EEG Waveform identification based on Deep Learning Techniques

Brian Ezequiel Ail

Supervisor: Rodrigo Ramele

Final Project

Computer Engineering Degree

Abstract

The use of Brain-Computer Interfaces can provide substantial improvements to the quality of life of patients with diseases such as severe Amyotrophic lateral sclerosis that cause Locked-in syndrome, by creating new avenues in which these people can communicate and interact with the outside world. The P300 speller is an interface which provide the patients the ability to spell letters and eventually words, so that they can speak while unable to use their mouth. The P300 speller works by reading signals from the brain using an Electroencephalogram. Traditionally, these signals were plotted and interpreted by specialized technicians or neurologists, but the development of Machine learning algorithms for classification allow the computers to perform this analysis and detect the P300 signals, which is an Event Related Potential triggered when certain stimuli such as a bright light is triggered on a place that the patient is focused on. In this thesis we used a Convolutional Neural Network to train multi-channel EEG readings, and attempted to detect P300 signals from a P300 speller. The results are corroborated against a public ALS dataset.

Keywords – EEG, BCI, Brain-Computer Interface, Electroencephalogram, Machine Learning, Convolutional Neural Network, Deep Learning

Contents

List of Figures

List of Tables

1 Introduction

The understanding of the human brain has been one of the most exciting fields in recent history. The term psychokinesis and telekinesis has been around in science fiction for more than a century, from Luke Skywalker recalling his lightsaber using the Force, to Magneto or Jean Grey of the X-Men. And what is telekinesis if not the movement of a physical object through signals emanating from the human brain?[5] The technology is not quite there at the moment, but it does not seem too far fetched to imagine right now, and not at all in the realm of science fiction. Controlling a prosthetic limb with the mind, or a monkey playing Pong[36] with just his brain, are things that would have been unimaginable only a few years ago, but cutting edge companies such as Neuralink are already working on invasive "high-bandwidth brain-machine interface system" [34], which can make this kind of things a reality.

In terms of potential medical applications, patients with severe cases of Amyotrophic lateral sclerosis (ALS) can get into a 'Locked-in' state, in which they are unable to move any of their muscles to communicate. Creating a Brain-Computer interface can allow them to connect with the outside world and substantially improve their quality of life[22].

1.1 Amyotrophic lateral sclerosis

Amyotrophic lateral sclerosis (ALS) is a progressive neurodegenerative disease that affects nerve cells in the brain and spinal cord. It is often known as Lou Gehrig's disease, after a known baseball player was diagnosed with it[7].

The disease causes progressive and cumulative physical disabilities in the patients, eventually taking over the respiratory system and causing complications due to asphyxiation. The causes and the mechanism of the progression for the disease have not yet been understood, making it impossible to create a cure. The current treatment involves a multidisciplinary approach that can manage the symptoms and attempt to improve the quality of life of the patients, [38], Some of these treatments can reduce symptoms and extend the life expectancy by many years [7].

Some progressive cases of ALS develop Locked-in syndrome (LiS). These syndrome causes

people who are awake and conscious to be a state of almost complete immobility and at a loss of verbal communication [31]. This brings forth a field of research called augmentative and alternative communication (AAC), which would create a means to compensate for the progressive loss of verbal and gestural communication. These can take the form of eye-tracking (ET) or brain-computer-interfaces (BCI). In particular, BCI can be more effective in moderate to severe cases of ALS, as it does not require ocular-motor mobility, which is needed for ET applications.[12]

1.2 Brain Computer Interfaces

The study of Brain Computer Interfaces (BCI) attempts to develop non-muscular channels to communicate and send messages to the external world [53]. These studies are specifically aimed at developing new communication channels for patients with severe neuromuscular disorders, such as ALS, brainstem stroke, and spinal cord injury, and provide them with a pathway to communicate their desires to their caregivers, nurses, doctors and family.

BCI is implemented by reading different electrophysiological signals emanating from the patient, these signals can be read using non-invasive devices such as the Electroencephalogram (EEG), or invasive ones, which involve placing parts of the equipment inside the head, that have the potential to record the activity of even a single neuron [26].

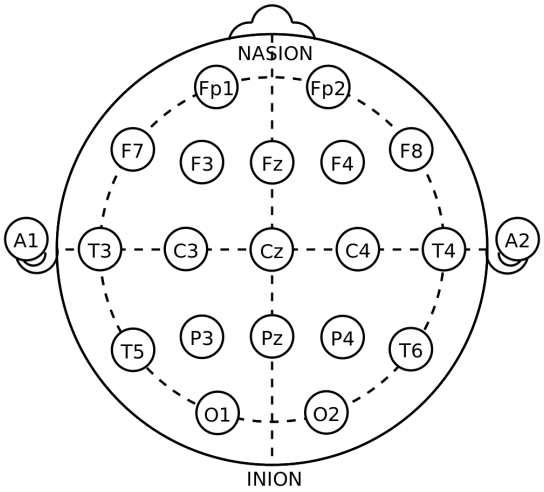
In BCI, these signals were translated into real-time commands, requiring the patients to 'encode', or train the brain to fire specific signals, and then the BCI could translate these signals into commands. Although in recent times the field has advanced with new applications, such as Brain Spelling [15], or controlling a wheelchair[1]. But the ultimate goal is to take it as far as using BCI to restore all the lost mobility and motion [52].

1.3 Electroencephalogram

This thesis focused on signals collected from a non-invasive tool called the Electroencephalogram or EEG, this tool has been around for almost a century, and is right now considered the de-facto non invasive method to read [40] signals from the human brain. In recent times, portable and low-cost EEG have been available such as

Neurosky MindWave or Emotiv Insight and EPOC[34]. The combination of being low cost, non-invasive and highly portable create the opportunity of every person interested on the field to purchase then, generating a very low entry barrier. The EEG consists of multiple electrodes placed on the human scalp. The placement of these electrodes has been standardised as the 10/20 international system, which places the electrodes on the head such that there is a 10%-20% distance of the front-back and left-right of the skull between each of the electrodes.

(a) Example of an EEG being connected. Patient is using a headset map for the 10/20 international system of electrode placement[47]



(b) Electrode placement of the 10/20 international system[8]

1.3.1 EEG Signal composition

The EEG signals are usually composed by[39]

- Artifacts: Signal sources not generated by the Nervous System, such as movement from the muscles of the face or the eyes, which are endogenous sources. Or they can be exogenous sources when they are generated by outside electromagnetic fields.
- Non Stationary: The signals from the EEG are non-stationary, the source from the signal can come from different parts of the brains at different times, and have varying frequencies and amplitudes.
- DC Drift: The Direct Current (DC) drift is a phenemoenon in which the base value around which the EEG oscilates changes over time. This is caused by several factors,

but the most important one is called electrode polarization. This is an effect caused by having a metallic electrode placed in contact with an electrolyte, which can be either the conductive gel or the scalp. This causes a chemical reaction which generates a charge accumulation on the electrode. This effect happens at different rate and quantities on the different electrodes, which is meassured as a difference of potential on the EEG[44]. This effect however, is not always undesired, as some cognitive functions in infants such as Slow Cortical Potentials or Slow Activity Transients can be understood and measured thanks to the DC drift[37],

- Basal EEG Activity: While the P300 signal is the one we are trying to identify, there is all sort of activity going on in the nervous system, such as thinking or processing of outside stimuli.
- Inter-Subject variability: Not all human brains are the same, and not all EEG signals are the same, they can vary significantly from person to person.[16].

1.3.2 The P300 signal

This study focuses on identifying an Event Related Potential (ERP), these are stereotyped signals generated as a response to special types of stimuli. These events are time-locked, which means that they are triggered a fixed amount of time after the stimuli.

We will focus on an ERP called P300 which is triggered by an experimental design called the *oddball paradigm*, this paradigm consists in showing repetitive stimuli which are suddenly interrupted by a deviant stimuli. The reaction to this 'oddball' stimuli is then recorded. The oddball paradigm was first used to trigger ERP by Nancy Squires, Kenneth Squires and Steven Hillyard at the UC San Diego (1977)[48]. And it was found that the event occurs around 300 ms after the stimuli was presented, hence the name P300.

This P300 Speller [15] is a type of AAC, that types characters based on the detection of the P300 signal when characters are flashed in a Matrix.

1.3.3 P300 Speller

The P300 speller was first described in [15] and consists of a Matrix of 6 rows and 6 columns, each of these containing either a character or a number or a symbol. The spelling

Figure 1.2: An example of a P300 speller[45]

is done by randomly flashing the columns and rows of the matrix while the subject is fixated on a letter, triggering the P300 signal when the letter is lit up.

The recordings are then divided or *segmented*, creating a different segment that begins when each of the rows or columns is lit, and ends after a certain time t_{max}, this time should be at least 300ms, so that the P300 signal is contained within the segment. And it is possible that the segments can be overlapped if the speed of the flashing lights is fast enough, or if the inter stimulus interval (ISI) is too short.

The generated segments will be 12, as the matrix is composed of 6 rows and 6 column. 2 of these segments contain the P300 signal, while the remaining 10 do not.

A successful speller can accurately predict which of these 2 segments contain the signal, as finding the row and column will pinpoint the letter that the subject was attempting to spell, so in order to make this, we have to create an algorithm to which we can give the 6 signal segments from the rows, and the 6 signal segments from the columns. This algorithm would then have to *classify* these signals, and find the row and the column with the P300 signal. Once we find the row and the column with the P300, we can pinpoint the letter that the patient was attempting to spell.

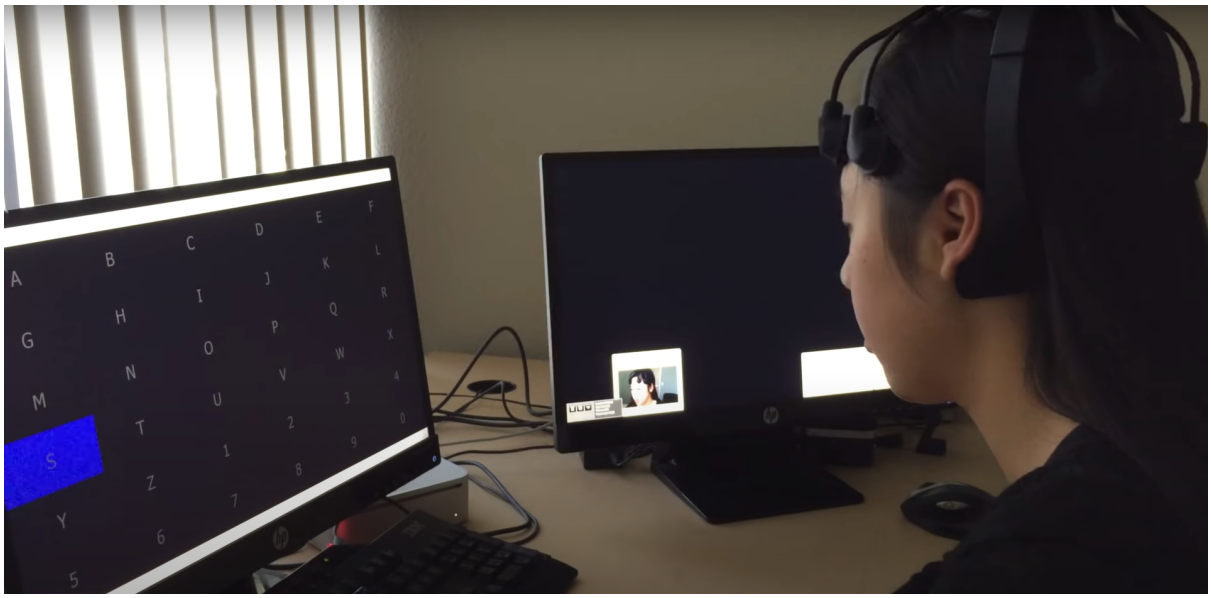


Figure 1.3: A P300 speller being operated while wearing an EEG[11]

1.3.4 Objectives and Contributions

The problem of classifying different signal segments can be solved with a group of algorithms called binary classifiers. These are a group of algorithms that receive different data samples to train or "learn", and will then be able to generalize this information to new examples, so if we can train a classifier with some segments, and tell it which one have a P300 signal and which one do not, it will be able to predict this pattern on future signals, thus allowing to identify which rows and columns of the P300 spellers have the P300 signal, and allow the speller to function, we will cover some examples of binary classifiers in section 2.

However, as mentioned on 1.3.1, the EEG signal is composed by different artifacts and noises. All of these pose different problems at the time of classifying the signals. The inter-subject variability makes the data sets for classifying much smaller, if every person had the exact same signal shapes, algorithms could be trained with data from several people, and then used to predict the readings for different people, thus creating a huge dataset. As it stands, the training must be done only with the readings of the same person that we want to predict. The noises and artifacts on the other hand, will sometimes be too dominant in the signal compared to the P300 signal that we want to find. If the signal-to-noise ratio becomes too low, finding the signal will be much harder.

Historically, EEG signals were interpreted by neurologists or specialized EEG technicians

which could interpret the waveform of the signals picked up by the EEG. Our proposed approach involves plotting the EEG signals so that we can emulate this waveform recognition using Deep Neural Networks (DNN), which have been the defacto algorithm for computer vision and speech recognition[10]

One of the biggest drawbacks and pushbacks against using Neural Networks models is the concept of Explainable AI (XAI) [21]. What happens if we have a self driving car ran by a Neural Network? Can we trust the decisions it takes even though we do not understand them? Can we fully trust a positive diagnosis of cancer by using Neural Network? The concept of explainability in the model involves that users/programmers can understand the motives behind the decisions taken by the Network, and Neural Network models can be seen as a kind of Black Box, we train them with the data and then generalize to other examples, but we don't really know why. To address this issue, our proposal in this thesis was to plot the EEG signals and in this way, the network can analyze the same picture as Neurologists that try to interpret EEG signals. So instead of analysing the raw data, the network will analyse the image of the plot from the EEG segments. This way, the network can be used to corroborate, or be asserted by specialized neurologists or EEG technicians. This idea was explored by Ramele et. al. in [39] and [40], and was our starting point to improve upon.

In the following sections we will take a look at how this classification can be performed using different algorithms such as Neural Networks, Support Vector Machines, or Scale Invariant Feature Transform, as well as the current state of the art involving the use of such methods in recent years. Then we will discuss about our proposal for classification, which involves plotting the signals, and training a Convolutional Neural Network to identify the images, and finally compare these results to other studies performed using a common dataset of ALS patients.

2 AI Methods for signal identification

2.1 Machine Learning

Machine learning (ML) is a branch of Artificial Intelligence (AI), where it's objective is the study of algorithms that update free parameters, based on previous data, in order to achieve a classification, regression or generation tasks. According to Tom. M Mitchell, a formal definition of machine learning is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"[33].

Nowadays, one of the main objectives for these algorithms is classification of data: 'Does this email contain spam?', 'Does this tweet carry a positive or a negative sentiment?', 'What species does this plant belong to?', 'Does this EEG segment contain a P300 signal?'. In these cases, for example, we can train the program the *task* of classifying a set of pictures, by training it with many different emails, which would serve as *experience*, and then we can *measure* it's performance by counting how many of a new set of emails, or a 'testing set' were correctly classified.

This would require both the training set and the testing set not only to be sets of emails that may or may not contain spam, but we must also know which of these emails contain spam and whichs do not beforehand. In this way, the algorithm can use these examples as experience. In this example, 'has spam' and 'does not have spam' are called the labels of the data. Labels are required in a kind of learning called 'Supervised learning', in which we know the labels beforehand, train the program with these labels, and the ultimate goal is to be able to automatically predict new examples with as much accuracy as possible.

There is also a process in Machine Learning known as 'Unsupervised learning'. In this kind of training the data does not have labels and the algorithms usually try to find similarities between the examples and group them together. These can be used for recommendations systems by platforms such as Netflix or Amazon that recommend similar products or anomaly detection such as fraud detection.

In this thesis however, we focused on supervised learning, as it was our objective to classify whether EEG segments contain a P300 signal or not. Since the EEG waveforms

are different for each person, a calibration process must be run, in which the subject must "type" certain words which we know beforehand. By doing this, we will have both the expected outcome (the labels) and the EEG signals. This gives us everything we need to train the algorithm, and use it to generalize and be able to predict other words based only on the EEG signals.

