


Image Classification from EEG Brain Signals using Machine Learning and Deep Learning techniques

by
Pieter Johannes Uys



Thesis presented in partial fulfilment of the
requirements for the degree of Master of
Engineering (Mechatronic) in the faculty of
engineering at Stellenbosch University

Supervisor: Prof. D. Van den Heever

December 2019



Departement Meganiese en Megatroniese Ingenieurswese
Department of Mechanical and Mechatronic Engineering



DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 2019

Copyright © 2019 Stellenbosch University
All rights reserved

Abstract

Currently the dynamics and working of the brain are not fully understood. The detection of brain-wave patterns has been possible for decades, but the missing element was the ability to interpret them. Enabled by the state of the art technologies, we can finally start to decode these brain-wave patterns and get an understanding of what is going on inside people's minds, thanks to artificial intelligence (A.I.). In past studies functional Magnetic Resonance Imaging (fMRI) has been used to reconstruct natural visual images perceived by a person's brain activity. However, this hasn't been fully achieved by Electroencephalography (EEG), EEG is a much cheaper technology than fMRI and can be used to measure in real time. Also, in the past the models were built using receptive field models.

In this study, we decoded the measured brain activity by reducing the EEG signals into a video classification problem, which is designed to help preserve some of the multimodal information of the EEG brain signals. We trained a Deep Neural Network (DNN) using Convolutional Neural Networks (CNN) as well as other traditional machine learning techniques such as logistic regression and K-Nearest Neighbor (K-NN) for the EEG classification task by using EEG videos.

In the study, we used the 128-channel EEG to record the brain activity of 10 participants while they looked at images of four geometrical shape classes. The proposed K-NN model used for discriminating between the geometrical shapes using the brain signals reached a high accuracy of 93 %, which outperforms the CNN model which reached an accuracy of 74 %. This was achieved after adding GAN generated synthetic images to the training set. Our study shows that K-NN is a very powerful method and should not be overlooked when choosing a model for an application and go directly into Convolutional Neural Networks (CNN). Deep Neural Networks (DNN) don't always necessarily make better predictions than the traditional machine learning methods.

In the study we attempted to explore to what extent each region of the brain plays a role in human visual processing. In this part of the investigation - it was necessary to use the 128-channel EEG. We determined that the frontal cortex plays a role in visual processing undeniably and will definitively aid in understanding the brain better.

Opsomming

Tans word die dinamika en werking van die brein nie ten volle begryp nie. Die opsporing van breingolfpatrone is al dekades moontlik, maar die ontbrekende element was die vermoë om dit te interpreteer. Aangesien kunsmatige intelligensie (K.I.) deur die moderne tegnologie gebruik word, kan ons uiteindelik breingolfpatrone begin dekodeer en 'n begrip kry van wat in mense se gedagtes aangaan. In vorige studies is funksionele magnetiese resonansbeelding (fMRI) gebruik om natuurlike visuele beelde wat deur 'n persoon se breinaktiwiteit waargeneem word, te rekonstrueer. Dit is egter nie ten volle bereik deur die gebruik van elektro-ensefalografie (EEG) nie, wat 'n baie goedkoper tegnologie is as die voormalige fMRI en wat in reële tyd gebruik kan word. Ook, in die verlede is die modelle gebou deur die gebruik van reseptiewe veldmodelle.

In hierdie studie dekodeer ons die gemete breinaktiwiteit deur die EEG-seine in 'n videoklassifikasieprobleem te verminder, wat ontwerp is om sommige van die multimodale inligting van die EEG-breinseine te bewaar. Ons het 'n Diep Neurale Netwerk (DNN) opgelei met behulp van “convolutional” neurale netwerke (CNN) asook ander tradisionele masjienleertegnieke soos logistieke regressie en K-Nearest Neighbor (K-NN) of “naaste buurman” vir die EEG-klassifikasietak deur gebruik te maak van EEG-video's.

In die studie het ons die 128-kanaal EEG gebruik om die breinaktiwiteit van 10 deelnemers op te neem terwyl hulle na beelde van vier meetkundige vormklasse kyk. Die voorgestelde K-NN model wat gebruik is om die meetkundige vorms te onderskei met behulp van die breinseine, het 'n hoë akkuraatheid van 93 % bereik, wat beter is as die CNN-model wat 'n akkuraatheid van 74 % bereik het. Dit is bereik nadat GAN-gegenereerde sintetiese beelde by die opleidingsstel gevoeg is. Ons studie toon dat K-NN 'n baie kragtige metode is en nie oorkyk moet word as 'n model vir 'n toepassing gesoek word nie en direk na Diep Neurale Netwerk (DNN) gaan kyk nie. Diep Neurale Netwerk (DNN) maak nie noodwendig altyd beter voorspellings as die tradisionele masjienleermetodes nie.

Hierdie studie het ook ondersoek ingestel om te bepaal tot watter mate elke deel van die brein 'n rol speel in die visuele prosessering van die mens en daarom is dit noodsaaklik om al die 128-kanale van die EEG te gebruik. In die studie het ons ook vasgestel dat die frontale korteks wel 'n rol speel in visuele prosessering en sal beslis help om die brein beter te verstaan.

Acknowledgements

At the outset my sincere appreciation to my supervisor, Prof. D. Van den Heever for his support and inspirational guidance throughout this project and to my father, Pieter Uys, for his support and guidance throughout my studies since 2014 when I started my engineering studies. I also thank Pieter Holtzhauzen for assistance with neural networks and Herman Kamper for assistance with K-NN.

Dedicated to my parents for all their help, love and support throughout my studies.

Contents

1	Introduction	1
1.1	Problem statement	1
1.2	Research questions	1
1.3	Aim and objectives of the research	1
1.4	Anticipated outputs	2
2	Technical Background	3
2.1	The brain	3
2.1.1	Visual processing in the brain	3
2.1.2	The visual cortex	4
2.1.3	The Brodmann area mnemonic of the brain	4
2.2	Electroencephalography	6
2.2.1	EEG vs. fMRI	7
2.2.2	Interpreting the EEG data from each part of the cortex	7
2.2.3	Types of EEG signals	8
2.2.4	EEG data acquisition	10
2.2.5	Clean EEG data and artifacts	12
2.2.6	Independent Component Analysis (ICA)	15
2.3	Artificial Intelligence	16
2.3.1	Introduction to AI	16
2.3.2	Types of machine learning systems	18
2.4	Machine Learning Techniques	20
2.4.1	Logistic Regression	20
2.4.2	Linear Support Vector Machine	21
2.4.3	K-Nearest Neighbor	23
2.4.4	Random Forest	25
2.5	Artificial Neural Networks	26
2.5.1	Introduction	27
2.5.2	Fine-tuning model hyperparameters	28
2.6	Deep Learning	31
2.6.1	Convolutional Neural Networks	32
2.6.2	Generative Adversarial Networks	37
2.6.3	Training deep neural nets	39
2.6.4	Histogram of Orientation Gradients	42
2.6.5	Performance measures	44
2.7	Related work	46

2.7.1	A novel approach for classifying EEG signal with multi-layer neural network	46
2.7.2	Multimodal classification with deep convolutional-recurrent neural networks for electroencephalography	46
2.7.3	Deep learning human mind for automated visual classification	47
2.7.4	Multi-task generative adversarial learning on geometrical shape reconstruction from EEG brain signals	47
2.7.5	Summary	48
3	Methods and results	49
3.1	Experimental setup	49
3.1.1	The data	49
3.2	EEG data preprocessing	52
3.2.1	Importing data	53
3.2.2	Removing bad channels and interpolating	53
3.2.3	Artifact correction	54
3.2.4	Filtering	55
3.2.5	Six-sigma clipping	55
3.3	Making images from EEG time-series data	56
3.3.1	The fast fourier transform	56
3.3.2	2D projection	58
3.4	Implementation	60
3.4.1	Logistic regression model	60
3.4.2	Support Vector Machine model	60
3.4.3	HOG-SVM model	61
3.4.4	K-Nearest Neighbor model	62
3.4.5	Random Forest	63
3.4.6	Convolutional Neural Network model	63
3.4.7	Generative Adversarial Network model	65
3.4.8	How brain regions affect CNN predictions	67
3.4.9	Generalization across participants	69
3.5	Results	69
3.5.1	CNN Model vs. CNN model with GAN training data	70
3.5.2	Traditional machine learning techniques vs. convolutional neural networks	70
3.5.3	Brain regions comparison	71
4	Discussion and future work	73
5	Conclusion	77

Bibliography	84
Appendices	84
A Convolutional Neural Network	85
A.1 CNN Architecture	86
B GAN Architecture	87
B.1 Generator	87
B.2 Discriminator	88
C Classifying EEG signals with CNN graphs	89
C.1 Representation of occipital cortex (28 channels), and of frontal cortex (56 channels)	89
C.1.1 Electrode locations	89
C.1.2 2D projections	90
C.1.3 Accuracy graphs	90
C.1.4 Loss graphs	91
C.1.5 Confusion matrices	92
C.2 Representation of occipital and parietal cortex (72 channels), and of whole cortex (128 channels)	93
C.2.1 Electrode locations	93
C.2.2 2D projections	94
C.2.3 Accuracy graphs	94
C.2.4 Loss graphs	95
C.2.5 Confusion matrices	96

List of Tables

2.1	Advantages and disadvantages of each imaging methodology adapted from [7].	7
2.2	Confusion Matrix	45
2.3	Experiment results % showing [66]’s approach’s superiority above other traditional methods [66].	47
2.4	Experiment results of Zhang’s approach showing qualitative comparison of inception score and inception accuracy [69].	48
3.1	Models’ loss and accuracy results.	70
4.1	The comparison of classification results of brain signals using deep learning models and traditional machine learning models obtained in this study.	76

List of Figures

2.1	The cerebral hemispheres of the human brain with Brodmann’s “areas” applied. These images are taken from Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues [6].	5
2.2	Locations of the different lobes [16].	8
2.3	EEG Frequency Bands [19]	10
2.4	Overview of a typical experimental EEG recording protocol [20].	11
2.5	The 10-20 International Standard System for EEG recordings [24].	12
2.6	Example of eye movements and blinks, the x-axis presents time and the y-axis present microvolts [25].	13
2.7	A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology [35].	17
2.8	A labeled training set for supervised learning (e.g., spam classification) [5].	18
2.9	An unlabeled training set for unsupervised learning [5].	19
2.10	Semi-supervised learning [5].	19
2.11	Learning Methods with examples of each method.	20
2.12	The sigmoid function	21
2.13	Infinity Hyperplanes and Maximum Margin Classifier [39].	22
2.14	Support Vectors with a small margin and a large margin [38].	22
2.15	A simple K-NN example	24
2.16	Decision Tree for Temperature Prediction [40].	25
2.17	The biological neuron [5].	27
2.18	Neural network elements [44].	28
2.19	Local receptive fields in the visual cortex [5].	33
2.20	The convolution operation [53].	34
2.21	Feature map result and a 3-D convolution operation example [53].	34
2.22	Two feature maps stacked along the depth [53].	35
2.23	Max pooling layer (2 x 2 pooling kernel, stride 2, no padding) [53].	36
2.24	A typical CNN architecture [5].	37
2.25	The working of the process between training the discriminator and the generator [58].	39
2.26	An example of Data Augmentation [61]	42

2.27	On the left is the RGB image and on the right is the 8 x 8 patch with arrows that represent the gradients. Retrieved from one of Udacity's lecture videos [62].	43
2.28	Histogram of Orientation Gradients. Retrieved from one of Udacity's lecture videos [62].	43
3.1	Builder interface flow	50
3.2	One of the participants during a EEG recording	51
3.3	(a) The 4 geometrical shapes as stimuli, (b) EEG recording paradigm for one of the stimuli.	52
3.4	The identifying of ocular components	55
3.5	Time domain to frequency domain [71]	56
3.6	The EEG time series data from time domain to frequency domain	57
3.7	(a)The 13 x 13 electrode location mesh used to merge the three maps to form the three-channel image. (b)The three-channel image generated after the projections.	58
3.8	Overview of the approach of the EEG time series projections. The approach is based on the results of Bashivan et al. [67].	59
3.9	Example of the shades of gray image and the corresponding HOG image	61
3.10	Plot of misclassification error versus number of neighbors K.	62
3.11	CNN model optimization: The training and validation loss over the entire dataset is evident.	64
3.12	CNN model optimization: The training and validation accuracy cover the entire dataset.	65
3.13	The image on the left is the real 2D projected EEG image and that on the right is the generated image.	66
3.14	CNN model optimization: The training and validation loss over the entire dataset with generated images are included.	66
3.15	CNN model optimization: The training and validation accuracy over the entire dataset with generated images are included.	66
3.16	A brain image from a fMRI scan during one of the tests on the role of the frontal cortex in vision [76].	68
3.17	Mapping between EEG channels and the brain cortices [77].	68
3.18	CNN model's accuracy plot.	69
3.19	CNN model's loss plot.	69
3.20	A comparison of the CNN model's performance with the CNN model with added GAN training data.	70
3.21	71
A.1	Proposed CNN architecture.	86
B.1	Proposed Generator architecture.	87
B.2	Proposed discriminator architecture.	88

C.1	Electrode locations of the occipital cortex and the frontal cortex.	89
C.2	2D projected images generated.	90
C.3	28 Channel accuracy plot	90
C.4	56 Channel accuracy plot	90
C.5	28 Channel loss plot	91
C.6	56 Channel loss plot	91
C.7	28 and 56 channel confusion matrices.	92
C.8	Electrode locations of the occipital + parietal cortex, and the whole cortex.	93
C.9	2D projected images generated.	94
C.10	72 Channel accuracy plot	94
C.11	128 Channel accuracy plot	94
C.12	72 Channel loss plot	95
C.13	128 Channel loss plot	95
C.14	72 and 128 channel confusion matrices.	96

List of abbreviations

Symbol	Meaning
AEP	<i>Azimuthal Equidistant Projections</i>
ADAM	<i>Adaptive moment estimation</i>
AI	<i>Artificial Intelligence</i>
ANN	<i>Artificial Neural Network</i>
BCI	<i>Brain Computer Interface</i>
BN	<i>Batch Normalization</i>
CNN	<i>Convolutional Neural Network</i>
DL	<i>Deep Learning</i>
DNN	<i>Deep Neural Network</i>
EEG	<i>Electroencephalography</i>
EMG	<i>Electromyogram</i>
FEMG	<i>Facial Electromyography</i>
FFT	<i>Fast Fourier Transform</i>
fMRI	<i>functional Magnetic Resonance Imaging</i>
GAN	<i>Generative Adversarial Networks</i>
GPU	<i>Graphics processing unit</i>
GRU	<i>Gated Recurrent Unit</i>
HOG	<i>Histogram of Oriented Gradients</i>
ICA	<i>Independent Component Analysis</i>
LDA	<i>Linear Discriminant Analysis</i>
LSTM	<i>Long-Short Term Memory</i>
ML	<i>Machine Learning</i>
MNE	<i>Magnetoencephalography (MEG) and Electroencephalography (EEG)</i>
NN	<i>Neural Network</i>
PCA	<i>Principal Component Analysis</i>
RGB	<i>Red Green Blue</i>
RNN	<i>Recurrent Neural Network</i>
SVM	<i>Support Vector Machine</i>

Chapter 1

Introduction

1.1 Problem statement

My research project will strive to understand the functionality and structure of the brain better by classifying geometrical shapes as received and perceived by the brain. The goal is to decode brain activity of the visual system and to build computational models that can accurately predict the geometrical shapes observed by the subject by decoding their brain-wave patterns. By transforming these brain-wave patterns into meaningful content has a wide potential in creating brain-machine interfaces.

1.2 Research questions

- To what extent can CNN assisted by Generative Adversarial Network (GAN) and other ML techniques be used to discriminate the visualization of four geometrical shapes using image-based features extracted from EEG data?
- Does the Frontal Cortex play a role in vision?
- Will Convolutional Neural Networks (CNN) make better predictions than traditional Machine Learning (ML) techniques?
- Does the model generalize across participants?

1.3 Aim and objectives of the research

The aim of this research project is to get a better understanding how the brain encodes basic visual shapes by developing a computational model of brain activity that is evoked by static natural vision.

Objectives:

- Use EEG to acquire brain activity data recordings.
- Develop a thorough understanding machine learning and deep learning by applying different techniques for discriminative feature extraction.

- Build a Neural Network (NN) that can accurately predict specific brain activity focused on simple geometric shapes.
- Use traditional machine learning techniques such as Support Vector Machines to compare with Neural Networks.
- Interpret the results and draw conclusions.
- Understand how the brain encodes basic visual shapes.

1.4 Anticipated outputs

One of the main goals of this research project is to build computational models that can predict brain activity measurements during vision. Predictive models are the gold standard of computational neuroscience, according to Professor Jack L. Gallant, from Berkeley at the University of California [1] a leader in the research field of Cognitive Neuroscience. Using EEG data for image classification by building these predictive models which have great theoretical and practical potential. There are many potential applications for decoding brain activity: In principle it should be possible to decode visual content of dreams and memories. It can at a later stage be potentially used in detective work in forensic analysis for facial reconstruction or gathering eye-witness information on potential suspects rather than relying on a sketch artist [2]. All of these potential applications can help aid in our understanding of visual encoding in the brain.

Once the research project succeeds in accurately classifying the geometrical shapes and learn how and where the brain processes visual content, extending the scope of the research project to more complex shapes or improving generalization of such models will definitely be possible as a further research topic.

Chapter 2

Technical Background

This chapter starts by introducing the visual processing in the brain and some insight on how the brain came to be divided into distinct regions. Henceforth, the basic principles and characteristics of Electroencephalography are introduced, and continues by leading to Artificial Intelligence, Machine Learning techniques, followed by explaining the concepts of Neural Networks and Deep Learning. In addition, this chapter presents similar projects that are in line with this study.

2.1 The brain

2.1.1 Visual processing in the brain

The brain is a highly complex, non-linear, dynamic system, with so much still unknown about its structure and functioning. The brain is a very dense, interconnected network of neurons processing data at high speed. According to Kobinian Brodmann, the brain is divided into 47 local areas, where each area is processing specific types of data. Humans connect with their surrounding environment with their five senses. Sight or visual data is the sense that plays the largest role, as sensed by the eyes [3]. A very large part of the brain is responsible for vision and visual processing. When light enters the eye and passes through the cornea, it reaches the visual receptors of the eye that transduces the light into electrical nerve impulses [4].

These impulses are sent to the brain via the optical nervous system, to an area where these impulses are processed. The human eye has 125 million visual receptors which are composed of rods and cones. There are three types of cones which are sensitive to red, blue and green colors, and these colors are identified under the right lighting conditions. The rods do not sense color such as the cones, instead they produce grayscale data since they are more sensitive to dim lighting conditions [4].

2.1.2 The visual cortex

The visual cortex region in the posterior part of the brain known as the occipital lobe is responsible for processing the visual data. It is believed that understanding the structure and functioning of the visual cortex will provide the fundamental insights into how the neocortex operates [3]. The visual cortex consists of three areas (area 17, 18, 19 on the Brodmann area Mnemonic, also known as V1, V2, V3 as seen in figure 2.1) The Brodmann area Mnemonic of the brain is explained in section 2.1.3. The primary visual cortex (area V1) is the largest single processing module in the brain and responds to simple local features in images such as edges, color and texture [5] [4]. Signals that leave area V1 are distributed to the other visual areas, however, the functions of these higher visual areas are not fully known, but are believed to be responsible for extracting more complex higher order information from the visual data [4].

To uncover some of the brain's mysteries and neural dynamics in the study of visual processing, it should in principle be possible to understand the brain better as a computer, and discover the underlying computations that govern its functions. The goal of this research project is to capture the static natural visual experiences perceived by the primary visual cortex using electroencephalography (EEG), decode the EEG data, and develop computational models that can predict the pattern of the brain activity which are then used to reconstruct the stimulus [4]. From these predictions a lot can be learnt about how the brain processes images and create the mental images we call shapes.

2.1.3 The Brodmann area mnemonic of the brain

Korbinian Brodmann is best remembered for his classification of cortical areas which are based on cytoarchitecture. He was greatly influenced by Alzheimer, Vogt, Edinger, Nissl and Weigert. Although Brodmann's "mapping was first presented only in 1903, it continues to be the lingua franca of cortical localization. His writings on this topic have become a neuro- logical classic, evidenced by the many researchers who have built on his work and ideas, [6].

Brodmann published a series of seven communications on comparative mammalian (over 64 different species) cytoarchitecture between 1903 and 1908. He defined cytoarchitecture as "the localization of the individual histological elements, their layering, and their parcellation in the adult brain. His most well known papers [6] published in 1908 contain the famous map in which he organized the unique histological regions of the human cortex.

In 1918, Brodmann moved to Munich to take charge of the Department of Topological Anatomy which was managed by Emil Kraepelin. Kraepelin saw the importance of

neurohistology and cytoarchitecture in the future and formed a powerful collaboration by inviting Brodmann and Nissl to Munich. Unfortunately, Brodmann died on August 22, 1918, of sepsis just as the group came together. Brodmann identified 47 histological distinct regions in the brains of humans (see figure 2.1) using novel staining techniques introduced by Nissl, and in primates, Brodmann described 52 different regions [6].

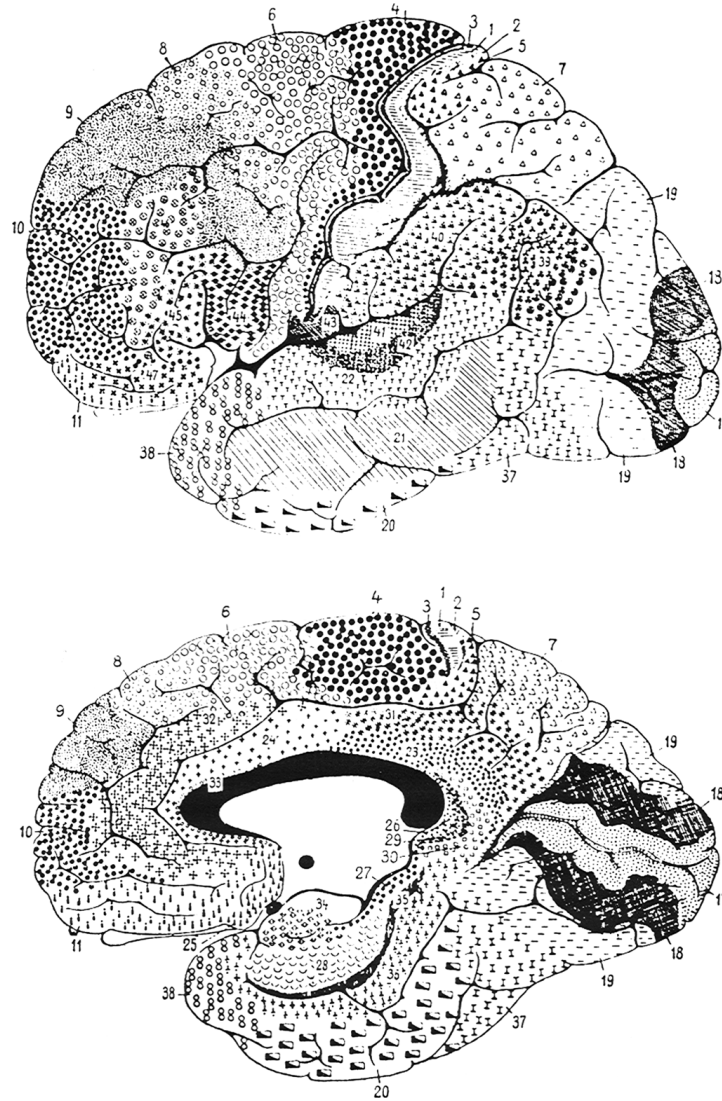


Figure 2.1: The cerebral hemispheres of the human brain with Brodmann's "areas" applied. These images are taken from *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues* [6].

At that time there were three schools of thought for mapping of the brain contributed by men such as Meynert, Betz, Edinger, Hammarberg, Lewis and Clarke [6]. Some of them sought to localize function on the basis of the presence of individual histological

elements, while others believed that each cortical layer of the brain was associated with a specific function. Brodmann was opposed to these ideas of assigning functions to specific areas of the brain [6]. The third school of thought which Brodmann was a supporter of, was in some way a combination of the first two. It claimed that regions that contained similar structures in both layering and cell type could produce specific functions. The data that Brodmann produced for modern researchers to associate specific functions with many of the different cortical regions he defined. Therefore, Brodmann's "areas provided the ability to apply more discrete terminologies to the lobes of the brain [6].

2.2 Electroencephalography

Electroencephalography, also known as EEG, is the study of the brain functions reflected by the brain's electrical activity and is considered one of the basic tools to image brain functioning. Our thoughts are generated through a network of neurons, that send signals to each other with the help of electrical currents. To be able to collect the brain's electrical signals, electrodes of an EEG headset is placed on the scalp. In addition a conductive paste is used to improve the conduction of the electrical signals¹. The EEG headset used in this study is an elastic cap similar to a bathing cap with the electrodes mounted to the cap. The electrodes are mounted systematically on the cap using the international 10-20 system for electrode placement to ensure that the data can be collected from identical positions across all respondents. These electrodes detect the electrical changes of thousands of synchronized neurons simultaneously. The voltage fluctuations measured by the electrodes are very small, typically in microvolts. The signals are digitized and sent to an amplifier where the signals are amplified [7].

Once the signals are amplified, the signals are sent to a computer, where these can be recorded. Using different methodologies, the signals can be represented as a vector array or matrix array for data processing utilities [8]. In addition, various maps of the brain activity can be generated, with a rapid temporal resolution. A drawback for EEG is the spatial resolution. It is difficult to tell whether the signals that were measured by the electrodes were produced near the surface, or in deeper regions of the brain. The cost of EEG systems depend on the following: (1) The number of electrodes on the headset, (2) The quality of the amplifier, (3) The sampling rate, measured in Hz [9].

Analyzing EEG data can be quite a challenge. Signal processing, artifact detection and attenuation, feature extraction, and computation metrics all effect the quality of EEG data. To properly identify and extract the valuable information from the collected data requires a certain level of expertise, the rest of this section introduces ways to ensure the quality of these data [9].

¹There are different types of EEG headsets, ones that use the conductive paste and ones that are dry electrodes.

Table 2.1: Advantages and disadvantages of each imaging methodology adapted from [7].

	EEG	fMRI
Temporal resolution	High	Low
Spatial resolution	Low	High
Level of experience needed	Some training	Extensive training
Measures brain activity	Directly	Indirectly (BOLD response)
Cost	Accessible to many researchers	Requires extensive funding
Portability	Both fully portable and semi-portable devices available	Not portable

2.2.1 EEG vs. fMRI

In the past, functional magnetic resonance imaging (fMRI) was used for these applications, for example [10] - [14]. fMRI measures the brain activity by detecting changes in the blood flow of the brain. fMRI has good spatial resolution, meaning that it can capture finer details in specific areas of the brain, but on the other hand, has poor temporal resolution and cannot measure events in real time [15]. Deciding whether to use EEG or fMRI depends on the research question. If you're more concerned with structural and functional detail, then fMRI will be the best choice. For quicker, more affordable, and accessible insights about the brain, with temporal resolution, then EEG would be the preferred method of choice. Each of the imaging methodologies has distinct advantages and disadvantages, some of these are summarized in figure 2.1 [7].

