represent the two and four class classifications respectively, all only use $\tau = 100$.

EEG signals during a real-time experiment with a subject playing the BrainRunners video game. Time constraints meant that this was not possible. However, a different test was devised that could approximate turning off the BCI and turning it back on, and checking that the model could still accurately predict user intent.

This test was performed by holding out an entire trial and training the model for each individual as in the previous manner, with the remaining trials chunked by the time slice $\tau = 100$ and divided into the training, validation, and test sets. Only the Morlet transformed data were used. The held out trial was then chunked in the same manner and the trained model was used to predict the held out trial's examples. In both the left and right and in the four-class motor imagery data this produced less than stellar results with no trained model for any individual performing significantly better than a model that randomly guessed the class. Note that in each case there is only one class the model needs to guess. The results are presented in Table 4.8.

These results are very disappointing and a strong indication of over-fitting being a major problem in the process described in previous sections. On second consideration, the culprit is most likely the implicit assumption that the states of the brain will be independent after a given amount of time has passed. In this way, examples drawn from the same trial would not contain sufficiently similar information apart from the motor imagery signal, which would allow them to be put into the training, test, and validations sets without concerns. At first glance, this may not be all that unreasonable of an assumption, since EEG signals are notoriously noisy and non-stationary, making prediction of the next time slice of EEG readings very difficult. Unfortunately, it seems that there do exist some identifying features which allow the convolutional networks to distinguish which trial an example comes from and then to predict the label of the trial instead of learning any generalisable features of the motor imagery task.

To reduce the potential for over-fitting, it is necessary to redesign the way the test, validation, and training sets are generated. Figure 4.13 presents the redesigned paradigm for the grey, green, yellow, and purple motor imagery data. In the new system, at the start of each training session, a random 60% of each type of **trial** is selected. These are then appended to form the training set. 50% of the remaining trials of each type are then selected and appended to form the validation set. The remaining trials of each type are then appended to form the test set. The trials are then chunked in the manner described in Chapter 3 for the desired τ and filter type to form the examples. This system has the desirable trait of keeping the training, validation, and test sets relatively balanced between each class type. It also ensures that no two sets have examples from the same trial, thus eliminating the ability of the network to over-fit to a given trial. For the left and right hand motor imagery experiment, this means that subject MM must be dropped as they only have four valid trials for each of the two tasks.

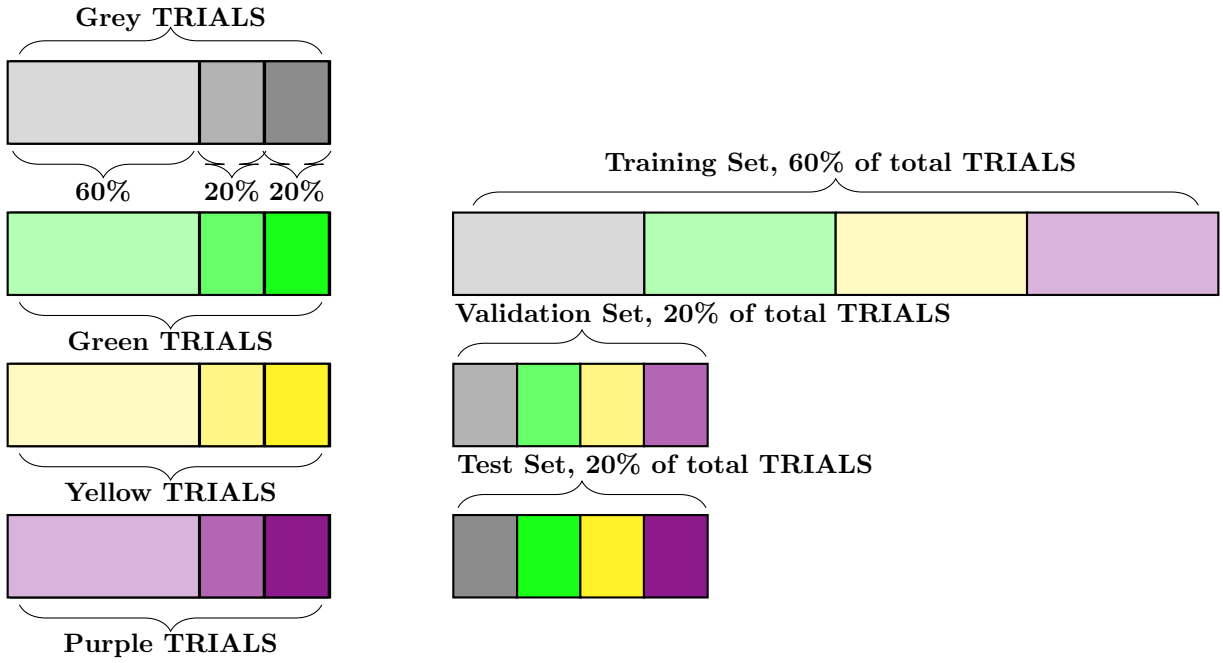


Figure 4.13: Revised training schedule for the Grey, Green, Yellow, and Purple motor imagery models: every run of the model, a random 60% of each imagery task's TRIALS is taken, these are appended to form the training set. Of the remaining 40% of TRIALS, a random 50% of each imagery task's examples is taken, these are appended to form the validation set. The remaining 20% forms the test set. This process ensures an equal proportion of classes among the three sets. Within each set, the trials are then chunked to form the examples as discussed in Chapter 3.

