ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Brain Computer Interface Driving Movement in a Virtual Reality Game based on EEG Signals

Georgios Ramiotis

Thesis Committee:

Professor Aikaterini Mania

Professor Michail G. Lagoudakis

Professor Georgios Karystinos

Technical University of Crete

School of Electrical and Computer Engineering

April 2024

Abstract

In the last decades there has been a growing research interest on how we can connect the human brain with a computer and how we can benefit from that. The research field of Brain Computer Interfaces (BCIs) tries to achieve just that. BCIs are systems that rely on external devices that read users' bio-signals through Electrocardiography (ECG), Electromyography (EMG) and Electroencephalography (EEG). Signal processing aims to distinguish between varied induced user states, which are then translated into actions for the BCI. For now, there are three standard BCI paradigms that induce user states: the P300 paradigm which is an event-related potential (ERP) component that measures a user's reaction to a stimulus; the Steady State Visually Evoked Potential (SSVEP) which are signals that are natural responses to visual stimulation at specific frequencies; and the Motor Imagery paradigm which is a mental process by which an individual rehearses or simulates a given action. This emerging technology has seen an expanding use in medical rehabilitation, neurofeedback, control of an exoskeleton and quite recently neurogaming.

This thesis focuses on the use of a Brain Computer Interface using the Motor Imagery paradigm and based on EEG signals in a virtual reality environment for neurogaming. There are existing software tools, such as OpenVibe, that offer a straightforward methodology of developing BCIs based on EEG signals. However, the classification system of these tools are based on supervised machine learning algorithms, such as Support Vector Machines and Linear Discriminant Analysis, to classify EEG patterns induced by BCI paradigms. These machine learning methods tend to perform poorly on EEG data derived from users with no experience in motor imagery, due to the difficulty of simulating motor actions and, thus, not producing easily separable EEG patterns. For this reason, we have developed a BCI that controls movement in a virtual reality maze game that is based on OpenVibe for EEG signal processing. This BCI replaces the machine learning-based classification system with a Convolutional Neural Network to classify EEG features. Since, neural networks require a large amount of data for training, we have designed a Wasserstein Generative Adversarial Network to learn from the recorded EEG features and augment the dataset to produce more samples. We have also developed a brain-controlled interaction system that enables the player to interact with in-game props and User Interface menus using the BCI. We put this system to the test by comparing the performance and accuracy our neural network offers to that of OpenVibe's machine learning algorithms. We also measure the performance of the system with a varied number of available user states. The results have shown that using Deep Learning further improves the accuracy of the system, compared to OpenVibe.

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Διεπαφή Εγκεφάλου με Υπολογιστή για Κίνηση σε Παιχνίδι Εικονικής Πραγματικότητας βασισμένο σε Σήματα Εγκεφαλογραφήματος (EEG)

Γεώργιος Ραμιώτης

Εξεταστική Επιτροπή:

Καθηγήτρια Αικατερίνη Μανιά

Καθηγητής Μιχαήλ Γ. Λαγουδάκης

Καθηγητής Γεώργιος Καρυστινός

Πολυτεχνειο Κρητης

Σχολη Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων

Απρίλιος 2024

Περίληψη

Τις τελευταίες δεκαετίες υπάρχει ένα αυξανόμενο ενδιαφέρον για το πως μπορούμε να συνδέσουμε τον ανθρώπινο εγκέφαλο με έναν υπολογιστή και πως μπορούμε να επωφεληθούμε από αυτό. Το ερευνητικό πεδίο των Διεπαφών Εγκεφάλου με Υπολογιστή προσπαθεί να πετύχει ακριβώς αυτό. Οι διεπαφές εγκεφάλου υπολογιστή είναι συστήματα που βασίζονται σε εξωτερικές συσκευές, οι οποίες καταγράφουν τα βιοσήματα χρηστών μέσα από την Ηλεκτροκαρδιογραφία, την Ηλεκτρομυογραφία και την Ηλεκτροεγκεφαλογραφία. Η επεξεργασία σήματος στοχεύει στην διάκριση μεταξύ ποικίλων επαγομένων καταστάσεων, οι οποίες μετά μεταφράζονται σε εντολές για την διεπαφή εγκεφάλου υπολογιστή. Επί του παρόντος, υπάρχουν τρία τυποποιημένα παραδείγματα διεπαφών εγκεφάλου υπολογιστή, που προκαλούν αυτές τις καταστάσεις: το P300 παράδειγμα, το οποίο είναι ένα δυναμικό που σχετίζεται με συμβάντα (ERP) συστατικό που μετρά την αντίδραση ενός χρήστη σε ένα ερέθισμα· τα Οπτικά Προκαλούμενα Δυναμικά Σταθερών Καταστάσεων, τα οποία είναι σήματα παραγόμενα από την φυσική αντίδραση σε οπτικά ερεθίσματα συγκεκριμένων συχνοτήτων· και το παράδειγμα Κινητικής Αναπαράστασης, το οποίο είναι μια νοητική διαδικασία στην οποία ένα άτομο επαναλαμβάνει ή προσομοιώνει μια δοθείσα κίνηση. Αυτή η αναδυόμενη τεχνολογία παρουσιάζει επεκτεινόμενη χρήση στην ιατρική ανάρρωση, νευροανάδραση, έλεγχος εξωσκελετού και πρόσφατα, στα νευροπαιχνίδια.