2.2 Artificial Neural Networks

Artificial Networks are a Machine Learning technique which attempts to imitate a simplified version of the mechanism of biological neurons. The human nervous system is composed by neurons, these are cells that are connected between each other in different kinds of ways, creating a network of neurons which transmit electrical impulses between each other. The neuron is composed by a nucleus, an Axon an dendrites. The dendrites receive impulses from other neurons, while the Axon of a neuron has many branches, and is connected to the dendrites of different neurons, this connection is known as *synapses*. These connections are used to transmit impulses of electricity across the body, which can have as many as 86 billion neurons.

In artificial networks, the neurons are modelled as units which are connected between each other with 'weights', each input of a unit is multiplied by a weight, which will affect the output of the unit. The artificial networks usually have architectures of layers of these units, with the first layer being called the input layer, the last layer is known as the output layer, while all the layers in the middle are called hidden layers. Each layer must have at least one unit, but there is no limit to how many units it can have. We can say that a Network is deep when it is a "model consisting of multiple layers or stages of nonlinear information processing"[14].

The networks also needs *stimuli* to learn, in this case the stimuli comes in the form of the training set, these will be the experience used by the network to learn. They can be a set of emails with it's corresponding labels of whether or not they contain spam. These examples however must be formalized into data that the network can understand, so some preprocessing of the data must be done. This is usually referred to as 'feature extraction', a process in which several features are extracted from the training data that can be fed into the neural network. In the case of emails, we can for example use features such as

the amount of time that the word 'MONEY' or 'FREE' appear in the email. If we only take these two features, we can have an input layer of only 2 units.

The input data is then propagated throughout the network through the weights, and each unit will perform mathematical operations to its inputs, and then propagate it to the next layer. The output layer will usually depend on what we are trying to classify. In the case of 'spam' or 'no spam', we have a binary classification problem. An output layer with a single unit is sufficient for this classification. During training, this prediction is then compared to the label, which holds the expected value for the prediction, and if there is an error in the prediction, the network is adjusted to better predict that particular input. This adjustment is done in the form of modifying the weights that connects the neurons of different layers. We can say that a network has been trained when the weights have been completely adjusted to the training data.

So for a network with a single neuron and an input with d features, the forward propagation is defined as:

$$\hat{y} = \sum_{i=1}^{d} w_i \cdot x_i \tag{2.1}$$

Where w_i is the weight connecting the input to the neuron, and x_i is the feature i of the input. So it is essencially doing a linear combination, but before this is complete, two more things must be added, a bias and an activation function. The activation function maps the output of the neuron to the prediction that we want to make. So for example if we are trying to make a binary classification, we can use the sign function as the activation, and predict class A if the sign is negative and class B if the sign is positive.

We can think of the bias as an additional neuron that always outputs a 1. This adds an independant term to the equation, if we think about in terms of a linear function, without an independant coefficient, it would always be anchored to the origin, while an independant term allows it to move up or down along the y axis. After adding the bias and the activation function, we can define the prediction as:

$$\hat{y} = sign(\sum_{i=1}^{d} w_i \cdot x_i + b) \tag{2.2}$$

The process of adjusting the weight is done through a method called backpropagation.

After the network receives an input and makes a forward pass, by propagating throughout all the neurons, the output unit will compare against the label and calculate an error. This error is measured by a loss function, $L = f(\hat{y}, y)$. Where $\hat{y}$ is the output and y is the label. This function can take the form of squared loss $L = (y - \hat{y})^2$ or logistic $L = \log(1 + exp(-y \cdot \hat{y}))$. This latter one usually used for a learning method called logistic regression. But the loss function is very important, as it's the result that the network will try to optimize. So if we have a set D of training samples, the loss minimization function can look like:

$$\min_{\bar{W}} L = \sum_{(\bar{X},y)\in D} (y - \hat{y})^2 = \sum_{\bar{X}\in D} (y - sign(\sum_{i=1}^{d} w_i \cdot x_i + b))^2 \tag{2.3}$$

To update the weights, once the error E has been calculated, the weights are updated by:

$$\bar{W} = \bar{W} + \alpha E(\bar{X})\bar{X} \tag{2.4}$$

The parameter α is called the learning rate, of the network. It works as a regulator of how impactful each input is when updating the weights, there are different reasons for choosing an appropriate learning rate, if is too big the algorithm will fail to converge, and if it is too small it will take too much time to converge.

While the previous loss function is defined on the whole dataset D, the algorithm for the network is called stochastic gradient descent, as it trains the network with randomly selected inputs. Each time an input is used to train, we call it an *epoch*. An alternative to feeding single samples to the network is to train by mini batches. The advantage of mini batches is that it brings a good trade off between speed, stability, and memory requirements [2]. The size of the batches is another parameter that must be adjusted, just as the learning rate, to achieve a better performance for the network.

Once the network has been trained, predictions can be made with the testing set, this will be data that *has not been used in any part of the training*. The performance of the network can then be measured by how well it performs on the testing data. It is possible, however, that the network performs really well on the training data, but really badly on the testing data. This means that the network has failed to generalize it's learning. There can be many reasons for this, one of them, is that the problem cannot be generalized. For

example, if the labels are totally randomized, there is no logical connection between the training set and the labels, so while the network may try to find some kind of weights that adjust to this data, it will not follow any kind of logic that can be generalized.

Another common reason why networks perform poorly on the testing set, is called overfitting. What this means is that the network became too tightly adjusted to the training data, that this adjustment cannot be generalized in the rest of the data. This can be caused by many factors, such as having too many layers in the network, making the model too complex. There can be too much noise in the data, or the training dataset may not be big enough to be able to generalize. Ironically, later in development of this thesis we will run into most of these problems, as the datasets are very small and full of noise.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are biological-inspired networks from the work of Hubel and Wiesel[23], in which they study the organization of the neurons in the cat's visual cortex. The convolutional networks are based off the 'neocognitron', which in turn, was based on this study. They are primarily used for computer vision and image classification.

In a convolutional network, each layer is 3-dimensional, it has a spatial extend and a depth that corresponds to the number of features. Typically, in the input layer, the depth will correspond to the color channels (RGB), and in the hidden layers these features represent different encodings of the shapes in the image. If the input is greyscale, the input layer will have a depth of 1, but subsequent hidden layers will still be 3-dimensional. In this thesis, we created 8-dimensional input images that combined all of the channels of the EEG, making the input layer have a depth of 8.

For a better understanding, suppose we have a 150x150 pixels image in RGB colors, if used it to train a convolutional network, the input layer would have have a spatial dimension of 150x150 and a depth of 3. This would constitute what is known as 'pixels' for the CNN, in the previous example, the input image would have 150x150x3 pixels.

For any particular layer q of the CNN, we can say that it has a L_q, B_q and d_q, where L_q and B_q correspond to the spatial dimensions, while d_q corresponds to the depth. In most

cases, the value of B_q and L_q are the same for image-centric networks. In the previous example, the input layer 1 would have $L_1 = 150$, $B_1 = 150$ and $d_1 = 3$.

In convolutional networks, instead of weights, the connections between the layers are 3-dimensional structures called 'filters'. Much in the same way as the layers' dimensions, the first two dimensions are spatial dimensions, while the third one represents various shapes in the image, and should match d_q from the layer. The spatial shapes are always squared, so we can say that a filter for layer q has a F_qxF_qxd_q size, Where F_q is usually much smaller in dimensions than L_q and B_q. It is most usual that F_q is small and odd, and typical examples are 3 or 5.

When an input reaches a layer, it performs a *convolution* operation. This places a the filter on each of the possible positions of the input image, so that the filter perfectly overlaps with the segments of the image, and then performs a dot product between the parameters in the filter and that segment of the input. The amount of different segments in the image that can be overlapped with the filter will define the spatial dimensions of the next layer. The amount of possible overlappings can be thus calculated as $L_{q+1} = L_q - F_q + 1$ and $B_{q+1} = B_q - F_q + 1$, so in our previous example of a 150x150x3 input, if we use a 3x3x3 filter on layer 1, the spatial dimensions of layer 2 would be $150 - 3 + 1$, resolving in a 148x148 layer. The depth of the second layer however, will depend on the amount of different filters that we want to use on the layer, this is defined in the architecture of the network, using more filters will create more complex networks, as more image shapes and features will be extracted from the image. In our previous example, if we use 30 3x3 filters on the first layer, the second layer size would be 148x148x30. It is normal that the hidden layers have many more (over 500) feature maps, or depth.

To formally define the convolution operation, the pth filter on the qth layer has the parameters $W^{p,q} = [w_{ijk}^{p,q}]$ where i, j, k denote the length, width, and depth of the filter respectively. The feature maps or inputs on the qth layer are denoted by $H^{(q)} = h_{ijk}^{(q)}$. And when q is 1, $H^{(1)}$ represents the input layer. Thus the convolution operation from the qth

layer to the $q+1$th layer is defined by:

$$h_{ijp}^{(q+1)} = \sum_{r=1}^{F_q} \sum_{s=1}^{F_q} \sum_{k=1}^{d_q} w_{rsk}^{d,q} h_{i+r-1,j+s-1,k}^{(q)}$$

$$\forall i \in \{1..., L_q - F_q + 1\}$$

$$\forall j \in \{1..., B_q - F_q + 1\}$$

$$\forall p \in \{1..., d_{q+1}\}$$

Before moving on to the next layer, two more operations are done on the output. Firstly, Rectified linear unit (ReLU) is an activation function, which converts all negative values to 0, and leaves the positive values as they are. This operation does not modify the size of the output, only changes it's negatives values to 0. The RelU activation function has shown tremendous improvements to speed of calculations as has been used as the defacto activation function in CNN.[2]. This operation is not tipically shown in visual representations of the network's architecture.

The last operation is called Pooling, it is more similar to filters, in that it works on small regions of size P_qxP_q of the output, and 'summarizes' it by taking the maximum value in that area as the output, This operation is called MaxPooling. This 'summary' of the output will reduce it's dimensions to $L_q - P_q + 1$ and $B_q - P_q + 1$, the depth will remain intact, as the pooling operation operates independently on each of the channels of the output. This substantially reduces the spatial size of subsequent layers. There are other types of pooling in older network architectures such as average pooling, but they are rarely used nowadays.

The last two important concepts for CNN are padding and stride. When convoluting on the borders of the image, some information is lost, as for example the top left pixel only participates in one convolution operation, while a pixel in the middle can participate in many more convolutions, so some information may be lost. Padding is the technique that solves this problem, by adding $(F_q - 1)/2$ "pixels" all around the borders of the input, to mantain the spatial footprint from layer to layer. The value of these added pixels is set to 0, and the added size to the width and height of the image is $(F_q - 1)$, which is exactly the amount that was lost due to the convolution operation.

Stride on the other hand, is a technique to reduce the spatial footprint of the layers, the idea is that instead of placing the filters on each possible overlapping position, it 'skips' S_q pixels each time the filter is moved, where S_q is the value of the stride. This results in a reduction of the spatial size of the layer of $L_{q+1} = (L_q - F_q)/S_q$ and $B_{q+1} = (B_q - F_q)/S_q$. So if the stride value is 1, the filters would move one pixexl at a time and not skip any, but with higher values of striding, pixels will be skipped. It is most common to use a striding value of 1, and in some cases a striding of 2 can be used, higher than 2 is rarer.[2].

2.4 Support Vector Machines

Support Vector Machine (SVM) are an alternative type of algorithms for data classification. The goal of the SVM is to create an hyperplane that separates the data into classes, For example, if our dataset was as shown in figure 2.1

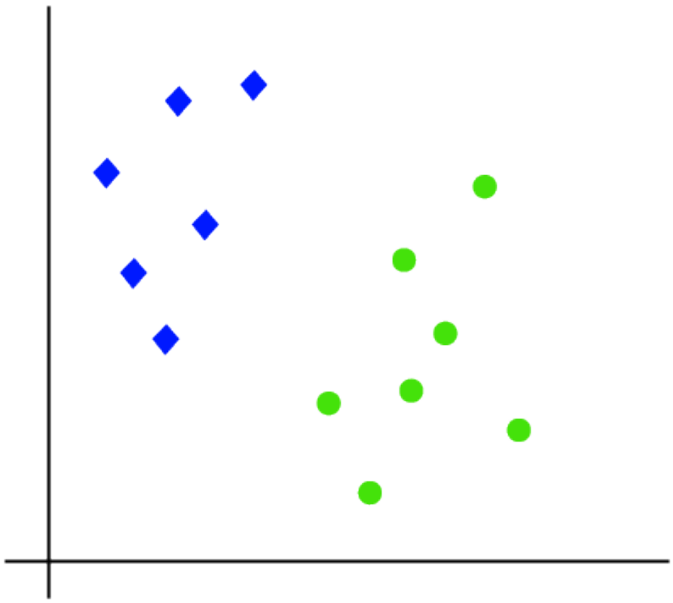


Figure 2.1: SVM Dataset[49]

We can see that there are many different lines that can separate this groups. For example figure 2.2 we see both a red and a green line that achieve this separation. While both work for this training data, the red line appears a bit too close to the blue group, and it will probably fail to generalize.

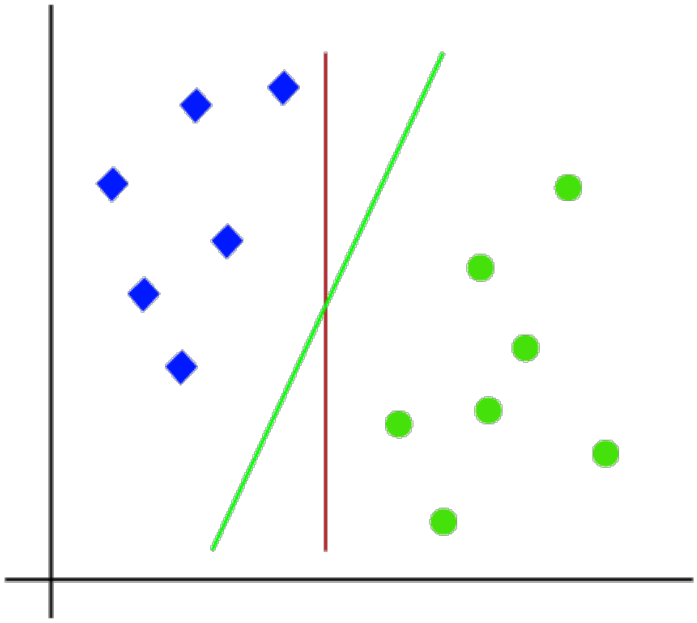


Figure 2.2: SVM Dataset with candidate separations[49]

The approach of the SVM is to identify the support vecctors, which are the two datapoints nearest to the ideal hyperplane which best separates both classes and draw a line that separates them. Then calculate the distance between both these points and the line, this is called the margin. The hyperplane for which the margin is maximum will be the optimal hyperplane (see figure 2.3).