The revised training sessions in the following sections all use this method for creating the training, validation, and test sets. Each experiment is carried out only once because of time constraints, but given the problems with over-fitting based on trials, it may be the case that this could bias the results. It is possible that entire trials are significantly different from trials even of the same type, so which set trials

Table 4.9: Accuracy rate at minimum validation error, L/R imagery, Morlet-transformed data

Subject	Epoch	Validation	Test
AF	27	49.889	49.889
AM	497	69.040	74.289
AS	165	60.760	53.622
BM	230	60.220	49.229
GZ	500	71.900	56.117
KP	483	82.340	71.200
MX	267	79.360	69.400

The table shows the epoch and accuracy rates in the validation and test sets for each subject at the point where the validation set accuracy is maximized. The model is given a maximum of 500 training epochs with $\tau = 100$.

are placed into could determine whether the network can accurately classify its examples or not. Therefore, the following results should be viewed as preliminary, but nevertheless indicative of the potential for the application of convolutional neural networks to EEG data for classification of motor imagery tasks.

4.4 Revised Left and Right Hand Motor Imagery Results

All results in this section use the left and right hand motor imagery data from [7] and a convolutional neural network of the structure described in Section 3.6. Additionally, a batch size of 500 and a learning rate $\eta = 0.01$ are used with 6 seconds of data per trial.

4.4.1 Morlet Wavelet Transformed

The following results are generated by application of the Morlet wavelet and then chunked with $\tau = 100$. The results are presented in Table 4.9 as well as Figures 4.14 through 4.20. The results are not as resounding as before, but there are several aspects which are positive. All subjects, except for AF, do exhibit significant learning. Additionally, several subjects do attain test set accuracies significantly higher than the 50% one would expect from classifier that makes random choices, with AM, KP, and MX all around 70% accurate. These results are encouraging since they are produced from the application of a filter that can be computed before run-time and only uses 100 slices of this transformed data. This means, at 500 Hz sampling frequency, the BCI could still be very responsive to the user's wishes. Additionally, there are many simple tweaks that could be made to improve the accuracy of these models, which is discussed further in Chapter 5.

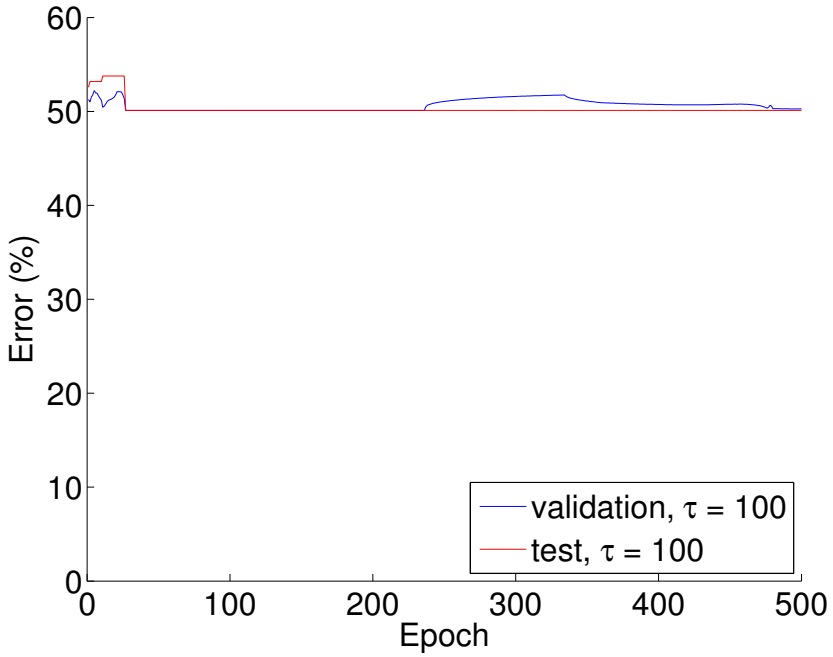


Figure 4.14: Validation and test set error for subject AF using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