Αυτή η διπλωματική εστιάζει στην χρήση μιας Διεπαφής Εγκεφάλου με Υπολογιστή χρησιμοποιώντας το παράδειγμα Κινητικής Αναπαράστασης και βασισμένο σε σήματα εγκεφαλογραφήματος (EEG) σε περιβάλλον εικονικής πραγματικότητας για νευροπαιχνίδι. Υπάρχουν εργαλεία λογισμικού, όπως είναι το OpenVibe, τα οποία προσφέρουν μια ευθύ μεθοδολογία για την ανάπτυξη διεπαφών εγκεφάλου υπολογιστή βασισμένο σε σήματα εγκεφαλογραφήματος. Ωστόσο, το σύστημα κατηγοριοποίησης αυτών των εργαλείων βασίζονται σε αλγορίθμους εποπτευόμενης μηχανικής μάθησης, όπως είναι οι Support Vector Machines και Linear Discriminant Analysis, που κατηγοριοποιούν τα EEG μοτίβα προκαλούμενα από τα παραδείγματα διεπαφών εγκεφάλου υπολογιστή. Αυτοί οι αλγόριθμοι μηχανικής μάθησης τείνουν να παρουσιάζουν χαμηλή απόδοση σε EEG δεδομένα προερχόμενα από χρήστες με μηδενική εμπειρία στην κινητική αναπαράσταση, λόγω της δυσκολίας προσομοίωσης κινήσεων, και έτσι, παράγοντας μη εύκολα διαχωρίσιμα EEG μοτίβα. Για αυτόν τον λόγο, έχουμε αναπτύξει μια Διεπαφή Εγκεφάλου με Υπολογιστή, η οποία ελέγχει κίνηση σε παιχνίδι λαβυρίνθου εικονικής πραγματικότητας, βασισμένο στο OpenVibe για την ανάλυση του σήματος. Επίσης, αυτή η διεπαφή εγκεφάλου υπολογιστή αντικαθιστά το σύστημα κατηγοριοποίησης βασισμένο σε αλγορίθμους μηχανικής μάθησης με ένα Συνελικτικό Νευρωνικό Δίκτυο για την κατηγοριοποίηση των EEG χαρακτηριστικών. Εφόσον, τα νευρωνικά δίκτυα απαιτούν μεγάλο αριθμό δεδομένων για εκπαίδευση, έχουμε σχεδιάσει ένα Wasserstein Παραγωγικό Ανταγωνιστικό Δίκτυο το οποίο μαθαίνει τα καταγεγραμμένα EEG χαρακτηριστικά και επεκτείνει το σύνολο

δεδομένων για την παραγωγή περισσότερων δειγμάτων. Επιπλέον, έχουμε αναπτύξει ένα εγκεφαλικά ελεγχόμενο σύστημα αλληλεπίδρασης που επιτρέπει στον παίχτη να αλληλεπιδρά με αντικείμενα και μενού διεπαφής χρήστη εντός του παιχνιδιού με την χρήση της διεπαφής εγκεφάλου υπολογιστή. Εξετάσαμε την απόδοση του συστήματος συγκρίνοντας την ακρίβεια που προσφέρουν τα νευρωνικά δίκτυά μας, με αυτήν των αλγορίθμων μηχανικής μάθησης του Οπενΐβε. Επίσης, καταγράψαμε την απόδοση του συστήματος με ποικίλο αριθμό καταστάσεων χρήστη. Τα αποτελέσματα παρουσιάζουν αύξηση στην ακρίβεια του συστήματος με την χρήση Βαθειάς Μάθησης, σε σχέση με το OpenVibe.

Acknowledgements

Initially, I would like to thank my supervisor Professor Katerina Mania for her continuous guidance and support throughout this thesis and for giving me the opportunity to work on such an interesting subject.

Next, this work would not be possible if not for my brother Lefteris, my friends Dimitris, Rafail and Michalis, who were more than eager to participate in the testing experiments for this thesis.

Last, but not least, I am truly grateful to my father Kostas and my mother Vana, who supported me in every way possible throughout my undergraduate studies.

Contents

List of Figures

Acronyms

BCI Brain Computer Interface

CNN Convolutional Neural Network

ECG Electrocardiography

EEG Electroencephalography

EMG Electromyography

fNIRS Functional Near-Infrared Spectroscopy

GAN Generative Adversarial Network

LDA Linear Discriminant Analysis

MI Motor Imagery

SPL Signal Power Log

SSVEP Steady State Visually Evoked Potential

SVM Support Vector Machine

TCP Transmission Control Protocol

UI User Interface

VR Virtual Reality

WGAN Wasserstein Generative Adversarial Network

1 Introduction

1.1 Brief Description

A Brain Computer Interface (BCI), sometimes also called Brain Machine Interface (BMI), is a direct communication pathway between the brain and an external device, typically a computer or an exoskeleton, such as a robot or robotic limb. They rely on standard paradigms to produce unique electrical patterns that can then be classified into varied actions for the external device. The most widely used paradigms are the P300 and Steady State Visually Evoked Potential (SSVEP) paradigm that measure the reaction of the brain to stimulus; and the Motor Imagery (MI) paradigm in which a user imagines physical movement of a limb. They are most commonly used in researching and mapping a brain's electrical activity and also in augmenting or repairing human cognitive or sensory-motor functions. The main philosophy behind a BCI is to skip the intermediary process of the physical movement of body parts and also to erase the distinct line between the human brain and a machine.

BCIs have now been an emerging research field, with researchers, universities and companies trying to explore the full potential of this technology. There has already been conducted many research trials of BCIs in the fields of medical rehabilitation, robotics, neurofeedback, user authentication using brain signals. In recent years, there has been another emerging use case of BCIs, that is neurogaming, which tries to offer an even more immersive experience than that of conventional games by eliminating the intermediary devices, such as a keyboard or a controller, and by directly controlling the game's mechanics with a brain's electrical signals. BCIs are complex systems that require accuracy and fast performance and are often challenging to implement. This has created the need to develop software tools that offer a straightforward methodology to implement a BCI. One of the most popular software platforms for the development of BCIs is OpenVibe. OpenVibe offers a variety of algorithms for recording, visualising, filtering, extracting and classifying a brain's electrical signal. Many of the existing software tools, including OpenVibe, rely on traditional machine learning methods to classify a brain's electrical activity and, thus, lack in accuracy that is crucial for a BCI.