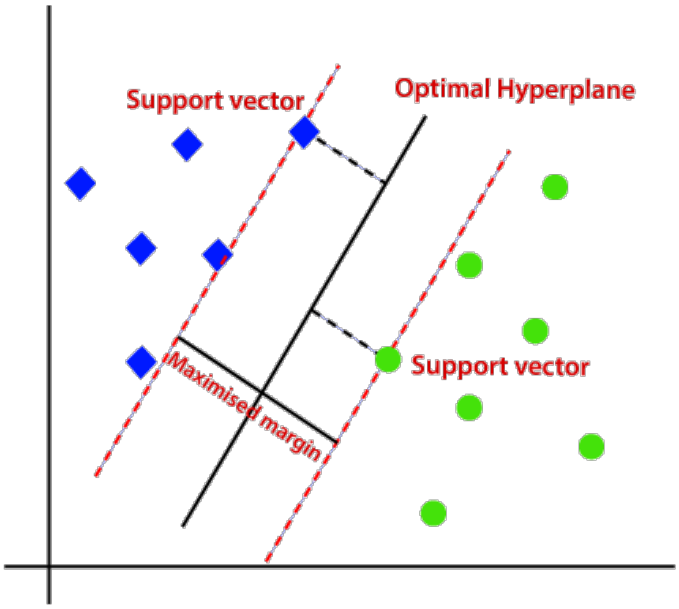


Figure 2.3: SVM Dataset separated by Support Vectors[49]

This simple example however can get more complicated, as the data cannot be linearly separable as in figure 2.4

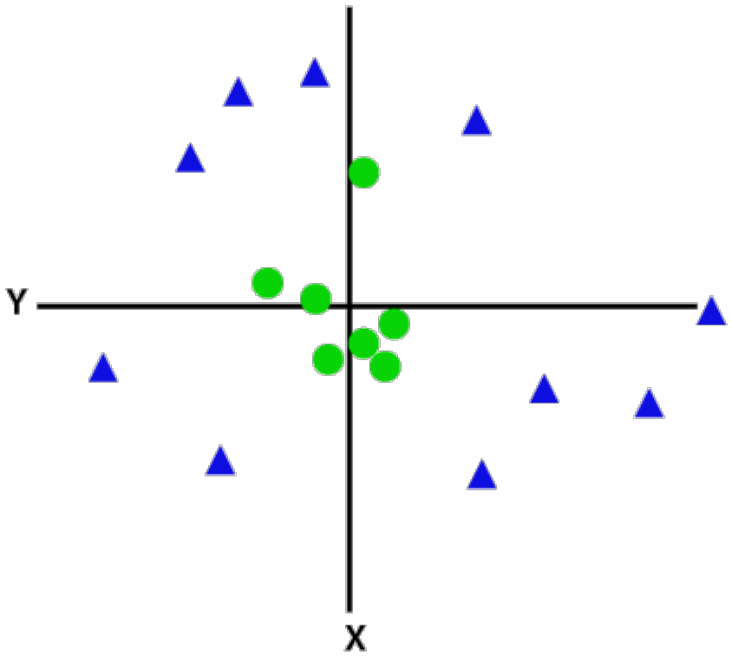


Figure 2.4: Non linearly separable dataset[49]

The solution approach for this cases, is to create a higher dimension for the data, in which the sets are in fact linearly separable. So if we add a third dimension to this example such that

$$z = x^2 + y^2 \tag{2.5}$$

We get the following result:

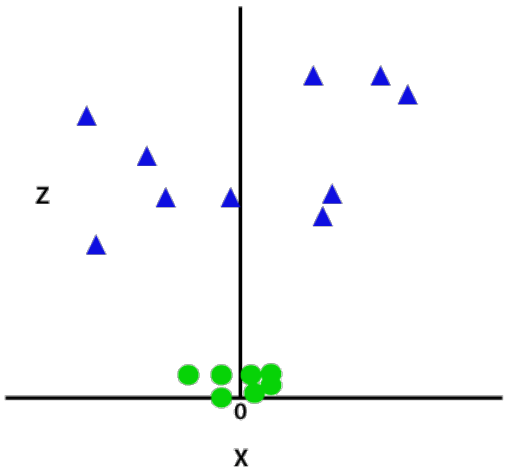


Figure 2.5: Non linearly separable dataset elevated to a third dimension[49]

And we can see that in this case the data is once again separable, the process of adding higher dimensions is called Kernelization[20]. By plotting this separation back in the original dimensions we get the following:

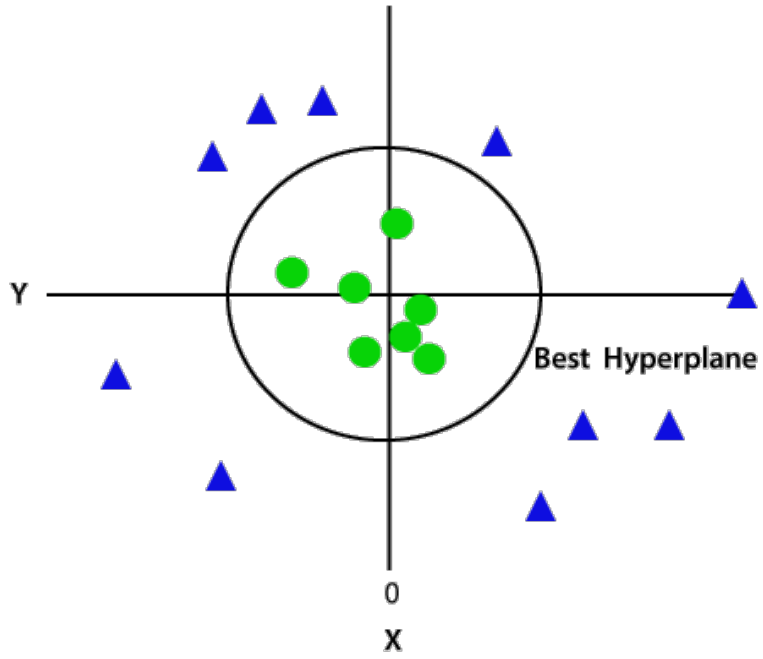


Figure 2.6: Non linearly separable dataset separated by a higher dimension kernel[49]

Adding more dimensions however can bring us overfitting problems, as the model can become more and more complex as we add higher dimensions.

2.5 Scale invariant feature transform

Scale invariant feature transform (SIFT) is an image descriptor for image-based matching and recognition. It was first developed by David Lowe [30] in 2004.

Originally, this method was designed to identify interest points in greyscale images by using a concept known as Gaussian Pyramids, originally formulated by Burt and Adelson[9] and by Crowley and Stern[13]. This method consists of repeatedly smoothing and subsampling the image. Each time the image is smoothed and subsampled constitutes a level in the Gaussian pyramid. A difference of Gaussian Pyramids is computed from the difference in adjacents level of the Gaussian Pyramid. Thus, the points of interest of the image are obtained by finding the points in which the difference in Gaussian Pyramids reach extremes values when compared to adjacent spatial coordinates points and the scale level

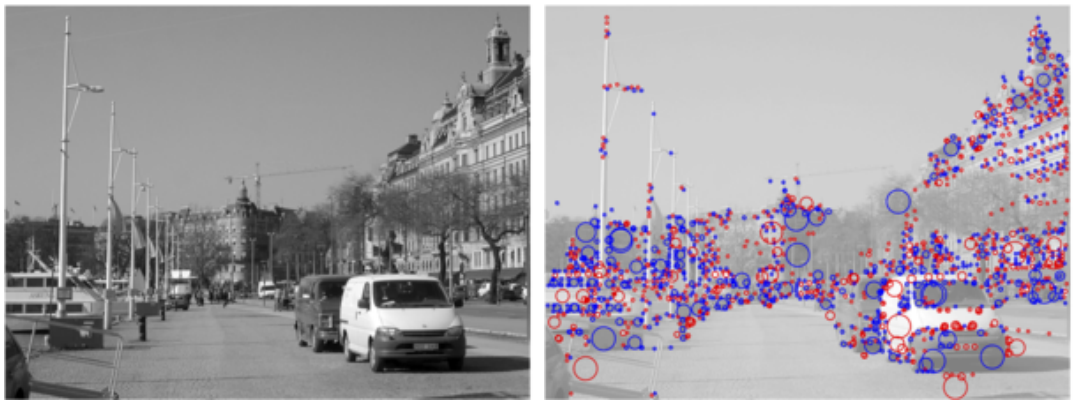


Figure 2.7: Scale-invariant interest points detected in a greyscale image. The radius of the circles indicate the intensity of the interest in the point. The blue circles represent dark image features. The red circles indicate bright image features[28]

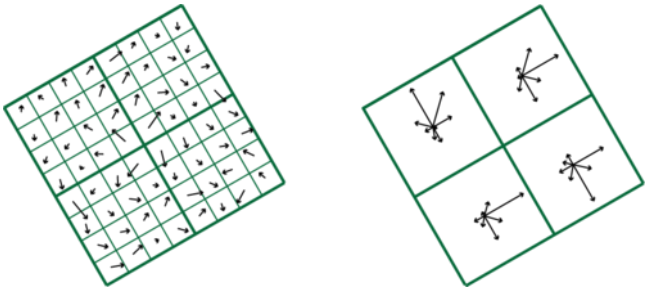


Figure 2.8: The left image shows the gradients around an interest point, orientation and magnitude. The right image shows the resulting vectors accumulated in each of the quadrants [28]

in the pyramid.

This method is called Scaled Invariant, as it can sustain and work despite alterations such as scaling, rotations, translations and robust regarding perspective transformations and illumnation variations[28].

Once the interest points are identified, the second step of this method is called the Image Descriptor. This method calculates the gradients of the pixels around the points of interests of the image by calculating the orientation in a pixel matrix around the interest point.

Once the descriptors are calculated, it is possible to compare the descriptors of different images or different interest points in the same image and attempt to find matches. This can be done in various ways, the simplest one being the L2 distance:

$$d(H_1, H_2) = \sqrt{\sum_k (H_1(k) - H_2(k))^2} \tag{2.6}$$

With H_1 and H_2 being two different histograms of size k. If the distance is 0, the histograms are a perfect match.[35]

2.5.1 Histogram of gradient orientations

Histogram of gradient orientations (HIST) is an alternative to SIFT developed in [40] to work specifically on signal plots. Instead of having a SIFT detector to identify the points of interest, it takes every line of the plot as a point of interest.

The other key difference with SIFT, is that the squared grid for the descriptors is changed to a rectangular flexible grid, which can better adapt to the shape of the plot, otherwise, direction of the plot image could not be entirely covered.

Other changes include removing some of the SIFT features, such as smoothing, rotations and orientations detection. These did not make any sense as the plots all have the same direction and rotation.

This method was used to classify the ALS public dataset and was compared against the work of Riccio et al[42], using both the HIST method and a SVM algorithm, and served as our benchmark to measure the performance of the deep learning algorithm.

3 BCI based on Electroencephalography and Neural Networks

As mentioned in section 1, the advance of hardware technologies, as well as the cheap accessibility of portable EEGs have significantly advanced the study of EEG signals using DNN. According to a 10 year study of Deep Learning in EEG[19], the amount of published papers that matched the searching criterion n [”Deep Learning” AND ”EEG” AND ”Classification” OR ”Recognition” OR ”Identification”] after manually curating the list, 213 papers had been published by march 2020. Out of these the vast majority of them were published from 2019 to 2020, and the trend is increasing rapidly. This papers include all range of studies of EEG, varying from Brain-Computer Interfaces to other applications such as Disease Detection, Sleep stage classification and Emotion recognition.

When discussing applications of Deep Networks in BCI, they were a little hesitant, as Convolutional Networks are often used for computer vision or speech recognition, and did not match the requirements needed to analyze EEG signals. However they found many papers of successful applications of Convolutional Networks such as the work of Li et al.[27], which uses 3 different blocks of convolutional networks to decode motor imagery, which if successful would allow patients with loss of motor function to recover this lost mobility by the use of external devices. Liu et al. [29] used Batch Normalization to speed up the training and alleviate the overfitting. There are many other papers describing not only Deep Architectures for Brain Computer interfaces, but also for disease detection, mainly focusing on Epilepsy (with almost 50% of the published papers) and Depression in second place with about 10%.

Another review by Roy et al.[43] specifically about using Deep Learning on EEG signals review a total of 156 papers. First of all, there is a very similar graph showing an increasing trend in the amount of published papers over the years, rising from less than 10 in 2014 to over 50 in 2018.

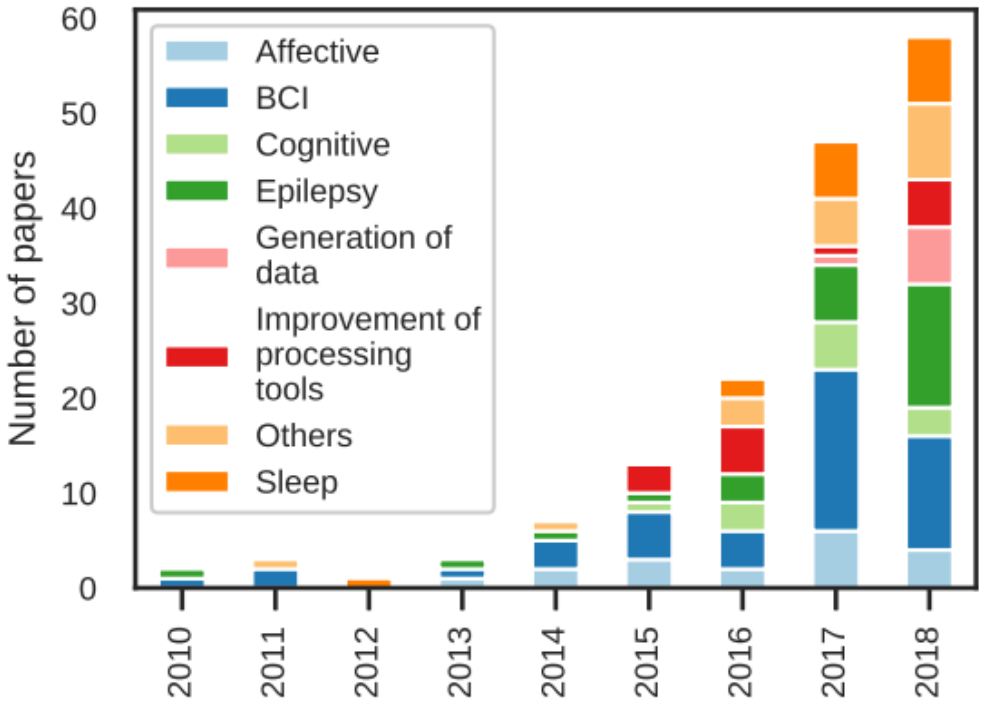


Figure 3.1: The increase of published papers in the recent years [43]

It also goes over most of the common problems, which we have also run into in this work. First of all regarding data imbalance, most papers with imbalanced data, such as epileptic seizure detection, or sleep stage transitions, used some form of class balancing, either by subsampling the majority class or by resampling the minority class on training.

For the DL architecture, the majority of the works used convolutional neural networks, with an overwhelming 41%, while other architecture types such as autoencoders (AE) or recurrent neural networks (RNN) are under 14%.

Regarding the amount of layers used, most architectures used from 2 to 7 layers, with few examples over 7 layers,

Regarding regularization, 76 papers used some form of regularization, such as L1 or L2, dropout, or early stopping. While the other 80 do not mention regularization.

Finally, most papers (30%) used the Adam method for optimization, while 17% used Stochastic Gradient Descent, and 6% used different optimizers.

This review gives us two big takeaways, first of all, it validates most of the choices used in

the architecture proposed in our research. Another takeaway from both reviews, is that while Deep Learning seems like a big promise for the field of EEG, the field is still in it's early steps, the Deep Networks do not perform nowhere as good on EEG signals as they perform on other applications such as computer vision or speech recognition. So it is still too early to say if Deep networks can perform as good as other learning algorithms, but if they can reach the same accuracy as they do on other applications, it can be a practical tool for EEG analysis.

4 Materials And Methods

4.1 Software and Hardware

The code ran on a HP Pavillion laptop with a Intel I7 @2.8GHz processor. The available RAM was 16Gb, although the software could run on much less RAM. A GPU was available, but the software did not run on a GPU.

The software for the signal segmentation, processing, and Signal drugging was written in python, using a modified version of the EEGWave repository from codeocean [39]. It uses the MNE library for segmenting the data and some operations with numpy for signal processing. The signal plotting is written in C++ using OpenCV, And the NN was run using Tensorflow's API for C++, based on Benny Friedman's article [18]. The full code is public and can be found at https://github.com/shipupi/BciSift/.

4.2 P300 Experiment

4.2.1 Experiment Description

The experiment was performed by Riccio et al., 2013[42]. where a group of eight individuals with confirmed ALS disease were tasked to spell 7 5-letter words (35 total letters) with a P300 speller.

The first three words were used for calibration/training, while the remaining four were used for testing with visual feedback.

Each time a letter was attempted to be spelled is called a Trial. Each trial contained 10 flashes in each of the 6 rows, and 10 flashes in each of the 6 columns. The flashes lasted a total of 0.125 seconds each, followed by an inter-flashing pause of the same duration. After each 120 flashes, an inter-trial pause was performed before moving to the next letter.