One of the major advantages of using EEG is the fact that it has excellent temporal resolution, meaning that it can measure in fine detail events happening in real-time. According to Neuroscience News, researchers believe that it takes about 17 milliseconds for the brain to form a representation of a human face making EEG the perfect candidate, as EEG can capture activity at a time scale down to milliseconds [15].

2.2.2 Interpreting the EEG data from each part of the cortex

Each area of the cortex is responsible for processing specific information at a given time (Figure 2.2 shows the location of each lobe):

Occipital cortex

The occipital cortex is primarily responsible for processing visual information, therefore EEG experiments with visual stimuli such as videos and images will focus on the effects in this region [9].

Parietal cortex

The parietal cortex is primarily responsible for motor functions and is active during self-referential tasks such as when coming in contact with an object or information that is of importance to humans [9].

Temporal cortex

The temporal cortex is responsible for language processing and speech production, the inner regions of the temporal cortex become more active during spatial navigation [9].

Frontal cortex

The frontal cortex is all about executive functions, it helps with maintaining control, plan for the future, and monitor daily behavior [9].

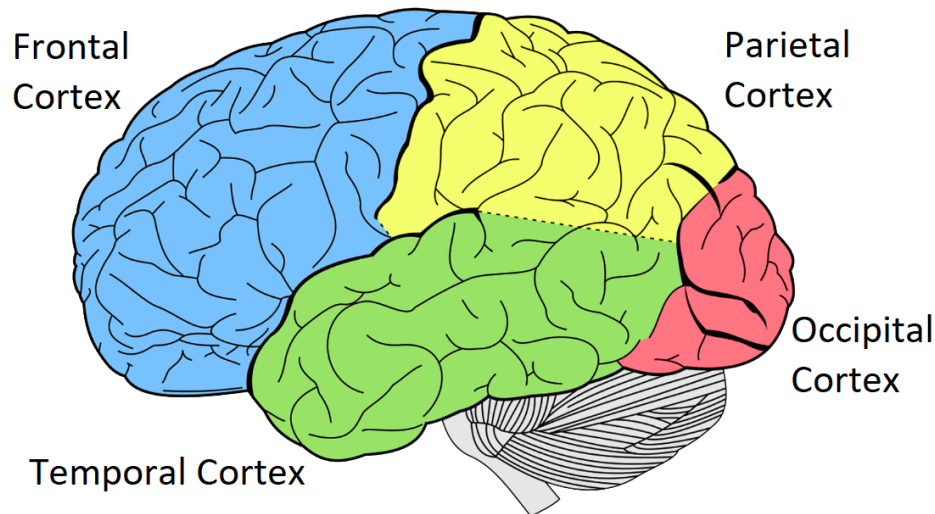


Figure 2.2: Locations of the different lobes [16].

2.2.3 Types of EEG signals

EEG is traditionally typically assessed in five standard frequency bands, which are shown in figure 2.3. Each frequency band is believed to play a role in different functions, the frequency patterns change each time the brain is in a different state, giving more insight into cognitive processes.

- **Gamma** wave frequency bands are considered the fastest brain waves have namely, a frequency range between 31 - 100 Hz. They are believed to be responsible for simultaneously processing information from different areas of the brain , rapid eye

movements called microsaccades² which are considered the integral parts of sensory processing and information uptake, to help with learning, perception, insight, expanded consciousness and peak focus [17].

- **Beta** wave frequency bands are associated with cognition, concentration and alertness, they have a frequency range between 12 and 40 Hz. Beta waves become stronger as we plan or execute movements of any of the body parts [9]. They are characterized in states of cortical activation [18]. Beta waves are believed to help with problem solving, memory and conscious focus at an optimal level [17].
- **Alpha** wave frequency bands have a frequency range between 8 - 12 Hz and are characterized by the state of consciousness and physical and mental health [18]. Alpha levels are increased when in the state of relaxed wakefulness. They are associated with visualization, inhibition, attention, relaxation and creativity [17] [9].
- **Theta** wave frequency bands have a frequency range between 4 - 8 Hz and are characterized by the state of deep and normal sleep in childhood, and also involve daydreaming [18]. Theta waves are present during meditation, creativity, emotional connection, intuition, relaxation and memory [17]. They are also associated with a wide range of cognitive processing, they become more prominent when a person is confronted with some sort of difficult task [9].
- **Delta** wave frequency bands have a frequency range between 0.1 - 4 Hz and are considered the slowest brain waves. They are characterized by indicative pathological states of neuronal difficulty and occur during deep sleep [18]. Babies and children produce the most delta waves, and as they age into adulthood, these waves are produced less. Delta waves are believed to help with natural healing, sleep, strong immune system and detached awareness [17].

²Small, jerk-like, involuntary eye movements.

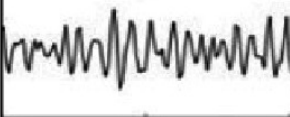

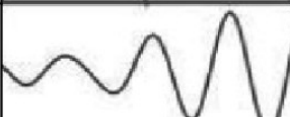
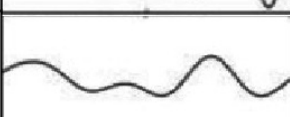

EEG Band	Waveforms	Function
GAMMA 31 - 100 Hz		Insight Peak focus Expanded Consciousness
BETA 16 - 30 Hz		Alertness Concentration Cognition
ALPHA 8 - 15 Hz		Relaxation Visualization Creativity
THETA 4 - 7 Hz		Meditation Intuition Memory
DELTA 0.1 - 3 Hz		Detached awareness Healing Sleep

Figure 2.3: EEG Frequency Bands [19]

2.2.4 EEG data acquisition

An experimental protocol is defined to meet the needs of the experiment, the protocol for recording useful data sets is the most important task in order to systematically accumulate all the training data. The protocol helps to ensure that the same amount of stimuli are shown for each class, set how long each stimulus is shown, how long a blank screen needs to be displayed between each stimulus and to give the subjects a break after each phase etc. Further more, raw EEG signals suffer from interference from various noise sources, such as power line noise, electrode movements, electromyography (EMG), eye movements, eye blinks, skin artifacts and swaying and swinging. These unwanted components may bias the analysis of the EEG, and may lead to wrong conclusions. Therefore it is common practice to clean the EEG data prior to any feature extraction or classification steps.

EEG recording protocols

Although EEG protocols differ from project to project, the basic principles stay the same. A typical EEG experimental protocol is explained in an example and is shown in figure 2.4.

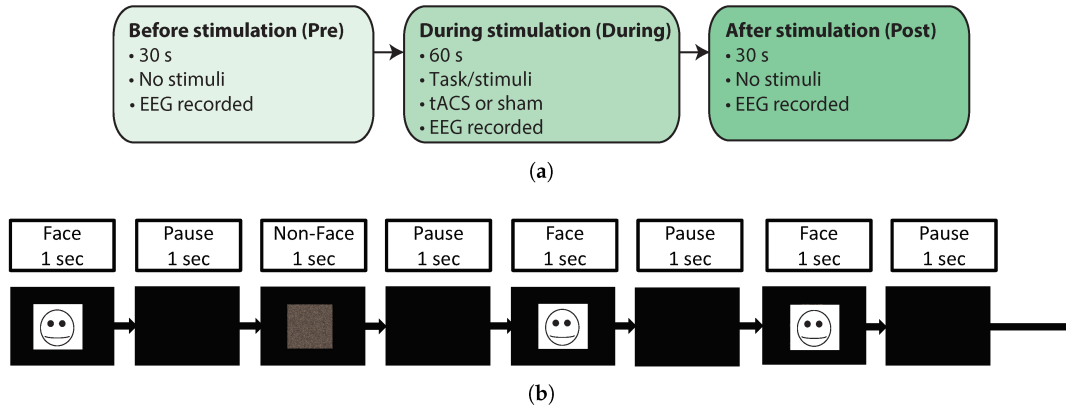


Figure 2.4: Overview of a typical experimental EEG recording protocol [20].

In this example paradigm, there are two classes presented, faces and non-face stimuli. The data are typically recorded with a sampling frequency of 250 - 500 Hz. Each trial starts off with a blank screen for 30 seconds. Usually the blank screens contain a small fixation cross in the middle of the screen. Next, a stimulus is presented (face/non-face stimuli) on the screen for one second. This is followed with a short break of one second before presenting the next stimuli. The stimuli are shown in a randomized order and this pattern continues until all 30 stimuli have been shown (15 face and 15 non-face stimuli). This protocol is repeated 12 times for each participant [20]. Each project differs, but typically such a project would use between 5 - 30 subjects to ensure the data set is big enough. After the experimental trials, all the recorded data are analyzed in a python script or Matlab to remove artifacts that reside in the recorded data, for example, remove eye blinking using ICA. Section 2.2.5 gives more insight on artifacts.

Electrode placement

The 10-20 International Standard was first introduced by Dr. Herbert H. Jasper in 1958, which is the most common standard for EEG electrode placement known today. It's popularity has increased significantly the past two decades since it ensures comparability and reproducibility of results [21]. The technique is based on standard landmarks of the skull. These landmarks are namely the nasion, inion and the left and right pre-auricular points as seen in figure 2.5. The first measurement is from the nasion to the inion in the anterior-posterior plane through the vertex. The measurement is then divided into five separate areas. The '10' and '20' refers to the electrodes being distributed in these five areas in a 10 % and 20 % relationship [22]. In other words, the electrodes are positioned such that 10 % are placed on the pre-frontal and occipital planes, with the rest divided into four parts consisting of 20 % each. The number of electrodes that are used and where they are positioned all depends on the particular signal that is to be

analyzed [18], although most relevant EEG potentials are typically in the central lobe and occipital lobe [23].

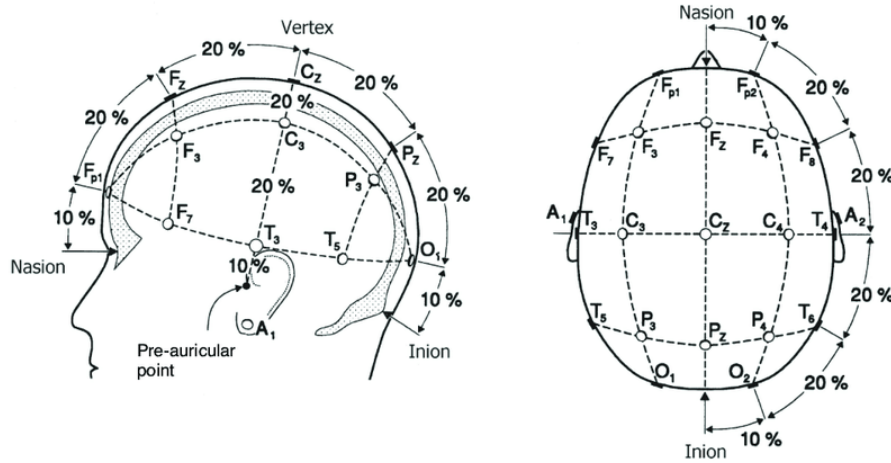


Figure 2.5: The 10-20 International Standard System for EEG recordings [24].

2.2.5 Clean EEG data and artifacts

It is very important to avoid or at least minimize artifacts as much as possible to ensure that the EEG data are clean (collect EEG data that reflect brain signals only). Artifact recognition and elimination pose the biggest challenge when recording and monitoring EEG data, these artifacts are either patient related artifacts or external sources of artifacts. These artifacts must all be handled differently, there are some tools that could be used for finding artifacts such as Facial Electromyography (FEMG), which are tiny electrical impulses that are generated by the facial muscle fibers when they contract.

Subject related artifacts

- **Muscle activity (Electromyogram (EMG)):** Muscle movements generate electrical activity that is picked up by the electrodes, the most severe effects are from muscle movements in the facial and neck regions such as jaw movements when talking, chewing, swallowing, smiling etc. [25]. Jaw clenching should therefore be avoided at all times. Another muscle activity that can effect the EEG data is the pulsating heart that produces an electric field. The pulse artifact occurs when electrodes are placed on a pulsating vessel.
- **Eye movements:** The movement of the eyeballs (horizontal and vertical) cause a

change in the electric field that gets picked up by electrodes positioned nearby [25]. Vertical eye movements look more sinusoidal, while horizontal eye movements look more box-shaped [27]. Eye movements are usually tracked using an eye tracker or placing more electrodes surrounding the eyes, making it more easy to remove. Figure 2.6a gives an example of how eyeball movements effect the EEG data.

- **Blinks:** The movements of eyelids also cause interference with the brain signals, although these artifacts are somewhat easier to detect because of their typical shape. There are algorithms developed to detect these artifacts automatically, namely Independent Component Analysis (ICA). ICA is explained more in depth in Section 2.2.6. Figure 2.6b gives an example of how eye blinks effect the EEG data.
- **Skin artifacts:** Another difficulty arises with the properties of certain layers of skin. The change of skin potential causes artifacts. In other words, a significant DC potential exists between the stratum corneum and the stratum granulosum and any local deformation of the skin will change this potential [26]. Sweat is also a common cause of changes in impedance between the skin and the electrodes creating a slow baseline drift. The only way to eliminate these artifacts is good electrode placement and to create low impedance pathways through the layers of skin to the electrodes by skin cleaning using an alcohol swab.

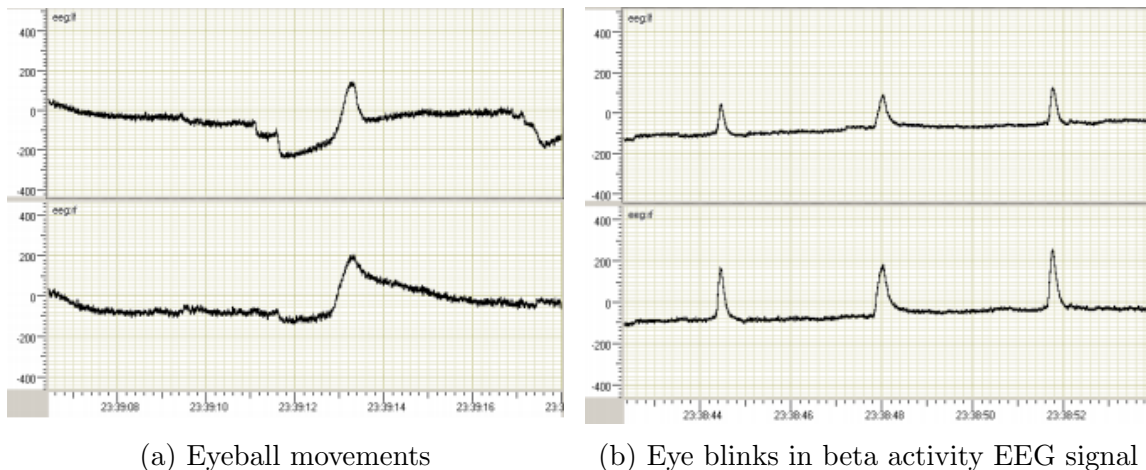


Figure 2.6: Example of eye movements and blinks, the x-axis presents time and the y-axis present microvolts [25].

External sources of artifacts

External artifacts are often a result of unsatisfactory technology which consists of apparatus and connections [25].

- **Electrode movements:** Electrode movements can cause severe artifacts in the channels that are affected or in all the channels, if the headset has moved. The reason for electrode movement can occur due to an electrode losing contact, because the headset might be becoming loose. It is always recommended to ensure that all the electrodes are securely attached to the skin and that the headset sits snug on the participants head [27].
- **Power line noise:** Power line interference can have strong artifacts on EEG recordings. The frequency of the mains cause an artifact appearing at 50 Hz³, the surrounding wall and electric cables produce a constant electric field [25].
- **Swaying and swinging:** Swaying and swinging can have strong effects on the EEG recordings, especially when swinging the head and or banging it against some object. This changes the water distribution, which changes the electrical properties and fields generated by the brain [27]. To prevent these effects in the EEG recordings, ensure that the participant don't move their heads too much.

Important matters to keep in mind when capturing data from subjects:

- Train the subjects to minimize the artifacts in the signals for example by showing them what happens to the signal when they move around, clench their jaw, smile etc.
- Minimize artifacts while collecting the data by monitoring the data every 30 seconds. If the duration of the trial is longer than an hour, re-gel some of the electrodes with the conductive paste as they get dry and don't make contact.
- Eye blinking doesn't destroy the EEG data, but is rather summed linearly on top of the EEG data. This can be removed during post processing of the EEG data by making use of an Independent Component Analysis (ICA) tool.
- Ensure that the stimuli are repeated many times (for example 50 times per class).
- Keep in mind that there will be differences in the EEG data for each subject (electrode placement, time of the day etc.). A subject will have slightly different data on different days, therefore it might be better to test on a participant for a longer periods on one day.
- Ensure that the participants are really engaged in the experiment and not day dreaming for example, it is useful to include a check point every now and then. For instance, the check might be to ask the participant after every 10-15 stimuli, what the last stimulus shown was? Or even better, to which category does the last stimuli belong to. By doing so, one can ensure that the participants have been paying attention.

³In South Africa it is 50 Hz, in the United States power line interference is at 60 Hz.

2.2.6 Independent Component Analysis (ICA)

ICA is a widely-used blind source separation technique that separates and localizes independent signals that have been added together, and finds directions in the feature space corresponding to projections with high nongaussianity [28]. It has been applied to a wide range of applications and is usually utilized as a black box, without understanding its internal detail. ICA is considered an extension of the principal component analysis (PCA) technique which finds uncorrelated components while ICA finds independent components [29].

To find these independent components, we can define ICA by using a statistical “latent variables” model. The mathematical derivations are adapted from [30]. Assume to be observing n linear mixtures x_1, \dots, x_n of n independent components

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \dots + a_{jn}s_n, \text{ for all } j. \quad (2.1)$$

The time index t has now been dropped in the ICA model and it is assumed that each mixture x_j as well as each independent component s_k is a random variable, instead of a proper time signal. Without the loss of generality, it can be assumed that both the mixture variables and the independent components have zero mean. Let \mathbf{x} denote the random vector whose elements are mixture x_1, \dots, x_n and \mathbf{s} the random vector with elements s_1, \dots, s_n . Let \mathbf{A} denote a matrix with elements a_{ij} . All of the vectors are understood as column vectors, thus, \mathbf{x}^T or also known as the transpose of \mathbf{x} , is a row vector. Using the vector-matrix notation, the above can be written as

$$\mathbf{x} = \mathbf{A} \cdot \mathbf{s}. \quad (2.2)$$

Equation 2.2 is called the independent component analysis, or ICA model. The ICA model is a generative model, meaning that it describes how the observed data is generated by a process of mixing the components s_i . The mixing matrix is assumed to be unknown, the only thing known is the random vector \mathbf{x} and both \mathbf{A} and \mathbf{s} must be estimated by using \mathbf{x} . The starting point for ICA is the simple assumption that all the components s_i are statistically independent. It must also be assumed that the independent components must have nongaussian distributions.

After the matrix \mathbf{A} has been estimated, the inverse can be computed, say \mathbf{W} , and all the independent components are obtained:

$$\mathbf{s} = \mathbf{W} \cdot \mathbf{x} \quad (2.3)$$

ICA has become a popular application in EEG signal processing, as it provides a powerful way of removing artifacts from the EEG data and also help disentangle otherwise mixed signals [31].

2.3 Artificial Intelligence

2.3.1 Introduction to AI

Artificial intelligence (AI) has captured the attention of a global audience during the last couple of years [32]. Hollywood has painted a multicoloured science fictional world on the big screens for us and now it is quickly becoming a reality. The phrase Artificial Intelligence was coined in 1956 by John McCarthy, who is widely recognized as one of the godfathers of AI, he defined it as "the science and engineering of making intelligent machines". He became the leader of AI for many decades. McCarthy was a computer science professor at the University of Stanford. This is where McCarthy founded an AI laboratory where he worked on early versions of a self-driving car. He produced many publications, some of which were on robot consciousness and free will. McCarthy worked in the AI field continuously until semi-retirement from Stanford in 2000. McCarthy unfortunately passed away on October 24, 2011 [33].

AI is the capability of a machine to imitate intelligent human behavior. Today, AI is being integrated into expert systems, speech recognition, facial recognition, autonomous vehicles, robotic process automation and the list just keeps going on [32].

How does Machine Learning and Deep Learning relate to Artificial Intelligence? One can think of it as a set of Russian dolls nested within each other, starting with the smallest and working outwards [34]. Deep learning is a subset of machine learning, and machine learning is a subset of AI.

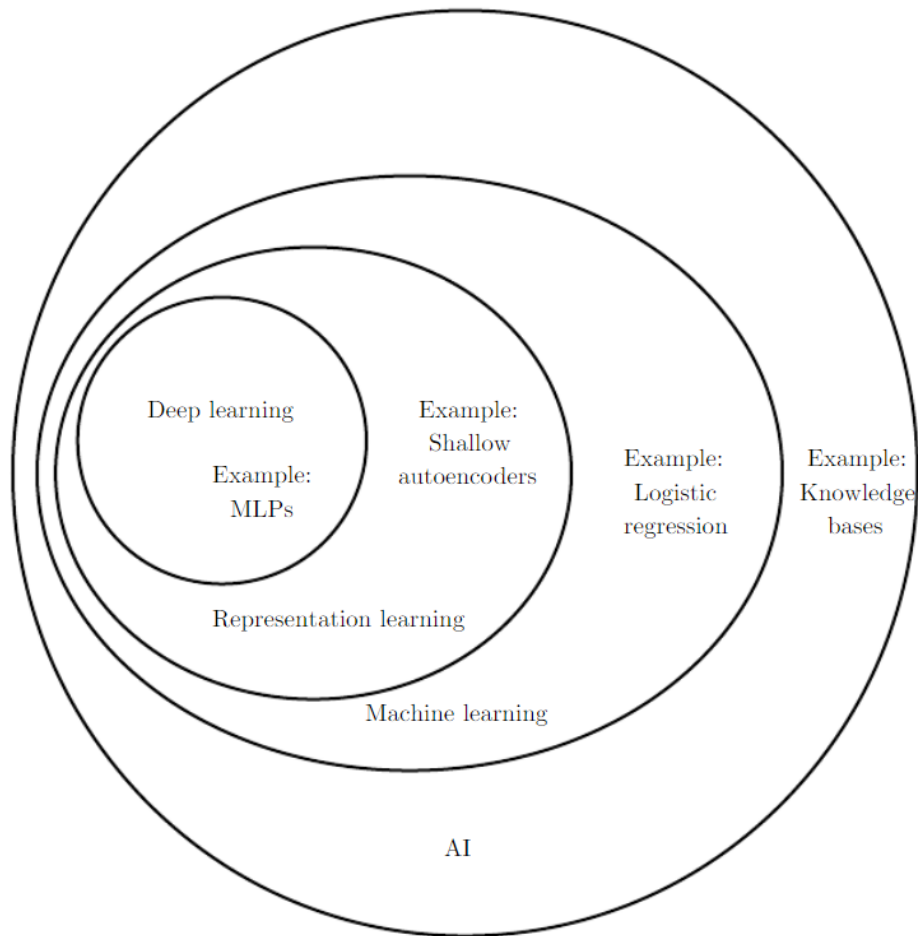


Figure 2.7: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology [35].

When most people hear “Machine Learning”, they picture some fictionally futuristic fantasy such as the Terminator or the technology from Star Trek. Truth be spoken, it’s not that futuristic any more, it’s already here. But what is machine learning? Machine Learning is the science of programming computers such that they are able to learn from the data. The “learning” part of machine learning means that machine learning (ML) algorithms attempt to optimize along a certain dimension. This is usually done by minimizing the error or maximizing their likelihood of their predictions being true. ML is being implemented across all business verticals and we can no longer imagine a

world without AI. This is due to big data⁴ and opening the doors to ML. ML is able to perform tasks that human beings are not capable of performing, by analyzing the large volumes of data and identifying patterns⁵ [32].

There is a very nice saying that dates back to 1913, “figures don’t lie, but liars do figure”- Mark Twain. Figures don’t have to lie i.t.o. suggest things that aren’t true when finding patterns in data. We as humans are so dependent on patterns that we tend to see patterns even if they aren’t there at all, this is because we need to find patterns, and the biggest challenge is to decide which are useful and which are not [32]. This is where AI excels.

2.3.2 Types of machine learning systems

There are many different types of machine learning systems, these learning systems are usually distinguished into the following three types: supervised learning, unsupervised learning and semi-supervised learning.

Algorithms that fall under supervised learning are trained with labelled datasets. The training data are composed of labelled training examples where each example consists of an input vector and desired output value. A typical example for a supervised learning task is classification i.e. the spam filter where it is trained with many emails along their class (spam), and must learn how to classify new emails as seen in figure 2.8 [5].

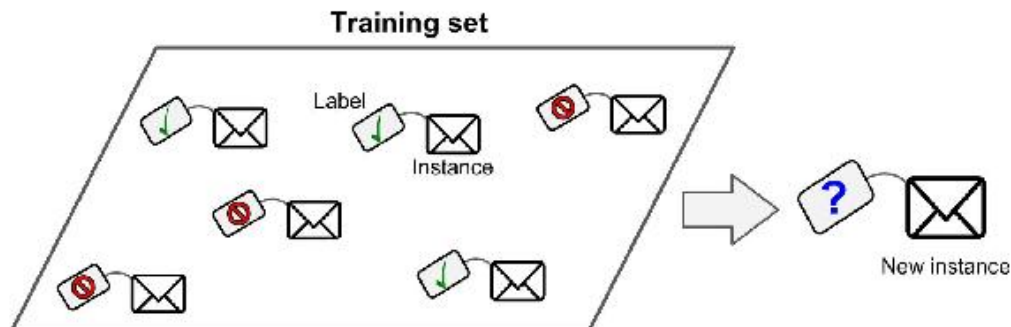


Figure 2.8: A labeled training set for supervised learning (e.g., spam classification) [5].

Unsupervised learning is where the training data are composed of only input parameters and the algorithm is left in the dark, and doesn’t know the correct answer. The goal

⁴Big data is extremely large data sets that may be analyzed computationally to reveal patterns.

⁵Temporal, spatial, structured and unstructured patterns.

is to learn representations and uncover some hidden structures for all the observations that make up the input space [32]. Figure 2.9 gives an example of an unlabelled training dataset.

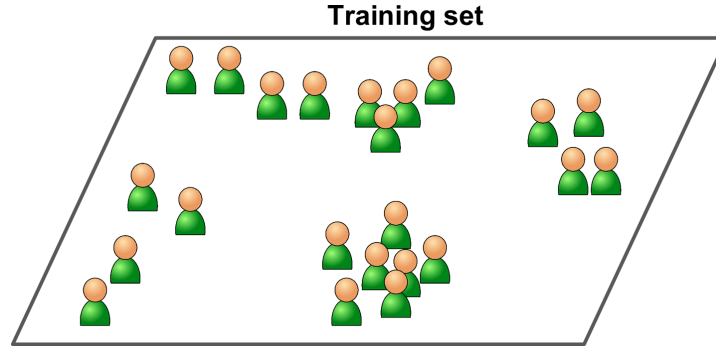


Figure 2.9: An unlabelled training set for unsupervised learning [5].