Virtual Reality (VR) is another field that has been growing rapidly in recent years. Virtual Reality offers the user, the ability to explore other worlds, be part of them and interact with them, through a VR headset. It offers a unique and an immersive experience that cannot be replicated with standard computer screens. This has raised the question whether it is possible to further augment the experience Virtual Reality gives. For this reason, there has been a highly growing demand for VR games that can be controlled with a more direct method. BCIs

can be paired with VR games to implement a more direct method of control, while augmenting the experience of the user. Many of the BCIs that are used in VR games are based on software tools that, as stated before, lack in accuracy and performance, which degrades the experience making it unrealistic.

In this thesis, we developed a Brain Computer Interface using the MI paradigm and based on Electroencephalography (EEG) signals, to directly control movement in a VR maze game. This BCI is based on the OpenVibe software platform for processing the EEG signal and replaces its classification system a Convolutional Neural Network to improve the classification accuracy of EEG features. This makes the experience of the game more accurate and direct, while improving the immersiveness of the virtual environment.

More specifically, we employed the use of an Enobio 8 EEG headset and the NIC2 software it provides, to record the user's electrical brain activity. We utilized the MI paradigm to induce unique electrical signal patterns. In the MI paradigm the user simulates physical movement of a limb indicated by visual cues. The number of simulated actions is varied.

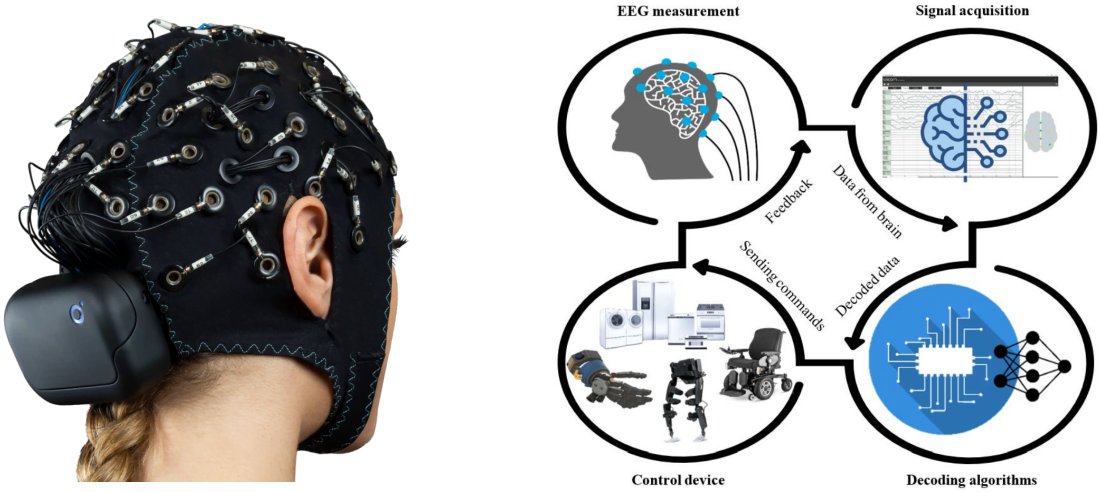

Figure 1: Images of an EEG Device(left, Soterix Medical Mobile EEG) and a simple BCI setup(right) [1]

After the signal is recorded, we used the BCI software platform, OpenVibe, to process the signal and extract EEG features related to the electrical patterns of simulated movements. We used OpenVibe's large library of algorithms to filter the signal from noise and other artifacts derived from eye blinking, minor muscle movements and external stimulus from the environment. Then, we implemented an algorithmic pipeline to extract important features of the filtered signal. The extracted features are then fed to a machine learning algorithm that tries classify

the simulated action.

Then, we employed the use of neural networks to improve the classification accuracy of OpenVibe while also producing stable results. In more detail, we implemented a Wasserstein Generative Adversarial Network (WGAN) to learn from the distribution of EEG features recorded in OpenVibe, and produce artificial data to augment the recorded dataset. Additionally, we designed a Convolutional Neural Network (CNN) to read the augmented dataset and classify the simulated actions more accurately. Both of the Neural Networks were written in Python using the powerful Deep Learning framework PyTorch.

For the VR game, we used the Unity game engine that offers a feature-rich environment to develop realistic 3D computer graphics and Virtual Reality environments. We implemented three levels for the game, a main menu scene and a training scene that will integrate with the OpenVibe software for the training of the user using the MI paradigm. All of the associated scripts were written in Microsoft's C# programming language, which Unity integrates into its game engine to facilitate the creation of complex scripts that dictate the behavior of the games.

As mentioned before, the number of simulated actions for the MI paradigm in BCI training can be varied. We tested our BCI for 2, 3 and 4 classes/actions and for our VR game, each implemented level was designed to accommodate the controls of a 2, 3 and 4 class system respectively. For all class systems, the simulated movement of feet and the simulated state of rest was needed to represent the basic states of forward movement and rest. The 3 class system adds the simulated movement of right hand for extended functionality in the game and the 4 class system adds left hand movement. The reason for adding classes in our BCI and what it achieves for the control system of our game is explained in more detail in Chapter 5.