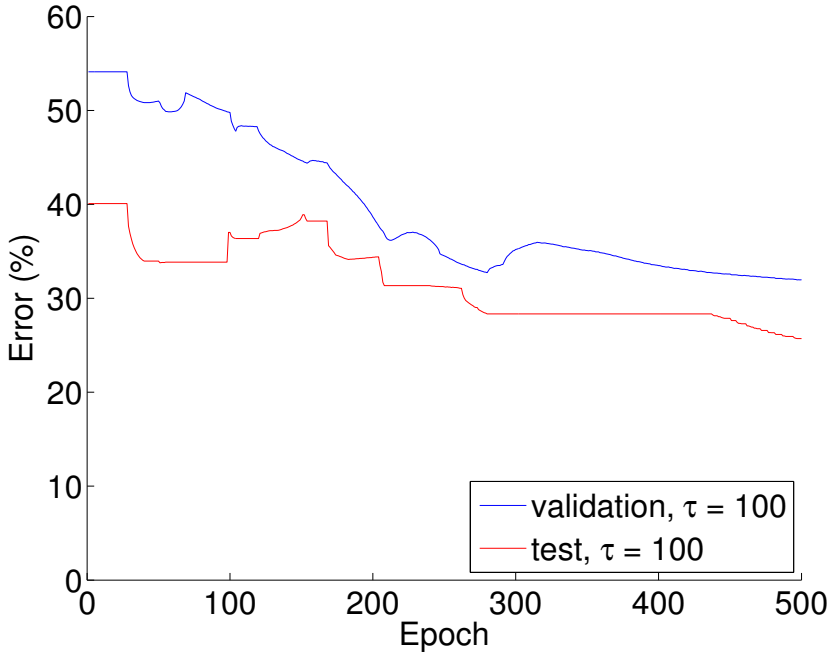


Figure 4.15: Validation and test set error for subject AM using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

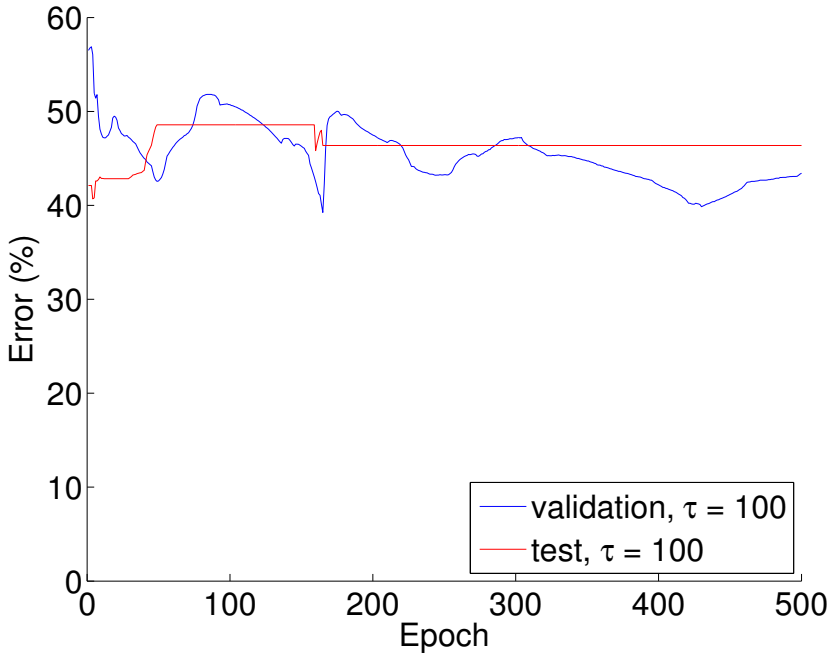


Figure 4.16: Validation and test set error for subject AS using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

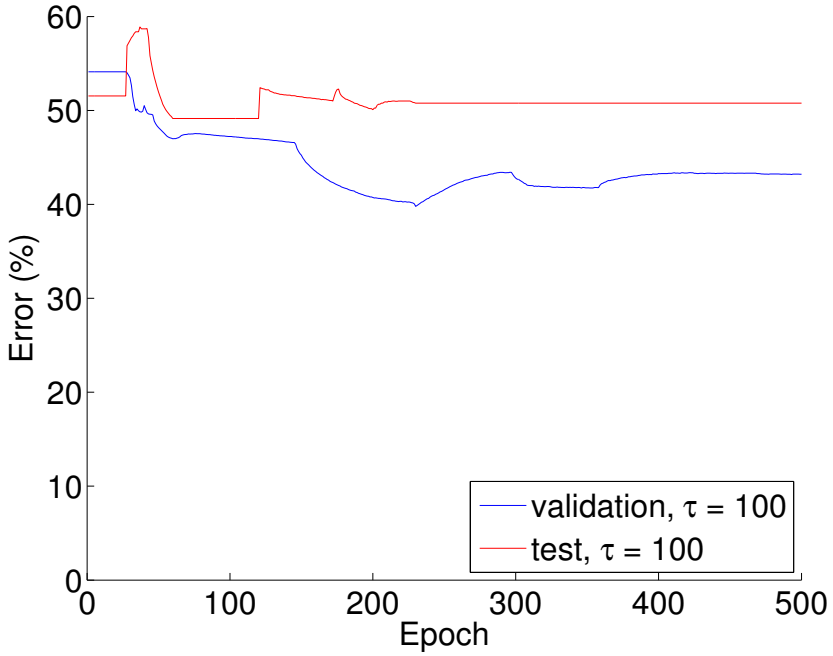


Figure 4.17: Validation and test set error for subject BM using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