Semi-supervised learning is very similar to supervised learning, the difference being that both labeled and unlabeled datasets are used as the input. Figure 2.10 gives an example of a semi-supervised learning dataset. It consists of small amounts of labelled data and larger amounts of unlabeled data. Photo-hosting services is a good example of semi-supervised learning, such as Google Photos. Once one's family photos have been uploaded to the service, it will automatically recognize if a person shows up in more than one photo i.e. Person A shows up in photo 1, 2 and 4 and person B only in photo 2 and 3 [5]. Most of the time semi-supervised learning will be a combination of both supervised and unsupervised learning. Figure 2.11 shows some examples for each learning method. Some of the other methods that are not explored in this paper are reinforcement learning, batch and online learning, instance-based learning and model-based learning.

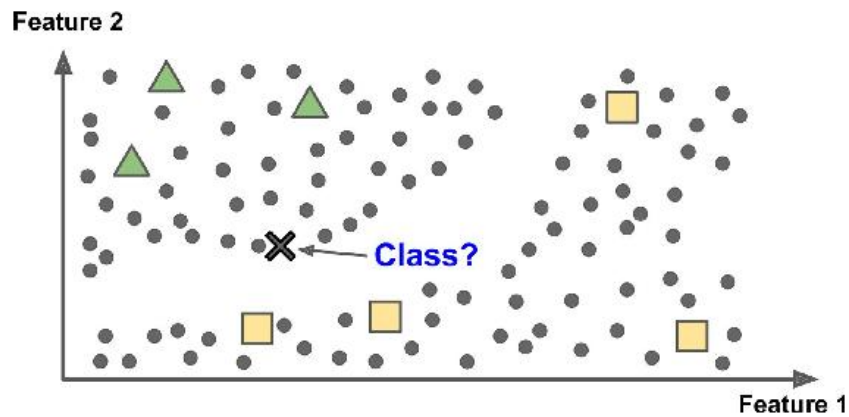


Figure 2.10: Semi-supervised learning [5].

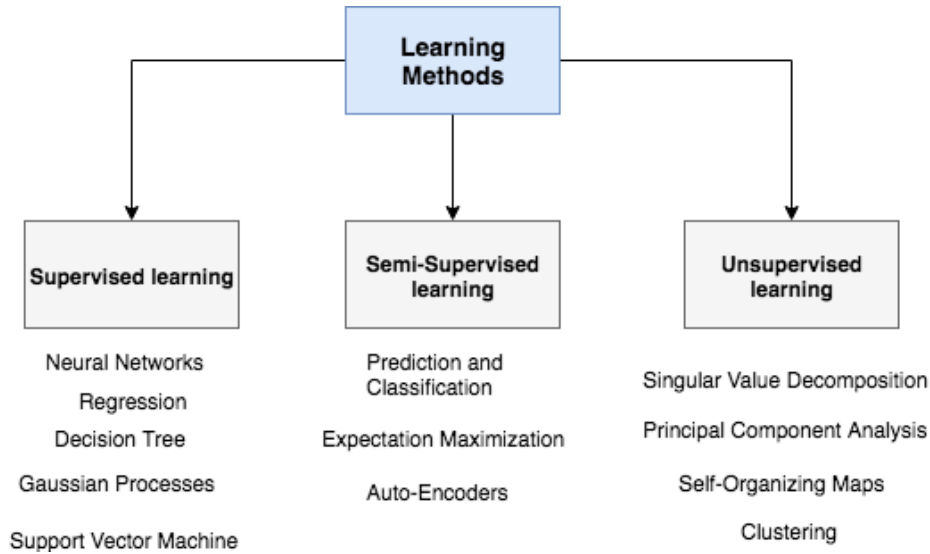


Figure 2.11: Learning Methods with examples of each method.

2.4 Machine Learning Techniques

Machine learning is a data analytics technique that teaches computers how to do tasks that come naturally to humans. There is a lot of algorithms that can be applied to almost any data problems, these include Linear Regression, Logistic Regression, Decision Trees, SVM, Naive Bayes, K-NN, K-Means, Random Forest etc.

This section looks at some of these machine learning techniques that are commonly used for classification tasks.

2.4.1 Logistic Regression

Logistic Regression is a classification algorithm, even though the name suggests that it's a regression algorithm [36]. Logistic Regression algorithms are used to estimate discrete values based on a given set of independent variables, it is therefore the go-to method for binary classification [37]. It is also known as logit regression since it fits the data to the logit function⁶ to predict the probability of the occurrence of an event. The sigmoid function (equation 2.4) maps any real value into another value between 0 and 1. $S(z)$ is the output between 0 and 1, z is the input to the function (the algorithm's prediction) and e is the base of natural log. The sigmoid function can be seen in figure 2.12.

$$S(z) = \frac{1}{1 + e^{-z}} \quad (2.4)$$

⁶The logit function is better known as the sigmoid function.

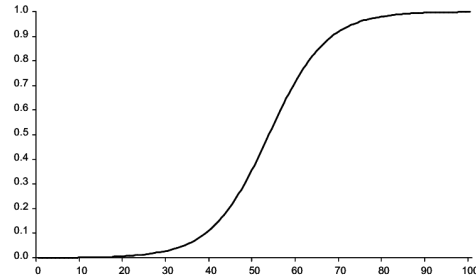


Figure 2.12: The sigmoid function

In order to map this to a discrete class, a threshold value is selected which will classify values into different classes. For example, if binary classification⁷ is used and the threshold was 0.5 and the prediction returned 0.7, then this observation is classified as class 0. If the prediction returned 0.2, then this observation is classified as class 1. In other words, if $p \geq 0.5$, $class = 0$, if $p \leq 0.5$, $class = 1$.

There are three types of logistic regression:

- Binary (Pass/Fail)
- Multi (Cats, Dogs, Sheep)
- Ordinal (Low, Medium, High)

Multiclass logistic regression is used in this project, since there are more than two classes. Instead of $y = 0/1$, the definition is expanded to $y = 0,1\dots n$ where n is the number of classes. Basically this means that the binary classification is re-run multiple times, once for each class

2.4.2 Linear Support Vector Machine

Linear Support Vector Machines (SVM) are simple algorithms that every ML expert should have in his/her arsenal. SVMs are very powerful and versatile Machine Learning models, capable of performing from regression problems to linear or nonlinear classification problems, and even outlier detections [5]. SVMs are particularly good at classification or regression tasks that are complex but derived from small- or medium-sized datasets.

⁷There are only two classes with binary classification.

Hyperplanes and Support Vectors

The objective of a SVM is to find a hyperplane in a N-dimensional space that distinctly classifies the data points. Hyperplanes are decision boundaries between the classes that help classify the data points. There are many possible hyperplanes that can be chosen to separate two planes, the objective is to find a plane that has the maximum margin, for example having the maximum distances between the data points of both classes [38].

As seen in figure 2.13, support vectors are the data points that are closer to the hyperplane and therefore influences the hyperplanes' orientation and position. These support vectors help build the SVM by maximizing the margin of the classifier. Figure 2.14 shows small and large classifier margins [38].

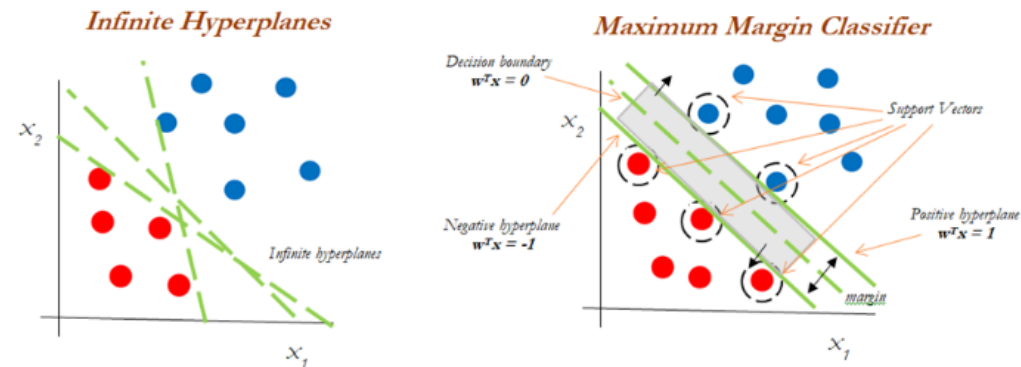


Figure 2.13: Infinity Hyperplanes and Maximum Margin Classifier [39].

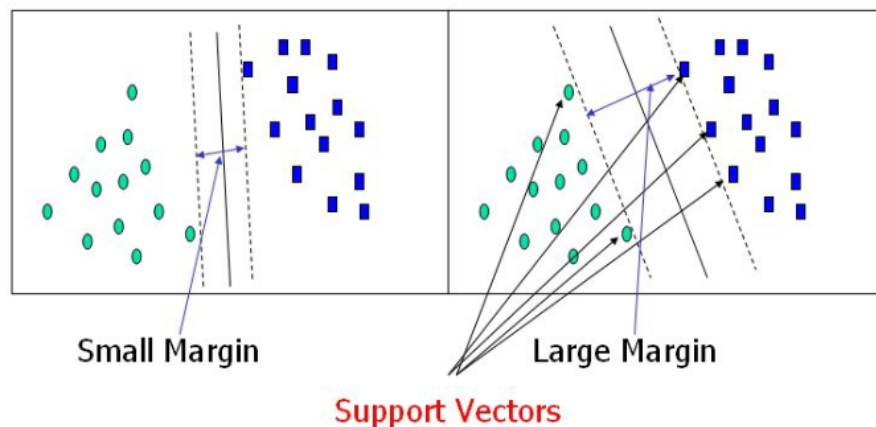


Figure 2.14: Support Vectors with a small margin and a large margin [38].

Cost Function and Gradient Updates

To maximize the margin between the data points and the hyperplane, the hinge loss function is used. The mathematical derivations are adapted from Ghandi's paper [38].

$$c(x, y, f(x)) = \begin{cases} 0 & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x) & \text{otherwise} \end{cases} \quad (2.5)$$

If the actual value and the predicted value are of the same sign, the cost is 0, if not, the loss value is calculated. A regularization parameter is also added to the loss function to balance the margin maximization and loss. The loss function with the regularization parameter added can be seen in equation 2.6.

$$\min_w \lambda \|w\|^2 + \sum_{i=1}^n (1 - y_i \langle x_i, w \rangle)_+ \quad (2.6)$$

Next, the gradients are obtained by taking the partial derivatives with respect to their weights (equation 2.7 and 2.8). Using the gradients, the weights are updated.

$$\frac{\delta}{\delta w_k} \lambda \|w\|^2 = 2\lambda w_k \quad (2.7)$$

$$\frac{\delta}{\delta w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0 & \text{if } y_i \langle x_i, w \rangle \\ -y_i x_{ik} & \text{otherwise} \end{cases} \quad (2.8)$$

If there is no misclassification and i.e the model correctly predicts the class of the data points, only the gradient from the regularization parameter has to be updated (equation 2.9 and 2.10).

$$w = w - \alpha \cdot 2\lambda w \quad (2.9)$$

If there is a misclassification and i.e the model makes a mistake on predicting the class of the data points, both the loss and regularization parameter are included to perform a gradient update.

$$w = w - \alpha \cdot (y_i \cdot -2\lambda w) \quad (2.10)$$

2.4.3 K-Nearest Neighbor

K-Nearest Neighbor, also known as K-NN is a simple non-parametric supervised classification algorithm, it gives pretty high accuracies, is versatile⁸ and is one of the most used learning algorithms. K-NN being non-parametric means that it does not make any assumptions on the underlying data distributions (the model structure is determined by the data). When there is very little or no prior knowledge about the data

⁸It is useful for classification and regression.

distributions, K-NN should be one of the first choices for a model. K-NN is also a very lazy algorithm, this does not mean that the algorithm does nothing. It only means that it does not use the training points to do any generalizations. There is no explicit or very little training, making it very fast, most of the training data is used during the test phase.

The principle behind K-NN is to find a predefined number of training samples closest in distance to the new point, and then predict the labels from these. In other words, K-NN computes the distance between new data points with every training example using Euclidean distance, Hamming distance or Manhattan distance to compute the distance measurements. Euclidean distance is one of the most popular methods used. It is essentially the magnitude of the vector obtained by subtracting the training data point for the point to be classified, the general formula is shown in equation 2.11. The model then picks K entries from the database which are the closest to the new data points and takes a majority vote being the most common labels among those K data points that will be the class of the new data point.

$$E(x, y) = \sqrt{\sum_{i=0}^n (x_i - y_i)^2} \quad (2.11)$$

A few applications of K-NN include credit ratings, political voting, handwriting detection, image recognition and even video recognition. K-NN requires a lot of memory since it stores all of the training data which it uses as its representation, making it computationally expensive. The algorithm is very good at predicting on small or medium sized datasets, but once the data size gets fairly big, it becomes very slow at making predictions. Figure 2.15 shows a simple example of K-NN algorithm. With $K = 3$, Class B will be assigned and with $K = 6$, Class A will be assigned.

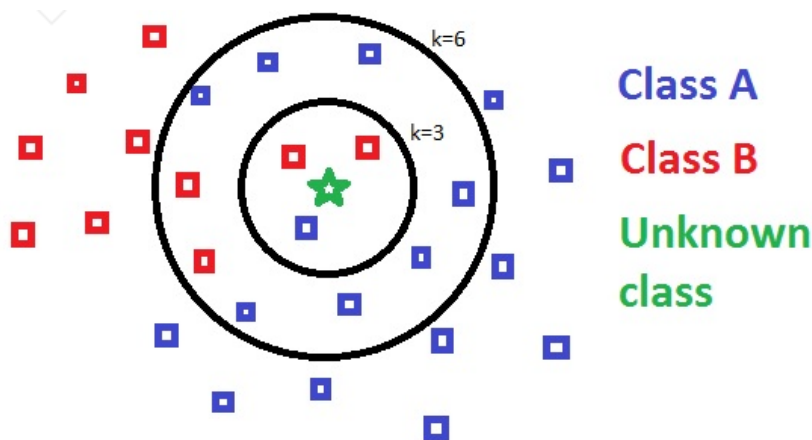


Figure 2.15: A simple K-NN example

2.4.4 Random Forest

The random forest algorithm is a more advanced version of the decision tree algorithm using multiple decision trees. As the name suggests, a decision tree is a tree-like model of decisions. The random forest algorithm is an easy to use, flexible machine learning algorithm. One of its big advantages is that it can be applied to both classification and regression problems. The name ‘random forest’ comes from the fact that each decision tree in the forest considers a random subset of features when forming the questions while only having access to a random set of the training data points [40]. The hyperparameters in a random forest can either be used to increase the model’s predictive power or to increase the model’s speed, although this isn’t necessary since the default hyperparameters often produce good prediction results.

With random forests, the problem of overfitting can be avoided given there are enough trees in the forest. The main limitation with the random forest algorithm is that a large number of trees can make the algorithm slow and ineffective for real-time predictions [41]. Figure 2.16 shows a simple temperature prediction example of the building blocks of a random forest, a single decision tree. For each question there is only one of two possible answers, true or false. Each answer is the start of a separate branch and eventually ending with a prediction.

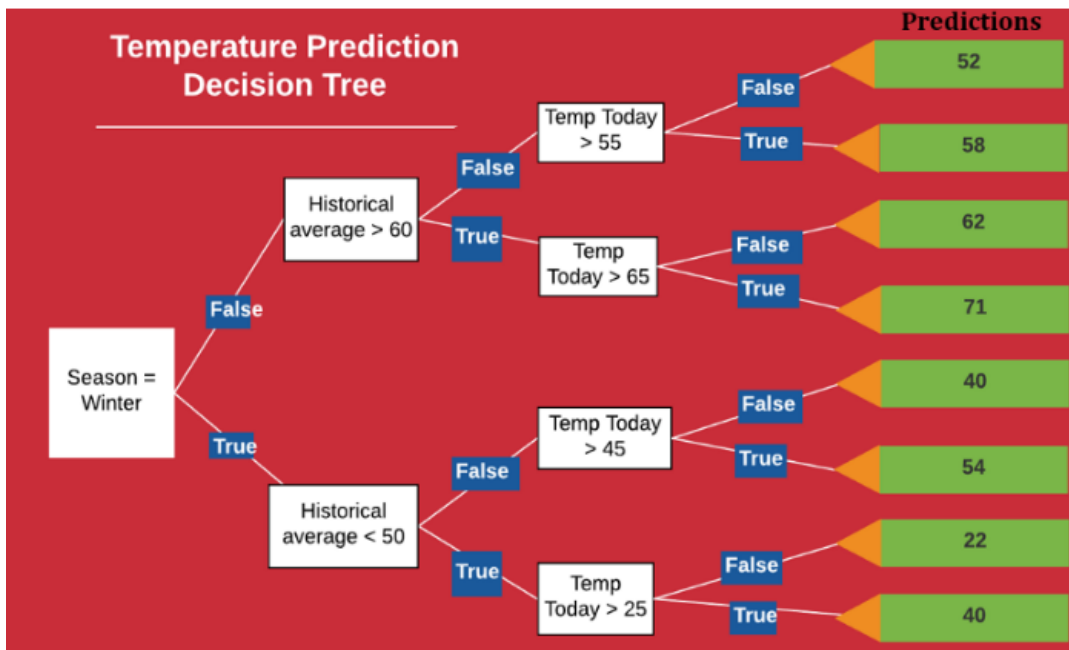


Figure 2.16: Decision Tree for Temperature Prediction [40].

So what makes a random forest algorithm better than a single decision tree? Each individual decision tree brings forth its own background experience and information sources to the problem. This increases the diversity in the random forest and makes the algorithm more robust on all of its predictions. When it comes to the point of making a prediction, the random forest algorithm takes an average of all of its individual decision tree estimates and makes a prediction in the case of a regression problem. In the case of a classification problem with discrete class labels such as cat or dog, the random forest will take a majority vote for the predicted class [40]. The algorithm for random forests is taken from [42] and shown in Algorithm 1.

Algorithm 1: Random Forest for Regression or Classification [42].

1. For $b = 1$ to B :
 - (a) Draw a bootstrap sample^a Z^* of size N from the training data.
 - (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
 - i. Select m variables at random from the p variables.
 - ii. Pick the best variable/split-point among the m .
 - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $T_{b_1}^B$.

To make a prediction at a new point x :

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of both the b th random-forest tree. Then $\hat{C}_{rf}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$.

^aThe bootstrap method involves iteratively resampling a dataset on random with replacement.

2.5 Artificial Neural Networks

This section introduces the concepts of artificial neural networks (ANN).

2.5.1 Introduction

Birds have inspired us to fly, fish have inspired us to swim and there are so many other inventions inspired by nature. Therefore it only makes sense to look at the architecture of the brain to get inspiration on building intelligent machines, this brings us to the area in computer science and engineering, namely Artificial Neural Networks.

The human brain is composed of about 86 billion nerves cells which are called neurons. These neurons are composed of a cell body containing a nucleus and most of the cell's complex components, as well as many branching extensions called dendrites. The cell body connects to thousands of other cells surrounding it through very long extensions called axons [43]. At the tip of the axon branches are the synaptic terminal, also called synapses. The neurons send or receive short electrical impulses called signals via these synapses. If a neuron receives a signal within a few milliseconds, it reacts if the incoming signal crosses a certain threshold, by sending its own signal. Each neuron typically connects to thousands of other neurons, and even though these neurons act in a fairly simple way, these vast network of billions of neurons can perform highly complex computations with their combined effort [5]. Figure 2.17 shows an example of the biological neuron.

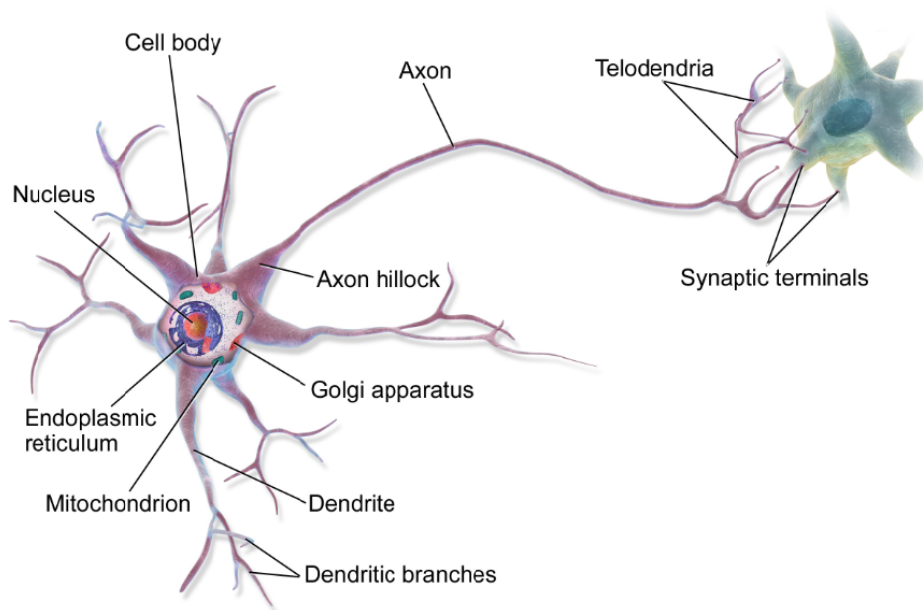


Figure 2.17: The biological neuron [5].

ANNs are composed of multiple interconnected nodes (also known as neurons), which imitate the biological neurons of the human brain. Each neuron communicates with

other neurons through the connections known as links. These links are each associated with a weight, and the ANN is capable of learning by altering the weight values. Each neuron serves as a simple processing unit. The input-weight products are summed and then passed through an activation function, this sum determines whether and to what extent the signal progresses further through the NN and ultimately affect the outcome as seen in figure 2.18a. If the signal gets passed through a neuron, that neuron is said to have been “activated” [44].

The neurons within an NN are structured into layers, a typical NN consists of an input layer, one or more hidden layers, and an output layer as seen in figure 2.18b. Each layer receives an input from the previous layer before passing its output to the next layer and finally reaching the output layer.

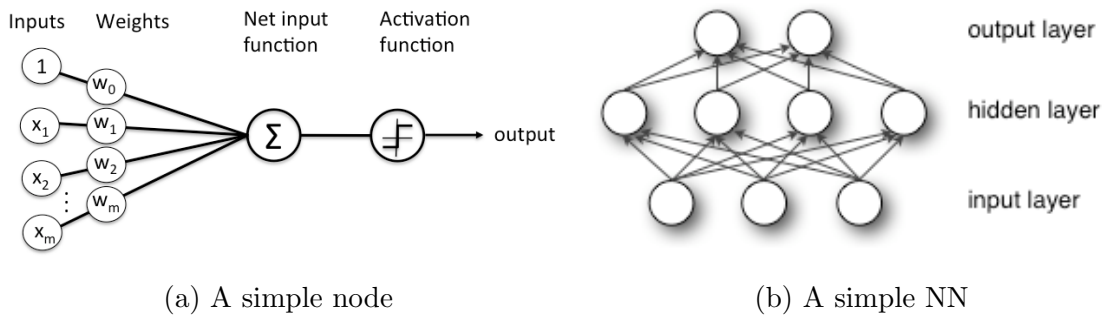


Figure 2.18: Neural network elements [44].

The width of a NN is determined by the number of neurons in each layer and the depth of a NN is determined by the number of hidden layers, when a NN has more than 3 hidden layers, it is considered a deep neural network (DNN). This project focusses on classification, therefore the output layer consists of one neuron per class.

2.5.2 Fine-tuning model hyperparameters

Getting used to a guitar out of tune is like creating a bad habit, tuning a guitar is crucial during the stage of learning, because you are creating connections between your different senses. So is the case with hyperparameter tuning for machine learning and deep learning. Hyperparameters are variables that needs to be set before a learning algorithm can be applied to a dataset. However, the challenge with hyperparameters is that there is no magic number that will work everywhere.

Hyperparameters can be divided into two categories:

1. Optimizer hyperparameters

2. Model specific hyperparameters

Optimizer hyperparameters include learning rate scheduling, minibatch size and the number of epochs. They are more related to optimization and training and are discussed in section 2.6.3. On the other hand, model specific hyperparameters are more involved in the structure of the model [45]. There are many hyperparameters to tweak in a NN, furthermore, there are an inconceivable number of different network topologies that might work on a certain problem and are therefore commonly seen as a type of art. Since it takes a long time to train on large datasets, a grid search⁹ with cross-validation¹⁰ won't be able to find all the right hyperparameters. This subsection explains ways to restrict the search space for finding the right hyperparameters.

Number of hidden layers

A network with a single hidden layer would give a reasonable result for many problems, a single hidden layer can model some of the most complex functions provided it has enough neurons. Although shallow networks work, DNNs have a much higher parameter efficiency. Multiple hidden layers do not only help DNNs to converge faster to a good solution, but also improves their ability to generalize better on new data sets. To summarize, for many problems such as a model for the MNIST dataset¹¹ only one or two layers are required. As the complexity of the problem increases, hidden layers can be added until the model starts overfitting the training set and just try to memorize the dataset. Overfitting is when the model performs very well on the training data, but does not generalize well. Underfitting is the opposite of overfitting, it is when a model is too simple to learn the underlying structure of the data [5].

Number of neurons per hidden layer

The number of hidden units is the main measure of a model's learning capacity [45]. The type of input and output the task requires determines the number of neurons the input and output layers require, for example, the MNIST task requires $28 \times 28 = 784$ input neurons (the MNIST images are 28×28 pixels) and 10 output neurons (one neuron per class) [5]. As for the hidden layers, it is a common practice to have fewer and fewer neurons at each layer until reaching the number of neurons equal to the number of classes. The network usually takes the shape of a funnel. Unfortunately, it is not an easy task of finding the perfect number of neurons, evidently it remains rather somewhat of a black art [5].

⁹A grid search is the process of performing hyperparameter tuning in order to find the optimal values for a given model.

¹⁰See section 2.6.4 on Cross-validation.

¹¹The MNIST dataset is a large database of handwritten digits that are commonly used for training and testing machine learning algorithms.

A simple approach would be to choose a model with more neurons and layers than what the task requires and then to use some sort of regularization technique such as dropout or early stopping which can be used to stop it from overfitting [5]. Dropout and early stopping are explained in section 2.6.3.

Activation functions

An activation function computes a scaled output of the weighted sum of the inputs. In most cases the ReLU activation function is used in the hidden layers since it is a bit faster to compute than other functions, and the Gradient Descent does not get stuck as much on plateaus [5]. Gradient descent is an optimization algorithm used to minimize some sort of function by iteratively moving in the direction of the steepest descent which is defined by the negative of the gradient [46]. The most popular activation functions used as well as the ReLU activation function are listed below:

- **Linear function**

$$A(x) = cx \quad (2.12)$$

A straight line function where activation is proportional to the input. They are used on very limited occasions since linear functions are usually not a good option for using as an activation function. The purpose of using an activation function is to introduce non-linearity into the network.