After finishing the implementation of the BCI and the VR game, we needed a connection pathway among the OpenVibe software, our Neural Networks and our VR game. This was accomplished by establishing a local network of Transmission Control Protocol (TCP) sockets. This local network is responsible for transmitting data from the OpenVibe software to our Neural Network for classification; and for transmitting the prediction from our Neural Network to the VR game where it is responsible for translating the prediction into an appropriate command.

We established different evaluation systems for the BCI and the VR game. More specifically, we added visualization plots, in OpenVibe, to demonstrate the effects on the signal, due to filtering and feature extraction algorithms. For the the Neural Networks, we implemented a statistical metric system to test the accuracy of the system and validate the integrity of the results. To evaluate our VR game we prepared a short questionnaire that inquires the user about the overall experience, the flexibility of the control system and the accuracy of the BCI.

The main contributions of this thesis is a Deep Learning based classification system for classifying EEG signals that improves performance over algorithms implemented in OpenVibe and a VR game that utilizes this classification system of the BCI to drive movement, control UI menus and in-game props.

1.2 Thesis Structure

Chapter 1 is brief introduction to Brain Computer Interfaces, Virtual Reality and a presentation of existing tools that are used to develop BCIs. This Chapter also, briefly demonstrates our implementation of a BCI and a VR game, their purpose and their requirements.

Chapter 2 describes the research that was conducted in order to better conceptualise what a BCI and VR is. In this chapter, the history of BCIs and VR is presented and the principle behind them ,as well as, their use cases, is explained in more detail. Some of the most popular VR headsets are presented and the difference in functionality is described. Furthermore, the use of Deep Learning techniques in modern systems and the impact in performance and stability is analyzed. More explicitly, this chapter offers a more detailed explanation on the methodology of Generative Adversarial Network (GAN)s and Neural Networks.

Chapter 3 outlines, in more detail, the requirements for this thesis as well as the general system design. The EEG headset used in this thesis is presented and a report on its technical specifications is provided. This section also offers a more detailed look into the OpenVibe BCI platform and the Unity game engine that were used. In addition, the python PyTorch framework is presented and it's powerful tools for making Deep Neural Networks are analyzed.

Chapter 4 analyzes the implementation of the BCI based on the tools and frameworks we used, in full detail. All the processing stages of the EEG signal are explained and a direct comparison between OpenVibe's machine learning algorithms and our Deep Neural Networks is offered. The different statistical metrics and visualization techniques that were used to validate the performance of our implementation are also stated in this section.

Chapter 5 provides a detailed technical layout of our VR game. A presentation of all the scenes of the game, the control system of movement, as well as the interaction of in-game props and in-game menus is offered. In this section, the impact of the varied number of classes on the freedom of movement; and freedom of interaction with User Interface (UI) menus, is also interpreted.

Chapter 6 provides the evaluation system of the BCI and how well it performs with machine learning and Deep Learning algorithms. The evaluation of the VR game is also provided in this chapter, presenting a satisfaction score.

Chapter 7 presents a summary of the results and conclusions derived from our work and ending with a few remarks and suggestions on further improving this thesis.

2 Research Overview

2.1 Introduction

For the preparation of this thesis, research was conducted on various topics about BCIs, VR and Deep Learning. This chapter will provide an essential overview of the history of BCIs; a detailed explaination of the principles behind BCIs; a thorough presentation of the use cases of BCIs as well as many recent examples of related work. Furthermore, a brief outline of the history of VR will be presented with explanation on how immersion is achieved along with some examples on the most recent VR headsets. Finally, an exhaustive analysis will be provided on Deep Learning, specifically Neural Networks and Generative Adversarial Network, where their principles, architecture and use cases will be discussed.

2.2 Background on Brain Computer Interfaces

2.2.1 Definition

A brain–computer interface (BCI), sometimes called a brain–machine interface (BMI) or smartbrain, is a direct communication pathway between the brain's electrical activity and an external device, most commonly a computer or robotic limb. BCIs are often directed at researching, mapping, assisting, augmenting, or repairing human cognitive or sensory-motor functions. They are often conceptualized as a human–machine interface that skips the intermediary component of the physical movement of body parts, although they also raise the possibility of the erasure of the discreteness of brain and machine. Implementations of BCIs range from non-invasive (EEG, MEG, MRI) and partially invasive (ECoG and endovascular) to invasive (microelectrode array), based on how close electrodes get to brain tissue. [8]

2.2.2 History

The early history of BCIs dates back in 1924, when German psychiatrist Hans Berger recorded the first samples of a brain's electrical activity which led to the development of Electroencephalography (EEG). Berger was able to identify the alpha and beta frequency waves of the brain and published his findings in his paper titled "Über das Elektrenkephalogramm des Menschen", in 1930. [9]