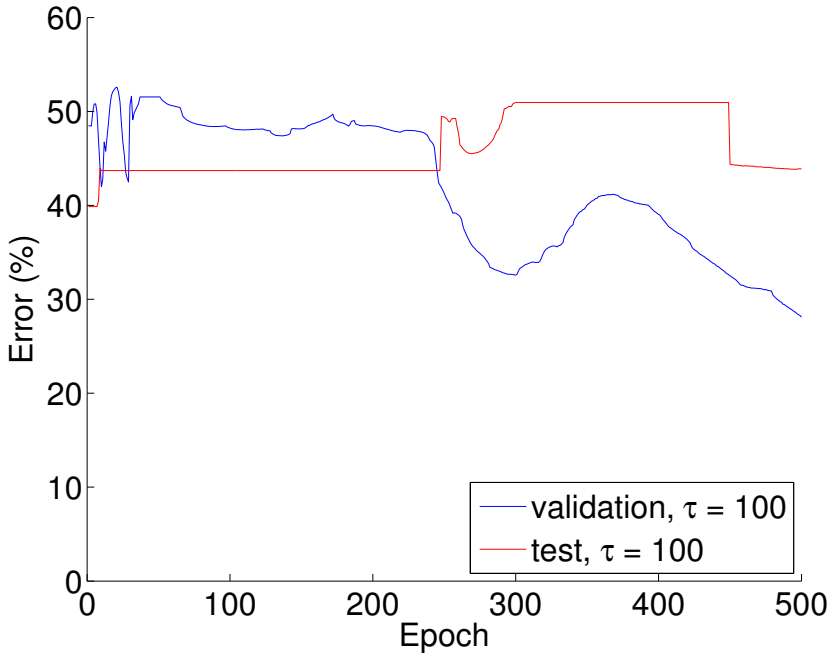


Figure 4.18: Validation and test set error for subject GZ using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

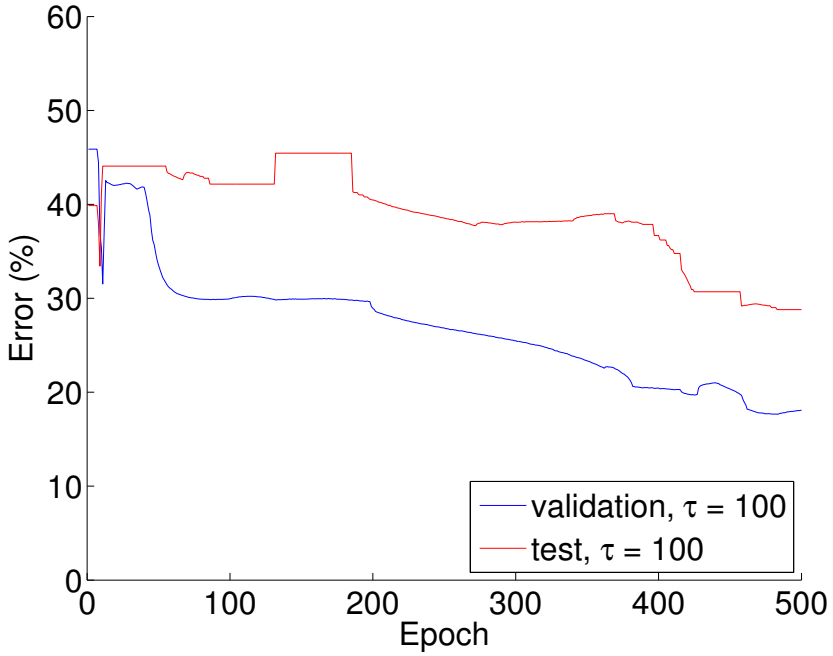


Figure 4.19: Validation and test set error for subject KP using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

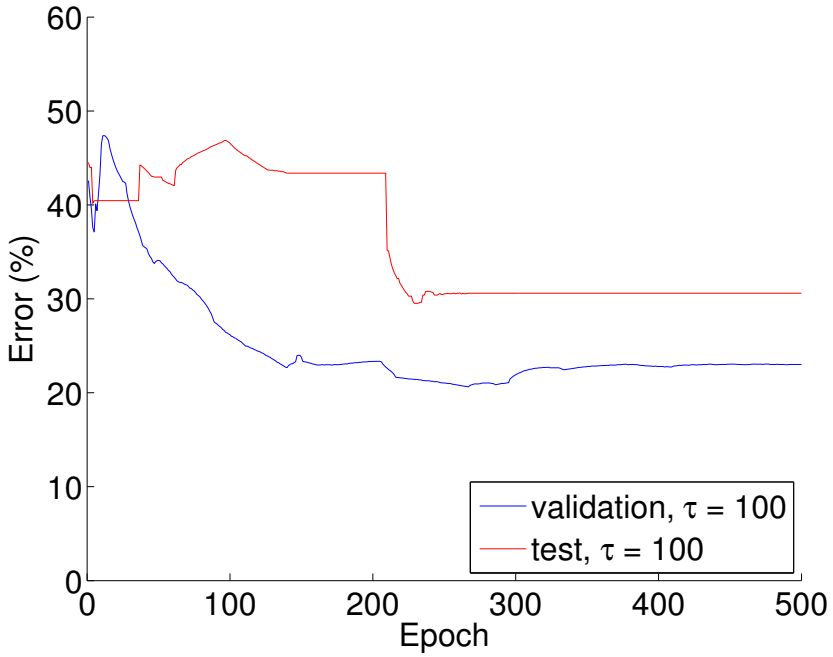


Figure 4.20: Validation and test set error for subject MX using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

4.4.2 Voltage Data

The results for voltage data are presented for three subjects in Table 4.10 as well as Figure 4.21. Once again, this is a three-class classification: no thought, left, and right. While one can see that the BM and GZ results are disappointing, perhaps due to poor initialization, the results for KP are encouraging. The model manages to achieve about 80% accuracy on the test set. This is particularly encouraging because the input is the noisier, untransformed data and there are three classes, meaning a random guess would have around 33% accuracy. These results are further discussed in Chapter 5.