The experiment was performed with an 8 channel EEG, with electrodes placed at the Fz, Cz, Pz, Oz, P3, P4, PO7, and PO8 channels according to the 10/20 international system. The software used for the flashing was the BCI2000 (open source) [46]

The subjects were instructed to perform a copy-spelling task, which means that they have

to spell a predetermined set of words that was instructed beforehand.

4.2.2 Signal Processing

The first step is to segment the data, to do this, the MNE library was used to divide the signal into segments of 800ms. It requires the raw data to have an additional channel which marks the starts of each event that must be segmented, which was already provided by the original dataset. Each of these segments start the moment that the stimuli started, as the rows or columns of the matrix were lit. The 800ms duration allows for the P300 signal to be contained within the segment.

4.2.3 Dataset Structure

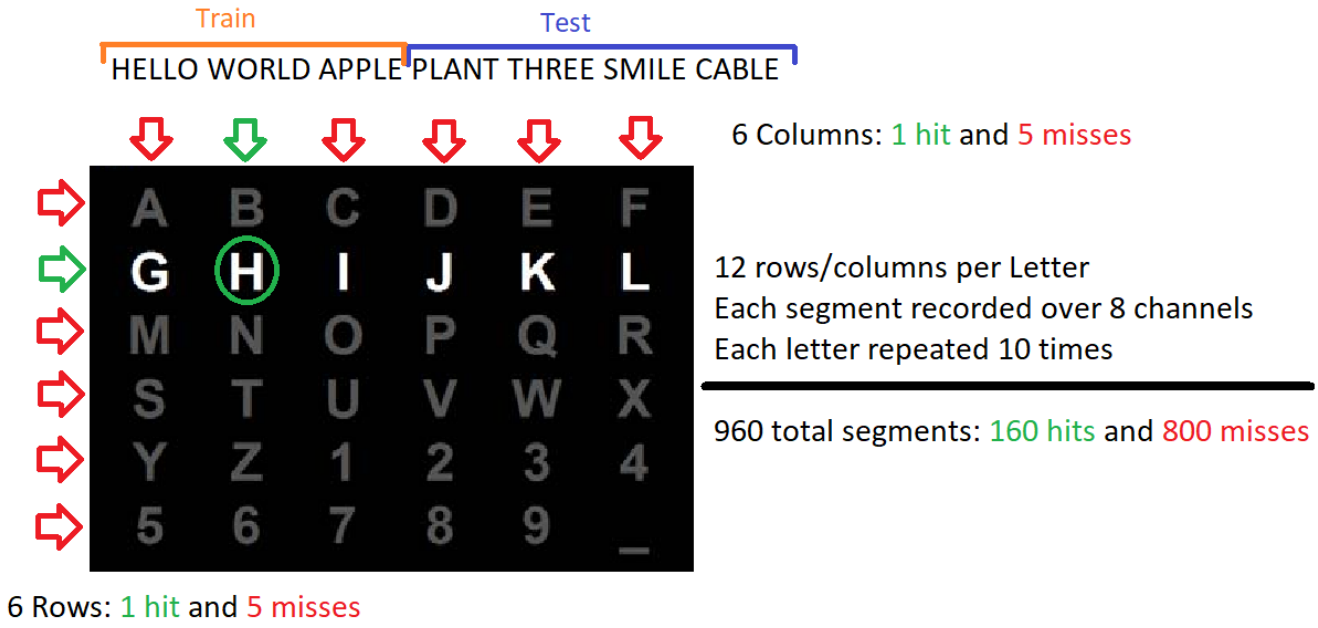


Figure 4.1: For each of the 35 letters, each row and column is highlighted 10 times, to allow for signal averaging. In each of these stimulus, if the column or row contains the target letter, we consider it a 'hit', while the columns and rows without the target letter are considered misses or 'nohits'. Since our EEG reads a total of 8 channels, the amount of total segments is calculated as 12 rows/columns x 10 repetitions x 8 channels giving a total of 960 segments per letter. Since only a row and a column contain a hit while the others contain misses, we have 160 segments with hits and 800 segments with nohits.

After processing and segmenting the raw data. We can calculate the amount of segments generated by a single letter as follows:

$$N_l = (n_r + n_c)rc \tag{4.1}$$

Where n_r and n_c are the number of rows and columns in the P300 matrix, c is the amount of channels i.e. the amount of electrodes on the EEG, and r is the amount of repetitions done per letter.

In the ALS experiment, the data was recorded using an 8-channel EEG, and 10 repetitions were performed per letter. The P300 matrix contained 6 rows and 6 columns, this results in a total of 960 segments per single letter. There is a single row and a single column with the P300 signal, and 5 rows and 5 columns without it. This results in each letter being segmented into 160 hits, and 800 nohits.

The experiment was performed using 7 5-letter words, resulting in 35 letters, and a total of 33600 segments. Out of these, 15 letters were used for the training of the networks, and the remaining 20 were used to test the accuracy.

4.2.4 Signal Averaging

As explained on section 1, the EEG signal is composed by many different components, while our main interest was to detect the P300 signal, there are other signals picked up on the EEG that are considered to be noise from sources such as artifacts of basal EEG activity. The relation between our signal of interest, the P300, and the rest of the signals is called signal-to-noise (SNR). When the SNR is low, the target signal is harder to detect, since it is obfuscated by the noise.

The proposed solution for this problem is called *signal averaging*. This technique can be applied for time-locked signals where the timing of the signal is known, the noise and signal are not correlated, the signal is consistent if the experiment is performed multiple times and the noise is truly random with zero-mean[50].

Assuming these conditions, our raw signal can be described as:

$$x_i(t) = n_i(t) + s_i(t) \tag{4.2}$$

Where x_i is the signal read from the EEG on the ith repetition of the trial. These signal is

composed by a noise component n_i and a timed locked signal component s_i. If we repeat the same experiment multiple times and average them, the resulting signal would be:

$$X(t) = \frac{1}{N}\sum_{i}^{N}(n_i(t) + s_i(t)) \tag{4.3}$$

Since the noise can be considered random with a zero-mean, and the time locked signal has a similar pattern throughout the segment, we can approximate:

$$X(t) = 0 + S(t) \tag{4.4}$$

Leaving just $S(t)$ which is just the average of the time locked signal. In this case, we've removed the noise and have a clearer signal. It should be noted that this equation can only be considered valid once N is big enough, since we need an infinite number of repetitions to truly eliminate the noise, as the amount of repetitions increase, the clearer the signal will be.

It should be noted that when averaging the signals, the variance of the averaged set is lower than a single set X of size n which has a variance of σ^2, this can be proved by:

$$Var(\frac{\sum_{i=1}^{n} X_i}{n}) = \frac{1}{n^2}Var(X_i) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n} < \sigma^2 \tag{4.5}$$

The downside of signal averaging is that repetitions of the experiment take more time, thus slowing down the Information transfer rate (**ITR**). ITR can be measured in Bits Per Second. Out-of-the-box BCI systems could get very low transfer rates, of around just a few 5 bits per minute or 0.08 bps [52] More sophisticated systems, tailored for specific persons using brainwaves in non-invasive SSVEP Speller can get 13 bps[41]. BCI systems have a wide inter-subject variability so there are papers reporting faster rates but they may be hard to generalize. At the same time it is tough to reach the 13 bps value for everyone and for the same person in different settings and across sessions. Invasive systems with implanted electrodes are faster, reaching 15 bps or even more. You need a very high bandwidth to control all the DOF of an arm prosthesis[4]. For reference, Keyboard typing is 16-20 bps (200 words per minute) while Speaking is 39 bps [17]. Facebook had a neural system project which aimed to achieve 100 wpm (8 bps). Neuralink, on the other hand,

aims for 40 wpm (3 bps)[51]

4.2.5 Multichannel classification

The EEG used in the ALS dataset was a 8-channel EEG for the Fz, Cz, Pz, Oz, P3, P4, PO7 and PO8 channels according to the 10-20 international system of electrode placement. In principle, each of the channels can be used separately to train the NN, and it is possible to determine the most accurate channel for each subject.

Alternatively, we tested the hypothesis that the combination of all 8 channels would yield higher accuracy than the best individual channel. To test this we tried a Majority system; The NN was trained separately on each of the 8 channels. When predicting new letters, each channel cast a vote for the selected row and column. The column/row with the most votes were selected. A random one was selected in case of a tie.[3]

4.2.6 Training scheme

To compare the results with *Riccio et al* and *Ramele et al.* The NN was trained with 15 out of the 35 letters, leaving the other 20 letters for testing. The metric used to compare the results is the letter accuracy, which means how many of the 20 letters were correctly predicted.

4.2.7 Data imbalance

Our dataset was composed of 35 letters, repeated 10 times, and in 8 different channels, as shown on figure 4.1. This means we had 960 segments per letter, and a total of 33600 segments (960 x 35).

Out of the 960 per letter, 160 of the segments contained the P300 signal, while the remaining 800 did not, Resulting in a total of 5600 hits, and 28000 nohits.

This means that by the nature of the experiment the data was not balanced, this is a problem because imbalanced datasets are not good for learning algorithms such as NN that use the gradient descent method [32].

To address this issue, once the 33600 segments were plotted into images, we replicated the 5600 hits a total of 4 times. Leaving us with 5600 x 5 = 28000 hits, thus equalling the

amount of hits and nohits. This technique is similar to a method of data balancing called resampling, but resulted in an easier implementation, albeit requiring higher disk space.

4.3 Signal plotting

The signal segments used as inputs for the NN are segments of 800ms sampled at a frequency of 256Hz, resulting in a total of 206 datapoints. The first step was z-score normalization

$$Z = \frac{x - \mu}{\sigma} \tag{4.6}$$

Then the height and width of the image were determined by

$$height = 2\gamma_h(max(s) - min(s)) \tag{4.7}$$

$$width = \gamma_w len(s) \tag{4.8}$$

$$zerolevel = \frac{height}{2} \tag{4.9}$$

This generates an image where the plot fills the whole width, and leaves 25% of air on the top and bottom. γ_w and γ_h are scaling parameters which were set at 2 and 30 respectively, and zerolevel is the vertical center of the image.

(note: as the max and min of each signal is different, the height of the generated images varied image to image)

The plot was then made by drawing the dots at their newly-scaled positions (centered with zerolevel and scaled by γ_w and γ_h), and joined by Bresenham's algorithm[6].

Finally, a rescale was made using Bilinear interpolation to 150x150 pixels. This is due to the fact that CNN require that all inputs have the same dimension, as an mentioned earlier, the height of each image depended on the max and the min of the signal. This rescaling provides a standard size for all of the signals.

4.4 Synthetic Signals

The synthetic signals were used as the first step of the experimental protocol. They are not based of real EEG data, and were exclusively generated in order to establish the Neural Network structure and test the generated images as inputs.

4.4.1 Synthetic simple signals

The first signals used to train the NN were manually generated, they were made as simple as possible to establish that the NN was working, with the synthetic images. A 100% accuracy was expected for this first step, as the images were extremely simple and visibly easy to classify. The formula for the simple signals that contain the ERP is:

$$s(x) = \begin{cases} 1500, & \text{if } x = 20 | x = 188 \\ -1500, & \text{if } x = 103 \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

The rationale behind this formula is to exaggerate the ERP. The peak value placed at the middle of the segment simulates the evoked potential on real signals. Our synthetic signal segments contain 206 datapoints, so negative peak is placed at the middle of the segment, and two other positive peaks are placed on the roughly at the first and third quarters of the segment, these peaks make the segments that contain the ERP easier to classify for the NN.

(a) A simple synthetic signal without P300

(b) A simple synthetic signal with an artificial P300-like signal

Figure 4.2: Simple synthetic signals: The width of the plot is dependant on the duration of the segment, and the height is dependant on the minimum and maximum peaks it has so that there is 25% of "air" above the max and below the min

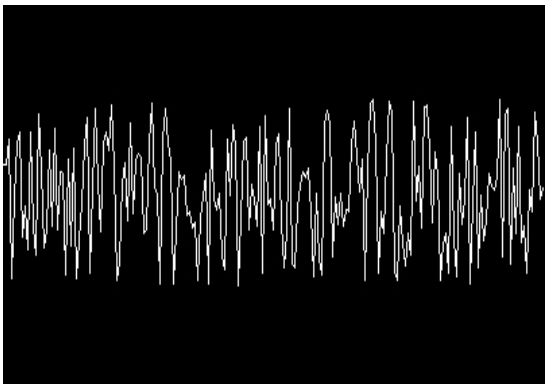
4.4.2 Synthetic noisy signals

The next step was to add some noise to the synthetic images, these new signals are closer to the real EEG signals, and were a safety step before moving on to pseudo-real data. The formula for the noisy signals is:

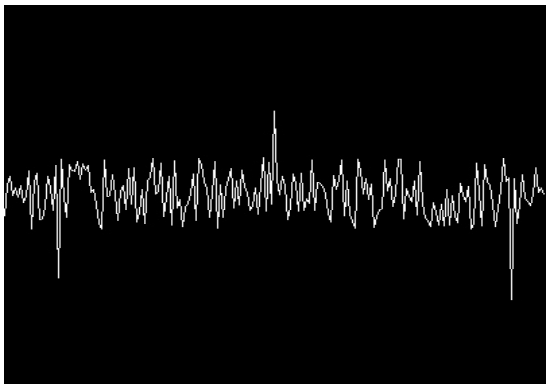
$$s(x) = \begin{cases} rand(1000, 1500), & \text{if } x = 20 | x = 220 \\ -rand(1000, 1500), & \text{if } x = 128 \\ rand(-500, 500), & \text{otherwise} \end{cases} \tag{4.11}$$

Where rand is a pseudo random number generator function with a uniform distribution, where the first parameter is the minimum value that it can take, and the second parameter is the maximum value. A 100% accuracy was expected on the noisy signals as well.

For noisy signals not containing the P300, the formula used is simply: $s(x) = rand(-500, 500)$



(a) A noisy synthetic signal without P300



(b) A noisy synthetic signal with an artificial P300-like signal

Figure 4.3: The noisy synthetic signals try to imitate a more real EEG segment, which is why a noise component is added, we also synthetically added 3 peaks with a higher amplitude than the rest of the noise that represent the P300 signal we want to find, while the amplitude of the noise components is the same for both images, the scaling is different because the height of the image is scaled depending on the min and max of the signals.

4.5 Drugged signal

4.5.1 Generation

Before testing on the ALS dataset, a previous step was taken by training with a pseudo-real dataset. The generation of this dataset is detailed in [39] and the code is available in the online repository of the Code Ocean platform under the name EEGWave, but in summary, it is a basal EEG stream that had a P300 template injected. This template was obtained from the subject 8 of the ALS dataset, by averaging all of the segments with the P300 signal, cancelling out the basal EEG noise and leaving the P300 segment. This template was multiplied by a boost β. So that the resulting drugged signal was

$$S_\beta = S + \beta T \tag{4.12}$$

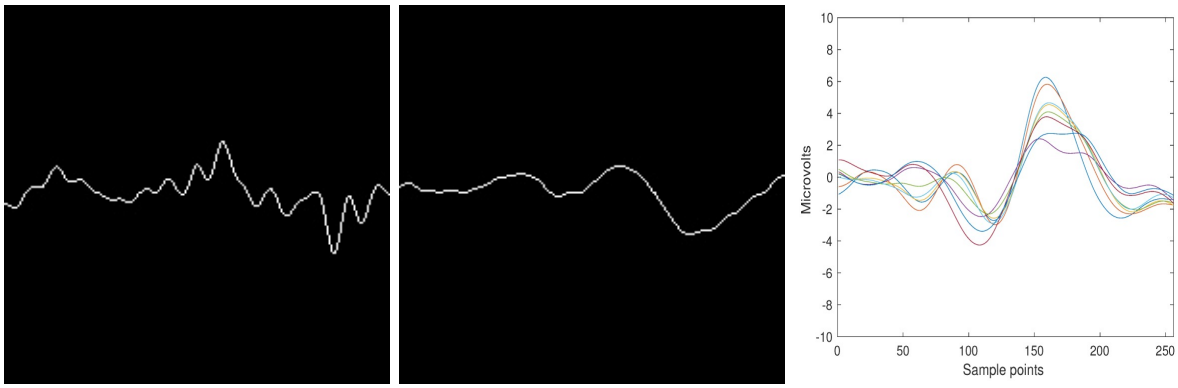
Where T is the P300 template, and S is a Basal EEG signal without any P300 stimuli. When β was big enough, the P300 template dominated the basal signal, and became easier to classify. This step was essential to test the pipeline from a raw signal to a fully plotted training dataset. With a high enough β, the accuracy was expected to be 100%, as beta decreased, it started to resemble a real signal, and the accuracy was expected to decrease.