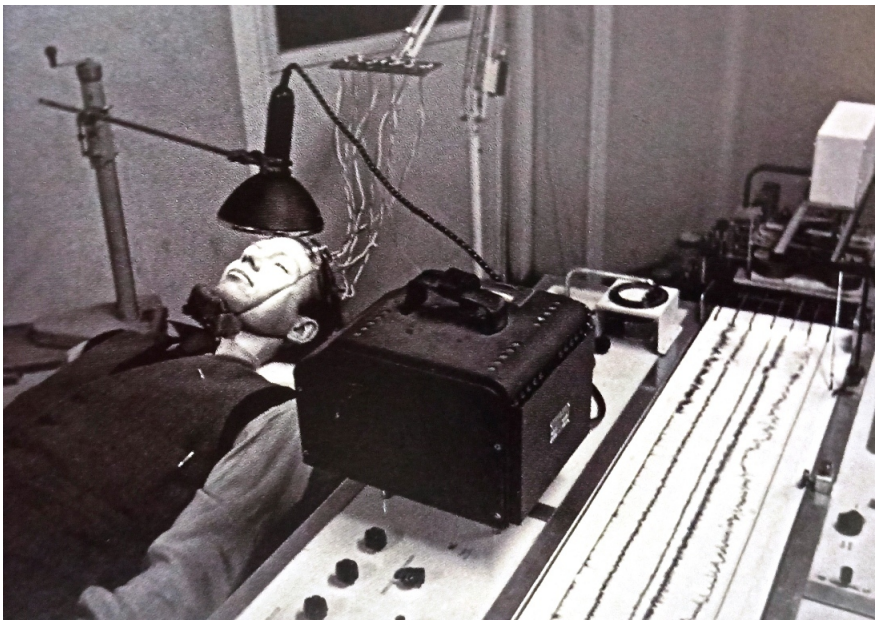

Figure 2: The Electroencephalogram – 1924. A person undergoing an EEG Test in the 1950s. Around this time the procedure was first used to look foreign of epilepsy in patients

However, the main direct research on BCIs began in 1976, when scientists from the Brain-Computer Interface Laboratory, UCLA in California provide evidence that single-trial visual evoked potentials could be used as a communication channel effective enough to control a cursor through a two-dimensional maze. In 1987, Phillip Kennedy builds the first intracortical brain-computer interface by implanting neurotrophic-cone electrodes into monkeys. In 1988, EEG signals are used to control a mobile robot. [10]

In recent decades, studies on BCIs focus mainly on medical applications, specifically on restoring sensory motor functions of the human body [11, 12] but they also have shown that BCI technologies can be used to enable functionality such as controlling the navigation of a wheelchair. [13]

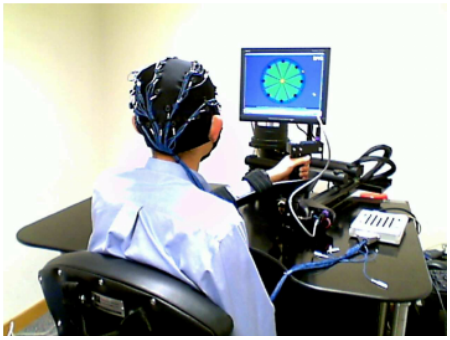

Figure 3: BCI-based robotic rehabilitation system [2]

Figure 4: BCI Control of a Motorized Wheelchair for Disabled Individuals using a calibrationless SSVEP system `https://www.youtube.com/watch?v=qhK572LJhSc`

2.2.3 Principles of BCIs

As mentioned before, a BCI is a system where the bio-signals of a user can be recorded, processed and then translated into commands for an interactive application. For this to be done, the BCI system must first be trained with recorded bio-signals from a user. This is accomplished by a long procedure of recording the user's signal, processing it and finally training a classifier that will classify the signal into classes/actions for the BCI application. The most popular ways of extracting a user's bio-signals are Electrocardiography (ECG), Electromyography (EMG), Electroencephalography (EEG) and Functional Near-Infrared Spectroscopy (fNIRS). In Electrocardiography a heart's electrical activity through repeated cardiac cycles is recorded. It is an electrogram of the heart which is a graph of voltage versus time of the electrical activity of the heart using electrodes placed on the skin. These electrodes detect the small electrical changes that are a consequence of cardiac muscle depolarization followed by re-polarization during each cardiac cycle (heartbeat). Electromyography is a technique for evaluating and recording the electrical activity produced by skeletal muscles. EMG is performed using an instrument called an electromyograph to produce a record called an electromyogram. An electromyograph detects the electric potential generated by muscle cells when these cells are electrically or neurologically activated. The signals can be analyzed to detect abnormalities, activation level, or recruitment order, or to analyze the bio-mechanics of human or animal movement. Functional near-infrared spectroscopy is an optical brain monitoring technique which

uses near-infrared spectroscopy for the purpose of functional neuroimaging. Using fNIRS, brain activity is measured by using near-infrared light to estimate cortical hemodynamic activity which occur in response to neural activity. Alongside EEG, fNIRS is one of the most common non-invasive neuroimaging techniques which can be used in portable contexts. Furthermore, Electroencephalography, which is the focus of this thesis, is a method to record an electrogram of the spontaneous electrical activity of the brain. The bio-signals detected by EEG have been shown to represent the postsynaptic potentials of pyramidal neurons in the neocortex and allocortex. It is typically non-invasive, with the EEG electrodes placed along the scalp (commonly called "scalp EEG") using the International 10–20 system, or variations of it. [14, 15, 16, 17]

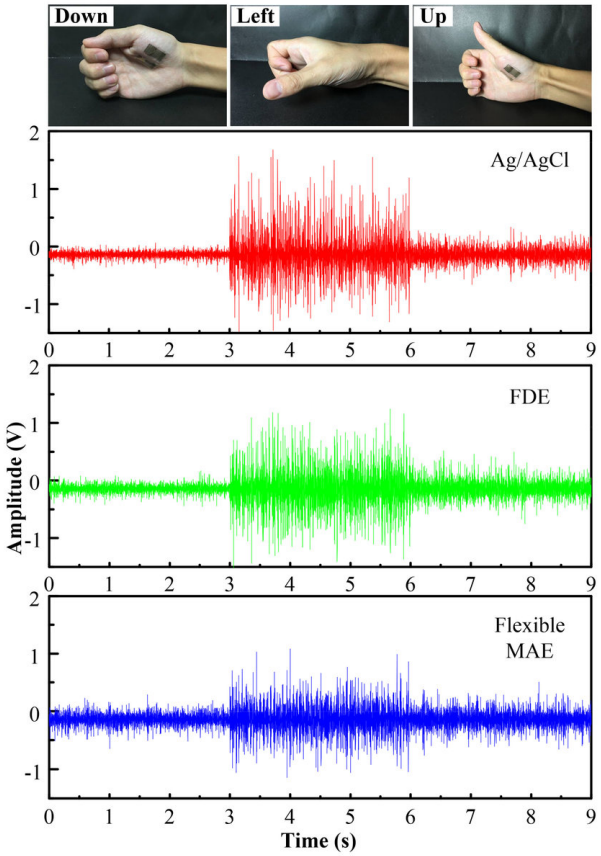

Figure 5: EMG signals of three thumb motions recorded by Ag/AgCl electrode, FDE and flexible MAE [3]