Table 4.10: Accuracy rate at minimum validation error, L/R imagery, voltage data

Subject	Epoch	Validation	Test
BM	1408	57.600	50.700
GZ	1	49.665	42.207
KP	1469	76.626	80.082

The table shows the epoch and accuracy rates in the validation and test sets for each subject at the point where the validation set accuracy is maximized. The model is given a maximum of 4000 training epochs with $\tau = 100$.

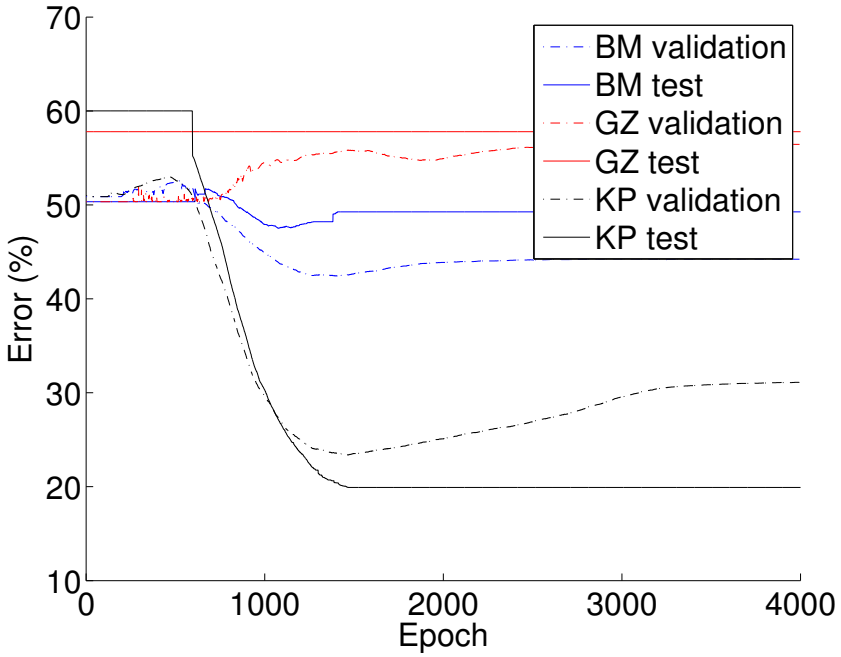


Figure 4.21: Validation and test set error for subjects BM, GZ, and KP using the new methodology, for $\tau = 100$. The test errors are the solid lines. The test error is only calculated when a new minimum of the validation error is reached.

4.5 Revised Grey, Green, Yellow, and Purple Motor Imagery Results

All results in this section use the 4-class motor imagery data from [29] and a convolutional neural network of the structure described in Section 3.6. Additionally, a batch size of 500 and a learning rate $\eta = 0.001$ are used with 6 seconds of data per trial.

4.5.1 Morlet Wavelet Transformed

The results can be seen in Figures 4.22 and 4.23. The model does not manage to learn much in either case and the error remains high. This is perhaps due to poor initialization and requires further research, as discussed in Chapter 5. Of course, this is also a more complex learning task, as the model must learn four classes with less fine-grained data (200 Hz) and fewer data points.

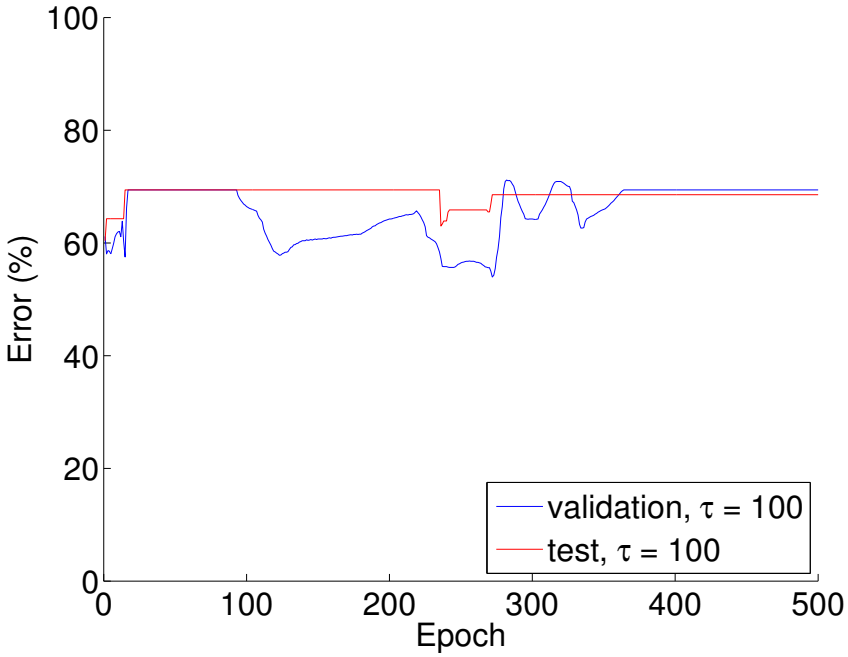


Figure 4.22: Validation and test set error for subject EG using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