4.5.2 Experimental protocol

The drugged signals were generated with the same structure as the ALS signals, this proved useful as it helped develop a pipeline from raw signal to a structured dataset.

We generated different sets of drugged signals with boost levels of 0, 1, 3, 5, 10 and 30. These sets contained 35 letters on the same 8 channels of the ALS public dataset, and with 10 repetitions each to allow for signal averaging.

In figure 4.4 we can see the drugged signals for β values of 5 and 30, and see how as the beta increases, the signals become more and more similar to the P300 template



(a) Drugged signal with $\beta = 5$

(b) Drugged signal with $\beta = 30$

(c) P300 template extracted from [39]

Figure 4.4: We can see that a β value of 5 makes the resulting signal more similar to the template, albeit with some noise, while the signal with a *beta* value of 30 totally eliminates the noise and is very similar to the P300 template

4.6 P300 Plot classification based on NN

The objective of this work was to use Deep Learning techniques to classify and detect P300 ERP in EEG plots, so different neural network architectures were implemented. The first architecture was based on an established and recognized Deep Network, and subsequent versions were improvements made to address the problems that arose in the first version.

4.7 VGG16 Neural Network

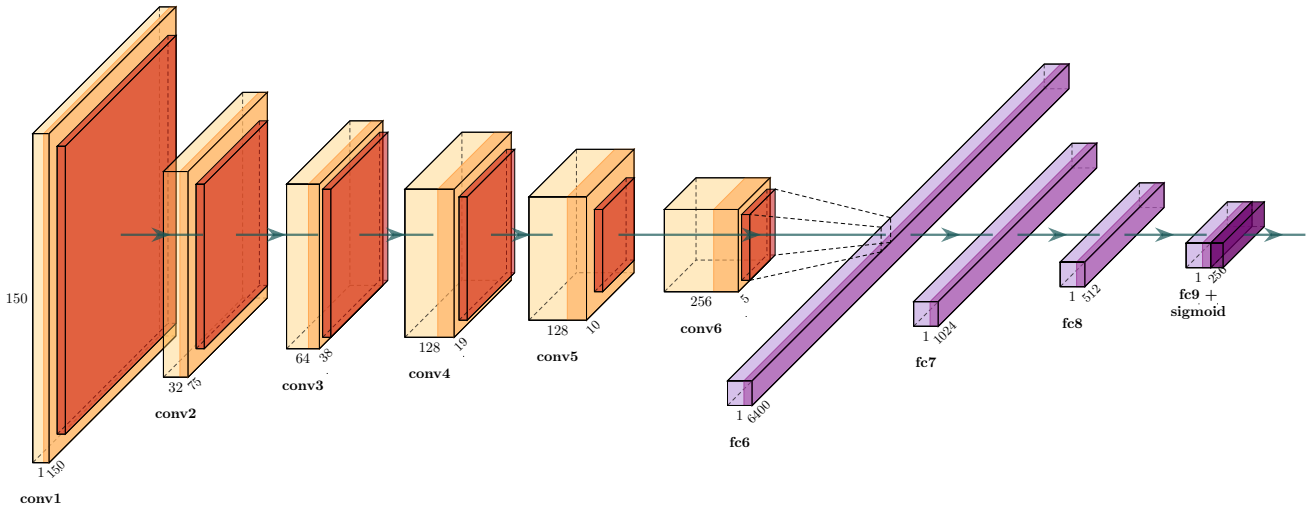


Figure 4.5: First version of the NN, a set of 5 convolutional layers is followed by 4 fully connected layer, and finally activated using a sigmoid function

The first version of the neural network was based on VGG16, a deep network which was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVR) competition in 2014[24]. It uses a 3x3 filter with a stride of 1 and uses padding to keep the same spatial dimensions, followed by a MaxPool layer of 2x2 size and stride 2, and follows this arrangement of convolution and maxpool layers all the way throughout the whole architecture, for 5 convolutional + maxpool sets. In the end, the VGG16 has 2 fully connected (regular neural network) layers, while we are using 4. In the end, the last layer is activated with a sigmoid function to make the binary classification. The advantage of the VGG16 approach, is that it reduces the amount of hyperparameters that must be

tinkered with, as all the filters and strides are kept constant.

Our network has 6 convolutional layers with a depth of 1, 32, 64, 128, 128 and 256. The stride in the maxpool reduces the spatial size in each iteration, leaving a spatial size of 150, 75, 38, 19, 10, 5 (the images are squared, so width and height are the same).

After the convolutional layers, a dropout layer with a drop rate of 0.5, and a flatten layer to make the data ready for the dense layers. The 4 dense layers have a size of 6400, 1024, 512 and 256 units, with the last one having a sigmoid activation.

The learning algorithm was the stochastic gradient descent method called Adam[25], and the loss function was the means squared difference. And it was done in batches of size 20.

The learning rate used was $5\,10^{-4}$, which deviates from the recommended $3\,10^{-3}$. This parameter was tinkered with for a bit, and was the suspect of the Network not convering at first, so decreasing the learning rate helped a bit.

Ultimately, the convergence issue was solved by balancing the dataset, first by prunning some of the misses in, and in later versions by duplicating the hits, as this latter method allowed the use of all the misses.

After training with the first 15 letters, the network was used to predict the remaining 20 letters, and the performance of the network was calculated by finding the row and column with the highest predicted change of containing the P300 signal and comparing the predicted letter to the original expected letter from the experiment.

These predictions were done by training each of the EEG channels separately, and then finding the channel that best perform. As mentioned early, another prediction was used by making a non-weighted voting system by each of the channels, and selecting the row and column with the most votes.

Since we are running what is considered a 'BCI Simulation'. The training was performed only once, instead of multiple runs doing cross validation or calculating mean and standard deviation in the accuracies. This is because we want to simulate a real use case of a patient actually using our interface.

4.8 Small VGG16

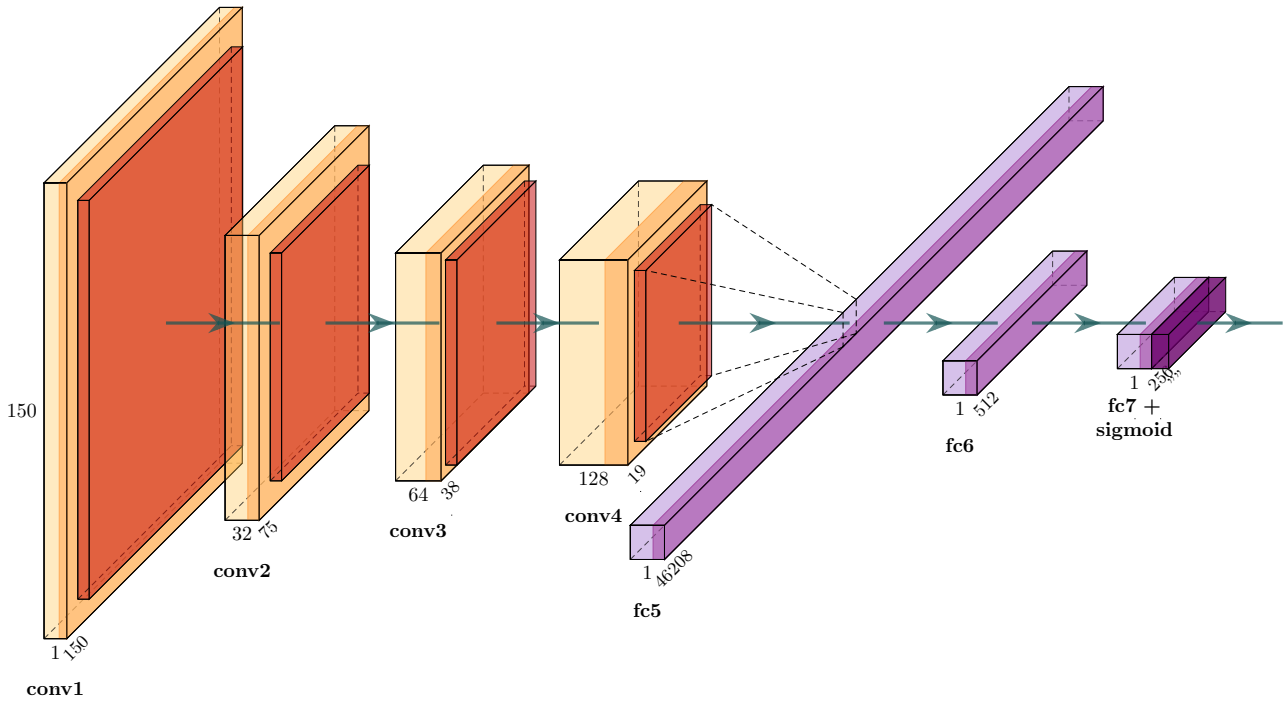


Figure 4.6: SV16 is similar to VGG16, but has two less convolutional layers and two less dense layers

The second approach was the Small VGG16 (SV16). The first change was to reduce the network size, so 2 convolutional layers and 2 fully connected layers were removed. Leaving the convolutional layers depth at 1, 32, 64 and 128, with spatial sides of 150, 75, 38, and 19. The flatten and dropout layers are left untouched, and then finally 3 fully connected layers of sizes 46208, 512 and 256. The overall philosophy of the architecture was left untouched, i.e. the padding, filter size and stride.

Another addition in this version was early stopping at 7 epochs to help decrease overfitting.

4.9 Multichannel Small VGG16 (MSV16)

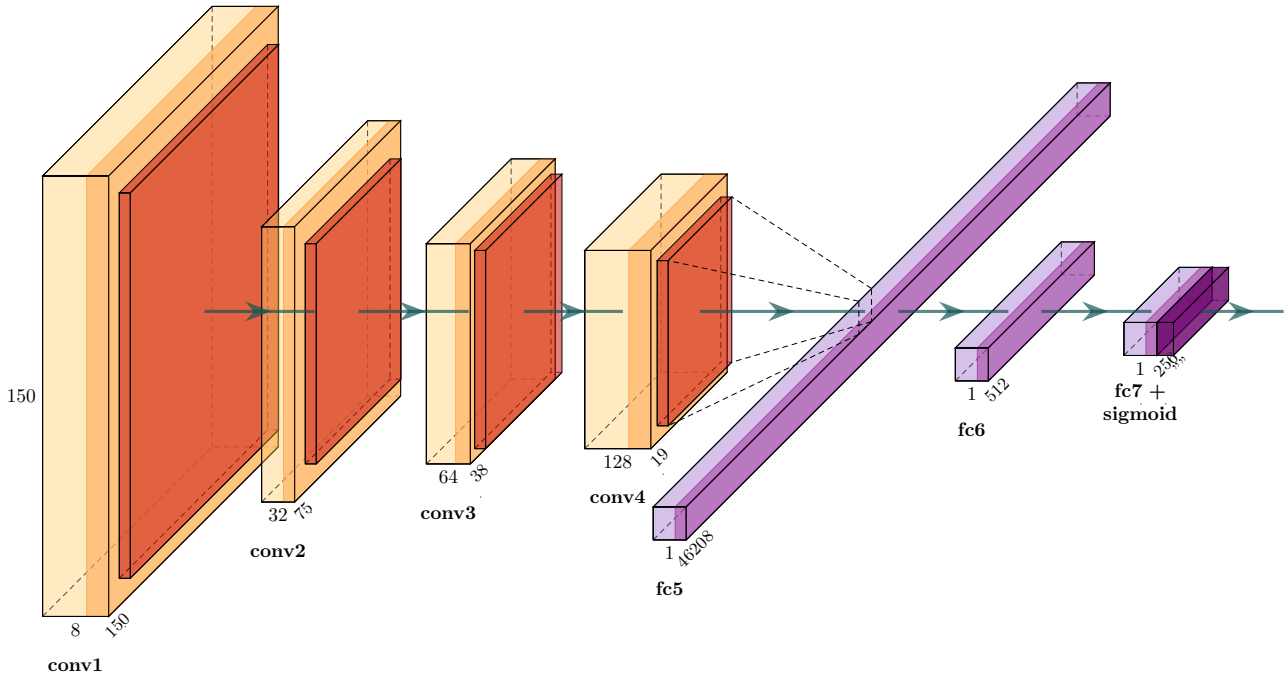


Figure 4.7: MSV16 has the same architecture as the second one, but the input layer is modified for an 8-channel input

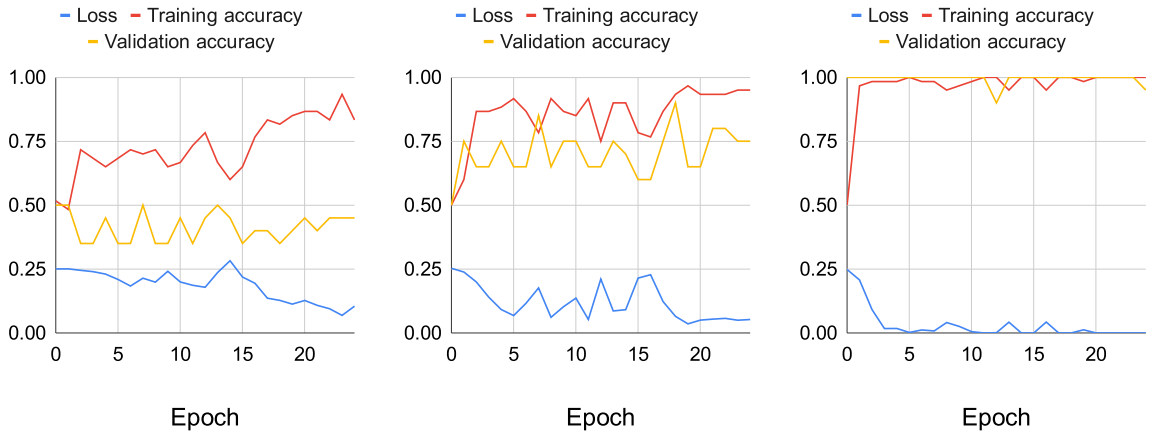
With this multichannel network, instead of training all of the EEG channels separately, they were merged into a single 8-channel image on the preprocessing stage. And the network was trained with this image as input. This modified the input layer to have a depth of 8, but the subsequent layers were left intact. The best improvement was that it allowed for the combination of all the different EEG channels to work together, achieving a higher performance than each channel separately.

To make this change work however, extra changes had to be made to the network parameters. The batch size was reduced to 6, as having a greater batch size generated memory issues. The last change was reducing the learning rate to the recommended value of $3 \cdot 10^{-}3$.

5 Results

5.1 Drugged Signals

The VGG16 network was first used to classify the artificially generated drugged signals. As explained on the previous section, this served multiple purposes. The first one was to establish a working pipeline from raw signals to structured datasets, while the others involved reaching consistency in the network in a controlled environment, where we can adjust the 'ease of classification' of the data, and check that the network actually fails to classify the 'unclassifiable' sets i.e. boost level of 0, and that it reaches perfect accuracy on the easily classifiable sets i.e. boost level 30.



(a) Learning curve with a boost level of 0, the network can somewhat learn the data, but since it is just basal EEG, it cannot generalize and the validation accuracy remains on random levels (50%).

(b) Learning curve with a boost level of 1, we can see that the validation accuracy is no longer on random levels, and the network is beginning to generalize

(c) Learning curve with a boost level of 3, the validation accuracy is already close to 100%

Figure 5.1: Drugged signals' learning curves, the red line represents the training accuracy, the yellow line is the validation accuracy and the blue line is the loss, which is the means squared difference of the errors in each iteration. These trainings were done at an intensification level of 10

We can see that this is the case in figure 5.1a, where we trained the network with a drugged dataset with a 0x boost, which means that the P300 template is not injected

at all and it is purely a basal EEG signal. The training accuracy (seen in the red line) manages to increase, meaning that the network successfully adapts to the training data. However, as the basal EEG signal has no underlying event with a specific waveform, it is basically pure noise, and thus the network cannot generalize it, and the accuracy levels for the validation set (yellow line) remain at a purely random level (50%).