More specifically, in the case of EEG, a researcher is more interested in the varied frequency bands that the human brain operates in. In more detail, the

frequency bands of the brain's electrical activity are the gamma band, the beta band, the alpha band, the theta band and the delta band. The gamma band operates on frequencies greater than 30hz, in the fronto-central areas of the brain, and is mostly found on normal adults. In this frequency band occurs the voluntary movement of body limbs by having the highest frequencies with the lowest amplitudes. The beta band operates in the frequency range of $14 - 30$hz, in parietal; somatosensory;frontal; and motor areas, with amplitudes of 30µV and indicates whether a human is in panic condition, when in high levels. In the frequency range of $8 - 13$hz in the occipital and parietal regions, resides the alpha band that indicates a relaxed state of mind, and can be used to assess a subject's anxiety and emotional tension. Lastly, the theta and delta bands in the frequency range of $0.1 - 7.5$hz operate in the hippocampus region and the thalamus region of the brain, and they suggest a deeply relaxed state. In these bands, the subconscious activity of the brain and artifacts from minor muscle movements, are occurring. [18]

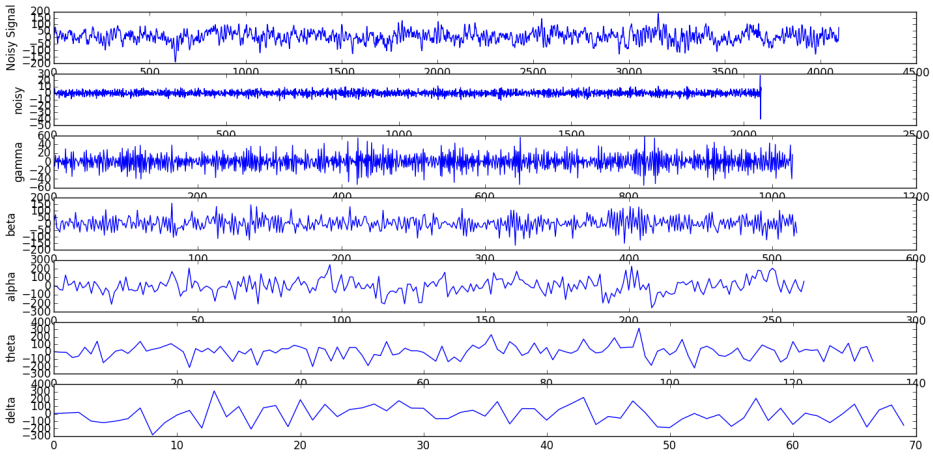

Figure 6: EEG frequency bands [4]

In the context of EEG-based BCIs, it is necessary to choose a paradigm that will dictate the procedure in which the user's brain will generate unique electrical patterns in the frequency bands that were mentioned above. [19] There are three main paradigms that most BCIs are based on, and are categorized as passive and active paradigms. In passive paradigms, the desired electrical patterns are produced from a user's reaction to external stimulus, with the most popular being the P300 and SSVEP. The P300 refers to the amplitude assessment of measured reaction of a user's brain to a stimulus, in the time window of $300 - 400$ms after

it was presented. It is a endogenous event-related potential and is thought to reflect processes involved in stimulus evaluation or categorization. [20, 21] Steady State Visually Evoked Potential (SSVEP) are signals that are natural responses to visual stimulation at specific frequencies. When the retina is excited by a visual stimulus ranging from 3.5 Hz to 75 Hz, the brain generates electrical activity at the same (or multiples of) frequency of the visual stimulus. SSVEPs are useful in research because of the excellent signal-to-noise ratio and relative immunity to artifacts. SSVEPs also provide a means to characterize preferred frequencies of neo-cortical dynamic processes. SSVEPs are generated by stationary localized sources and distributed sources that exhibit characteristics of wave phenomena. [22] The main active paradigm is the Motor Imagery (MI) paradigm, where an individual rehearses or simulates a given action. It is widely used in sport training as mental practice of action, neurological rehabilitation, and has also been employed as a research paradigm in cognitive neuroscience and cognitive psychology to investigate the content and the structure of covert processes (i.e., unconscious) that precede the execution of action. In some medical, musical, and athletic contexts, when paired with physical rehearsal, mental rehearsal can be as effective as pure physical rehearsal (practice) of an action. It uses visual cues to indicate the action that needs to be imagined. After that, a feedback period is given for the user to imagine the appropriate action and this procedure is repeated for a number of trials based on the number of actions/classes. [23] However, the MI paradigm requires a substantial amount of user training to yield EEG patterns that could be separable by machine learning algorithms such as Support Vector Machines and Linear Discriminant Analysis. [24]