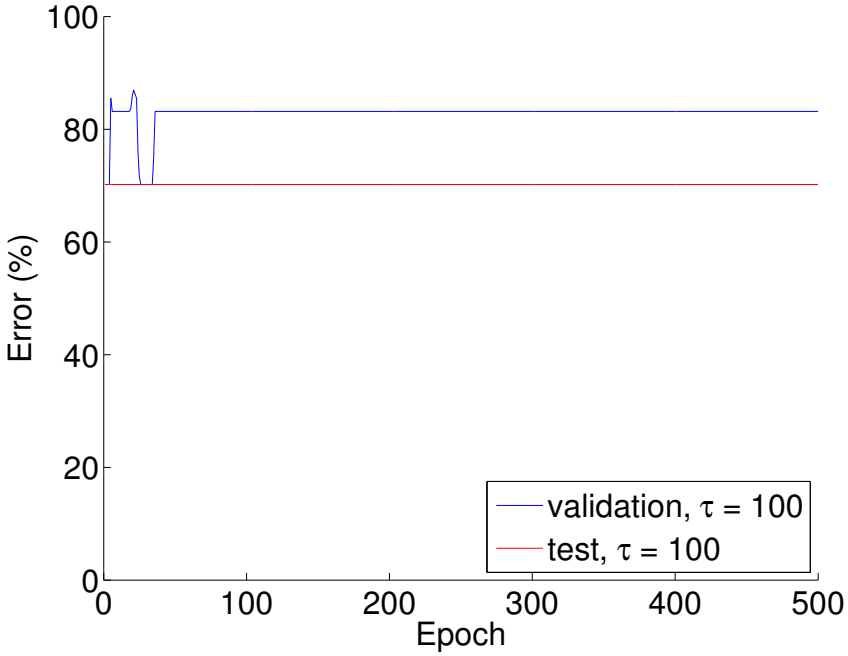


Figure 4.23: Validation and test set error for subject LG using the new methodology, for $\tau = 100$. The test error is the red line. The test error is only calculated when a new minimum of the validation error is reached.

Chapter 5

Conclusion and Future Research

This paper proposes a novel application of convolutional neural networks to classify user intent generated by motor imagery via EEG data for use in a real-time BCI. To the best of the author's knowledge, there do not exist any such applications in the literature. Convolutional neural networks have been applied to EEG data for the prediction of epileptic seizures, but for motor imagery classification very few researchers have attempted to apply deep learning algorithms. It is the only paper to attempt to classify more than two types of motor imagery using deep learning techniques. Additionally, this paper proposes a novel method for defining convolutional filters along the scalp to disjoint groups of electrodes that cover similar regions of the brain. Although the initial results were discovered to demonstrate significant overfitting very late into the project, the preliminary results from a revised experimental set-up are still significant and offer many opportunities to be easily improved.

The highest accuracy achieved in the revised results is 80.08% for subject KP in the left and right hand imagery experiments, a significant improvement over a random model, which would be expected to have a 50% accuracy. Additionally, higher accuracies may be easily attainable by a more thorough search for valid hyperparameters for the model. It must be emphasized that this 80.08% accuracy is achieved when using $\tau = 100$ using untransformed voltages. This is a three class classification task distinguishing itself from the research which has focused solely on the binary distinction between two motor imagery tasks. This raises the potential of a very responsive BCI, since the voltage readings need not be passed through time consuming preprocessing. $\tau = 100$ means that only 100 EEG observations need to be fed to the convolutional network for classification. At a sampling frequency of 500 Hz, this would mean that, at worst, a user would need to wait only 0.2 seconds for a correct interpretation of their intent. This makes the prospect of a BCI which can easily handle the BrainRunner challenge in the Cybathlon, and perhaps more tangible applications, tantalisingly close.

The research presented includes many choice points at each of which an educated, though by no means definitive given the complexities of the fields involved, decision was made. Thus, any future research should begin with tweaking the hyperparameters such as the learning rate and the time slice τ to attempt to improve on the reported results. However, these are not the only potential simple variations future research could test. Other potential questions include: are three electrodes

in the disjoint groups the optimal choice? Should different activation functions be used at each layer? Should the depth of the network be changed? Should a second convolutional layer be added? Are 3×5 convolutional filters the optimal choice for extracting meaningful features? How many units should be in each layer? Should the cost function be significantly modified to aid in convergence and discourage over-fitting? Et cetera. The list of minor modifications and tweaks to the model which could significantly increase its accuracy goes on. However, given the behaviour of the network in the revised data, there are three major steps that should be taken before other considerations.