- **Sigmoid function**

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.13)$$

The sigmoid function is also known as the logistic function, it is the go-to method in binary classification as mentioned in section 2.4.1. It's output ranges between 0 to 1.

- **Hyperbolic function (tanh)**

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2.14)$$

The hyperbolic function looks very similar to the sigmoid function, as a matter of fact, it is just a scaled version and it is bound to range (-1,1). It is also a very popular activation function, deciding between the sigmoid and tanh function will depend on the requirement of the gradient strength [47].

- **Rectified linear unit function (ReLU)**

$$A(x) = \max(0, x) \quad (2.15)$$

ReLU is not bounded, its range is from [0,inf). ReLU enables sparse and efficient representations inside the NN, by ideally not activating some of the neurons [47].

the output clips negative values of x to 0. ReLU is less computationally expensive than tanh and sigmoid activation functions, since it involves simpler mathematical operations.

- **Exponential linear unit function (ELU)**

$$A(x) = \begin{cases} x & \text{if } x \geq 0 \\ \sigma(e^x - 1) & \text{otherwise} \end{cases} \quad (2.16)$$

ELU is known to converge cost to zero faster, increasing the learning process. ELU is very similar to the ReLU function except for negative values. Unlike ReLU, ELU produces negative outputs, setting the lower bound to $-\alpha$, where alpha is a tunable hyperparameter with $\alpha \geq 0$ [48].

- **Softmax function**

$$A(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.17)$$

For the output layer, the softmax activation function is generally a good choice for classification tasks. The softmax for a vector x is defined for each of its components x_i to satisfy the following two conditions [49]:

1. $A(x_i) \geq 0$
2. $\sum_j A(x_m) = 1$

Summary

Hyperparameters cannot be learned from data, and have to be entered manually. Finding the right values for hyperparameters and choosing the right activation function to use can be a very frustrating task, which can lead to underfitting or overfitting the machine learning models. The right values can be chosen by setting different values, training different models, and choosing the values that test better.

2.6 Deep Learning

As mentioned before, there are many traditional machine learning techniques such as logistic regression, decision trees, SVM, KNN, random forests, and more that are incredibly powerful when used in the right ways and in the right applications. They can make accurate and efficient predictions when using small datasets, because they are computationally inexpensive. This is not the case when using large datasets since they quickly become much more time consuming to compute and the errors also tend to increase. For computing large datasets in modern neural networks with high level features can be a difficult task to analyze without sophisticated mathematical

techniques, introducing a fast trending approach, the so-called Deep Learning (DL) [50].

The “deep” part of deep learning refers to creating deep neural networks. In other words, a network with a large number of layers with additional weights and biases. This improves the networks ability to approximate much more complicated functions. Deep learning is a branch of machine learning which is completely based on artificial neural networks. Deep learning algorithms have shown superior learning and classification performance in areas such as natural language processing, image processing, image recognition, visual art processing among many others [51], [35], [52]. The idea of deep learning is to automatically learn multiple levels of representations (by automatically extracting high- and low-level features necessary for classification purposes) of the underlying distributions of the data to be modelled [51]. Deep learning outperforms traditional machine learning techniques when the dataset size becomes very large and really excels when it comes to very complex problems such as natural language processing, speech recognition, and image classification.

Implementation of new ideas and architectures is enabled by novel software frameworks such as Caffe, Deeplearning4j, H2O, MXNet, Tensorflow, Theano, Torch and Keras [5]. In the following, deep learning models used in this thesis are explained, namely Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN). See section 2.6.1 and section 2.6.2.

2.6.1 Convolutional Neural Networks

Introduction

A study by David H. Hubel and Torsten Wiesel in 1958 and 1959 gave crucial insights on the structure of the visual cortex [5]. They were able to show that many neurons in the visual cortex have small local receptive fields, which means that they only react to visual stimuli located in that limited region of the visual field. These receptive fields of different neurons may overlap, and together they form the whole visual field. They showed that some neurons react only to images of horizontal lines, while others reacted to lines of other orientations. They also noticed that some neurons have larger receptive fields than others, and reacts to more complex patterns that are combinations of the lower-level patterns [5].

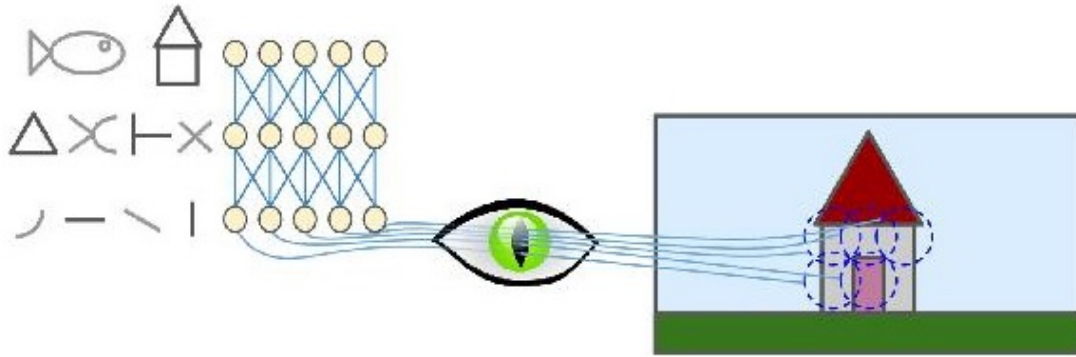


Figure 2.19: Local receptive fields in the visual cortex [5].

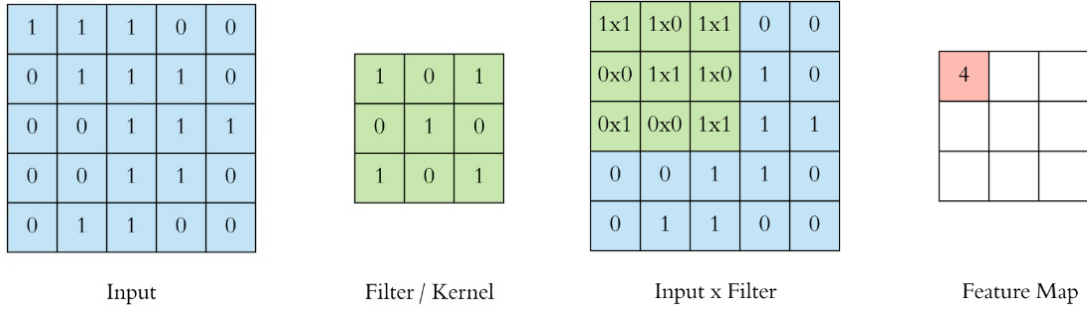
These observations led to the idea that the higher-level neurons are based on the outputs of their neighboring lower-level neurons. Figure 2.19 shows the local receptive fields as perceived by the eye and how each neuron is only connected to few of the previous layer. These studies of the visual cortex inspired the neocognition that was introduced in 1980, which gradually evolved into what today is known as convolutional neural network (CNN) [5].

CNNs are everywhere and arguably one of the most popular deep learning architectures due to its immense effectiveness, computational efficiency, and accuracy that blows competition right out of the water. Therefore, CNN is currently the go-to model for every image related problem. The recent surge in popularity started with AlexNet¹² in 2012 and has grown exponentially ever since. CNN is not only successful in image related problems, but also in natural language processing, recommended systems and more applications [53]. The main advantage of CNN compared with other architectures is that it automatically detects the important features without any human supervision [53].

Convolution

The convolutional layer is the main building block of CNNs. Convolution is a mathematical operation to merge two sets of information, in this case convolution is applied on the input data to produce a *feature map*.

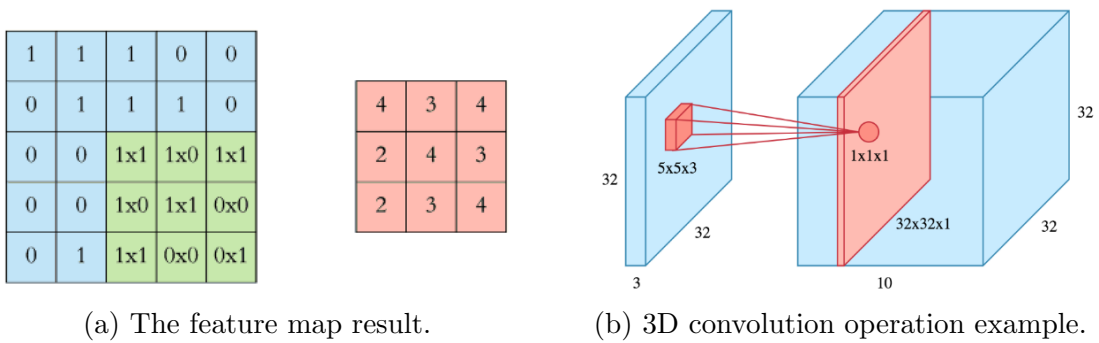
¹²The AlexNet CNN architecture achieved 17% top-5 error rate while the second best achieved only 26% [5].



(a) The Input and Filter/Kernel. (b) The Input x Filter and Feature Map.

Figure 2.20: The convolution operation [53].

On the left of figure 2.20a is the input to the convolution layer that represents for example the input image. On the right is the convolution filter, also known as the kernel. This is a 3 x 3 convolution due to the shape of the filter. The convolution operation is performed by sliding the filter over the input, an element-wise operation is performed at every location and the result is then summed. The summed result goes into the feature map. The green area in the left of figure 2.20b is called the receptive field with a size of 3 x 3. The filter is slid across the whole input, adding the convolution result to the feature map (See figure 2.21a). In reality, an image is represented in 3D with the dimensions of height, width and depth. The depth corresponds to RGB colors. A 3D convolution is explained in the form of an example, figure 2.20b shows a 32 x 2 x 3 image when a filter of size 5 x 5 x 3 is used. When the filter is at a particular location, it covers a small volume of the input, and the convolution operation is described above. The only difference is that this time the sum of matrix multiply is in 3-D. The feature map is of size 32 x 32 x 1 and is shown in figure 2.21b as the red slice on the right [53].



(a) The feature map result. (b) 3D convolution operation example.

Figure 2.21: Feature map result and a 3-D convolution operation example [53].

If ten different filters are used, ten feature maps of size 32 x 32 x 1 that are stacked together along the depth dimension give the final output convolution layer, a volume of

size $32 \times 32 \times 10$. The height and width stay unchanged and this is due to padding, which is explained shortly. Figure 2.22 shows two feature maps stacked along the depth dimension where the convolution operation is performed on each filter independently and the resulting feature maps are disjointed [53].

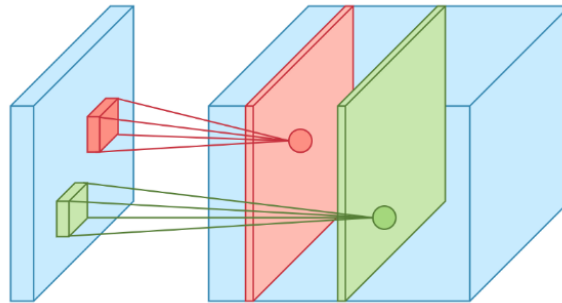


Figure 2.22: Two feature maps stacked along the depth [53].

Non-linearity

For any NN to be powerful, it needs some sort of non-linearity. The result of the convolution operations are passed through an activation function to achieve the non-linearity. So the values of the final feature map are sums with the activation function applied to them. Without applying an activation function, the CNN won't achieve its full potential [53].

Stride and padding

A *stride* defines how much the convolution filter is moved at each step, the default value is set to one. To have less overlap between the receptive fields, the *strides* can be made bigger [53]. If the dimensionality of the feature map needs to be maintained, *padding* can be used to surround the input with zeros. *Padding* is commonly used with CNNs to preserve the size of the feature maps, otherwise the size of the feature map would shrink after each layer [53].

Pooling

The pooling layers are the second common building block of CNNs, these techniques are usually performed to reduce the number of parameters, computational load and memory usage. These both shorten the training time and combats overfitting. Each neuron in the pooling layer is connected to the output of a limited number of neurons of the previous layer that are located in a small receptive field, just like with the convolution layer. Just like before, the size, stride and padding type have to be defined. The difference being that the pooling neurons have no weights, all it does is to aggregate the inputs using an

aggregation function such as max or mean [5]. The most common type is *max pooling*, it slides a window over its input and simply takes the maximum value at each window [53].

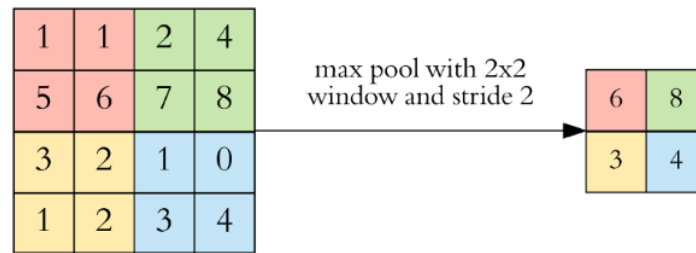


Figure 2.23: Max pooling layer (2 x 2 pooling kernel, stride 2, no padding) [53].

As an example, a 2 x 2 pooling kernel with a stride of 2 and no padding is used. Note how only the max input value makes it to the next layer in each kernel, the rest is dropped out. The window and stride configuration halve the size of each feature map as seen in figure 2.23. If the input to the pooling layer has the dimension of 32 x 32 x 10, using the same pooling parameters as described above, the feature map becomes 16 x 16 x 10. The height and width are halved while the depth dimension stays the same. If the CNN architecture contains millions of weights, this dimensionality reduction plays a pretty big role [53].

Hyperparameters

There are four important hyperparameters that need to be set:

- **Filter size:** A filter size of 3 x 3 is typically used, but 5 x 5 or 7 x 7 are also used depending on the application [53].
- **Filter count:** The more filters are used, the more powerful the model becomes, but the risk of overfitting increases due to the parameter counts. Usually a small number of filters are started with at the initial layers, and progressively increase the amount of filters moving deeper into the network. The number of filters is a power of two anywhere between 32 and 1024 [53].
- **Stride:** Strides are a four-element 1-D array, where the two central elements are the vertical and horizontal strides [5]. It is usually kept at default value 1.
- **Padding:** Padding must either be set to “VALID” or “SAME”. If set to “VALID”, the convolution layer does not use padding and when set to “SAME”, the convolution layer uses padding where necessary [5]. Padding is usually used.

CNN architecture

A typical CNN architecture consists of a few convolution layers, then a pooling layer, then another few convolution layers, then another pooling layer, and so on. After each convolution layer, an activation function is applied [5]. At the top of the stack, to wrap the CNN architecture up, fully connected layers are added. Flattening is simply arranging the 3-D volume of numbers into a 1-D vector (see figure 2.24) [53]. Finally, the final layer outputs the prediction (e.g. a softmax layer that outputs the estimated prediction of each class) [5].

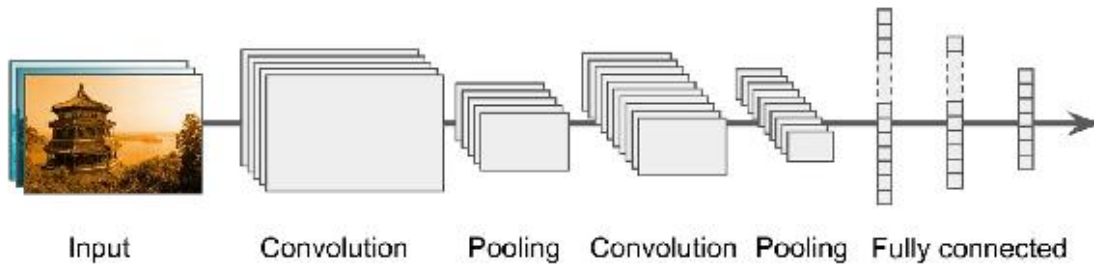


Figure 2.24: A typical CNN architecture [5].

2.6.2 Generative Adversarial Networks

Introduction

One of the biggest issues faced by machine learning is the lack of availability of large, labelled EEG datasets. Obtaining large EEG datasets for machine learning algorithms to make accurate predictions is not only expensive and time consuming, but is also highly dependent on the data not containing a lot of noise. The limited amount of data can inhibit the performance of supervised machine learning algorithms which most of the time need large amounts of data to train on to avoid overfitting. Data Augmentation is a common approach used in deep learning when the training data is limited. The number of training samples are increased by rotation, reflection, scaling etc. as explained in section 2.6.3. Obtaining more training data by augmenting the real training data can really reduce the chances of overfitting and not only improve the accuracy but also the ability to generalize of deep learning approaches [54].

Generative Adversarial Networks (GANs) offer a novel way to unlock additional information from an EEG dataset by generating synthetic sample images that look the same as the real images, thus increasing the size of the training dataset. Yann LeCun, a French computer scientist that works primarily in the fields of machine learning described GANs as “the most interesting idea in the last 10 years in machine learning”. It is always a

great advertisement for deep learning to receive such a compliment from such a prominent researcher in this field. GANs have had a huge success since it was first introduced by Ian J. Goodfellow and co-authors in the article Generative Adversarial Nets published in 2014 [55], [56].

Generative vs. Discriminative algorithms

Discriminative algorithms try to classify the input data to the corresponding targets as correct as possible. For example, given all the words in an email, a discriminative algorithm could predict whether the mail is SPAM or NOT. It learns the conditional probability distribution and can be expressed mathematically, the label is called y and the features are called x . The formulation $p(y|x)$, which means “the probability of y given x should be maximum” and would translate in this case to “the probability that an email is spam, given the words it contains”. An example of such a model is logistic regression or SVM’s [57].

Generative algorithms attempt to generate similar inputs and it’s labels from the target inputs. For example, the question generative algorithms try to answer is: Assuming that the email is SPAM, what likely are the features? It learns the joint probability distribution and can be expressed mathematically as $P(x, y) = p(x|y).p(x)$, the algorithms care about “how to get x and how the training data are generated/distributed”. An example of such a model is Naive bayes [57].

GANs Concepts

GANs are generative models that try to train the model to generate the input distributions as realistic as possible, the end goal is to predict the features given a label. A GAN consists of two neural networks: One of the neural networks, called the “Generator”, generates new data instances from some random uniform distribution. The goal is to create new similar type of fake data from the inputs. The other neural network is called the “Discriminator”, evaluates the authenticity of the fake data produced by the Generator from the real data [57].

The main idea behind GANs is to train the different networks to compete with each other using two different objective functions. The generator tries to fool the discriminator into believing that the input sent by the generator is real, while the discriminator identifies the data coming from the generator is fake. The generator then learns to produce a better, but similar type of training data inputs. This process continues until equilibrium is found, and is called Adversarial Training. As the generator gets stronger and stronger at generating real type of results, the discriminator also get stronger and stronger at

identifying the real and the fake ones [58]. Figure 2.25 illustrates the working process between the generator and the discriminator.

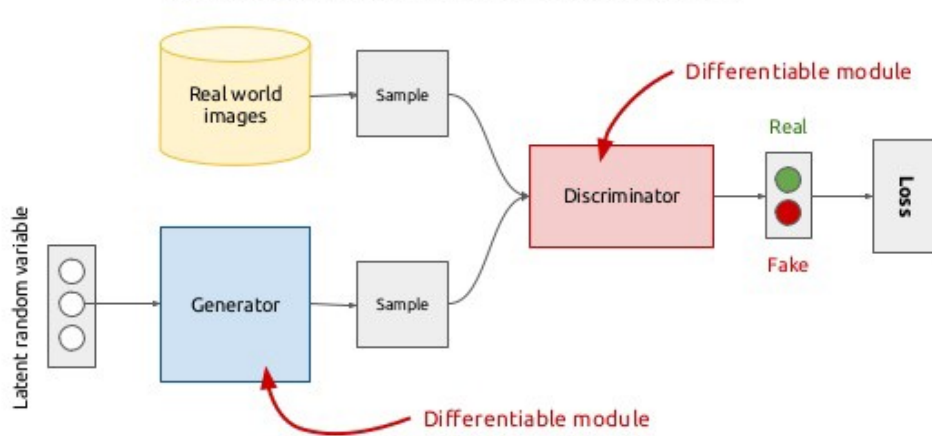


Figure 2.25: The working of the process between training the discriminator and the generator [58].

GANs objective function

Since the discriminator is a binary classifier, the model should produce a high probability for real data and low probability for fake data. The variables are defined as follows [58]:

$$z \Rightarrow \text{Noise vector}, \quad G(z) \Rightarrow \text{Generator's output} \Rightarrow x_{\text{fake}}$$

$$x \Rightarrow \text{training sample} \Rightarrow x_{\text{real}}$$

$$D(x) \Rightarrow \text{Discriminator's output for } x_{\text{real}} \Rightarrow P(y|x_{\text{real}}) \Rightarrow 0, 1$$

$$D(G(z)) \Rightarrow \text{Discriminator's output for } x_{\text{fake}} \Rightarrow P(y|x_{\text{fake}}) \Rightarrow 0, 1$$

As seen above, the functions $D(x)$, $D(G(z))$ gives a score between 0 and 1. The discriminator model should maximize the real data while minimizing the fake data and the generator model should maximize the fake data. At the discriminator, $D(x)$ and $D(G(z))$ should be maximized and at the generator. The final equation (equation 2.18) by Ian Goodfellow is in terms of the mathematical expectation from his paper [59].

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.18)$$

2.6.3 Training deep neural nets

There are many techniques that can help to improve the classification accuracy and/or generalization capabilities of a DNN. In this subsection, all the techniques that are most commonly used are briefly explained.

Adam optimization

There are many optimization algorithms available, some of the optimization algorithms are ADADELTA, ADAGRAD, ADAM, SGD, RMSPROP. ADAM performs well in practice and outperforms other adaptive techniques, and is therefore used in this thesis.

ADAM stands for Adaptive Moment Estimation and combines the ideas of Momentum optimization and RMSProp, it computes adaptive learning rates for each parameter. In addition, optimization methods such as AdaDelta which stores an exponentially decaying average of the past squared gradients, ADAM also keeps an exponentially decaying average of past gradients. In terms of convergence and learning speed, ADAM converges very fast and the learning speeds are quite fast compared to other adaptive learning techniques [60].

Learning rate scheduling

To obtain a good learning rate could be very challenging, if the learning rate is set too high, the training may actually diverge and if the learning rate is set too low, the training may eventually converge to the optimum after a long time. The ideal learning rate will converge quickly within the first few epochs, this can be done by starting with a high learning rate and reduce the learning rate as soon as the algorithm stops making fast progress. A good starting point is to set the learning rate to 0.01, if the learning rate is much smaller than the optimal value, it would take a much longer time to reach the ideal state. On the other hand, if the learning rate is much larger than the optimal value, then it would overshoot the ideal state and might not even converge [45]. There are many other strategies called learning schedules, the most common ones are the following [5]:

- **Performance scheduling** - Measure the validation error every N steps and reduce the learning rate by a factor λ when the error stops to drop.
- **Exponential scheduling** - Set the learning rate to a function of the iteration number t : $\eta(t) = \eta_0 10^{\frac{-t}{r}}$
The learning rate will drop with a factor of 10 for every r steps.
- **Power scheduling** - Power scheduling is very similar to exponential scheduling, but the learning drops much more slowly. The learning rate is set to $\eta(t) = \eta_0 (1 + \frac{t}{r})^{-c}$ where c is typically set to 1.

It is not necessary to add an extra learning scheduling when using AdaGrad, RMSProp or ADAM optimization since they automatically reduce the learning rate during training.

Minibatch size

Currently, a commonly used technique is to set the minibatch size. Stochastic training is when the minibatch size is set to 1, and batch training is when the minibatch size is set to the number of examples in the training set. A larger minibatch comes at the expense of needing more memory for the training process. However, it allows computational boosts that utilizes matrix multiplication in the training calculations. A smaller minibatch is useful in preventing the training process from stopping at a local minimum, but induces more noise in their error calculations. Minibatch values are of size 2^N , a good value to choose is usually 32 [45].

Number of epochs

To choose the right number of epochs for the training step, the focus must be on the validation error. The manual way of training is to let the model train as many numbers of iterations, as long as the validation error keeps on decreasing [45]. Early stopping is a technique that determines when to stop the training and is explained next.

Early stopping

A great solution to avoid overfitting during training is to make use of early stopping. Early stopping interrupts training when the performance on the validation set starts to drop or when the validation error has not improved the past 10-20 epochs [5].

Batch normalization

A technique called Batch Normalization (BN) was proposed in a paper by Sergey Ioffe and Christian Szegedy (2015) to address the vanishing/exploding gradients problem as well as the problem where the distributions of each layer's input changes during training, as the parameters of the previous layers change. It has been shown that BN improves generalization and accuracy, while significantly speeding up the learning process, especially with CNNs. BN also acts like a regularizer, which reduces the need for other regularization techniques such as dropout, described in section 2.6.3 [5].

Dropout

Dropout is probably the most popular regularization technique for deep neural networks. It is a fairly straightforward technique, at each training step, every neuron has a probability p of being disabled temporarily "dropped out". In other words, it will be disabled during that training step and may be active the next training step. Neurons are only dropped during the training process, and the hyperparameter is typically set to 50 % [5]. Dropout has been proven to be highly successful by even boosting the state-of-the-art neural networks with 1-2 % accuracy, it effectively makes sure that the network learns

generalized features rather than having to rely on each individual neural network's connection.

Data augmentation

One way to artificially boost the training set size is by generating new training instances from existing ones. The idea is to generate realistic training instances that humans should not be able to tell which ones are generated and which ones are not [5]. If for example the model is meant to classify pictures of a hamster, the training instances can be slightly shifted, rotated, resized etc. as seen in figure 2.26. Assuming the hamster is symmetrical, the image can be flipped as well. By combining these new training instances with the original, the training set size can be greatly increased. Another powerful way of using data augmentation to increase the training set size is to use Generative Adversarial Networks (GAN) as explained in section 2.6.2.

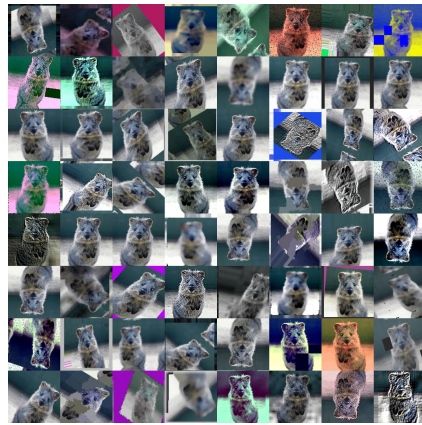


Figure 2.26: An example of Data Augmentation [61]

2.6.4 Histogram of Orientation Gradients

A standard approach to object recognition is the Histogram of Oriented Gradients (HOG). HOGs are used for feature reduction, to lower the complexity of the problem while also maintaining as much variations as possible. The idea behind HOG is to instead of using each individual gradient direction of each pixel in an image, the pixels are grouped into smaller cells (for example 8 x 8 pixels). For each cell, the gradient direction is computed and grouped into a number of smaller bins. The gradient magnitude is then summed up for each sample. So, stronger gradients contribute more weight than other to their bins, and effects of smaller gradients due to noise are reduced [62]. This histogram gives an idea of the more dominant orientations of that cell. Figure 2.27 shows an example of an 8 x 8 patch in the image and shows how the gradients look. Figure 2.28 shows how the histogram of orientations gradients are obtained for each cell.