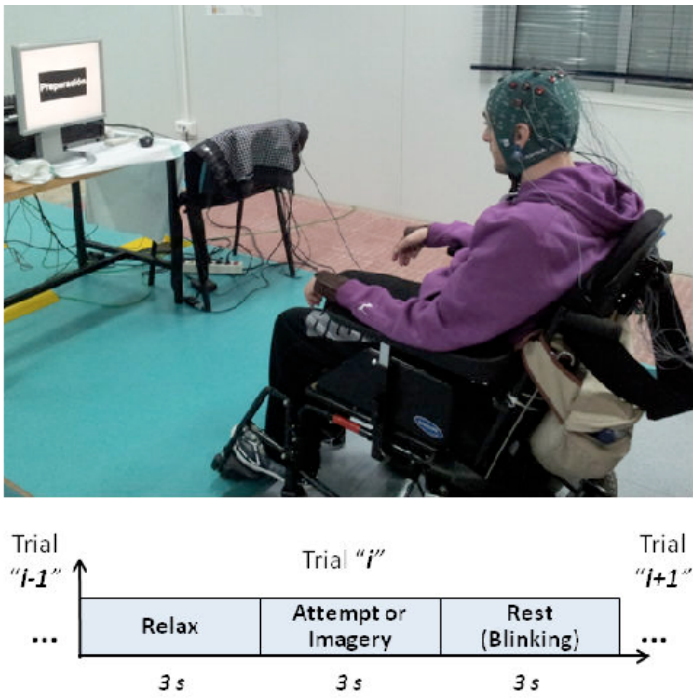

Figure 7: Example of Motor Imagery [5]

After the preferred bio-signal is generated and recorded, using a BCI paradigm, the signal analysis process follows, responsible for the translation of the bio-signal into appropriate commands for the BCI. According to Padfield et al [25], the signal analysis process is categorized as follows. After extracting the raw bio-signal data from the external device, a technique called pre-processing is employed to clean artifact noise and improve the quality of the collected data. Then the process of feature extraction is performed to remove redundant information and retain only the data that is related to the application. Finally, during the classification step the signal is categorized as a certain action and the corresponding command is performed by the system.

In the pre-processing stage, signal processing techniques are employed to denoise the signal. There are two type of artifacts that produce noise in the signal: extra-physiologic and physiologic artifacts. Extra-physiologic artifacts are occured with stimulus from the external environment and equipment during signal acquisition such as electrode artifacts, alternating current artifacts and artifacts from interference with other equipment. Physiologic artifacts occur from the human body rather than the brain, and come from minor muscle movements, jaw clenching, eye blinking, breathing and heart beats. Some of the most popular denoising algorithms are temporal filtering where the use of high-pass, low-pass or band-

pass filters is employed to cut off the frequencies where artifacts are usually found; spatial filtering which helps to recover the brain's original signal by gathering the relevant information that is spread over different channels,and reduces the dimensionality of the signal to less number of spatially filtered signals; and Cauchy filtering which targets the impulsive noise since it has a more powerful distribution than the Gaussian distribution to capture abrupt changes in noise. [26]

After the processing the signal comes the feature extraction stage. In this stage the denoised signal is pipelined to a series of feature extraction and feature selection methods. This is one of the most important stages in the signal analysis because the redundant information is discarded and the remaining signal is processed to extract feature patterns related to the activity we want to measure. Feature extraction algorithms operate in different domains of the signal and are categorized as: time-domain, frequency-domain, time-frequency domain and common spatial patterns techniques. In the time-domain, the most prominent techniques are Linear Prediction (LP) and Independent Component Analysis (ICA). LP is a technique where the extracted output feature of the signal is a linear combination of past output values along with past and present input values. The goal of the ICA algorithm is to linearly decompose a multidimensional data vector into statistically independent components, meaning that each produced independent component is a linear combination of the original input signal. This results in a separated signal where noisy components can be discarded. [27, 25] The most popular algorithm for frequency analysis is the Fast Fourier Transform (FFT). FFT is an algorithm that calculates the discreet Fourier transform of a signal focusing on efficiency. It is used to filter and estimate the power spectrum of the signal. [28, 29] For time-frequency analysis a modified version of FFT is used called Short Term Fourier Transform (STFT), with the difference being that the FFT is applied within a fixed time window that is moved along the time series in order to estimate the changes in power and phase of signals over time. Furthermore, Discrete Wavelet Analysis (DWT) is also a powerful time-frequency algorithm that works as filters that divide the signal into two sub-bands at each level called approximations and details. The approximations are the high-scale, low-frequency components of the signal and the details are the low-scale, high-frequency component. For every subsequent level, the signal is divided into two sub-bands and the number of samples is halved. [30, 31, 32, 33] Finally, the common spatial pattern technique is an algorithm that provides a set of spatial filters of the signal. These filters are obtained from a learning procedure, where the variance of spatially filtered signals is larger for one class compared to the other class. [34]

Lastly, the classification stage is responsible for learning from features, that were processed and extracted in earlier stages of the signal analysis, and classify them into different classes. The classification stage is the main factor of accu-

racy and performance of the BCI. The algorithms used for the classification of the signal, can be varied from machine learning algorithms to Deep Neural networks, depending on the BCI application. In this chapter, we will focus on traditional machine learning algorithms. Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA) are the most robust machine learning algorithms used in many BCI applications, and are recognized for their performance and efficiency. More explicitly, SVM aims to correctly classify actions by finding the optimal hyper-plane that separates two classes. The optimal hyper-plane is calculated by maximizing the margin between the classes, and is linearly represented by a vector w and a bias vector b. These two hyper-parameters are learned during the training phase. The reason that SVM chooses the hyper-plane with the maximum margin is that it is expected to generalize better in an unseen dataset. The optimal hyper-plane is calculated by solving the following minimization problem:

$$\begin{aligned} \min_{w,b,\xi} \frac{1}{2} w^T w + c \sum_{i=1}^{N} \xi_i \\ s.t: \ y_i(w^T \Phi(x_i) + b) \leq \xi_i + 1 \\ \xi_i \geq 0, \quad i = 1, \ldots, N. \end{aligned}$$