First, stochastic gradient descent is a relatively simple algorithm for learning the parameters relative to more recent research. Other algorithms exist, which may significantly improve learning, such as BADMM [34]. Momentum could also be implemented to aid in the convergence by modifying the stochastic gradient descent algorithm. Second, the networks suffer from poor initialization, simply randomising the weights is not sufficient for convergence. A simple solution to this problem would be to allow for more training epochs, but there is no way to know when it may converge. A better solution is to look into more advance deep learning techniques that can be pre-trained, such as deep belief networks. Another proposal for future research is to implement a stacked convolutional auto-encoder, such as one implemented by [35], which could learn high-level features to feed into a classifier. Recurrent neural networks could also offer some potential advantages when dealing with the time dimension of the data.

Third, and perhaps most fundamentally, future research should focus on what distinguishes the trials and how to improve out of sample performance. As observed, there are traits which can clearly distinguish the trials the model is trained on, which leads to the over-fitting problem demonstrated in the results, but even under the new system of creating the test, validation, and training sets, some trials demonstrate behaviour that indicates examples even with the same label may be significantly different, so much so that training produces widely divergent behaviour on the validation and test sets from minor changes to the parameters. This suggests that perhaps only using the Morlet wavelet or voltages as currently defined is insufficient for fully distinguishing motor imagery tasks. This opens up many interesting future extensions to the project, such as increasing the number of input *channels* to the neural network. Input channels typically refers to the intensity of red, green, and blue in pixels when convolutional neural networks are applied to colour images. Here, multiple input channels could mean feeding the neural network both the voltage data and the Morlet transformed value for a given time period, but could also include a moving average of the voltage data, or deviations from it. This could help focus the learning process on changes in the state of the brain, perhaps improving its ability to identify motor imagery.

More data collection could be necessary for two reasons. First, it may not be possible to train deep learning algorithms on only around 20 labelled trials in a way that generalizes well. However, subject LG in the data provided by [29] has around 360 labelled trials, and while more data is always useful, improving the results of this paper rests more on the extensions explained above before subjecting

individuals to longer data collection trials. Still, there is one potential pitfall of taking models trained on the given data and applying it to an real-time BCI: how the trials deal with transitions between motor imagery tasks. As defined, **every** trial is the transition from no thought to a motor imagery task. At no point have subjects been asked to switch from, for example, imagining opening and closing the right hand to opening and closing the left hand. Even in the BrainRunners video game participants will be asked to perform such transitions. Switching from one imagery task to another, with no training examples, will likely lead to slower response from the BCI, as a correct classification may only occur once the entire input time slice is entirely within a period of specific motor imagery and has the potential for numerous misclassifications during the transition and after. Future data collection could include such transitions to ensure the robustness of the BCI in real time. Training an automatic artefact rejecter which could remove muscular-skeletal movements and signals would further improve the robustness of the real-time BCI.

Finally, this project has focused solely on estimating separate models on different individuals, but future research could attempt to learn features from EEG data which are common among all individuals. Any success would be interesting for having found generalisable features from scalp EEG, but would also have the practical application of reducing training times for new users, as the model would only need to be honed to the specific characteristics of the new user.

Bibliography

[1] E. Zurich, "Cyblathon bci race," *www.cybathlon.ethz.ch/the-disciplines/bci-race.html*, 2015. pages 3

[2] J. N. Mak and J. R. Wolpaw, "Clinical applications of brain-computer interfaces: Current state and future prospects.," *IEEE reviews in biomedical engineering*, vol. 2, pp. 187–199, 2009. pages 4, 5

[3] D. McFarland and J. Wolpaw, "Brain-computer interface operation of robotic and prosthetic devices," *Computer*, vol. 41, pp. 52–56, Oct 2008. pages 5

[4] K. Tanaka, K. Matsunaga, and H. Wang, "Electroencephalogram-based control of an electric wheelchair," *IEEE Transactions on Robotics*, vol. 21, pp. 762–766, Aug 2005. pages 5

[5] B. H. Dobkin, "Braincomputer interface technology as a tool to augment plasticity and outcomes for neurological rehabilitation.," *The Journal of Physiology*, vol. 579.Pt 3, p. 637642, 2007. pages 5

[6] D. J. McFarland, W. A. Sarnacki, and J. R. Wolpaw, "Electroencephalographic (EEG) control of three-dimensional movement," *Journal of Neural Engineering*, vol. 7, no. 3, p. 036007, 2010. pages 5

[7] A. Ferrante, C. Gavriel, and A. Faisal, "Data-efficient hand motor imagery decoding in EEG-BCI by using morlet wavelets & common spatial pattern algorithms," pp. 948–951, 2015. pages 6, 7, 8, 19, 20, 21, 23, 25, 36, 44, 56

[8] C. Guger, H. Ramoser, and G. Pfurtscheller, "Real-time EEG analysis with subject-specific spatial patterns for a brain-computer interface (BCI)," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, pp. 447–456, Dec 2000. pages 7

[9] D. McFarland, L. Miner, T. Vaughan, and J. Wolpaw, "Mu and beta rhythm topographies during motor imagery and actual movements," *Brain Topography*, vol. 12, no. 3, pp. 177–186, 2000. pages 7