To find the magnitude and direction of the gradient of each pixel equation 2.19 and 2.20 are used:

$$g = \sqrt{g_x^2 + g_y^2} \quad (2.19)$$

$$\theta = \arctan \frac{g_y}{g_x} \quad (2.20)$$

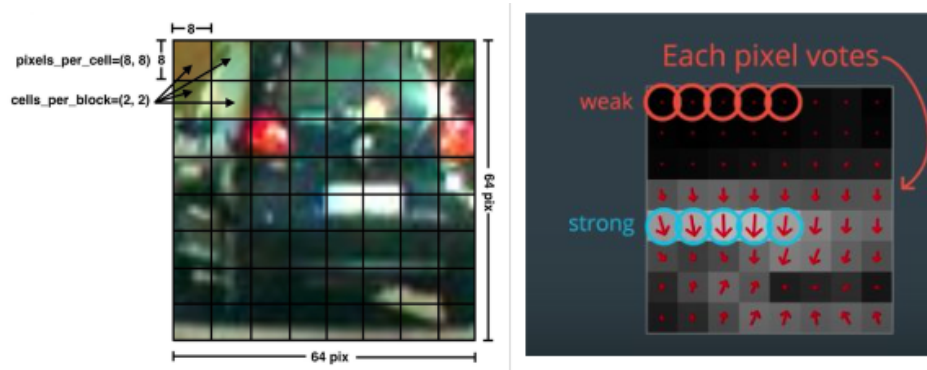


Figure 2.27: On the left is the RGB image and on the right is the 8 x 8 patch with arrows that represent the gradients. Retrieved from one of Udacity's lecture videos [62].

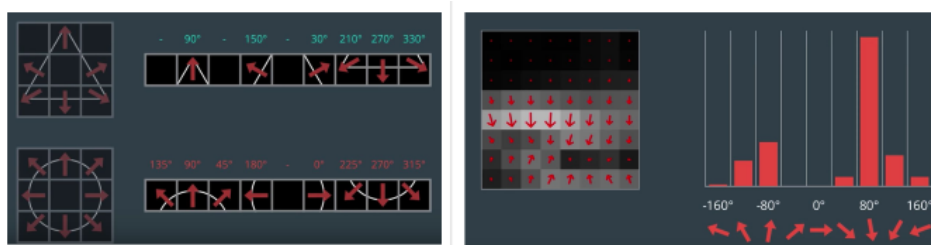


Figure 2.28: Histogram of Orientation Gradients. Retrieved from one of Udacity's lecture videos [62].

ℓ_1 and ℓ_2 Regularization

ℓ_1 and ℓ_2 can be used to constrain a neural network's connection weights and reduce overfitting by penalizing weights [5]. A model that uses ℓ_1 is called Lasso regression and a model that uses ℓ_2 is called Ridge regression. The key difference is the penalty term in the cost function. Specifically, ℓ_1 introduces the term $\lambda \sum_{j=1}^p |\beta_j|$ and ℓ_2 introduces the term $\lambda \sum_{j=1}^p \beta_j^2$ to the cost function as seen below in equation 2.21. The key difference is that Lasso regression shrinks the less important feature's coefficients to zero, therefore removing some features altogether [63].

$$\sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (2.21)$$

Cross validation

When training a machine learning model, one way of evaluating the model would be to use a `train_test_split` function that splits the training set into a smaller training set and a validation set. The validation set is used to measure the model's performance on unseen data. The model is then trained using the smaller training set and evaluated against the validation set. Therefore, *K-fold cross validation* is commonly used. When for example K is set to 10, it randomly splits the training set into 10 distinct subsets which are called folds. It then trains and evaluates the model 10 times, picking a different fold for every evaluation and training on the other 9 folds. This is done iteratively until every subset was used once for validation. The 10 evaluation scores are represented in an array giving the model's performance [5].

However, with deep learning, cross-validation is usually avoided since the cost associated with training K different models on large datasets is high. Therefore, it is common practice to rather split the dataset into training, validation and testing subsets chosen randomly. They are usually chosen as 70% / 15% / 15%. The model is trained as long as the validation error keeps on decreasing, early stopping can be used. The model is then tested on the test data split to measure the model generalization capabilities.

2.6.5 Performance measures

Evaluating the performance of machine learning algorithms is an essential part of any project. Most of the time classification accuracies are used to measure the performance of a model. However, it is not enough to truly judge a model's performance. This section covers different types of evaluation metrics that are available.

Classification accuracy

This is what is usually meant when using the term accuracy, classification accuracy is the ratio of the number of correct predictions to the total number of input samples. It works very well when there are equal amounts of samples belonging to each class.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \quad (2.22)$$

Confusion matrix

Confusion Matrix as the name suggests gives a matrix as an output that describes the complete performance of the model. Each row in the confusion matrix represents an actual class, while each column represents a predicted class [5]. Confusion matrices are explained in the form of an example (see Table 2.2). Assume having a binary classification problem having samples belonging to two classes: YES or NO. On testing the model on 165 samples, the following result is obtained.

Table 2.2: Confusion Matrix

n = 165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

There are four important terms [64]:

- **True Positives:** The cases where a YES was predicted and the actual output is YES.
- **True Negatives:** The cases where a NO was predicted and the actual output is NO.
- **False Positives:** The cases where a YES was predicted and the actual output is NO.
- **False Negatives:** The cases where a NO was predicted and the actual output is YES.

The accuracy for the matrix can be calculated by taking the average of the values across the main diagonal.

$$Accuracy = \frac{TruePositives + FalseNegatives}{TotalNumberOfSamples} \quad (2.23)$$

$$\therefore Accuracy = \frac{100 + 50}{165} = 0.91 \quad (2.24)$$

2.7 Related work

This section presents an overview of related work done in the field of classifying brain signals.

2.7.1 A novel approach for classifying EEG signal with multi-layer neural network

The purpose of the paper by Lam and Nquyen [65] was to classify EEG signals into 5 different classes (human-, animal-, landscape-, city- and flower images). The proposed technique that was used to convert the EEG signals is the Mexican hat Wavelet transform, then synthesize and normalize the input values for a multi-layer neural network to classify the signals. The experimental dataset was recorded from 21 participants over the course of two days using 32-channels. The proposed technique used in this thesis was implemented, using the MatLab and EEGLab toolbox, a high accuracy of 92.68 % was achieved with 40 hidden nodes in the hidden layer.

2.7.2 Multimodal classification with deep convolutional-recurrent neural networks for electroencephalography

The most notable features of EEG signals reside in the frequency dimension which is usually studied using a spectrogram. The feature vector formed by aggregating spectral measurements of all the electrodes is the traditional method of analyzing EEG signals [67]. These methods clearly ignore the locations of electrodes and the inherent information in spatial dimension. In this approach Tan et al. [66], the spatial structure is preserved by EEG images. The 3D locations of the electrodes are projected to 2D points by making use of Azimuthal Equidistant Projections (AEP) which interpolates and maps the positions to a 32 x 32 image¹³. The EEG images generated by AEP maintains the distance between the electrodes more accurately, which reflects more of the spatial information. The signals were recorded from nine participants using 22 electrodes, for each subject, two sessions were recorded on different days. The four participants had to perform four types of motor imagery, hence four classes. Their approach was compared with various classifiers commonly used in this field such as Support Vector Machines (SVM), Linear Discriminant Analysis (LDA) and Conv1D¹⁴. They constructed a deep network containing a CNN part and a recurrent neural network (RNN) part for the classification of the EEG signals. The EEG videos and optical flow were first fed into the CNN before fed into the RNN part.

¹³ [67] uses the same approach as [66] with AEP.

¹⁴Conv1D is a one dimensional convolutional neural network.

Table 2.3: Experiment results % showing [66]’s approach’s superiority above other traditional methods [66].

	S1	S2	S3	S4	S5	S6	S7	S8	S9	Avg	Std
SVM	78.8	51.7	83	61.8	54.2	39.2	83	82.6	66.7	66.78	15.25
CSP+LDA	78.1	44.4	81.9	59	39.6	50	80.9	68.4	77.1	64.38	15.62
Conv1D	78.8	53.1	82.6	60.4	59	43.8	82.6	83.3	81.2	69.42	14.45
Our approach(LSTM)	78.8	62.5	83	63.5	67.7	45.8	90.3	85.8	72.6	72.22	13.17
Our approach(GRU)	90.6	41	95.1	68.1	47.6	54.9	90.3	64.9	80.6	70.34	18.79

Furthermore, their approach of applying Long-Short Term Memory (LSTM) and Gradient Recurrent Unit (GRU) as the basic element of their RNN, achieved superior accuracy over the traditional methods. The LSTM approach achieved 72.22 % where as the GRU achieved 70.34 %. Figure 2.3 shows the experimental results achieved by the Tan et al. [66].

2.7.3 Deep learning human mind for automated visual classification

In 2017, Spampinato et al. [68] proposed the first human brain-driven automated visual classification method. Their method comprised of two stages: The first stage was a RNN-based method to learn visual stimuli-evoked EEG data as well as to find meaningful and more compact representation of such data. The second part was a CNN-based approach aiming at regressing images into the learned EEG representation, enabling automated visual classification. Seven participants were used by Spampinato et al.’s [68], the dataset used for the visual stimuli was a subset of ImageNet, containing 40 classes of easy recognizable objects (dog, cat, butterfly etc.). The images were shown to the participants in bursts of 0.5 seconds each using a 32-channel cap. Spampinato et al. [68] proposed approach of discriminating object classes using brain signals reached an average accuracy of about 40 %.

2.7.4 Multi-task generative adversarial learning on geometrical shape reconstruction from EEG brain signals

Recently, the advancements of deep generative models such as GANs have supported the object generation from brain signals. In July 2019, Zhang et al. [69] proposed a novel multi-task generative adversarial network that converts the individual’s EEG signals evoked by geometrical shapes to the original geometry. Firstly, they employed a CNN to learn highly informative latent representations for the raw EEG signals. Next, they adopted a multi-task discriminator with a task-specific classifier which assigns the geometrical shape that had evoked the individual’s EEG signals into the correct class for

Table 2.4: Experiment results of Zhang’s approach showing qualitative comparison of inception score and inception accuracy [69].

Models	GAN	C-GAN	ACGAN	Ours
Inception score	1.931	1.986	2.061	2.178
Inception Accuracy	0.43	0.67	0.79	0.83

the aim of improving the quality of the shape that has been recovered. In addition, the discriminator learns to distinguish and classify fake samples simultaneously, improving the quality of the recovered shape. Furthermore, they proposed a semantic alignment constraint in order to enhance the realism level of the reconstructed shape [69].

They conducted their experiment on eight healthy participants, where each participant was required to sit in a comfortable armed chair in front of a monitor. Their whole experiment consisted of two sessions, where each session had five trials. During each trial, five geometrical shapes¹⁵ are presented in a random fashion, where each shape lasted for 5 seconds. Their proposed approach was evaluated over a local dataset and the experiment showed prodigious results. They obtained an inception accuracy of 83 % with their model, which outperforms the state-of-the-art reconstruction methods both qualitatively and quantitatively. Figure 2.4 shows the experimental results achieved by the Zhang et al. [69].

2.7.5 Summary

Thanks to deep learning, other works as seen above have attempted to investigate structure and functionality of the brain by decoding the brain signals in a variety of applications. For example, in [69], a combination of GANs and CNNs have been used to learn highly informative latent representations for raw EEG signals. These works above have all shown and proven the potential of using brain signals and deep learning for classifying and reconstructing EEG signals. In this study, we explore not only the capabilities of deep learning, but also other traditional machine learning techniques for classifying geometrical shapes as perceived by the brain.

¹⁵Circle, star, triangle, rhombus, and rectangle.

Chapter 3

Methods and results

This chapter explains at first the general experimental design and processing of the EEG time-series data. The second part explains how the EEG time-series data are projected to EEG images. The third part explains proposed implementation for decoding the brain signals using machine learning and deep learning techniques. Finally, the performance of the models are evaluated.

3.1 Experimental setup

The experiment aims to investigate the potential of using deep learning methods to classify the geometrical shapes in the EEG recordings as seen by the participants. The investigated approach is to make use of traditional machine learning methods (logistic regression, KNN, SVMs etc.) as well as Convolutional Neural Networks (CNN) described in Section 2.4 and 2.6.1 respectively.

3.1.1 The data

Deep learning models usually consist of thousands of tunable parameters and effectively need huge amounts of training data to accurately optimize the model and achieve good performance on making predictions. EEG signals have been collected from 10 healthy participants over a time span of 12 months during the duration of this research project.

Protocol/Stimulus

PsychoPy is an open-source application psychology software implemented in Python programming language, covering a wide range of experiments from neuroscience, psychology and psychophysics experiments [70]. PsychoPy provides a unique choice between a Builder interface or a Coder interface. For the purpose of this project the Builder interface was used, since it is sufficient for building a rich and flexible experiment. Figure 3.1 shows the flow of the experiment in the Builder interface view. Each of the variables used is as follows:

- **Welcome:** The welcome screen that explains the experiment and gives instructions to the participant.

- `break_2`: Sets all the variable values for each phase/session¹.
- `trial_2`: Displays the stimuli on the screen.
- `trials`: Sets the number of stimuli to be shown for each phase/session.
- `trials_2`: Sets the number of phases/sessions.
- `end`: Displays the exit screen.

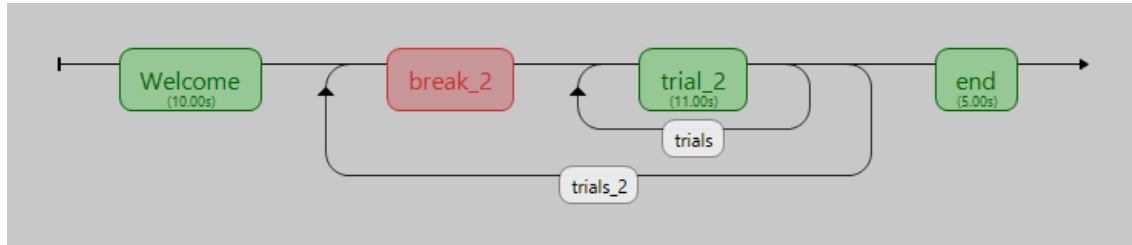


Figure 3.1: Builder interface flow

The “geometric” protocol

The EEG experiment was conducted in Stellenbosch, at the Neuromechanics lab. The 10 participants used were all Engineering students (six males and four females) between the ages of 20 and 30. The 10 participants had no history of mental issues and ethical approval for this study was obtained and the participants all gave their informed consent. During the experiment, the participants were seated in an arm chair a meter away from a LCD screen which they needed to watch. The participants were instructed to keep as still as possible during an active session: each participant was shown what happens to the EEG recordings when for instance they clenched their jaw. Each session consisted of six sessions, consisting of 72 trials in total per subject. Each session consisted of 12 trials, three for each class (square, triangle, frame, cross (see figure 3.3a)). The four classes were chosen because they are easily distinguishable². The participants were allowed to take a break after each session and to move around before continuing to the next session. Figure 3.2 shows the setup during one of the sessions.

¹Reason for 2 in the name is because it contains a variable with the same name.

²Only four classes were used to ensure the classification problem isn't over complicated

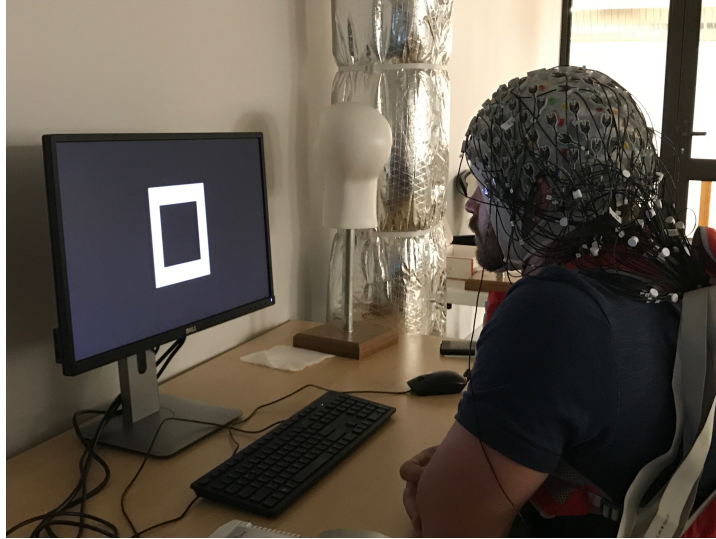


Figure 3.2: One of the participants during a EEG recording

Each trial has a duration of 15 seconds, at the beginning of each trial ($t = 0$ s), a black screen was shown for 1 second. The stimuli were shown in bursts of 500 ms. The geometrical shapes were all white in color and 8 inches in diameter. At $t = 1$ s a small fixation cross is displayed for 500 ms followed by the stimuli for 500 ms. This is repeated 10 times for each stimuli, at $t = 11$ s a black screen is displayed for 4 s before moving to the next trial. The EEG recording protocol can be seen in figure 3.3b for a better understanding. The background lights are dimmed during each session to eliminate any possible glare. The type of stimuli for each trial is randomly selected. The EEG data are recorded using the actiCAP 128 Channel standard from Brain Products with a sampling frequency of 500 Hz, the data recorded are saved without any filters applied. The bursts for each trial happened between $t = 1.5$ s and $t = 11$ s, resulting in a duration of 9.5 s. To project these time series data to images, a frame duration of 4.5 s was chosen with an overlap of 80 % for each image. For example, during one of the trials, a triangle was displayed in bursts. The time series data during which the bursts were shown was divided into frames (4.5 seconds each). From these frame durations, images were generated using an overlap of 80 %. Given the shape of the data as $A = (72, 128, 5250)$, where the first dimension is the number of trials, second the number of electrode channels and third the duration in which the stimuli was displayed in bursts. This data A is divided into a function that divides the duration into 4.5 s frames and generates images from these frames with a 80 % overlap. The resulting data shape will be $B = (6500, 28, 28, 3)$, where the dimensions are as follows:

1. Number of images.
2. Dimension 2 and 3 are the image shapes (28 x 28).

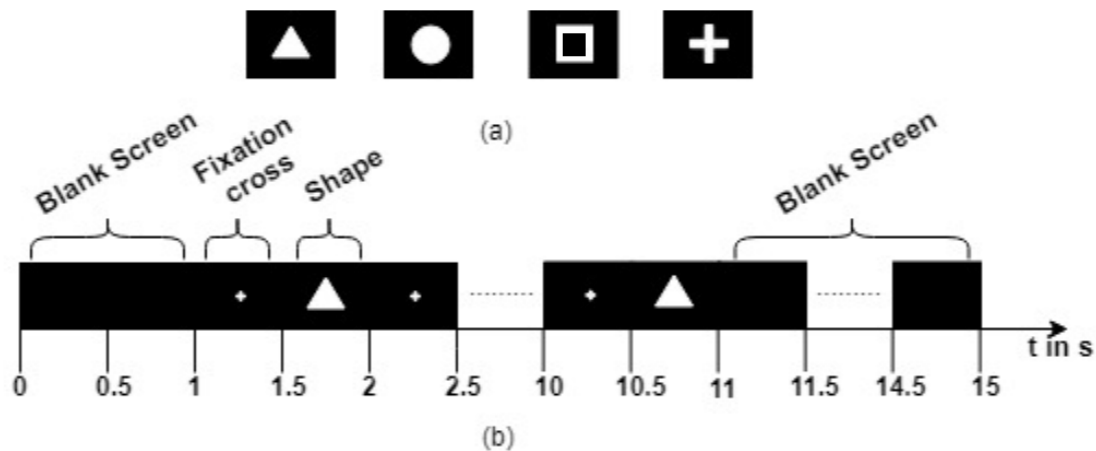
3. RGB³ colours.

Figure 3.3: (a) The 4 geometrical shapes as stimuli, (b) EEG recording paradigm for one of the stimuli.

3.2 EEG data preprocessing

This section explains how the preprocessing is implemented. Preprocessing is the procedure of transforming the raw EEG data into a more suitable format for further analysis which is more interpretable for the user. There is usually a lot of noise present in EEG data, which could obscure weaker EEG signals. The artifacts could be general background noise, natural noise that originates inside our brains due to the fact that our brains are constantly busy doing a lot of activities at a time, eye blinking or muscle movements as mentioned in Section 2.2.4.

If the artifacts are not properly removed before EEG analysis, the results are likely to be incorrect and misleading. Signals picked up from the scalp could also be an inaccurate representation of the signals originating from the brain since the spatial information gets lost. EEG preprocessing is still an area of research, meaning that there is no universal EEG preprocessing pipeline. This approach gives researchers a bit of freedom when choosing how they want to transform the raw EEG data.

³Red, green, blue

3.2.1 Importing data

MNE

MNE is a Python package for EEG/MEG processing, it is a very useful tool in EEG data analysis. Some of these tools include filtering, Independent Component Analysis (ICA), visualization, input/output and many more. It can perform much similar analysis as the well known EEGLAB implemented in Matlab.

File format

There are many different file formats for storing EEG data, since each manufacturer of EEG amplifiers uses its own file format. The BrainProducts amplifier stores data as an EEG/VHDR/VMRK file triplet.

The three separate BrainVision data file formats are as follows:

1. A text header file (.vhdr) containing meta data.
2. A text marker file (.vmrk) containing information about events in the data.
3. A binary data file (.eeg) containing the voltage values of the EEG.

Once the data are ready for analysis, a Python shell can be fired up from a terminal or Anaconda Navigator. In this research project, Jupyter notebook is used.

3.2.2 Removing bad channels and interpolating

Sometimes the EEG data will contain a ‘bad’ channel that does not provide accurate information, it is therefore important to remove these channels in early analysis. These ‘bad’ channels can exist because of a malfunction during EEG acquisition, an electrode didn’t have contact with the scalp, bridging between two channels or one of the electrodes got saturated. The most common way of detecting bad channels is by visualizing the raw data using MNE, these channels usually seem much noisier than others. If a channel is identified as a ‘bad’ channel, it can be excluded from further analysis by marking them as ‘bad’ and using the built-in MNE function to exclude them.

If a channel is removed, it is common practice to interpolate data for the bad channels based on the data from good channels surrounding the bad channels. Interpolation is a way of filling the missing data with other data that are available. This is usually done by spherical splines, a detailed description of this method can be found in reference [28].

It is also possible to remove some of the noise without removing the entire channel by using ICA as explained in Section 3.2.3. Fortunately during the experiments, bad channels were encountered only in a few cases, some of the noise could be removed using ICA without the entire channel being removed. Since the bad channels, happened so little, the bad channels were kept in the dataset, with the goal of making the classifier more robust and maybe contribute to its generalization capabilities.

3.2.3 Artifact correction

Artifacts are signals that are picked up by the EEG system that do not originate from the brain. There are many different sources of artifacts in EEG data, and can be classified as biological or environmental sources [28]:

- Biological artifacts originate from sources in the body, the most common biological artifacts are eye blinks, eye movement, muscle movements and heart beats.
- Environmental artifacts originate from interference from outside-world, this could be power line interference or electrodes that lose contact with the scalp during the experiment. Power line interference can be removed by applying a notch filter at 50 or 60 Hz as explained in Section 3.2.4.

Independent Component Analysis (ICA)

There are many different ICA algorithms such as FastICA, Infomax and projection pursuit. The independent components are extracted by minimizing the mutual information, maximizing the non-Gaussianity or using the Maximum Likelihood (ML) estimation method. The data are fitted with the FastICA algorithm, after which the task is to identify all the ocular components. Figure 3.4a shows an example of the first 20 independent components from one of the participant's data. Only the first 20 components are of importance, because ocular components are generally found amongst these components [29].

From the scalp projections in figure 3.4a, the component labeled as “ICA000” looks like it could represent eye movements because of its frontal location. To be sure that “ICA000” represents ocular activity, the component is further analyzed by looking at the power spectral density, the epochs image which shows typical intermittent activity as blue and red stripes, and the epochs variance as seen in figure 3.4b. The component is then removed once it has been confirmed to be an ocular activity by excluding it from the rest of the components.

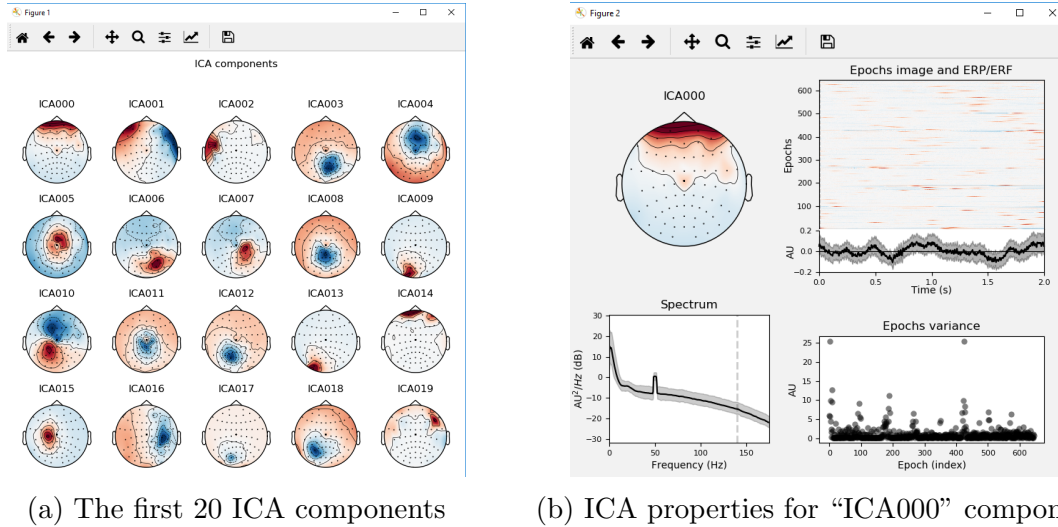


Figure 3.4: The identifying of ocular components

3.2.4 Filtering

When examining the frequency spectrum of EEG data, a popular action is to filter the data, this can be due to bad electrode placement on the scalp, or to remove the interference from power line noise etc. In this study, the following filters were implemented in Python using the MNE library.

1. High Pass Filter

The impedance between the electrodes and scalp slowly change due to small movements of the skin against the electrodes and the gel. This causes slow baseline drifts in the EEG data, the slow drifts can be removed by applying a high pass filter to the data with a cutoff frequency of 1 Hz.

2. Low Pass Filter

The signal characteristics that are typical for human brain visual experiences are usually found in the frequency ranges of theta-, alpha- and beta-waves (4-8 Hz, 8-15 Hz and 16-40 Hz respectively). A low pass filter was applied to the data with a cutoff frequency of 40 Hz.