Then we can see the learning curve with a boost level of 1, this boost level means that the generated signal is very similar to the signal of subject 8 of the ALS trial, as subject 8 was the best performing subject in all of the previous papers which classified this dataset (Ramele et al.[40], Riccio et al. [42]). We can see in figure 5.1b that the validation accuracy is much higher, this means that while not reaching perfect levels, the network can generalize this data.

When increasing the boost level to 3, we can see in figure 5.1c that both the training and the validation accuracy are almost at 100%, this means that the network can perfectly generalize this data. Subsequent curves for boost levels 5, 10 and 30 are not shown, since already at a boost level of 3 the network reaches perfects levels.

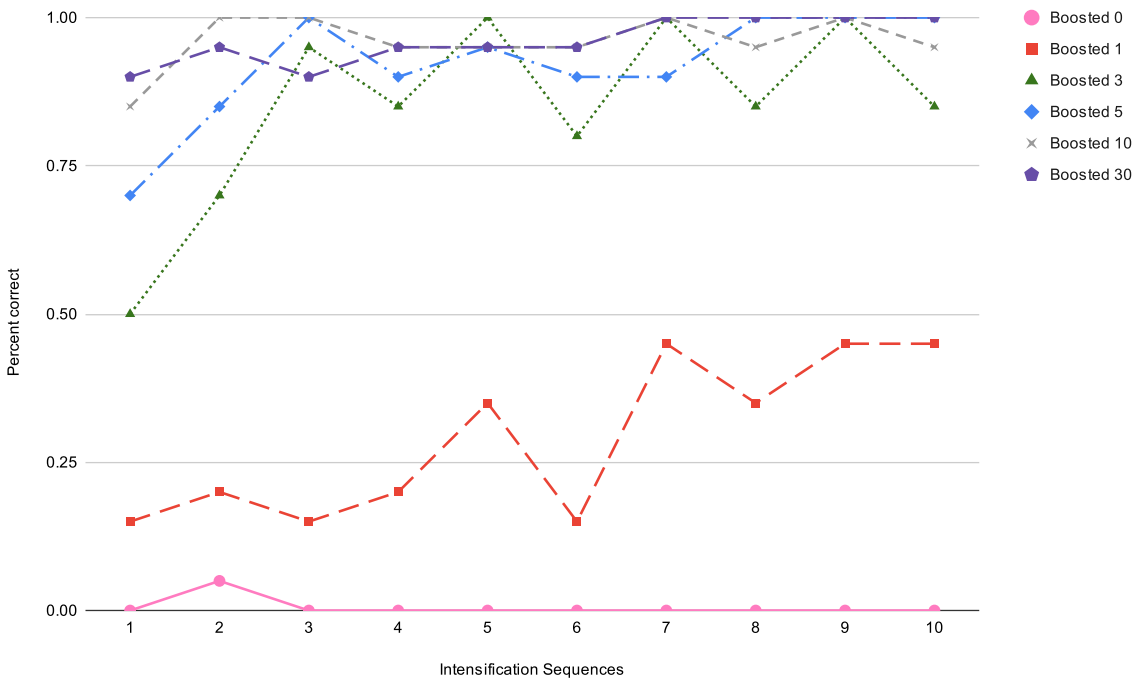


Figure 5.2: Accuracy levels for the different boosted signals at an increasing intensification level. We can see that the networks perform better as the intensification increases

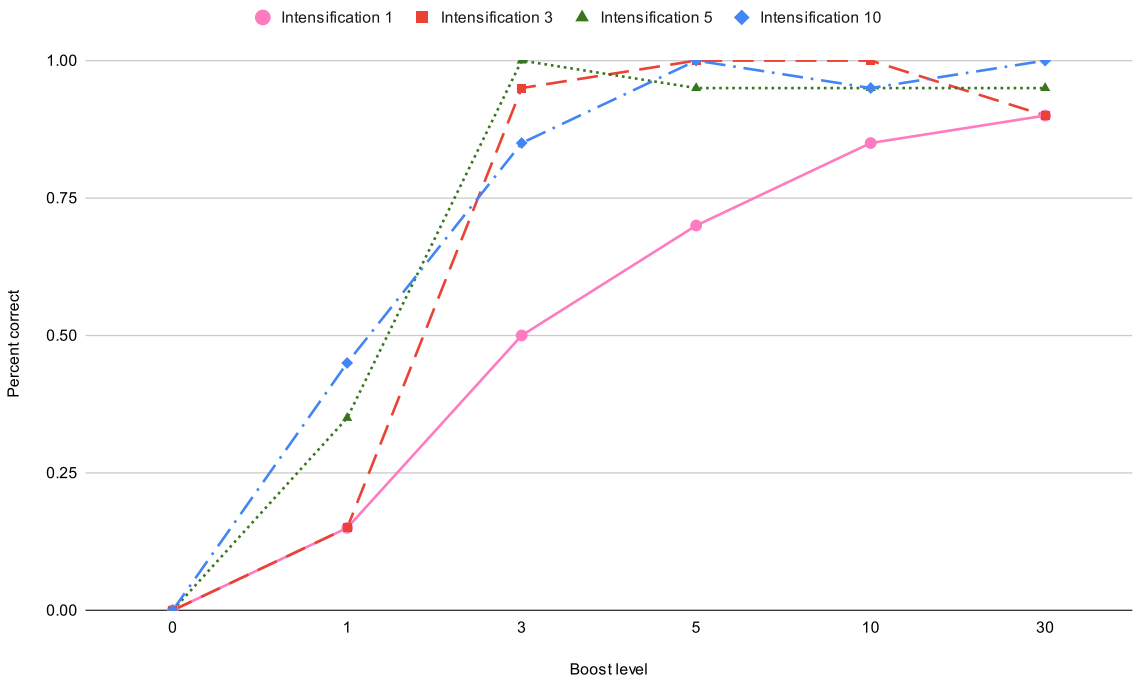


Figure 5.3: Letter accuracy shown as the boost level increases. As expected, as the boost level rises, the accuracy rises since the data is easier to classify

In figure 5.2 we can see how different boost levels performs as we intensify the signal, by averaging more repetitions of the experiment. This is the first evidence that the intensification of signals worked as expected, as we increased the intensification, all of the boost levels increased their performance, since the underlying basal EEG noise is being cancelled with signal averaging. The boost level 0 however, was unaffected by this, as the signal was entirely composed of basal EEG noise, in this signal the accuracy remains at or close to 0%, which is the random levels for letter accuracy (1/36).

In a very similar fashion, figure 5.3 shows that as the boost level increases, the accuracy increases as well, as discusses, this is due to the fact that the signals become easier to classify, since they are more dominated by the P300 signal and the noise decreases.

5.2 VGG16

Subject	bpc	VGG16(%)	majority voting	bpc	HIST(%)
1	Cz	15	5	Cz	35
2	Fz	70	50	Fz	85
3	Oz	30	25	Cz	25
4	Oz	30	15	PO8	55
5	Oz	35	40	PO7	40
6	Oz	45	40	PO7	60
7	Fz	70	50	PO8	80
8	PO8	90	80	PO7	95

Table 5.1: Character recognition rates for VGG16 network, on it's best performing channel (bpc) at an intensification level of 10. Compared to it's majority voting (mv) rate, as well as the HIST accuracy with it's own bpc[40]

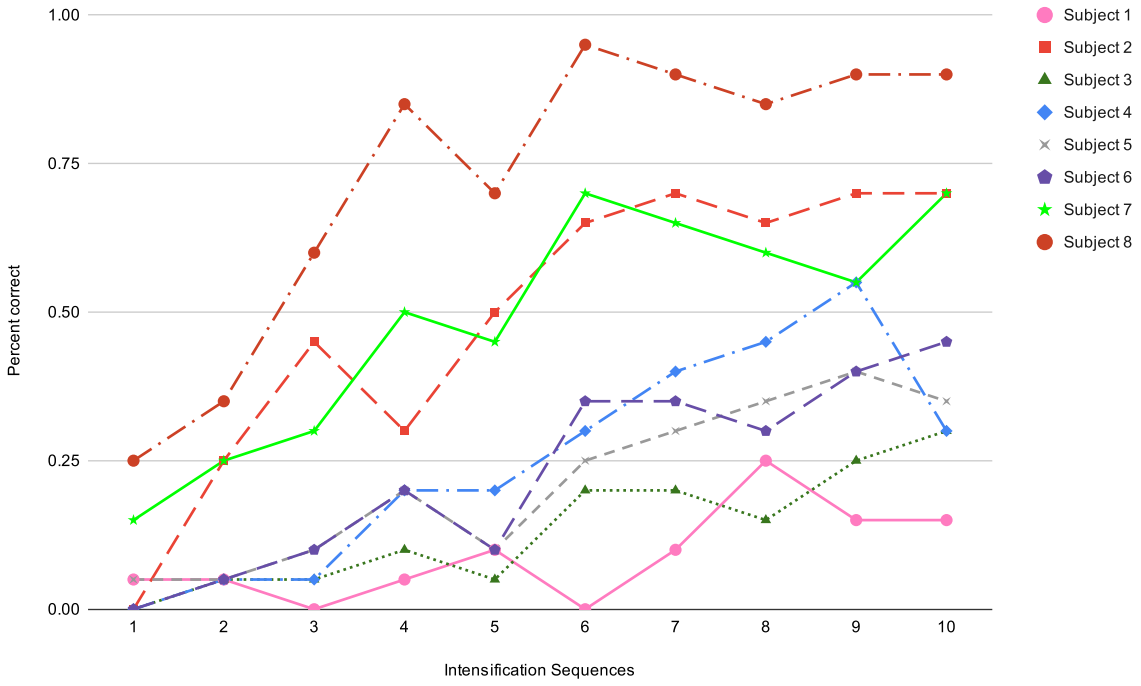


Figure 5.4: Accuracy curve for the participants with an increasing intensification level

Figure 5.4 shows the letter accuracy using the VGG16 NN, as explained on 4.2.4 we want the highest possible ITR, in order to do this, we must achieve a high letter accuracy on low intensification levels, since the higher the intensification level, the lower the ITR. Table 5.1 shows the accuracy of this version compared against the results from [40], where the Histogram of Gradient Descent method of features extraction was used to classify

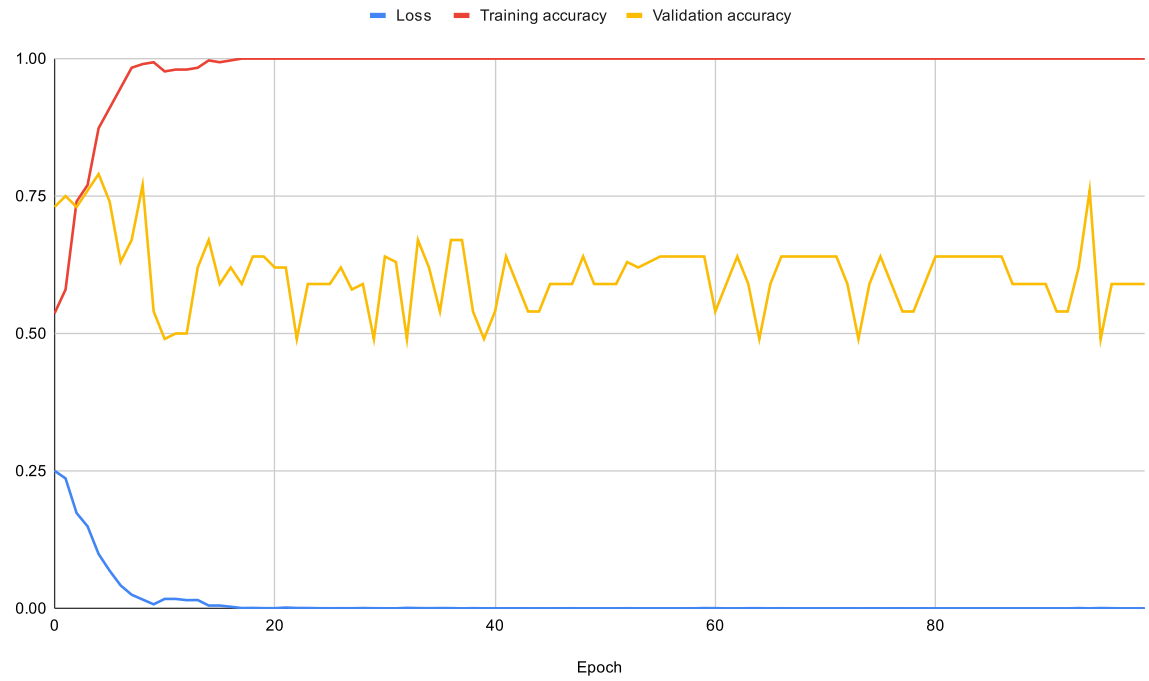


Figure 5.5: Learning curve of subject 5, with an intensification level of 4 on the PO8 channel

the signals. The VGG16 is less accurate than the HIST method, but the results are comparable, and they are even better for subject 3.

It should also be noted that while the results perform worse than the HIST method, they are consistent, the subject 8 performs with over 90% accuracy in both, and most of the other subjects have comparable values. This is an indication that while the network might still need improvements, the overall code and pipeline appear to be working correctly and bug-free

The table also shows the classification using majority voting, where the classification of each channel casts a vote for the selected row and column of the speller matrix. It can be observed that in all of the subjects, the majority voting method performs worse than the best performing channel. It was expected that the combination of the different channels would translate to a higher accuracy, but majority voting did not seem to make an improvement over the best performing channel.

5.3 SV16

Subject	bpc	VGG16(%)	SV16 (%)
1	Cz	15	10
2	Fz	70	50
3	Oz	30	30
4	Oz	30	40
5	Oz	35	50
6	Oz	45	40
7	Fz	70	65
8	PO8	90	95

Table 5.2: Character recognition rates for SV16, the PO8 channel at an intensification level of 10. Compared to the bpc in VGG16.

The first version of the CNN presented overfitting issues, as can be seen in 5.5. This was addressed in SV16, by pruning two convolutional layers and two dense layers of the network, and adding early stopping. 5.2 shows how SV16 performed on the same dataset in comparison to VGG16. This training was only performed on the PO8 channel, to save time without training with the whole dataset and look for small improvements. The second version performed better with subjects 8, 5 and 4, performed worse on subjects 1, 2, 6 and 7, and performed equally on subject 3. This does not appear to be a big improvement, but it has to be taken into consideration that only one channel was trained instead of taking the best performing channel, thus it was considered a positive outcome. However, further changes were made to the network before training on the full dataset.