[10] H. Yuan, A. Doud, A. Gururajan, and B. He, "Cortical imaging of event-related (de)synchronization during online control of brain-computer interface using minimum-norm estimates in frequency domain," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, pp. 425–431, Oct 2008. pages 7

[11] C. Park, D. Looney, N. Rehman, A. Ahrabian, and D. Mandic, "Classification of motor imagery BCI using multivariate empirical mode decomposition," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 21, pp. 10–22, 2013. pages 7

[12] Q. Zhao, L. Zhang, and A. Cichocki, "EEG-based asynchronous BCI control of a car in 3D virtual reality environments," *Chinese Science Bulletin*, vol. 54, no. 1, pp. 78–87, 2009. pages 7

[13] P. W. Mirowski, D. Madhavan, and Y. LeCun, "Time-delay neural networks and independent component analysis for EEG-based prediction of epileptic seizures propagation," in *AAAI*, 2007. pages 8

[14] P. Mirowski, Y. LeCun, D. Madhavan, and R. Kuzniecky, "Comparing SVM and convolutional networks for epileptic seizure prediction from intracranial EEG," in *IEEE Workshop on Machine Learning for Signal Processing, 2008. MLSP 2008.*, pp. 244–249, Oct 2008. pages 8

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998. pages 8, 9, 15

[16] X. An, D. Kuang, X. Guo, Y. Zhao, and L. He, "A deep learning method for classification of EEG data based on motor imagery," in *Intelligent Computing in Bioinformatics* (D.-S. Huang, K. Han, and M. Gromiha, eds.), vol. 8590 of *Lecture Notes in Computer Science*, pp. 203–210, Springer International Publishing, 2014. pages 8, 23

[17] F.-F. Li and A. Karpathy, "Stanford CS231n course materials." `http://cs231n.github.io/convolutional-networks/`. Accessed: 03-09-2015. pages 9, 12, 15

[18] University of Montreal LISA Lab, "Deep learning tutorials." `http://deeplearning.net/tutorial/`. Accessed: 03-09-2015. pages 9, 12

[19] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, and S. Tandon, "Stanford deep learning tutorial." `http://ufldl.stanford.edu/tutorial/`. Accessed: 03-09-2015. pages 9, 11, 12

[20] J. van Doorn, "Analysis of deep convolutional neural network architectures," 2014. `http://referaat.cs.utwente.nl/conference/21/paper/7438/analysis-of-deep-convolutional-neural-network-architectures.pdf`. pages 9, 12, 15

[21] A. Tveit and T. Morland, "DeepLearning.University – an annotated deep learning bibliography." `http://memkite.com/deep-learning-bibliography/`. Accessed: 03-09-2015. pages 9

[22] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2015. pages 9, 10, 11, 13, 15, 17

[23] H. R. Roth, J. Yao, L. Lu, J. Stieger, J. E. Burns, and R. M. Summers, "Detection of sclerotic spine metastases via random aggregation of deep convolutional neural network classifications," in *Recent Advances in Computational Methods and Clinical Applications for Spine Imaging*, pp. 3–12, Springer, 2015. pages 9

[24] V. John, S. Mita, Z. Liu, and B. Qi, "Pedestrian detection in thermal images using adaptive fuzzy c-means clustering and convolutional neural networks," in *Machine Vision Applications (MVA), 2015 14th IAPR International Conference on*, pp. 246–249, IEEE, 2015. pages 9

[25] T. D. Team, "Deep learning tutorials," *http://deeplearning.net/tutorial/index.html*, 2013. pages 9, 18

[26] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012. pages 11

[27] NVIDIA, "CUDA," *www.nvidia.com/object/cuda_home_new.html*, 2015. pages 17

[28] NVIDIA, "Tesla k40c," *http://www.nvidia.com/object/tesla-servers.html*, 2015. pages 17

[29] E. Gonzalez Ponferrada, "Design and implementation of an EEG-based brain-computer interface," 2015. pages 19, 20, 21, 37, 51, 61, 64

[30] S. Arroyo, R. P. Lesser, B. Gordon, S. Uematsu, D. Jackson, and R. Webber, "Functional significance of the mu rhythm of human cortex: an electrophysiologic study with subdural electrodes," *Electroencephalography and clinical Neurophysiology*, vol. 87, no. 3, pp. 76–87, 1993. pages 25

[31] D. J. McFarland, L. A. Miner, T. M. Vaughan, and J. R. Wolpaw, "Mu and beta rhythm topographies during motor imagery and actual movements," *Brain topography*, vol. 12, no. 3, pp. 177–186, 2000. pages 25

[32] C. DAvanzoa, V. Tarantinob, P. Bisiacchib, and G. Sparacinoa, "A wavelet methodology for eeg time-frequency analysis in a time discrimination task," *International Journal of Bioelectromagnetism*, vol. 11, no. 4, p. 185, 2009. pages 25

[33] "Easy cap." `http://easycap.brainproducts.com/`. Accessed: 02-09-2015. pages 29

[34] H. Wang and A. Banerjee, "Bregman alternating direction method of multipliers," in *Advances in Neural Information Processing Systems*, pp. 2816–2824, 2014. pages 64

[35] 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, Springer, 2011. pages 64