3.2.5 Six-sigma clipping

During the recording sessions, unintentional movements by the participants can lead to high voltage spikes in the EEG data. To rectify these outliers, sample values that exceed $\pm 6\sigma(x_i)$ were set to $\pm 6\sigma(x_i)$ where σ is the standard deviation for the EEG data of channel i [21]. Six-sigma clipping is applied to the EEG data after it has been filtered and normalized using numpy’s normalization method.

$$x_i = \begin{cases} +6\sigma(x_i), & \text{if } x_i \geq +6\sigma(x_i) \\ -6\sigma(x_i), & \text{if } x_i \leq -6\sigma(x_i) \\ x_i, & \text{otherwise} \end{cases} \quad (3.1)$$

3.3 Making images from EEG time-series data

This section introduces a method similar to the method proposed by Bashivan et al. [67] and also used by Tan et al. [66] to obtain a sequence of topology-preserving multi-spectral images, as opposed to standard EEG analysis techniques that ignores the spatial information⁴. The approach is designed to preserve some spatial, spectral, and temporal structure of the EEG data, and extract the features that are more robust to variations and distortions within each dimension.

3.3.1 The fast fourier transform

The Fast Fourier Transform (FFT) is performed on the time series data for each trial to estimate the power spectrum of the signals. As mentioned in section 3.2.4, oscillatory cortical activity related to visual experiences of the human brain is usually found in the frequency ranges of θ -, α - and β -waves (4-8 Hz, 8-12 Hz and 12-40 Hz), respectively [71].

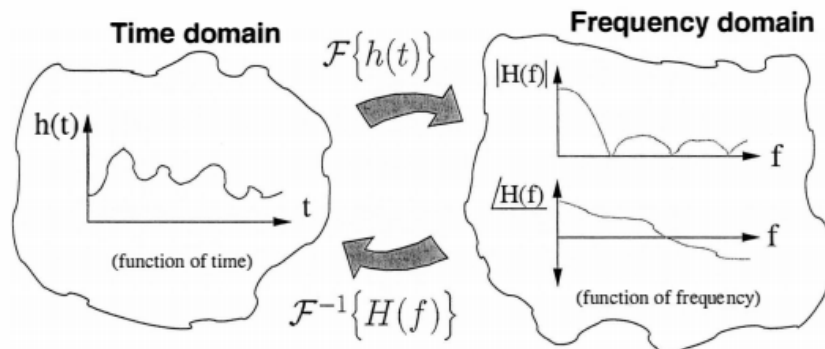


Figure 3.5: Time domain to frequency domain [71]

The FFT is a complicated- and one of the most powerful algorithm that operates by decomposing N sample points in the time domain signal consisting of single points into transformations of smaller length [72]. These N time domain points are then used to calculate the N frequency points, which are then synthesized into a single frequency spectrum composed of real and complex components [71]. Figure 3.5 illustrates the two frequency domains.

⁴Standard EEG analysis techniques represents the low-level EEG data as a vector

The FFT and the Discrete Fourier Transform (DFT) lead to exactly the same result, the difference is their structure of computation. The FFT is a more efficient/faster implementation of the DFT. It enables us to conveniently analyze and design systems in the frequency domain. It exploits periodicity and symmetry in the DFT to reduce the number of computations [71].

The FFT requires $\frac{N}{2} \log_2(N)$ complex multiplications and the DFT requires N^2 complex multiplications [71].

The FFT is an efficient implementation of the DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad (3.2)$$

Using equation 3.2 requires, for each k:

- N complex multiplications
- N-1 complex additions

and to compute all N values for X[k] requires:

- N^2 complex multiplications
- $N(N - 1) \approx N^2$ complex additions

The FFT is applied on the EEG time series data for each data frame (4.5 seconds) to transform the data from the time domain to the frequency domain to estimate the power spectrum of the signal (see figure 3.6).

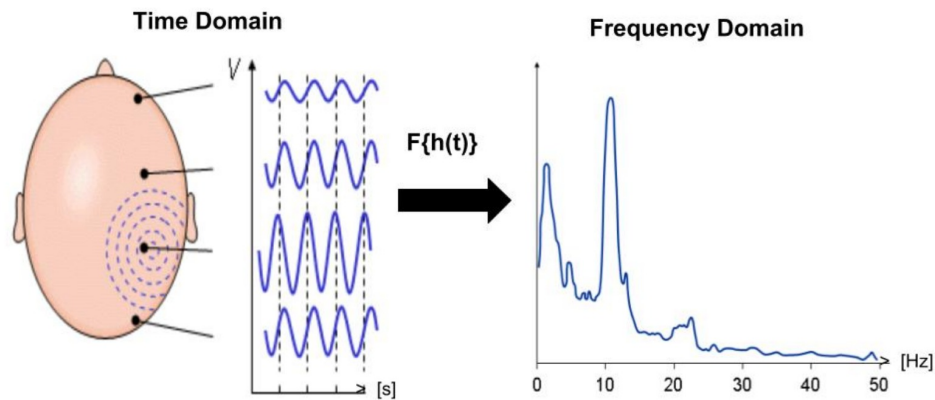


Figure 3.6: The EEG time series data from time domain to frequency domain

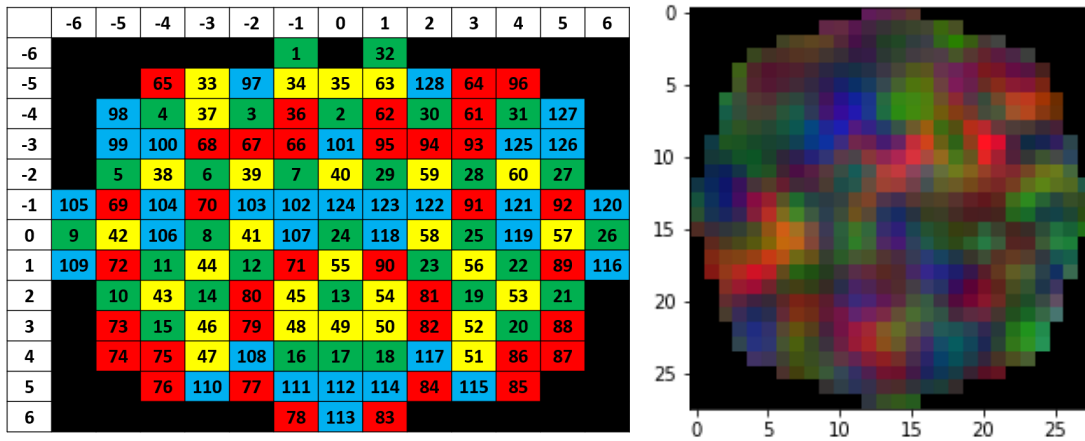
After the EEG time series data is transformed from the time domain to the frequency domain, the FFT amplitudes were grouped together into three groups, the theta, alpha and beta bands were considered for analysis.:

- Theta Band (4-8 Hz)
- Alpha Band (8-12 Hz)
- Beta Band (12-40 Hz)

The sum of squared absolute values within each frequency band were subsequently computed and used as separate measurement for each of the electrodes [67].

3.3.2 2D projection

Instead of using the standard approach of using the measurements from all the electrodes to form a feature vector, the proposed method is to transform the measurements into a 2-D image to preserve the spatial structure and use RGB color channels to represent the spectral dimensions. And finally, the sequence of images that are derived from the consecutive time windows are used to account for the temporal evolutions in the brain activity [67]. In order to transform the spatially distributed 3-D activity maps as 2-D images, the locations of each electrode from the 3-D space is projected to a 2-D surface. Similarly to this case where the cap that is worn on a human's head can be approximated by a sphere, the method can be used to compute the projections of the electrode locations on a 2-D surface that is tangent to the top point of the human head.



(a) The 13 x 13 electrode location mesh.

(b) Three-channel image

Figure 3.7: (a) The 13 x 13 electrode location mesh used to merge the three maps to form the three-channel image. (b) The three-channel image generated after the projections.

The proposed method can be considered as imagining a plane against (perpendicular to) a sphere, when a light source shines onto the sphere and all the points would be projected onto the plane, the result would be planar. The power measurements for each frequency band of interest is interpolated onto the 13 x 13 mesh as seen in figure 3.7a, resulting in three topographical activity maps corresponding to each frequency band. The legends for figure 3.7a are as follows:

- **Green:** Channels 1 - 32
- **Yellow:** Channels 33 - 64
- **Red:** Channels 65 - 96
- **Blue:** Channel 97 - 128

The three maps are then merged together to form an image with three color channels (see figure 3.7b). These three-channel images are used as input to the machine learning and deep learning techniques.

An overview of the approach is shown in figure 3.8. Firstly the EEG Time series data from multiple locations are required, the spectral power within the three prominent frequency bands are extracted and used to form the topographical maps for each time frame. The sequence of topographical maps is then combined to form a sequence of three-channel images which are then fed into for example a convolutional neural network that will make the output class predictions.

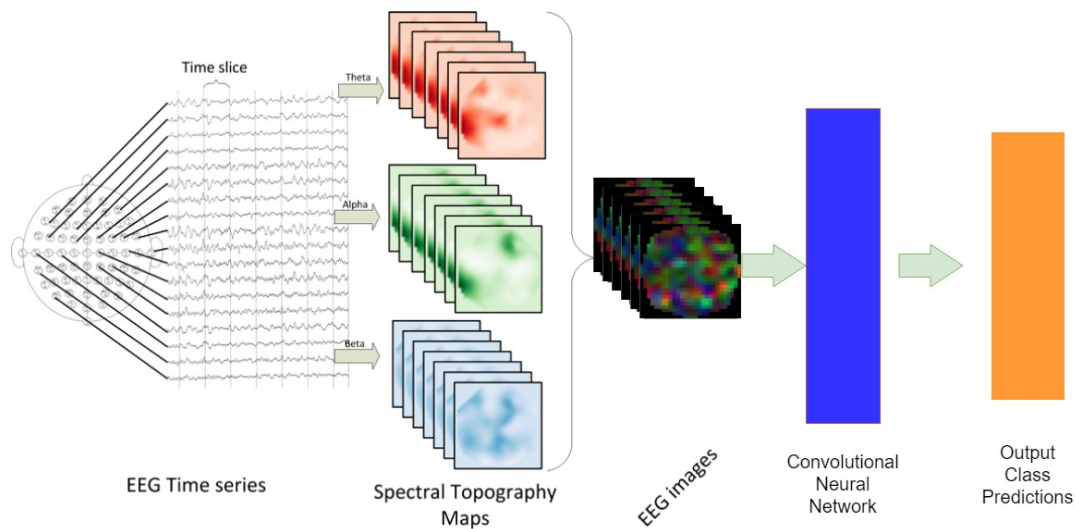


Figure 3.8: Overview of the approach of the EEG time series projections. The approach is based on the results of Bashivan et al. [67].

3.4 Implementation

This section explains the implementation of each proposed model for decoding the EEG brain signals. This work approaches the classification problem with different machine learning techniques as well as with deep learning techniques. First, the machine learning models are explained, followed by the deep learning models. The dataset for each participant was merged together and shuffled before fed into the models.

3.4.1 Logistic regression model

Logistic regression is one of the most fundamental machine learning algorithms and is usually one of the first techniques chosen when choosing a model for predictive learning. The three-channel EEG images are fed into the logistic regression model. The Architecture is described below.

Training

In this work, scikit-learn's built in Logistic Regression classifier was implemented with the following parameter settings, all other parameters not specified are set to their default values:

- `random_state = 0`
- `solver = 'saga'` For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss.⁵
- `multi_class='multinomial'`

The data are split using scikit-learn's built in `train_test_split` function into a training, validation and testing set (80 %, 10 % and 10 % respectively).

3.4.2 Support Vector Machine model

Support Vector Machines cover very high accuracies when compared to other classifiers such as logistic regression, and decision trees.

Training

To create the SVM classifier, scikit-learn's built in SVM classifier is implemented, the kernel is chosen to be a 'linear' kernel, which implements a "One-vs-all" multi-class strategy, thus training a n-class model. The C parameter is the penalty term, which represents misclassification or error term. This is used to control the trade-off between

⁵The best performing parameter is chosen iteratively.

the decision boundary and misclassification. The smaller the C value, the smaller is the margin hyperplane and vice versa. The value of C is set to 1. The data is split using scikit-learn's built in `train_test_split` function into a training, validation and testing set (80 %, 10 % and 10 % respectively).

3.4.3 HOG-SVM model

The idea of HOG is to lower the complexity of the problem while maintaining as much variations as possible. Scikit-learn comes with many built in transformers. The color images are first converted to shades of gray images, their HOGs are then calculated and finally scaled. The transformers used for this are `RGB2GrayTransformer`, `HOGTransformer` and `StandardScaler` respectively. Figure 3.9a shows the shades of gray image and figure 3.9b shows the corresponding HOG image.

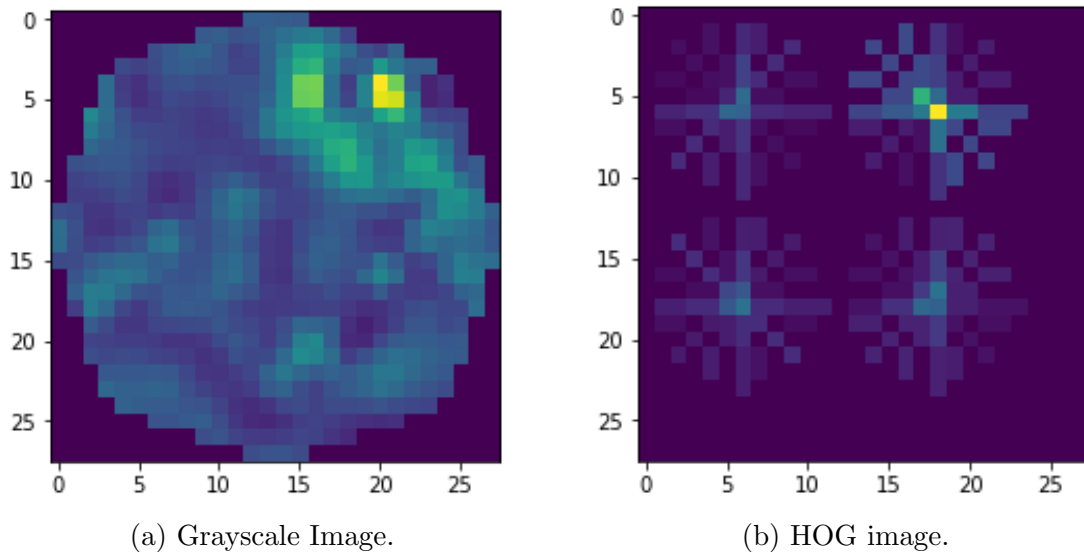


Figure 3.9: Example of the shades of gray image and the corresponding HOG image

Training

The number of orientations, `pixels_per_cell`, and `cells_per_block` for computing the HOG features are set as follows⁶:

- orientations = 8
- pixels_per_cell = 12
- cells_per_block = 2

⁶These parameters are adjusted until the optimal results were achieved

The data are split using scikit-learn's built in `train_test_split` function into a training, validation and testing set (80 %, 10 % and 10 % respectively). The datasets are then trained on a SVM classifier as before.

3.4.4 K-Nearest Neighbor model

The KNN algorithm is often used as the benchmark for more complex classifiers such as ANN and SVMs. Despite its simplicity, KNN can outperform some of the most powerful classifier algorithms available.

Training

In this work, scikit-learn's built in KNN classifier was implemented. The data are split first using scikit-learn's `train_test_split` function into a training, validation and testing set (80 %, 10 % and 10 % respectively). Next, the hyperparameter K is tuned using cross-validations. Cross-validation is used to estimate the test error associated with a learning method to evaluate its performance. A list of odd K 's is generated ranging from 1 to 30. A 10-fold cross-validation is performed and the misclassification error versus K is plotted. Figure 3.10 shows that the 10-fold cross-validation $K = 1$ results in the lowest validation error. Finally, the classifier is re-trained using the best K value that corresponds with the lowest test error rate and the labels of the test data are predicted.

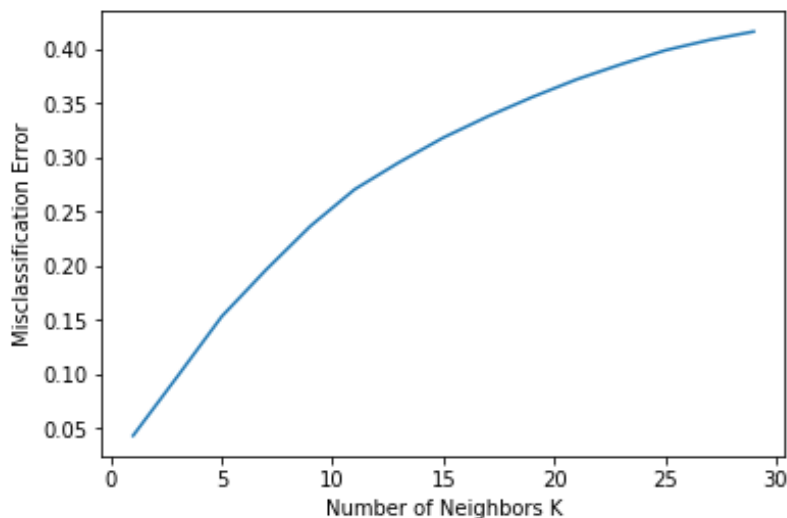


Figure 3.10: Plot of misclassification error versus number of neighbors K .

3.4.5 Random Forest

The random forest algorithm has a variety of applications, such as recommendation engines, image classification and feature selection. It is considered as a highly accurate and robust method which does not suffer from overfitting.

Training

The Random forest classifier is implemented using scikit-learn's built-in function. The features are firstly scaled using the `StandardScaler()` function before the data are split using scikit-learn's `train_test_split` function into a training, validation and testing set (80 %, 10 % and 10 % respectively). The number of trees parameter in the random forest classifier is set to 1000 trees, after which the algorithm is trained.

3.4.6 Convolutional Neural Network model

Convolutional neural networks are to date at the forefront of the most successful architectures when it comes to image recognition tasks and many other tasks. Image classification is the task of taking an image as input and outputting a class or a probability of the classes that best describes the input image. The computer needs to be able to differentiate between all the images that it's been given and figure out the unique features that makes a specific image fall under a certain class. This is the process that goes on in the human mind subconsciously as well. When a person looks at a picture of a dog, it can be classified as a dog if the image has identifiable features such as for example four paws. In a similar way, a computer is able to perform image classification by identifying simple low level features such as edges, texture and curves, and then building more abstract concepts through a series of convolutional layers.

Architecture

All CNN models follow a similar architecture, the proposed architecture is shown in figure A.1 in Appendix A.1. The network architecture uses stacked convolutional layers with an increase in size to the third layer after which it decreases again. After the first convolutional layer and last dense layer dropout is applied to improve robustness, generalization and reduce overfitting. Max-pooling is performed after each convolutional layer to down-sample the output from the previous layer by a factor of two to reduce the dimensionality. A ReLU non-linearity is then applied. The ensemble of convolution layer, max-pooling and ReLU activation is repeated three times. The output of the last convolutional ensemble is flattened and fully-connected to three layers where the last layer consists of K neurons with a softmax activation, K is the number of classes.

Other CNN configurations using different kernel sizes, number of neurons per layer, batch normalization added, the amount of stacked convolutional layers, the use of other

dropout ratios etc. have been tested until the architecture with the best performance was obtained. Since there is a large amount of tunable hyperparameters, it would be extremely time-consuming to do a grid-search for example over all possibilities to find the optimal values.

Training

Figure 3.11 and 3.12 visualize loss and accuracy during training, a visible overfitting pattern can be seen in figure 3.11. In the Loss plot, it is noticeable that even after adding dropout to the model, there is still overfitting present (Overfitting starts at 400 epochs). Early stopping could not be used, since the model only started learning significant features after more or less 50 epochs. The model was trained until the validation and training loss started to plateau. The model was run a couple of times until the model with the lowest validation loss was obtained and saved. This model was then used for testing and these results are presented in section 3.5. Overfitting is never a good thing, however, in this case the CNN model was still able to make relatively good predictions. The Loss indicates how good or bad the model's predictions are, the lower the loss, the better the predictions. The goal of training a model is to find a set of weights and biases that has a low loss. Training deep neural networks can range from days to weeks, training DNN involves a huge number of matrix multiplications and other operations which can be massively parallelized and thus sped up on GPUs. GPUs have many resources and faster bandwidth to memory. The model was trained on a GeForce GTX 1080 GPU, with CUDA 9.0 and cuDNN v7.5.0, using Keras 2.1.4 API.

The model is estimated using the fast and efficient Adam optimizer, with categorical cross-entropy as the loss function. The optimizer parameters were set to a learning rate of 10^{-3} and decay rates of first and second moments of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, respectively. The model was trained with a mini-batch size of 2^5 as described in section 2.6.3.

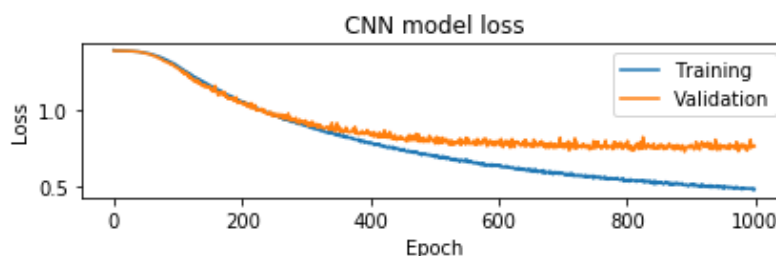


Figure 3.11: CNN model optimization: The training and validation loss over the entire dataset is evident.

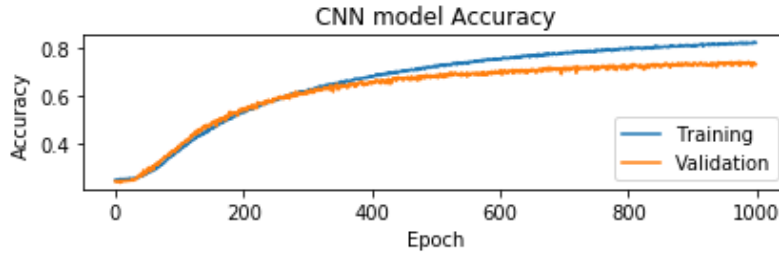


Figure 3.12: CNN model optimization: The training and validation accuracy cover the entire dataset.

3.4.7 Generative Adversarial Network model

Figure 3.11 shows that the CNN model starts overfitting at approximately 400 epochs, this happens because the model has too few training samples, resulting in the model having a poor generalization performance. A common case in machine learning applications, is that it is not an easy task to obtain new training data, especially with EEG datasets. One way to obtain more training data is to generate more training data in our current set, by making use of GANs as described in section 2.6.2.

Architecture

Figure B.1 in appendix B.1 shows the proposed Generator architecture. The network architecture uses stacked fully-connected layers that increases with power of 2 each layer. After each layer, Leaky ReLU activation function is applied followed by batch normalization. After the third fully-connected layer, the output is flattened and reshaped into 28 x 28 x 3 image, with tanh as activation function. Figure B.2 shows the proposed discriminator architecture in appendix B.2. The discriminator looks similar to the generators architecture, after the last fully-connected layer, the output is evaluated using a sigmoid activation function.

Training

The 2-D EEG image data are split the same as before using scikit-learn's `train_test_split` function and shuffled randomly. The training dataset is then split up into the four categories. The GAN model generates new training images for each category until the discriminator could not distinguish between what is fake and what is real data (Accuracy $\geq 95\%$). The GAN model then stores 2000 new training images for each of the four categories. The additional 8000 training instances are added back to the training samples, before the CNN model is trained again. Figure 3.14 and 3.15 visualize loss and accuracy during training, a visible overfitting pattern can still be seen in figure 3.14. Figure 3.13 shows an example of a generated image for one of the four categories compared to the

real 2D EEG images. It is clear that one can't distinguish between which one is fake and which one is real.

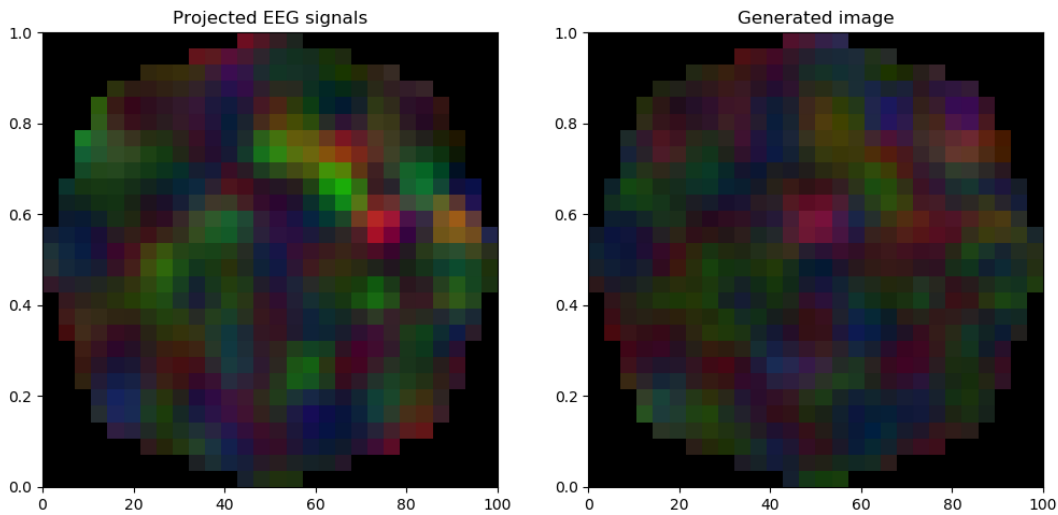


Figure 3.13: The image on the left is the real 2D projected EEG image and that on the right is the generated image.

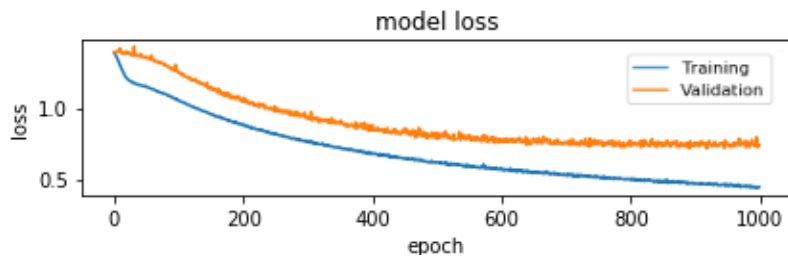


Figure 3.14: CNN model optimization: The training and validation loss over the entire dataset with generated images are included.

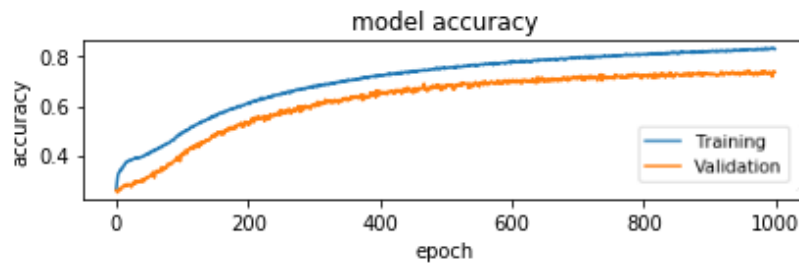


Figure 3.15: CNN model optimization: The training and validation accuracy over the entire dataset with generated images are included.