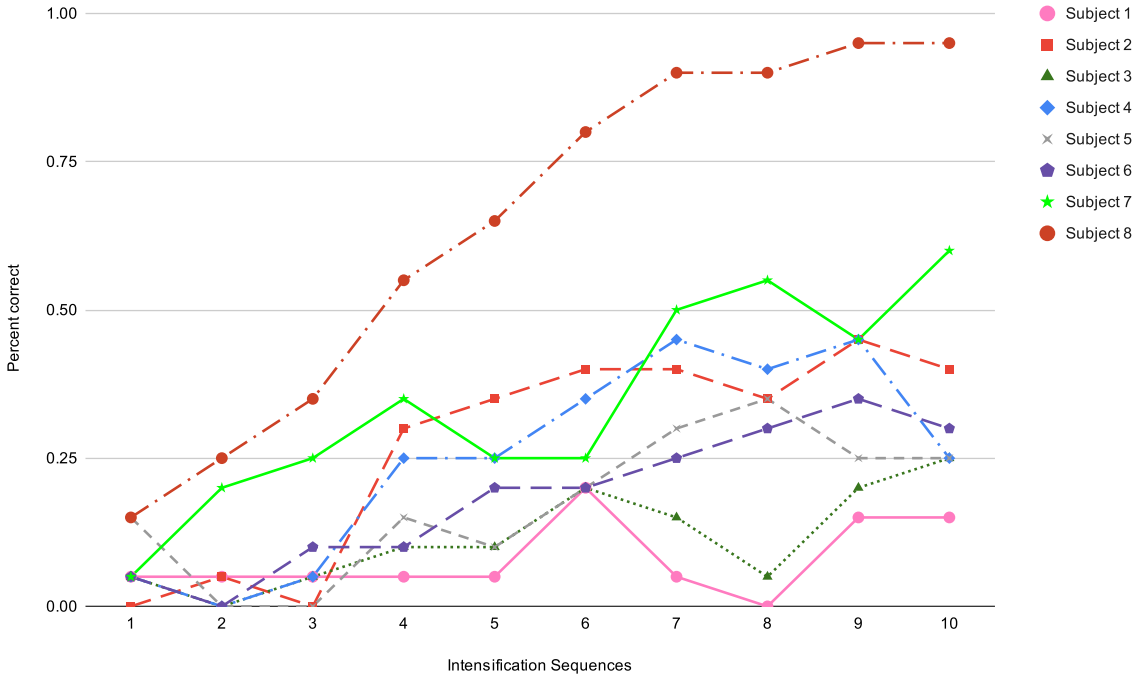


Figure 5.6: Letter accuracy for SV16 does not show significant improvements over VGG16, albeit these results are only for channel PO8 and not for the full dataset.

5.4 MSV16

Subject	VGG16(%)	SV16 (%)	MSV16(%)	HIST(%)
1	15	10	0	35
2	70	50	75	85
3	30	30	40	25
4	30	40	30	55
5	35	50	50	40
6	45	40	50	60
7	70	65	80	80
8	90	95	100	95

Table 5.3: Character recognition rates for VGG16, SV16, MSV16 and HIST

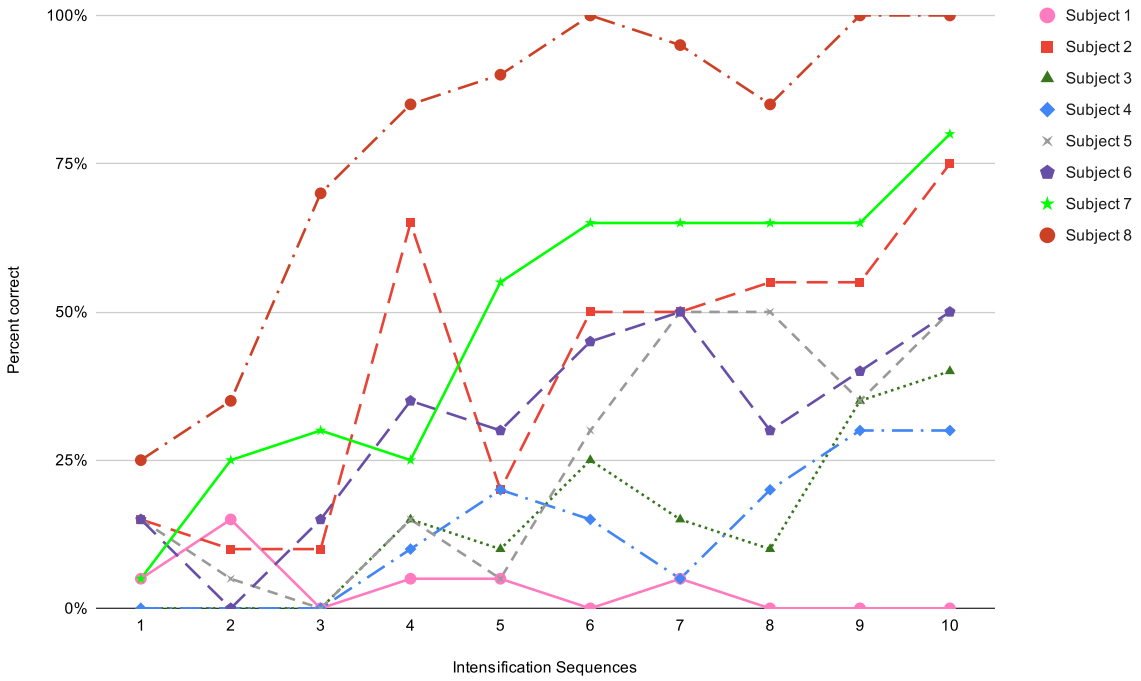


Figure 5.7: Letter accuracy for MSV16 shows a significant improvement over SV16

The third and final version of the network included both the improvements made in SV16, as well as the addition of multi-channel classification, and a smaller batch size. The results can be seen in 5.3. This last version showed an accuracy increase in 6 out of 8 channels over the other two versions. It surpasses HIST in 3 out of 8 subjects, and performs equally on 1. It does however seem to underperform on subject 1 and 4 compared to earlier versions and HIST. But it also reaches a 100% accuracy on subject 8, which none of the other methods had managed to achieve. Also, by looking at the accuracy per intensification level, subject 8 achieved over 90% accuracy in only 5 repetitions, meaning that a robust speller could be established with a much faster transmission rate.

The learning curve for subject 5 at an intensification level of 4 can be seen in 5.8. If compared to the same curve in the first network 5.5, the accuracy for the validation set is much higher, while the accuracy of the training set remains at or close to 100%. This suggests that there is still some overfitting left to solve, which would indicate that this method still has potential to find even higher success.

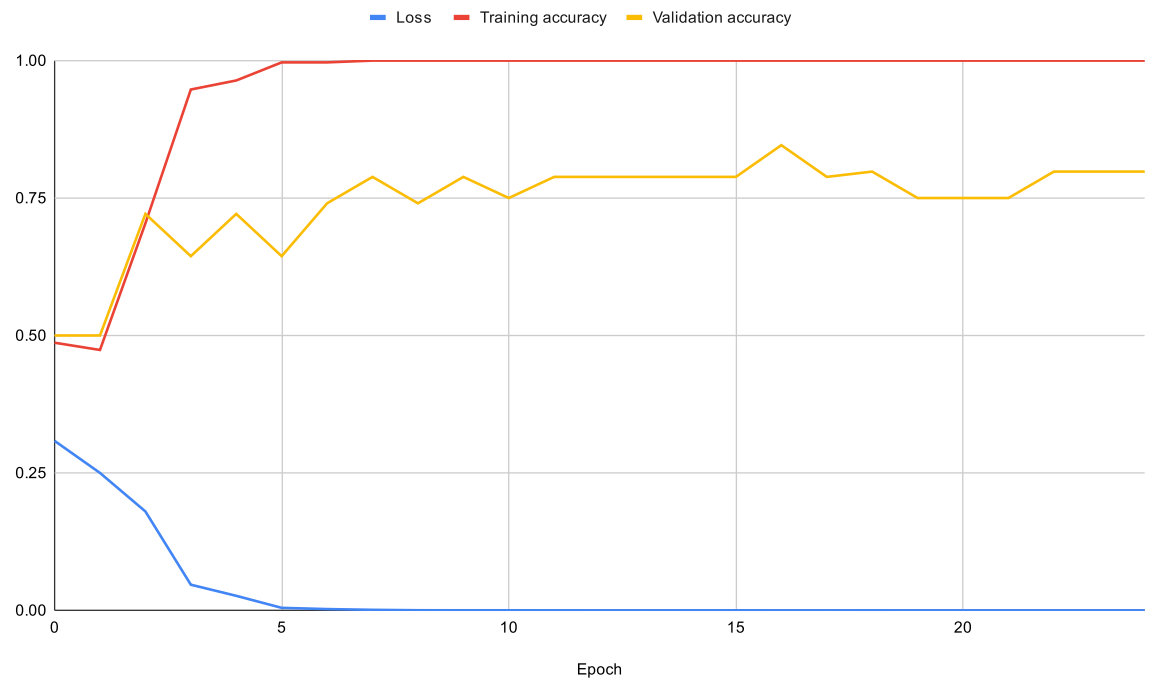


Figure 5.8: Learning curve of subject 5, with an intensification level of 4 using multichannel training

5.5 Discussion

5.5.1 Results validation

While there is no fail-proof way of assuring that the code is running bug-free, certain automatic and manual fail safes have been included to make the assertion that the results are valid.

When generating the training data, only the 15 letters are plotted. The remaining 20 letters are completely ignored, there is no possibility that the training used any of the testing set.

Generating the testing set follows a similar process, as the plotting is only performed on the last 20 letters, and the plots for the first 15 letters are deleted from the hard drive.

When averaging the signals, the standard deviation of the resulting signals is calculated, and it is asserted that the new standard deviation is indeed lower than the original one.

To corroborate that the results were not random, a training of the dataset was performed

by randomizing the labels, the expected outcome of this is that the predictions for the letters are completely random. We can see this in figure 5.9, the accuracy of the training done with random labels on subject 8 with MSV16 is oscillating around the random threshold, which is 1/36, or 3%, while the same training without randomizing the labels on the training goes back up to the regular accuracies. This shows that there is indeed a generalization being performed by the network, and the results are not 'by chance'.

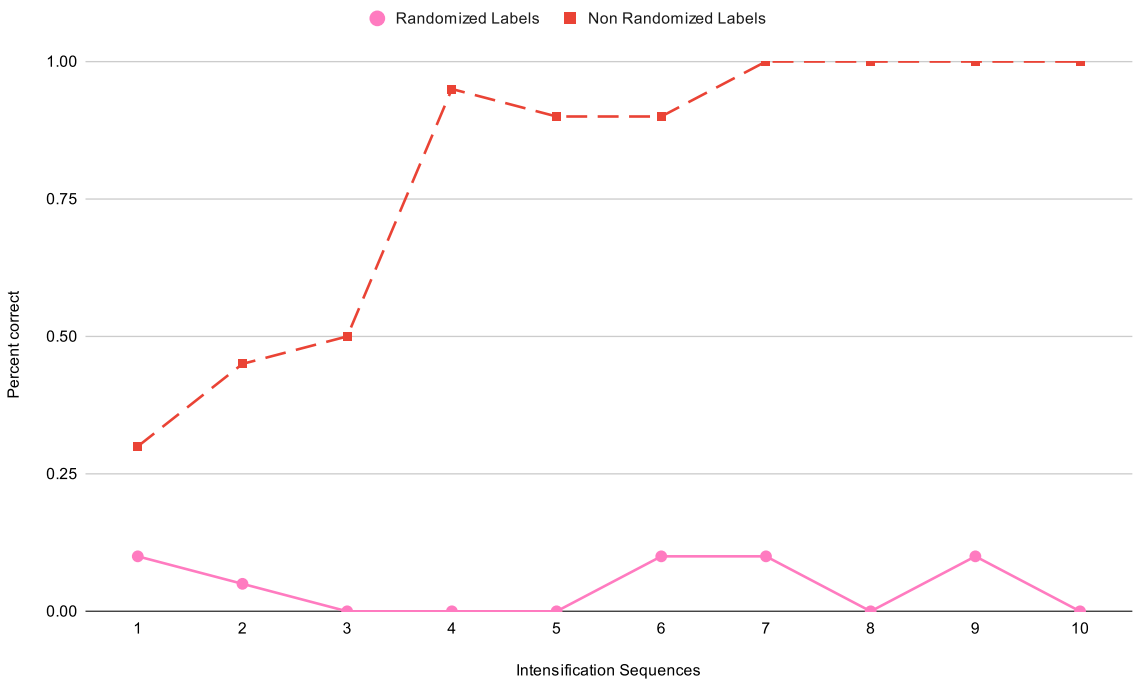


Figure 5.9: Accuracy of MSV16 on subject 8, the red line shows a normal prediction using regular labels on the training set, while the pink line shows the predictions of the NN by training it with randomized labels on the training set. We can see that the accuracy on the pink line hovers around the *random* range(3%)

The most telling assurance however, is that the results are consistent with previous works on the same dataset such as [40]. The accuracy curves for the different subjects, while not exactly identical, follow similar patterns, such as the subject 8 performing very well, while subject 1 performs badly. And the accuracy is consistently increasing along with the intensification level.

5.5.2 Drugged signals

When we generate a drugged dataset with a boost level of 0, we get a purely basal EEG signal, when encountering this dataset, the network should learn and converge, but the

accuracy on the testing set should perform very bad, close to random levels, as there is no pattern to generalize. This can be seen on figure 5.1a, while the training accuracy (red line) increases, the testing accuracy (yellow line) is stuck around 50%, which means that the prediction is completely random.

When moving on to a boost level of 1, the signal is very similar to the subject 8 of the ALS dataset, this gave us a preview of how the network behaved and learned on a real dataset. On figure 5.2 the boosted 1 curve (red line) performs pretty well, reaching almost 50% accuracy when averaging 10 signals (note: the threshold for random letter accuracy is $1/36 \approx 3\%$).

As the boost levels increased, the P300 became more dominant in the signal and the SNR decreased, making the images easier to classify, thus the expectation is that when the level of boost increases, the accuracy increases as well, which we can see on figure 5.3.

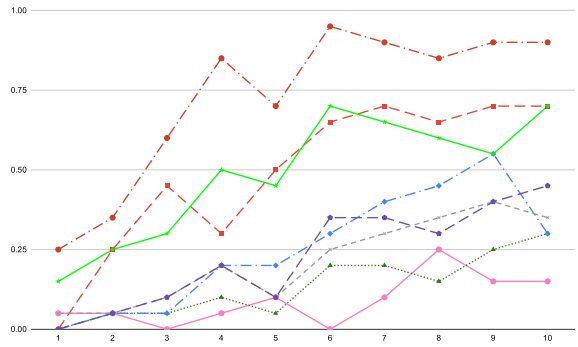
5.5.3 DL performance

Regarding the VGG16 architecture, we can see in figure 5.5 that the main issue is overfitting. The training sets are being learned perfectly, reaching a 100% accuracy, but the accuracy on the validation set does not increase, and oscillates around 60%, while this is a bit better than random level values, it still needs improvement. Figure 5.5 is just one of the many learning curves drawn for this version, but it showcases the overfitting problem.

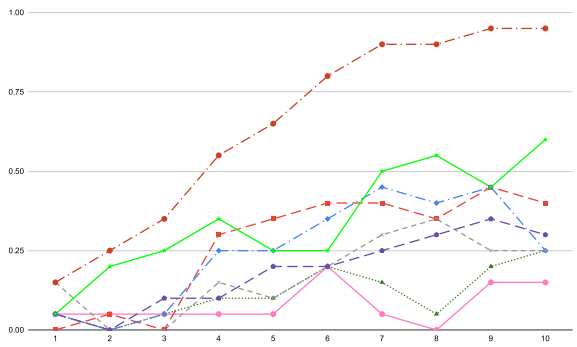
The SV16 version attempted to address this issue by adding early stopping and reducing the amount of layers, as shown on table 5.2, the improvement on letter accuracy from these changes seems small, as the results are barely better, but it should be taken into account that the results for VGG16 are taken with the best performing channel, while the results for SV16 are only taken from training on the PO8 channel, so even a similar result is a good indicator.

The biggest breakthrough of this work however is the MSV16 version, the inclusion of all the EEG channels into a single image, generated substantial improvements compared to both previous versions, and it even surpassed other proven methods such as SVM and HIST on some of the subjects, proving that using DL methods for this task is a very viable option. For some reason however, it performed very poorly on subject 1, although

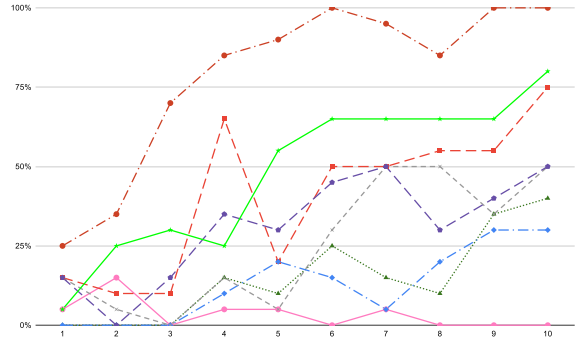
most of the DL methods did, while the HIST method can get a 35% accuracy on it.