3.4.8 How brain regions affect CNN predictions

Sight is a complex function of the brain that extends from the front to the back of the head [73]. As light enters the retina, it is sent via the optical nervous system to the back of the brain, where it is then processed by the occipital lobe. The optical nervous system is what connects the eyes to the brain, information from the left eye is sent to the right hemisphere and vice versa according to the Canadian Institutes of Health Research [74], this is because the nerve crosses at the optic chiasm, which causes the information from the optic nerve coming from each eye to be sent to the left and right hemisphere of the brain [73].

The occipital lobe forms the center of the visual perception system according to the Centre for Neuro Skills [75]. Each hemisphere has its own occipital lobe that processes the information received from the optical nervous system. There are nearly 30 different cortical areas in the occipital lobe that contribute to visual perception [74]. As mentioned in section 2.1, the primary visual cortex (area V1) is the largest and responds to simple local features such as edges, color and texture. Signals that leave area V1 are then distributed to higher order visual areas such as V2, V3 and V4, these are believed to be responsible for extracting more complex higher order information from the visual data [4].

The frontal cortex, located in the front of the head, has for long been understood as the seat of higher level cognition, it is associated with thinking and making decisions. It's therefore often referred to as our "thinking cap". Until recent research done by Dobromir Rahnev, a psychologist at the Georgia Institute of Technology and by researchers from the University of California, the frontal cortex has not commonly been connected with vision. "Some people believe that the frontal Cortex is not involved," according to Rahnev [76].

According to Rahnev, the thinking cap of the brain controls and oversees the whole process, making it as essential as to those other areas in visual processing. The brain is actively constructing the scenes we see and is making decisions about it. To test out the frontal cortex's involvement in visual perception, the researchers ran a two part experiment. The first part was to observe which parts of the brain lit up with activity while healthy volunteers completed visual tasks using a fMRI machine, observing the frontal cortex in particular. The second part was to observe those same regions with magnetic stimulations using a fMRI machine to confirm their involvement in the visual process. As a result, they were able to clearly demonstrate the frontal cortex's involvement in visual processing [76]. Figure 3.16 shows a brain image from a fMRI scan during the second part of the experiment, where they tested the role of the frontal

cortex in vision.

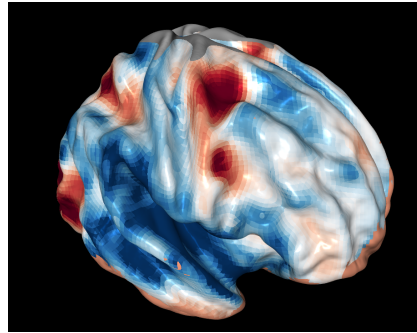


Figure 3.16: A brain image from a fMRI scan during one of the tests on the role of the frontal cortex in vision [76].

To establish if the frontal cortex plays a role in visual processing, four models were built in this study using different location regions of the brain, to determine how each region effects the prediction accuracies. Figure 3.17 shows the mapping between the EEG signals and the brain cortices.

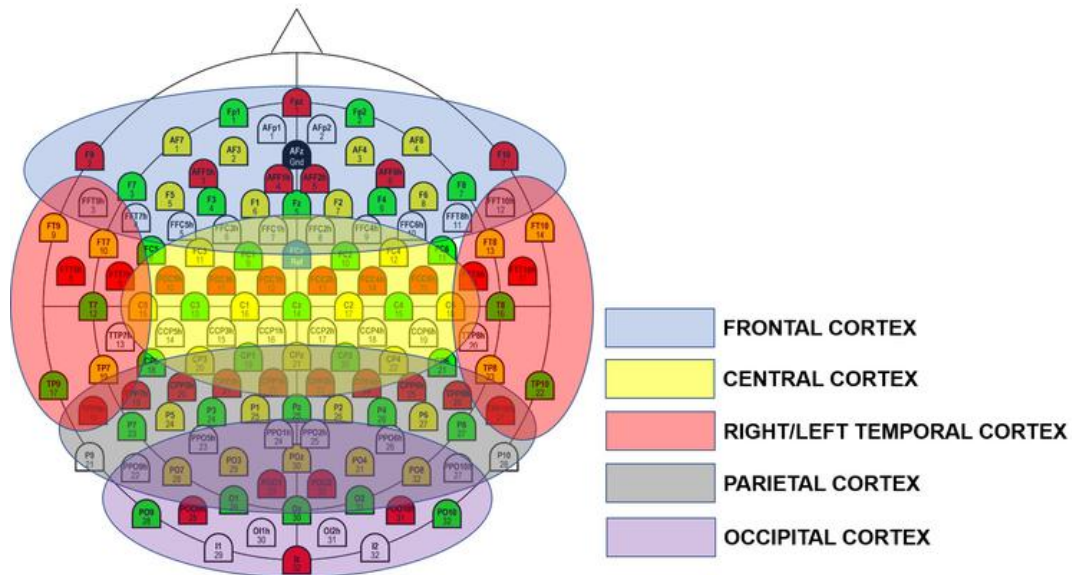


Figure 3.17: Mapping between EEG channels and the brain cortices [77].

The Four location regions that were defined for this experiment are as follows⁷:

1. Occipital Cortex

⁷The electrode location regions are highlighted in light blue in section C.1 and C.2 in appendix C.

2. Frontal Cortex
3. Occipital Cortex and Parietal Cortex
4. Whole Cortex

3.4.9 Generalization across participants

To establish if our models generalize well across participants, the two models that obtained the highest accuracies (K-NN and CNN model) were trained on nine of the ten participants, chosen at random. The training set was split into a training and validation set (80 % and 20 % respectively). The tenth participant's data was used for testing the models. The K-NN model achieved a high accuracy of 97 % and the CNN model achieved an accuracy of 82 % with a loss of 0.5. Figure 3.18 shows the CNN model's accuracy plot and figure 3.19 shows the CNN model's loss plot.

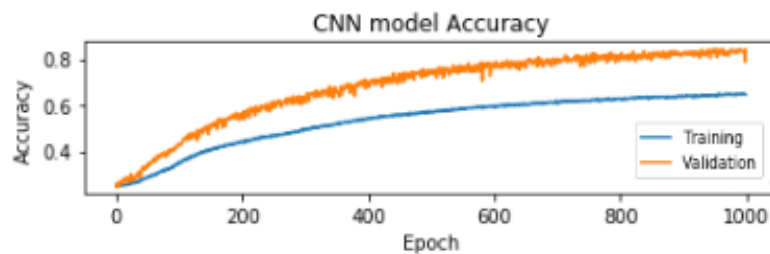


Figure 3.18: CNN model's accuracy plot.

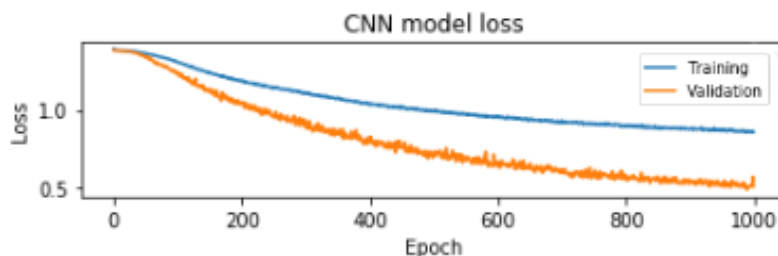


Figure 3.19: CNN model's loss plot.

3.5 Results

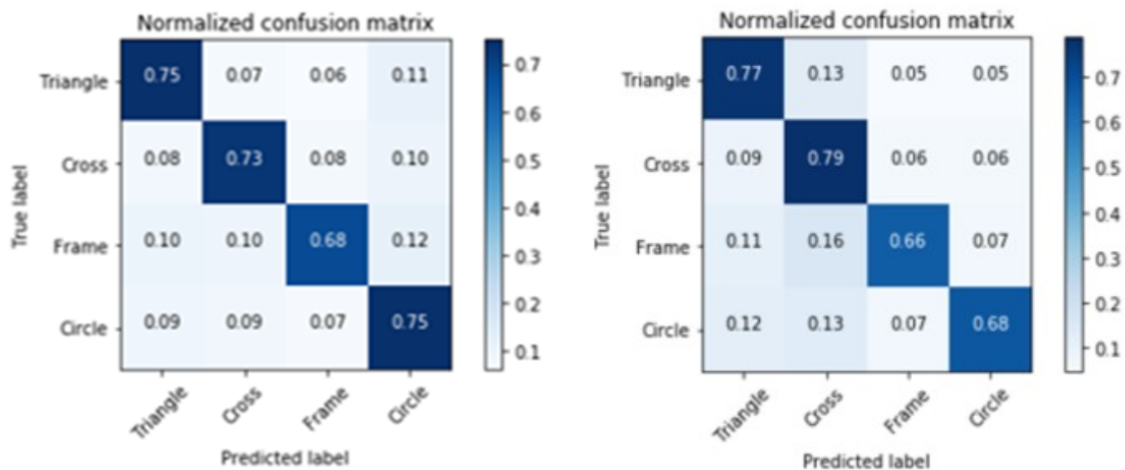
This section of the thesis starts by showing the results obtained by using GANs to generate new training samples to improve the accuracy of the models and to reduce overfitting. Secondly, the results of traditional machine learning techniques and deep learning techniques are shown. Finally, it shows how each of the brain regions forms part of visual processing.

3.5.1 CNN Model vs. CNN model with GAN training data

Table 3.1 shows the model loss and accuracy comparison between the CNN model and the CNN model with the added generated images. The CNN model achieved an accuracy of 73 % with a loss of 0.77. After adding the generated training instances to the CNN model, the model was able to increase the accuracy with 1 %, reaching an accuracy of 74 % with a loss of 0.65. The confusion matrices can be seen in figure 3.20.

Table 3.1: Models' loss and accuracy results.

	CNN	GAN-CNN
Model Accuracy	0.73	0.74
Model Loss	0.77	0.65



(a) CNN model confusion matrix

(b) GAN-CNN model confusion matrix

Figure 3.20: A comparison of the CNN model's performance with the CNN model with added GAN training data.

3.5.2 Traditional machine learning techniques vs. convolutional neural networks

We compared our CNN approach against various traditional machine learning classifiers in the field, including SVM, Logistic Regression, K-NN and random forest. In our experiments, we tested each of these models using the same datasets. The summary of the performance results are shown in figure 3.21.

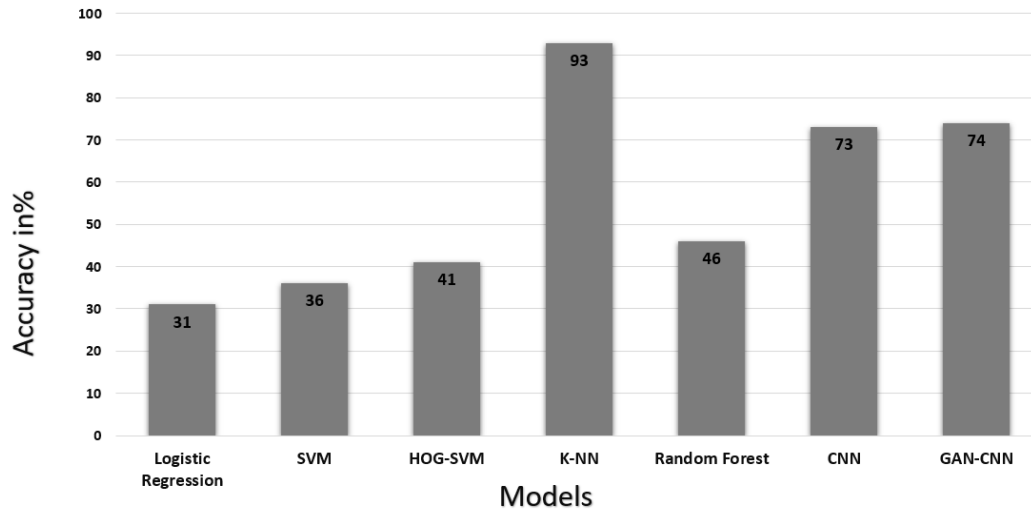


Figure 3.21

The experimental results show that the CNN, GAN-CNN and K-NN models achieved the highest accuracy. Even though K-NN falls under traditional machine learning methods, the CNN model achieved superior accuracy over the rest of the traditional methods. K-NN is the best performing model which achieved a high accuracy of 93 %, out performing the other traditional machine learning techniques and beating NNs by 20 %.

3.5.3 Brain regions comparison

Section C.1 and C.2 in appendix C show the four models that were built using the different location regions of the brain, to determine how each region of the brain effects the prediction accuracies.

Comparing the 2D projections

In the ‘Occipital Cortex’ (1) and the ‘Occipital Cortex and Parietal Cortex’ (3), the colors present in the 2D projections are quite dull compared to the colors observed in the ‘Frontal Cortex’ (2) and the ‘Whole Cortex’ (4), where the colors are more clearer and more vibrant. This is due to the frequencies present in those specific regions, as explained in Section 3.3.2, the higher the frequency present, the higher the color intensity. A possible reason for this might be that Delta waves are the majority frequency present right at the back of the human head. Moving forward, the frequencies increase, and the presence of Theta waves increases. Alpha and Beta waves have the highest frequencies, which is mostly present in the front half of the brain.

Comparing the accuracies

1. Occipital Cortex - Only looking at the 28 channels in the back of the head, gives clearly not enough information for accurate results, as seen in section C.1 (in appendix C), the model obtained 53 % accuracy with a quite high loss of 1.34.
2. Frontal Cortex - For the first 200 epoch, the train and test trends have the same values, before the test data points start plateauing. An astonishing 72 % accuracy was obtained with a lower loss than in (1).
3. Occipital Cortex and Parietal Cortex - The test and train data points start off with a better trend than in (1), but after 180 epochs, the test loss stops decreasing and starts oscillating around 1. This model resulted in the same accuracy as in (2).
4. Whole Cortex - Looking at the graphs in section C.2 (in appendix C), the models accuracy train and test data has the same trend and ends off with an small error less than 5 %. A high accuracy of 73 % was achieved when all the channels were used together.

It is clear from the results in section C.1 and C.2 (in appendix C) that the Frontal Cortex was able to predict which shape was shown with an accuracy of 72 %, therefore, confirming the frontal cortex's involvement in visual processing. This result also helps us see that all 128 channels are not necessarily needed for analyzing human visual processing. What is really interesting about these results are that the frontal cortex was able to predict the shapes that were shown with an accuracy of 72 %, which is exactly the same results as with the Occipital and Parietal Cortex which actually includes the whole visual cortex of the brain. These results bring us to a conclusion that the CNN model is able to identify patterns in the patients' minds.

Chapter 4

Discussion and future work

In this study different machine learning and deep learning models were developed to classify the geometrical shapes that had evoked a given activity pattern in the EEG data. This chapter shows an overview of the results of this study compared to similar approaches in literature.

The data from all the subjects were combined to form one dataset, although this lowers the classification results significantly, we were interested in building one classifier that will be able to accurately predict the stimuli observed by the participants. We were able to demonstrate that the model generalizes across the 10 participants, in addition, using GAN's helped improve the generalizability.

After adding the synthesized training instances generated by the GAN model, the CNN model's overfitting reduced while also slightly increasing the model's accuracy. The proposed SVM, HOG-SVM, and Random Forest models achieved accuracy between 31-41 % which is not considered to be a success for the models performance compared to the CNN model. Another possibility could also be that one of the participant's data brought down the accuracy of the models, because the participant maybe didn't sleep that well the previous night, or maybe wasn't paying enough attention. Therefore, it is recommended that a lot more participants should be used in future studies.

In section 3.4.8 we looked at to what extent each region of the brain plays a role in visual processing by building four models using different regions of the brain. The results that the frontal cortex is involved in visual processing, confirming the findings of Dobromir Rahnev [76]. Each of the four models were built using a different number of electrodes, the frontal cortex region used 56 electrode channels and obtained similar accuracies as the model with 128 electrode channels. Therefore, it is not always necessary to use all 128 EEG channels.

The proposed K-NN model shows a performance comparable to the Multi-Layer NN developed by Chuyen Lam et al. [65]. Also comparing the CNN model's results with the K-NN model shows how powerful traditional machine learning techniques could be in some cases, outperforming the NNs. The K-NN model is a very powerful traditional machine learning technique, so this high accuracy of 93 % is not surprising. This illustrates the fact that just because a NN is used, does not necessarily make it better.

The K-NN and CNN accuracies will both increase with the increase in training data size. The difference, however, K-NN will become slower and slower with the increase in training size, whereas this is not the case with CNN. In this study, 10 participants were used to build these predictive models, however, to get much better generalization and performance, a lot more participants are required.

Comparing proposed models with other work

Table 4.1 shows an overview of the results of my study compared to similar approaches in literature. An overview of each of these related studies can be seen in section ??.

The CNN and GAN-CNN model show a performance comparable to the LSTM and GRU model developed by Tan et al. [66] for classifying their four class motor imagery experiment. Considering CNN's suitability in image classification, the proposed CNN model did not perform as well as was expected in classifying the four geometrical shapes. The CNN model is overfitting despite the fact that dropout was used. The reason for this finding is that the model is training on 8000 images per category. In comparison using deep learning methods, 100 000 training samples are required at least. No matter what regularization techniques are used, the model will still overfit on such a small dataset.

In section 3.4.9, we trained the K-NN and CNN models on nine of the ten participants, chosen at random. The tenth participant's data was used for testing the models. The K-NN model achieved a high accuracy of 97 % and the CNN model achieved an accuracy of 82 % with a loss of 0.5. Figure 3.18 shows the CNN model's accuracy plot and figure 3.19 shows the CNN model's loss plot. Once again the K-NN model outperformed the CNN model, both the CNN and K-NN models did extremely well in generalizing across the participants. These findings support the idea that people's brains encode and process shapes in a similar manner. We believe the models will do even better with a larger dataset and more participants, further investigation is required.

In future work, researchers should focus on building larger EEG datasets which will allow the exploration of a larger and deeper machine learning model that will improve the decoding of human brain processes involved in visual recognition. In addition, more participants should be involved in the experiment in order to provide these larger datasets, producing more robust models that will in turn generalize better on each individual. Another interesting experiment would be to again test the participants on a different day and use this new data as the testing set and see if the model generalizes more effectively.

When selecting the stimuli for the experiment, there are too many attributes that need to be paid attention to, these attributes include color, size, background, shape etc. It is difficult to figure out which attributes the human brain are more sensitive to and which ones contribute to more accurate classification results. In this study, the goal was to present the stimuli a color that stood out from the background, which in this case was

white. It would be interesting to in future work to see how these attributes contribute to the classification accuracies. The study focused on only four simple geometrical shapes. Increasing the number of geometrical shape categories and consider adding more complex shapes to the dataset is definitely another potential research direction.

Table 4.1: The comparison of classification results of brain signals using deep learning models and traditional machine learning models obtained in this study.

Author	Reference	Model	Classification Type	Accuracy	Classes
This work	Section 3.4.1	Logistic Regression	Shapes	31	4
This work	Section 3.4.4	K-NN	Shapes	93	4
This work	Section 3.4.5	Random Forest	Shapes	39.93	4
This work	Section 3.4.2	SVM	Shapes	36	4
This work	Section 3.4.3	HOG-SVM	Shapes	41	4
This work	Section 3.4.6	CNN	Shapes	73.42	4
This work	Section 3.4.7	GAN-CNN	Shapes	74.13	4
Tan et al.	[66]	SVM	Motor imagery	66.78	4
Tan et al.	[66]	Conv1D	Motor imagery	69.42	4
Tan et al.	[66]	LSTM	Motor imagery	72.22	4
Tan et al.	[66]	GRU	Motor imagery	70.34	4
Spampinato et al.	[68]	CNN	Dogs, cats, butterflies etc.	40	40
Chuyen Lam et al.	[65]	Multi-Layer NN	Humans, animals, landscapes, cities, flowers	92.68	5
Zhang et al.	[69]	GAN-CNN	Shapes	83	5

Chapter 5

Conclusion

In this study, it was shown that both CNN and K-NN models were able to effectively perform image classification on the EEG data and predict what geometrical shape had evoked a given activity pattern. Furthermore, the work investigated the augmenting of training data using GAN derived synthetic images, and the results demonstrated that this can improve the CNN models. One of the biggest issues faced when using machine learning algorithms is the lack of availability of large, labelled datasets. This approach can help when the size of the dataset is limited and also helps reduce overfitting.

Instead of using the standard approach which uses the measurements from all the electrodes to form a feature vector, alternatively rather transforming the EEG time-series data into a two dimensional EEG videos could be considered. The optical flow has been introduced, which can characterize the variant of EEG signals in the temporal dimension. To preserve some of the spatial features, the electrode positions are projected to a two dimensional mesh and frequency filters are applied to represent the spectral information.

We also attempted building computational models of different regions of the brain to interpret the role of each region of the brain plays. We came to the conclusion that the whole cortex is involved in the human visual processing, in addition, the frontal region showed superior involvement in the role of vision. The results in section C.1 and section C.2 (in appendix C) shows that the model from the frontal region obtained the same accuracy as the model from the occipital and parietal region. Based on these results we can confirm the findings of the research teams of Dobromir Rahnev [76], namely associating the frontal cortex with visual processing.

The main advantage of using CNN is that it does not require any prior assumptions on the data, making it suitable for many classification challenges, without manually engineering complicated feature extraction algorithms. This can save a huge amount of time while still obtaining state of the art classification results of EEG signals. We showed that the K-NN model out performed the CNN model, giving us an intuitive insight that CNN's aren't always the best approach to a problem.

Bibliography

- [1] J. L. Gallant, “Jack I. Gallant’s research description,” <https://psychology.berkeley.edu/people/jack-l-gallant>, 2018.
- [2] T. N. Y. B. B. Y. . J. L. G. Shinji Nishimoto, An T. Vu, “Reconstructing visual experiences from brain activity evoked by natural movies,” *Nishimoto et al., Current Biology 2011*, 2011.
- [3] neural networks research group. The primary visual cortex. [Online]. Available: <https://www.cs.utexas.edu/users/nn/web-pubs/sirosh/pvc.html>
- [4] D. T. V. P. T. Kameswara Rao, M. Rajyalakshmi, “An exploration on brain computer interface and its recent trends,” *International Journal of Advanced Research in Artificial Intelligence*, vol. 1, pp. 17–22, 2012.
- [5] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* ” O’Reilly Media, Inc.”, 2017.
- [6] M. Loukas, C. Pennell, C. Groat, R. S. Tubbs, and A. A. Cohen-Gadol, “Korbinian Brodmann (1868-1918) and His Contributions to Mapping the Cerebral Cortex,” *Neurosurgery*, vol. 68, no. 1, pp. 6–11, 01 2011. [Online]. Available: <https://doi.org/10.1227/NEU.0b013e3181fc5cac>
- [7] P. Bryn Farnsworth, “Imotions: Eeg vs. mri vs. fmri what are the differences?” <https://imotions.com/blog/eeg-vs-mri-vs-fmri-differences/>, February 2018.
- [8] L. D. K. X. F. G. P. A. F. Cong, Q. H. Lin and T. Ristaniemi, “Tensor decomposition of eeg signals: A brief review,” *Neuroscience*, vol. 248, pp. 59–69, 2015.
- [9] P. Bryn Farnsworth, “Imotions: What is eeg (electroencephalography) and how does it work?” <https://imotions.com/blog/what-is-eeg/>, September 2018.
- [10] F. Pereira, T. Mitchell, and M. Botvinick, “Machine learning classifiers and fmri: a tutorial overview,” *Neuroimage*, vol. 45, no. 1, pp. S199–S209, 2009.
- [11] R. Zafar, A. S. Malik, A. N. Shuaibu, M. Javvad ur Rehman, and S. C. Dass, “Classification of fmri data using support vector machine and convolutional neural network,” in *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, Sep. 2017, pp. 324–329.
- [12] A. G. Huth, T. Lee, S. Nishimoto, N. Y. Bilenko, A. T. Vu, and J. L. Gallant, “Decoding the semantic content of natural movies from human brain activity,” *Frontiers in systems neuroscience*, vol. 10, p. 81, 2016.

- [13] K. N. Kay, T. Naselaris, R. J. Prenger, and J. L. Gallant, “Identifying natural images from human brain activity,” *Nature*, vol. 452, no. 7185, p. 352, 2008.
- [14] T. Naselaris, K. N. Kay, S. Nishimoto, and J. L. Gallant, “Encoding and decoding in fmri,” *Neuroimage*, vol. 56, no. 2, pp. 400–410, 2011.
- [15] U. o. T. Don Campbell, “Mind reading algorithm uses eeg data to reconstruct images based on what we perceive,” *Neuroscience News*, February 2018.
- [16] A. multi speciality hospital. (2018, Jul) Neurologist. [Online]. Available: <http://anshmultispecialityhospital.com/neurology-2/>
- [17] D. C. of Psychotherapy. (2018) Neurofeedback- qeeg brain map. [Online]. Available: <https://denvercenterofpsychotherapy.com/qeeg-brain-maps/>
- [18] A. Roman-Gonzalez, “Eeg signal processing for bci applications,” vol. 98, pp. 571–591, 01 2012.
- [19] T. Lisboa. Types of eeg signal. [Online]. Available: <https://psiibrainsensoryandprosthetic.weebly.com/eeg-signals.html>
- [20] S. Kohli and A. J. Casson, “Removal of gross artifacts of transcranial alternating current stimulation in simultaneous eeg monitoring,” *Sensors*, vol. 19, no. 1, 2019. [Online]. Available: <http://www.mdpi.com/1424-8220/19/1/190>
- [21] J. Fedjaev, “Decoding eeg brain signals using recurrent neural networks,” 2019.
- [22] U. Herwig, P. Satrapi, and C. Schönfeldt-Lecuona, “Using the international 10-20 eeg system for positioning of transcranial magnetic stimulation,” *Brain topography*, vol. 16, no. 2, pp. 95–99, 2003.
- [23] E. M. R. Das and P. Campisi, “Eeg biometrics using visual stimuli: A longitudinal study,” *IEEE Signal Processing Letters*, vol. 23, pp. 341–345, 03 2016.
- [24] S. F. on ResearchGate. (2019, Feb) Advanced forward models for eeg source imaging. [Online]. Available: https://www.researchgate.net/figure/The-10-20-system-for-EEG-recordings-Figure-adapted-from-4_fig9_294430587
- [25] Antti Savelainen, “An introduction to eeg artifacts.”
- [26] T. M. P. V. Laboratory, “Biomedical signals acquisition,” <https://machinelearning-blog.com/2018/02/06/the-random-forest-algorithm/>, 2019.
- [27] iMotions, *EEG, The Complete Pocket Guide*. iMotions, 2017.
- [28] NEUROTECHEDU. (2017, May) Preprocessing. [Online]. Available: <http://learn.neurotechedu.com/preprocessing/>

- [29] A. Tharwat, “Independent component analysis: An introduction,” *Applied Computing and Informatics*, 2018.
- [30] A. Hyvärinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.
- [31] M. Ullsperger and S. Debener, *Simultaneous EEG and fMRI: recording, analysis, and application*. Oxford University Press, 2010.
- [32] K. R. Holdaway and D. H. Irving, *Enhance Oil and Gas Exploration with Data-Driven Geophysical and Petrophysical Models*. John Wiley & Sons, 2017.
- [33] A. Peart, “Homage to john mccarthy, the father of artificial intelligence (ai),” <https://www.artificial-solutions.com/blog/homage-to-john-mccarthy-the-father-of-artificial-intelligence>, 2017.
- [34] A. Wiki. Artificial intelligence (ai) vs. machine learning vs. deep learning. [Online]. Available: <https://skymind.ai/wiki/ai-vs-machine-learning-vs-deep-learning>
- [35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [36] N. Donges, “The logistic regression algorithm,” <https://machinelearning-blog.com/2018/04/23/logistic-regression-101/>, 2018.
- [37] ml cheatsheet. (2017) Logistic regression. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html#multiclass-logistic-regression
- [38] R. Gandhi, “Support vector machine - introduction to machine learning algorithms,” <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>, 2018.
- [39] P. Dangeti, *Statistics for Machine Learning*. Packt, 2017.
- [40] W. Koehrsen, “Random forest simple explanation,” <https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d>, 2017.
- [41] N. Donges, “The random forest algorithm,” <https://machinelearning-blog.com/2018/02/06/the-random-forest-algorithm/>, 2018.
- [42] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.
- [43] tutorialspoint. (2019) Artificial intelligence - neural networks. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm

- [44] A. Wiki. A beginner's guide to neural networks and deep learning. [Online]. Available: <https://skymind.ai/wiki/neural-network>
- [45] S. Agrawal, "Hyperparameters in deep learning," <https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd>, 2019.
- [46] S. Ruder, "An overview of gradient descent optimization algorithms," <http://ruder.io/optimizing-gradient-descent/>, 2016.
- [47] A. S. V, "Understanding activation functions in neural networks," <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>, 2017.
- [48] A. Karpathy, "Cs231n convolutional neural networks for visual recognition," <http://cs231n.github.io/neural-networks-1/>.
- [49] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*, F. F. Soulié and J. Héroult, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236.
- [50] D. Mellouli, T. M. Hamdani, and A. M. Alimi, "Deep neural network with rbf and sparse auto-encoders for numeral recognition," in *Intelligent Systems Design and Applications (ISDA), 2015 15th International Conference on*. IEEE, 2015, pp. 468–472.
- [51] F. Q. Lauzon, "An introduction to deep learning," in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, July 2012, pp. 1438–1439.
- [52] K. R. Holdaway and D. H. Irving, *Enhance Oil and Gas Exploration with Data-Driven Geophysical and Petrophysical Models*. John Wiley & Sons, 2017.
- [53] A. Dertat, "Applied deep learning - part 4: Convolutional neural networks," <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>, 2017.
- [54] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert, "Gan augmentation: Augmenting training data using generative adversarial networks," *arXiv preprint arXiv:1810.10863*, 2018.
- [55] J. Rocca, "Understanding generative adversarial networks (gans)," <https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29>, 2019.

- [56] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [57] S. A. Wiki, “A beginner’s guide to generative adversarial networks (gans),” <https://skymind.ai/wiki/generative-adversarial-network-gan>, 2018.
- [58] M. Sanjeevi, “Ch:14 generative adversarial networks (gans) with math.” <https://medium.com/deep-math-machine-learning-ai/ch-14-general-adversarial-networks-gans-with-math-1318faf46b43>, 2019.
- [59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [60] A. S. Walia, “Types of optimization algorithms used in neural networks and ways to optimize gradient descent,” <https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>, 2017.
- [61] P. Popien, “How to improve your image classifier with googles autoaugment,” <https://towardsdatascience.com/how-to-improve-your-image-classifier-with-googles-autoaugment-77643f0be0c9>, 2018.
- [62] Mithi, “Vehicle detection with hog and linear svm,” <https://medium.com/@mithi/vehicles-tracking-with-hog-and-linear-svm-c9f27eaf521a>, Mar 2017.
- [63] A. Nagpal, “L1 and l2 regularization methods,” <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>, 2017.
- [64] A. Mishra, “Metrics to evaluate your machine learning algorithm,” <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>, 2018.
- [65] Q. C. Lam, L. A. T. Nguyen, and H. K. Nguyen, “A novel approach for classifying eeg signal with multi-layer neural network,” in *Proceedings of the 2017 International Conference on Robotics and Artificial Intelligence*. ACM, 2017, pp. 79–83.
- [66] C. Tan, F. Sun, W. Zhang, J. Chen, and C. Liu, “Multimodal classification with deep convolutional-recurrent neural networks for electroencephalography,” in *International Conference on Neural Information Processing*. Springer, 2017, pp. 767–776.
- [67] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, “Learning representations from eeg with deep recurrent-convolutional neural networks,” *arXiv preprint arXiv:1511.06448*, 2015.

- [68] C. Spampinato, S. Palazzo, I. Kavasidis, D. Giordano, N. Souly, and M. Shah, “Deep learning human mind for automated visual classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6809–6817.
- [69] X. Zhang, X. Chen, M. Dong, H. Liu, C. Ge, and L. Yao, “Multi-task Generative Adversarial Learning on Geometrical Shape Reconstruction from EEG Brain Signals,” *arXiv e-prints*, p. arXiv:1907.13351, Jul 2019.
- [70] J. Peirce. (2019) Psychopy 3. [Online]. Available: <https://www.psychopy.org/>
- [71] J. Proakis and D. Manolakis, “Digital signal processing, 4th-ed,” 2007.
- [72] S. W. Smith *et al.*, “The scientist and engineer’s guide to digital signal processing,” 1997.
- [73] L. Stannard. Parts of the brain that control sight. [Online]. Available: <https://www.livestrong.com/article/69157-medulla-brain-functions/>
- [74] B. Dubuc. (2002) The brain from top to bottom. [Online]. Available: http://thebrain.mcgill.ca/flash/i/i_02/i_02_cr/i_02_cr_vis/i_02_cr_vis.html#1
- [75] C. for Neuro Skills. (2019) Occipital lobes. [Online]. Available: <https://www.neuroskills.com/brain-injury/occipital-lobes.php>
- [76] G. I. of Technology, “Out of mind, out of sight: Brain’s frontal cortex controls vision: It leaves out things in plain sight, say researchers.” *ScienceDaily*, May 2016.
- [77] S. Palazzo, C. Spampinato, I. Kavasidis, D. Giordano, and M. Shah, “Decoding brain representations by multimodal learning of neural activity and visual features,” *arXiv preprint arXiv:1810.10974*, 2018.

Appendices

Appendix A

Convolutional Neural Network

A.1 CNN Architecture

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 28)	784
activation_1 (Activation)	(None, 26, 26, 28)	0
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 28)	0
dropout_1 (Dropout)	(None, 13, 13, 28)	0
conv2d_2 (Conv2D)	(None, 11, 11, 56)	14168
activation_2 (Activation)	(None, 11, 11, 56)	0
conv2d_3 (Conv2D)	(None, 9, 9, 56)	28280
activation_3 (Activation)	(None, 9, 9, 56)	0
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 56)	0
conv2d_4 (Conv2D)	(None, 2, 2, 56)	28280
activation_4 (Activation)	(None, 2, 2, 56)	0
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 56)	0
flatten_1 (Flatten)	(None, 56)	0
dense_1 (Dense)	(None, 36)	2052
activation_5 (Activation)	(None, 36)	0
dense_2 (Dense)	(None, 18)	666
activation_6 (Activation)	(None, 18)	0
dropout_2 (Dropout)	(None, 18)	0
dense_3 (Dense)	(None, 4)	76
activation_7 (Activation)	(None, 4)	0
=====		
Total params: 74,306		
Trainable params: 74,306		
Non-trainable params: 0		

Figure A.1: Proposed CNN architecture.

Appendix B

GAN Architecture

B.1 Generator

Layer (type)	Output Shape	Param #
dense_42 (Dense)	(None, 256)	25856
leaky_re_lu_28 (LeakyReLU)	(None, 256)	0
batch_normalization_16 (Batch Normalization)	(None, 256)	1024
dense_43 (Dense)	(None, 512)	131584
leaky_re_lu_29 (LeakyReLU)	(None, 512)	0
batch_normalization_17 (Batch Normalization)	(None, 512)	2048
dense_44 (Dense)	(None, 1024)	525312
leaky_re_lu_30 (LeakyReLU)	(None, 1024)	0
batch_normalization_18 (Batch Normalization)	(None, 1024)	4096
dense_45 (Dense)	(None, 2352)	2410800
reshape_6 (Reshape)	(None, 28, 28, 3)	0
=====		
Total params: 3,100,720		
Trainable params: 3,097,136		
Non-trainable params: 3,584		

Figure B.1: Proposed Generator architecture.

B.2 Discriminator

Layer (type)	Output Shape	Param #
flatten_7 (Flatten)	(None, 2352)	0
dense_39 (Dense)	(None, 512)	1204736
leaky_re_lu_26 (LeakyReLU)	(None, 512)	0
dense_40 (Dense)	(None, 256)	131328
leaky_re_lu_27 (LeakyReLU)	(None, 256)	0
dense_41 (Dense)	(None, 1)	257

=====
Total params: 1,336,321
Trainable params: 1,336,321
Non-trainable params: 0

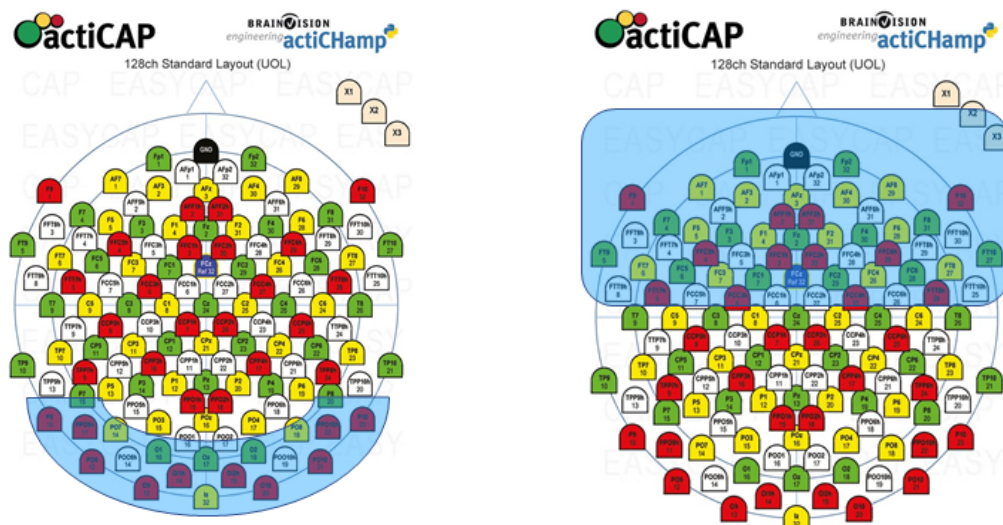
Figure B.2: Proposed discriminator architecture.

Appendix C

Classifying EEG signals with CNN graphs

C.1 Representation of occipital cortex (28 channels), and of frontal cortex (56 channels)

C.1.1 Electrode locations

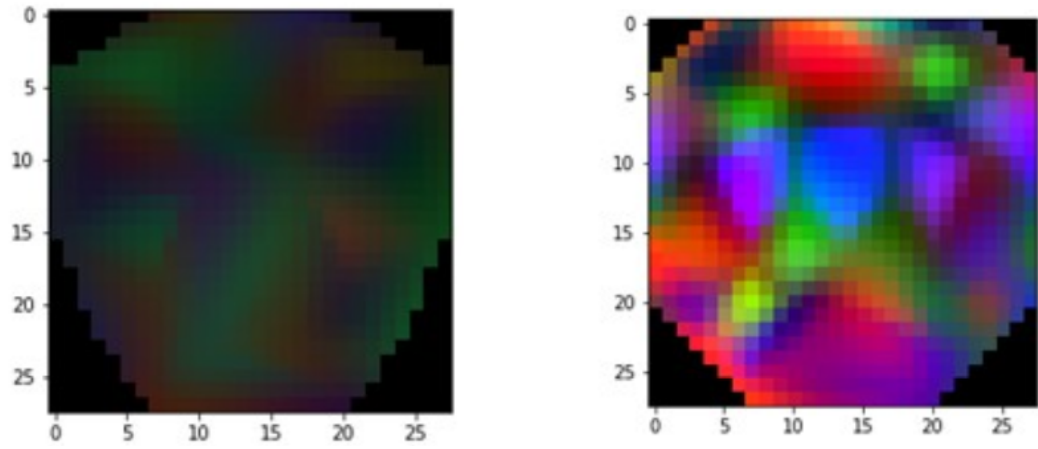


(a) Occipital cortex electrode locations.

(b) Frontal cortex electrode locations.

Figure C.1: Electrode locations of the occipital cortex and the frontal cortex.

C.1.2 2D projections



(a) 28 channels 2D image generated.

(b) 56 channels 2D image generated.

Figure C.2: 2D projected images generated.

C.1.3 Accuracy graphs

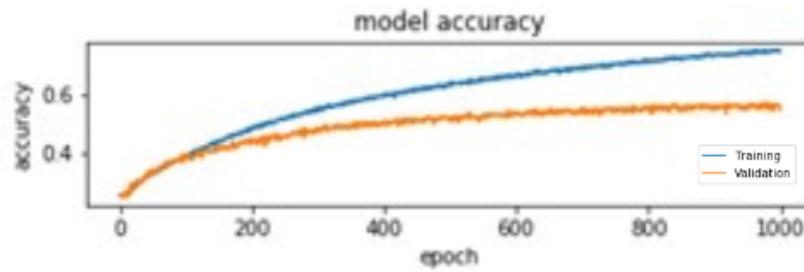


Figure C.3: 28 Channel accuracy plot

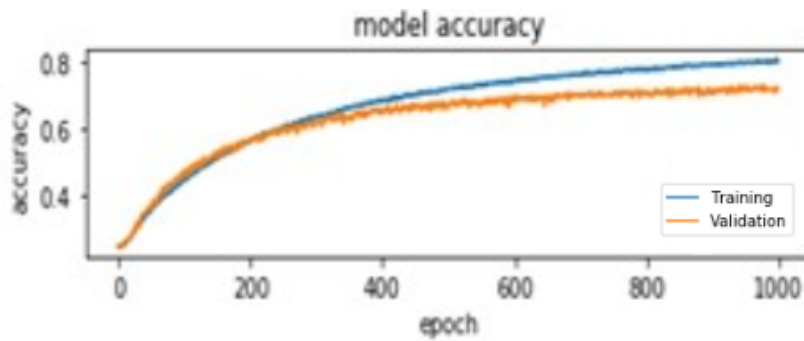


Figure C.4: 56 Channel accuracy plot

C.1.4 Loss graphs

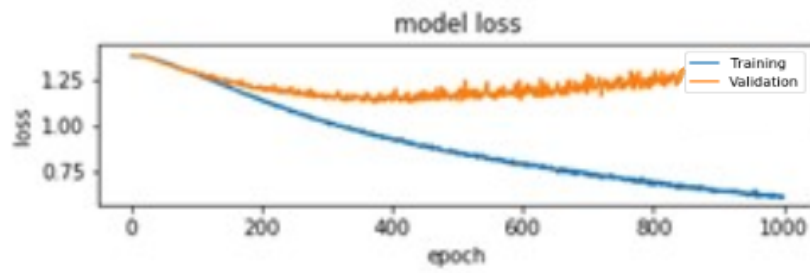


Figure C.5: 28 Channel loss plot

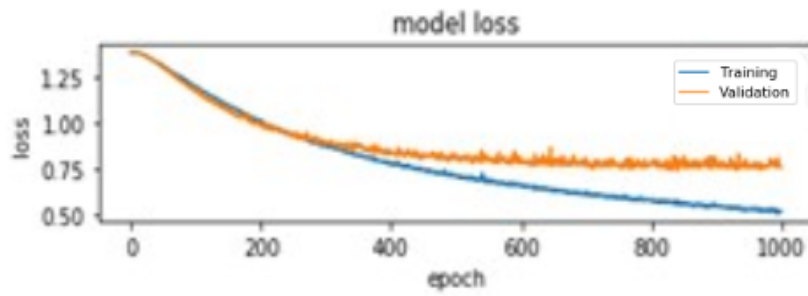
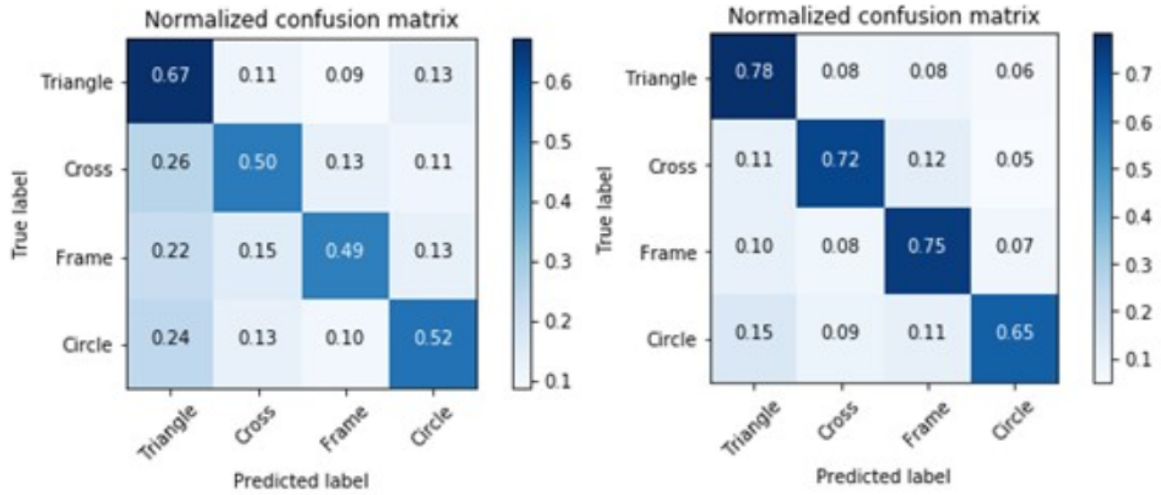


Figure C.6: 56 Channel loss plot

C.1.5 Confusion matrices



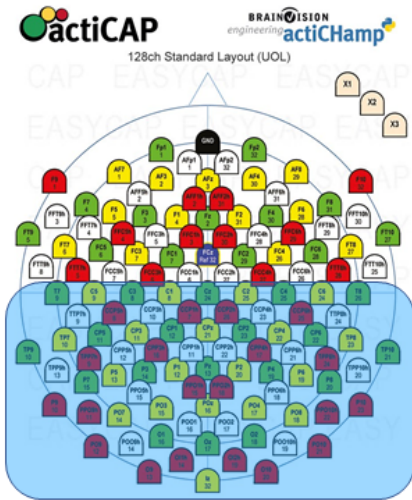
(a) 28 channels confusion matrix.

(b) 56 channels confusion matrix.

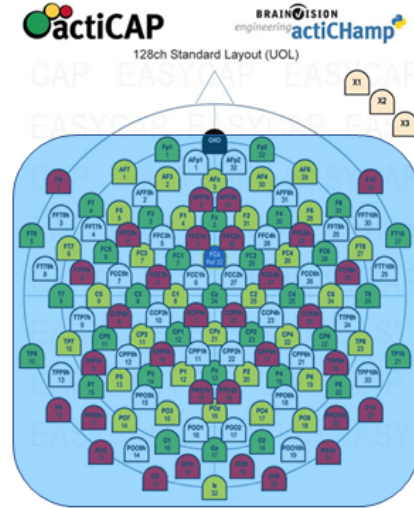
Figure C.7: 28 and 56 channel confusion matrices.

C.2 Representation of occipital and parietal cortex (72 channels), and of whole cortex (128 channels)

C.2.1 Electrode locations



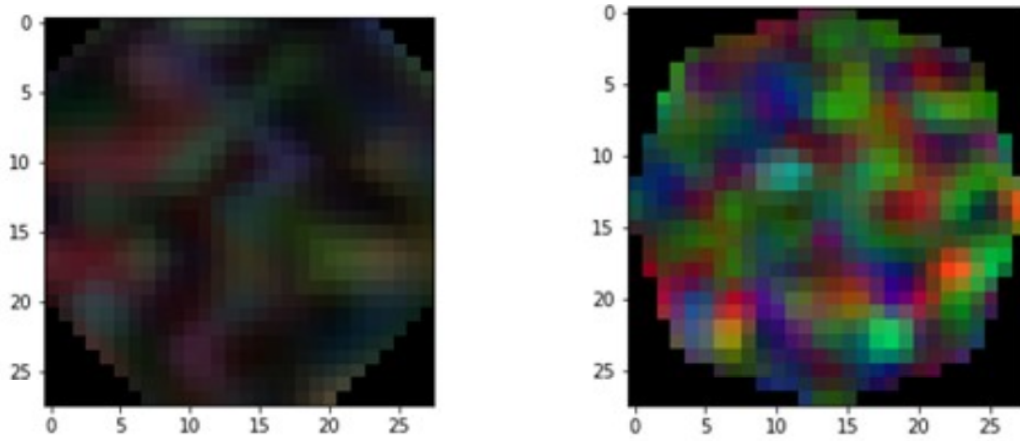
(a) Occipital + parietal cortex electrode locations.



(b) Whole cortex electrode locations.

Figure C.8: Electrode locations of the occipital + parietal cortex, and the whole cortex.

C.2.2 2D projections



(a) 72 channels 2D image generated. (b) 128 channels 2D image generated.

Figure C.9: 2D projected images generated.

C.2.3 Accuracy graphs

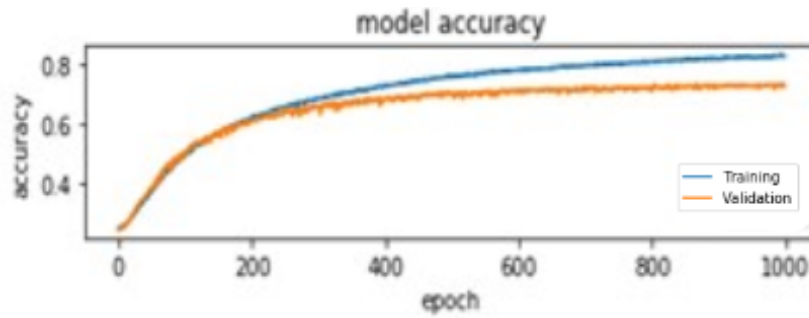


Figure C.10: 72 Channel accuracy plot

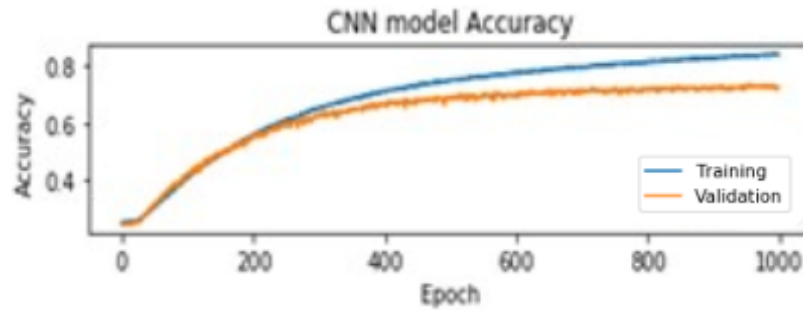


Figure C.11: 128 Channel accuracy plot

C.2.4 Loss graphs

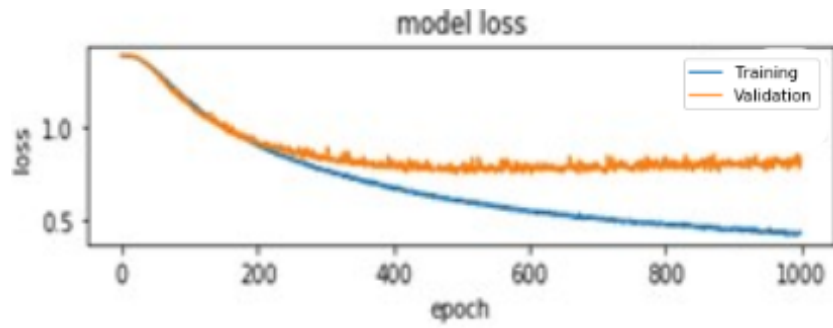


Figure C.12: 72 Channel loss plot

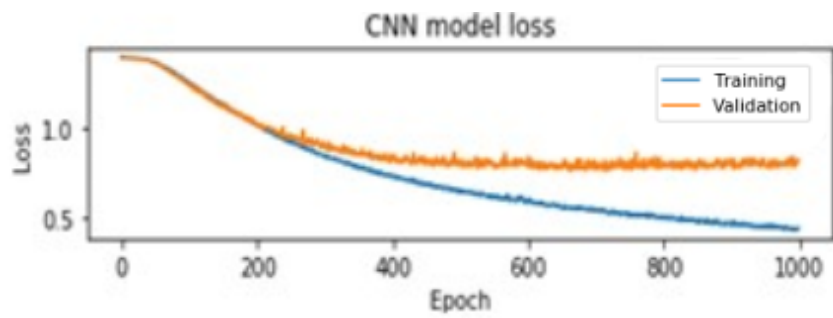
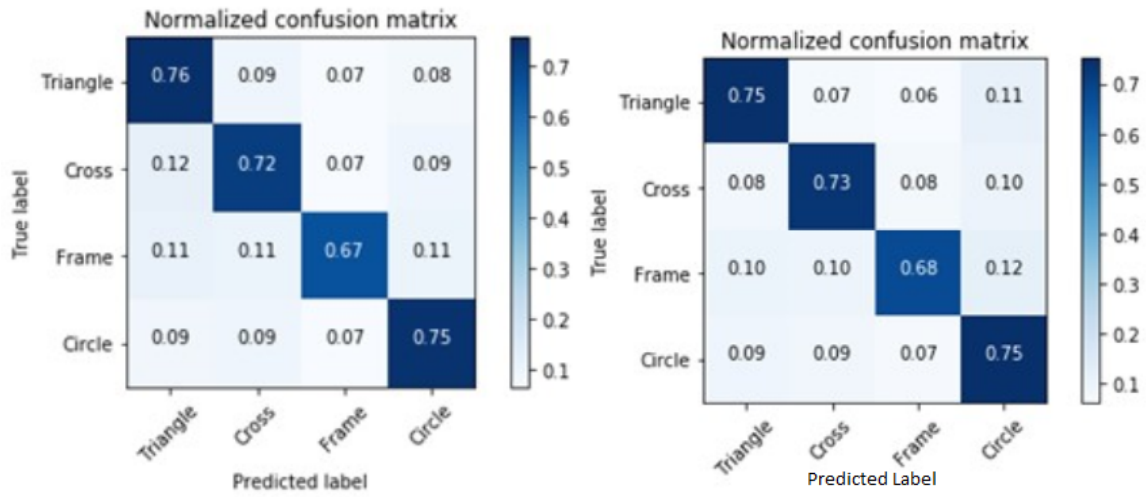


Figure C.13: 128 Channel loss plot

C.2.5 Confusion matrices



(a) 72 channels confusion matrix.

(b) 128 channels confusion matrix.

Figure C.14: 72 and 128 channel confusion matrices.