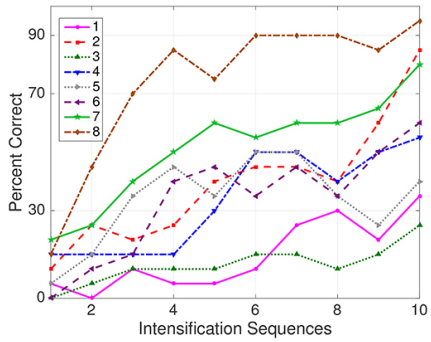
(a) Letter accuracy for VGG16



(b) Letter accuracy for SV16



(c) Letter accuracy for MSV16



(d) Letter accuracy for HIST method [40]

Figure 5.10: Side by side comparison of all the architectures, and finally a comparison to the HIST method, original figures can be found on section 5

6 Conclusion

The first conclusion can be drawn regarding the validation of our expectations. The drugged signals helped us validate that the data pipeline from a raw signal to a structured dataset was working as expected. And that the results were consistant, since as we increased the boost level, the accuracies improved substantially, and the same thing happened with the intensification levels, the increase of these two factors rose the accuracy in the predictions to 100%. Which confirms that this is the main issue that must be solved. The advantage of using the drugged signals first, is that solves the problem of null-signals (NS), these happen on non-synthetic datasets, when the subject is supposed to spell a letter but is not focusing enough or become distracted and the P300 is not triggered[40]. On these cases, the label of the segment would contain 'hit', but the actual signal would not contain the P300 signal. Synthetic methods such as signal drugging allow us to overcome this issues, by adding the signal ourselves, we are 100% sure that the ERP is present on all the segments with a 'hit' label.

It was the objective to determine whether deep neural networks could perform better than other classification methods for detecting the P300 signal from plots, such as HIST or SVM, since DL methods are very successful with dealing with visual information. Successfully detecting this signal could help improve the quality of life of advanced ALS patients and others with Locked-In syndrome. We have proved that deep networks have improved the accuracy in *some* patients over all other methods. This can potentially lead to further research in improving these deep networks for achieving an even higher improvement. Or an ensemble of different classifications methods, depending on the compatibility of the EEG signals of each person with each of the classification methods.

Furthermore, the addition of multiple EEG channels being merged into a single multi-channel image is a very interesting approach, and could hold the key to solving this and other classification problems with similar characteristics as the improvements made by this addition were significant.

Finally, different subjects have different accuracy rates by using different methods for predictions, some subjects seemed to perform better using the HIST method, while others performed better by using the CNN. It could be possible that an ensemble of networks is

made that is adjusted to each particular person, and find a classifier suitable for each one.

7 Future Work

As discussed on section 5, this method for classifying P300 signals has yielded very good results, and can be taken even further by tinkering with the Neural network architecture even more, trying several techniques for solving overfitting such as L1 or L2 regularization, data augmentation, or spending more time fine-tuning the parameters of the network such as the batch size, learning rate, and other meta parameters such as γ_w and γ_h, and whether to use or not z-score normalization for the plotting, or the halting cutoff.

If taken further, the multi-channel approach can be extended to more sophisticated EEGs with more than 8 channels, and will probably find an even higher success there, as evidenced by the increased accuracy when using all 8 channels of the EEG merged into a single image.

Different machine learning ensembles can also be tried, EEG signals are non stationary and have a huge inter-person variability, so for example we can set up a group of 10 learning algorithms ready so when a single subject attempts to use the speller, all 10 algorithms are trained in parallel, and according to the accuracies of each, use a different one depending on it's affinity with that particular subject. This can even be extended by having dynamically generated architectures that can adapt to each person, this has been briefly mentioned in [19], and it could fully unlock the potential of DL with EEG.

To fully accomplish this however, more data is needed. This experiment was only performed using a dataset of 10 patients and just 35 letters per patient. With the ease of access of EEGs, this can be substantially improved, and have thousands or millions of patients with hours worth of recording time, this can even lead to other interesting research such as clustering of different people with similar waveforms, allowing to maybe overcome the inter person variability problem.

8 Acknowledgements

First of all I would like to dedicate my thesis and my whole CS career to my girlfriend Celeste who provided me with continuous love, support, motivation and inspiration throughout these last five years, none of this would have been possible without you, I love you 3000 million infinites.

Secondly, I'd like to thank parents Gabriel and Roxana and my sister Karen for encouraging me in my pursuit of this career, despite the many,*many* setbacks that I had, never giving up on making this a reality.

To Apollo, for always providing company throughout these last years, bringing me tremendous amounts of luck and cheering me up whenever I'm sad. To Mushu, for keeping me focused on the work, giving me a clear goal and so much joy.

To my peers, Federico Bergagna, Manuel Rodriguez Brizi, Juan Baader, Lucio Pagni, Tomas Bacigalupo, Marina Fuster, Florencia Petrikovich, Gonzalo Hirsch, Lautaro Pinilla, Fernando Martin and so many others. For being such great classmates, helping me study and supporting me. Also to Federico Albanese, for helping me with some Machine Learning concepts throughout the thesis.

To my teachers, which there are too many too name, thank you for making me the professional that I am today.

And last but not least, to Rodrigo Ramele, for helping me achieve this work, always with an positive attitude even when I wasn't so optimistic. And for putting as much effort and passion on this project as I did.

References

[1] Rahib H. Abiyev et al. "Brain-Computer Interface for Control of Wheelchair Using Fuzzy Neural Networks". In: *BioMed Research International* 2016 (Sept. 2016), p. 9359868. ISSN: 2314-6133. DOI: 10.1155/2016/9359868. URL: https://doi.org/10.1155/2016/9359868.

[2] Charu C. Aggarwal. *Neural Networks and Deep Learning. A Textbook*. Cham: Springer, 2018, p. 497. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0.

[3] Amir Ahangi et al. "Multiple classifier system for EEG signal classification with application to brain–computer interfaces". In: *Neural Computing and Applications* 23 (Aug. 2012). DOI: 10.1007/s00521-012-1074-3.

[4] A Bolu Ajiboye et al. "Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration". In: *Lancet* 389.10081 (May 2017), pp. 1821–1830.

[5] Stevo Bozinovski and Liljana Bozinovska. "Brain–Computer Interface in Europe: the thirtieth anniversary". In: *Automatika* 60.1 (2019), pp. 36–47. DOI: 10.1080/00051144.2019.1570644. eprint: https://doi.org/10.1080/00051144.2019.1570644. URL: https://doi.org/10.1080/00051144.2019.1570644.

[6] J. E. Bresenham. "Algorithm for computer control of a digital plotter". In: *IBM Systems Journal* 4.1 (1965), pp. 25–30. DOI: 10.1147/sj.41.0025.

[7] R G Brotman et al. "Amyotrophic Lateral Sclerosis". In: *StatPearls. Treasure Island* (2021).

[8] via Wikimedia Commons Brylie Christopher Oxley CC0. *International 10-20 system for EEG-MCN*. 2017. URL: https://commons.wikimedia.org/wiki/File:International_10-20_system_for_EEG-MCN.svg.

[9] P. Burt and E. Adelson. "The Laplacian Pyramid as a Compact Image Code". In: *IEEE Transactions on Communications* 31.4 (1983), pp. 532–540. DOI: 10.1109/TCOM.1983.1095851.

[10] Junyi Chai et al. "Deep learning in computer vision: A critical review of emerging techniques and application scenarios". In: *Machine Learning with Applications* 6 (2021), p. 100134. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100134. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000670.

[11] Mike Chi. *Cognionics Dry EEG P300 Speller Demo*. 2015. URL: https://www.youtube.com/watch?v=XIr2cRKFolY.

[12] Pietro Cipresso et al. "The use of P300-based BCIs in amyotrophic lateral sclerosis: from augmentative and alternative communication to cognitive assessment". en. In: *Brain Behav.* 2.4 (July 2012), pp. 479–498.

[13] James L. Crowley and Richard M. Stern. "Fast Computation of the Difference of Low-Pass Transform". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.2 (1984), pp. 212–222. DOI: 10.1109/TPAMI.1984.4767504.

[14] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*. Tech. rep. MSR-TR-2014-21. Microsoft, May 2014. URL: https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/.

[15] L A Farwell and E Donchin. "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials". en. In: *Electroencephalogr Clin Neurophysiol* 70.6 (Dec. 1988), pp. 510–523.

[16] Faranak Farzan et al. "Standardization of electroencephalography for multi-site, multi-platform and multi-investigator studies: insights from the canadian biomarker

integration network in depression". In: *Scientific Reports* 7.1 (Aug. 2017), p. 7473. ISSN: 2045-2322. DOI: 10.1038/s41598-017-07613-x. URL: https://doi.org/10.1038/s41598-017-07613-x.

[17] Véronique Etienne François Pellegrino. *Similar information rates across languages, despite divergent speech rates*. 2019. URL: https://www.cnrs.fr/en/similar-information-rates-across-languages-despite-divergent-speech-rates.

[18] Benny Friedman. *Creating a TensorFlow CNN in C++*. 2019. URL: https://towardsdatascience.com/creating-a-tensorflow-cnn-in-c-part-2-eea0de9dcada.

[19] Shu Gong et al. "Deep Learning in EEG: Advance of the Last Ten-Year Critical Period". In: (Nov. 2020). DOI: 10.1109/TCDS.2021.3079712.

[20] Steve R. Gunn. "Support Vector Machines for Classification and Regression". In: 1998.

[21] David Gunning et al. "XAI—Explainable artificial intelligence". In: *Science Robotics* 4 (Dec. 2019), eaay7120. DOI: 10.1126/scirobotics.aay7120.

[22] Violaine Guy et al. "Brain computer interface with the P300 speller: Usability for disabled people with amyotrophic lateral sclerosis". In: *Annals of Physical and Rehabilitation Medicine* 61.1 (2018), pp. 5–11. ISSN: 1877-0657. DOI: https://doi.org/10.1016/j.rehab.2017.09.004. URL: https://www.sciencedirect.com/science/article/pii/S1877065717304104.

[23] David H. Hubel and Torsten N. Wiesel. "Receptive Fields of Single Neurons in the Cat's Striate Cortex". In: *Journal of Physiology* 148 (1959), pp. 574–591.

[24] *ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. https://www.image-net.org/challenges/LSVRC/2014/.

[25] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[26] Christian Klaes. "Chapter 28 - Invasive Brain-Computer Interfaces and Neural Recordings From Humans". In: *Handbook of in Vivo Neural Plasticity Techniques*. Ed. by Denise Manahan-Vaughan. Vol. 28. Handbook of Behavioral Neuroscience. Elsevier, 2018, pp. 527–539. DOI: https://doi.org/10.1016/B978-0-12-812028-6.00028-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128120286000288.

[27] Yang Li et al. "A Channel-Projection Mixed-Scale Convolutional Neural Network for Motor Imagery EEG Decoding". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27.6 (2019), pp. 1170–1180. DOI: 10.1109/TNSRE.2019.2915621.

[28] T. Lindeberg. "Scale Invariant Feature Transform". In: *Scholarpedia* 7.5 (2012). revision #153939, p. 10491. DOI: 10.4249/scholarpedia.10491.

[29] Mingfei Liu et al. "Deep learning based on Batch Normalization for P300 signal detection". In: *Neurocomputing* 275 (2018), pp. 288–297. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.08.039. URL: https://www.sciencedirect.com/science/article/pii/S0925231217314601.

[30] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[31] D Lulé et al. "Life can be worth living in locked-in syndrome". en. In: *Prog. Brain Res.* 177 (2009), pp. 339–351.

[32] Ran Manor and Amir B. Geva. "Convolutional Neural Network for Multi-Category Rapid Serial Visual Presentation BCI". In: *Frontiers in Computational Neuroscience*

9 (2015). ISSN: 1662-5188. DOI: 10.3389/fncom.2015.00146. URL: https://www.frontiersin.org/article/10.3389/fncom.2015.00146.

[33] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[34] Elon Musk. "An Integrated Brain-Machine Interface Platform With Thousands of Channels". In: *J Med Internet Res* 21.10 (Oct. 2019), e16194. ISSN: 1438-8871. DOI: 10.2196/16194. URL: https://doi.org/10.2196/16194.

[35] Shree Nayar. *SIFT Descriptor | SIFT Detector*. 2021. URL: https://www.youtube.com/watch?v=IBcsS8_gPzE.

[36] Neuralink. *Monkey MindPong*. 2021. URL: https://neuralink.com/blog/monkey-mindpong/.

[37] E. Niedermeyer and F.H.L. da Silva. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. LWW Doody's all reviewed collection. Lippincott Williams & Wilkins, 2005. ISBN: 9780781751261. URL: https://books.google.com.ar/books?id=tndqYGPHQdEC.

[38] Acary Oliveira and Roberto Pereira. "Amyotrophic lateral sclerosis (ALS): Three letters that change the people's life. For ever". In: *Arquivos de neuro-psiquiatria* 67 (Oct. 2009), pp. 750–82. DOI: 10.1590/S0004-282X2009000400040.

[39] Rodrigo Ramele, Ana Julia Villar, and Juan Miguel Santos. "EEG Waveform Analysis of P300 ERP with Applications to Brain Computer Interfaces". In: *Brain Sciences* 8 (2018).

[40] Rodrigo Ramele, Ana Julia Villar, and Juan Miguel Santos. "Histogram of gradient orientations of signal plots applied to P300 detection". In: *Frontiers in computational neuroscience* 13 (2019).

[41] Rajesh P. N. Rao. *Brain-Computer Interfacing: An Introduction*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139032803.

[42] Angela Riccio et al. "Attention and P300-based BCI performance in people with amyotrophic lateral sclerosis". In: *Frontiers in Human Neuroscience* 7 (2013). ISSN: 1662-5161. DOI: 10.3389/fnhum.2013.00732. URL: https://www.frontiersin.org/article/10.3389/fnhum.2013.00732.

[43] Yannick Roy et al. "Deep learning-based electroencephalography analysis: A systematic review". In: *Journal of Neural Engineering* 16 (May 2019). DOI: 10.1088/1741-2552/ab260c.

[44] Marc Saab. *DC-EEG in Psychophysiology Applications – A Technical and Clinical Overview*. 2009. URL: https://www.bmedreport.com/archives/3739.

[45] Carolina Saavedra and Laurent Bougrain. "Wavelet denoising for P300 single-trial detection". In: *Cinquième conférence plénière française de Neurosciences Computationnelles, "Neurocomp'10"* (Oct. 2010).

[46] Gerwin Schalk et al. "BCI2000: a general-purpose Brain-Computer Interface (BCI) system". In: *IEEE Trans. Biomed. Eng.* 51 (July 2004), pp. 1034–. DOI: 10.1109/TBME.2004.827072.

[47] Caitlin Shure. *What is EEG and what is it used for?* 2021. URL: https://www.bitbrain.com/blog/what-is-an-EEG.

[48] Nancy K Squires, Emanuel Donchin, and Kenneth C Squires. "Bisensory stimulation: Inferring decision-related processes from the P300 component". In: *J. Exp. Psychol. Hum. Percept. Perform.* 3.2 (1977), pp. 299–315.

[49] *Support Vector Machine Algorithm*. 2022. URL: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.

[50] Wim van Drongelen. "4 - Signal Averaging". In: *Signal Processing for Neuroscientists*. Ed. by Wim van Drongelen. Burlington: Academic Press, 2007, pp. 55–70. ISBN: 978-0-12-370867-0. DOI: https://doi.org/10.1016/B978-012370867-0/50004-8. URL: https://www.sciencedirect.com/science/article/pii/B9780123708670500048.

[51] Kyle Wiggers. *How Facebook's brain-machine interface measures up*. 2019. URL: https://venturebeat.com/2019/07/31/how-facebooks-brain-machine-interface-measures-up/.

[52] Jonathan Wolpaw and Elizabeth Winter Wolpaw, eds. *Brain-Computer Interfaces: Principles and Practice*. Oxford University Press, 2012.

[53] Jonathan R Wolpaw et al. "Brain–computer interfaces for communication and control". In: *Clinical Neurophysiology* 113.6 (2002), pp. 767–791. ISSN: 1388-2457. DOI: https://doi.org/10.1016/S1388-2457(02)00057-3. URL: https://www.sciencedirect.com/science/article/pii/S1388245702000573.