Where ξ_i is a slack variable relaxing the constraints of perfect separation of the two classes and $\Phi(x_i)$ is the function that maps variable x to a dimension. On the other hand, LDA tries to calculate the best discriminating projection direction so that the distance between the classes is maximized, while the distance within a class is minimized. In the case of two classes, this is accomplished by assuming that the conditional probability density functions $p(\overrightarrow{x}|y = 0)$ and $p(\overrightarrow{x}|y = 1)$ are both the normal distribution with mean and covariance parameters (μ_0, Σ_0) and (μ_1, Σ_1), respectively. Then the Bayes-optimal solution is calculated, for a threshold T, by the equation:

$$\frac{1}{2}(\overrightarrow{x_0} - \overrightarrow{\mu_0})^T \Sigma_0^{-1} (\overrightarrow{x_0} - \overrightarrow{\mu_0}) + \frac{1}{2} \ln |\Sigma_0| - \frac{1}{2}(\overrightarrow{x_1} - \overrightarrow{\mu_1})^T \Sigma_1^{-1} (\overrightarrow{x_1} - \overrightarrow{\mu_1}) - \frac{1}{2} \ln |\Sigma_1| > T \quad (1)$$

LDA also makes the assumption that the covariance matrix between the two classes has full rank (ie. $\Sigma_0 = \Sigma_1 = \Sigma$), thus from the equation above several terms cancel because:

$$\overrightarrow{x}^T \Sigma_0^{-1} \overrightarrow{x} = \overrightarrow{x}^T \Sigma_1^{-1} \overrightarrow{x} \quad (2)$$

$$\overrightarrow{x}^T \Sigma_i^{-1} \overrightarrow{\mu_i} = \overrightarrow{\mu_i}^T \Sigma_i^{-1} \overrightarrow{x}, \quad \text{because } \Sigma_i \text{ is Hermitian} \quad (3)$$

Finally, the margin between the classes can be calculated with the following

equations over a threshold constant c:

$$(1) \xrightarrow{(2),(3)} \overrightarrow{w} = \Sigma^{-1}(\overrightarrow{\mu_1} - \overrightarrow{\mu_0})$$
$$c = \frac{1}{2}\overrightarrow{w}^T(\overrightarrow{\mu_1} + \overrightarrow{\mu_0})$$

Other popular classification algorithms are k-nearest-neighbor, regression tress and fuzzy classifiers. [34, 35, 36]

Finally, after the classification stage is finished and a classifier is trained, the BCI system can read input signals from a user and output specific actions, that the classifier was trained with, which are then transmitted to an external application.

In this thesis, we implemented an EEG-based BCI using MI paradigm, because it relies on the user actively imagining body limb movement without the need for visual cues. This eliminates potential User Interface clutter such as in the P300, SSVEP paradigms where there needs to be constant visual stimulus, that can be exhaustive for the user. Furthermore, the processing of the EEG signal was implemented in the OpenVibe BCI platform as it offers an extensive library of implemented algorithms for filtering and feature extraction of the signal.

2.2.4 Use cases of BCIs

Using the set of techniques, mentioned above, opens a new way of utilizing brain activity in order to perform certain actions and thus create applications that can be beneficial to our daily lives. Most of the recent studies apply BCIs and EEG data processing to the medical field. An example of that is Martin Steinisch et al 's [37] work where they proposed a post-stroke rehabilitation BCI system using an EEG device in a VR environment. Their system consists of a passive robotic device (Trackhold) for kinematic tracking and gravity compensation and five dedicated virtual reality (VR) applications for training of distinct movement patterns. Their research aims to provide a system that will monitor task-related neural activity in an immersive VR environment, and that will augment the rehabilitation process of post-stroke patients. Another example of an application of BCIs in the medical field is Daniela Muñoz's et al. [38] research where they developed an EEG-based BCI in a virtual environment for patients with Parkinson's Disease. Their research focuses on implementing a BCI system for treating neuropsychologic function loss in Parkinson's disease by having patients engaged in a collection of serious BCI VR games.

Figure 8: Brain-Computer interface used for rehabilitation after a stroke: `https://www.youtube.com/watch?v=9rYPS8unLpE`

BCIs can also be applied for control of external devices. In Christoph Guger's et al. [39] research they developed an EEG-based BCI system using the MI paradigm where the user tries to imagine left or right hand movement to control an external prosthetic arm. They used the adaptive autoregressive technique for feature extraction, along with the LDA algorithm to classify the signal. Another example of prosthetic control is Daniel Elstob's et al. [40] work, where they developed a EEG-based BCI using the MI paradigm. The EEG signal was recorded with the Emotiv Headset and signal processing was implemented in the OpenVibe software platform.

Figure 9: BCI Control of a Prosthetic Hand with Haptic Feedback: `https://www.youtube.com/watch?v=2-AcO8kRsqM`

With the advancements of BCIs in the medical field, an interest has been developed for the use of BCIs for commercial and entertainment use. For this reason research has been conducted on BCIs for games. Studies like Marios Hadjiaros et al 's [41] research where they studied the effectiveness of BCIs and EEG as an alternate form of serious and commercial gaming. In Szczepan Paszkiel's [42] book, a detailed documentation is offered, on the viability of applying BCIs in the fields of neurogaming, VR/AR, in IoT applications and the use of BCIs for controlling mobile vehicles. Other examples of research on BCI for neurogaming is Qiang Wang's [43] et al paper, where they proposed two neurofeedback games, in which they process brain signals in order to read the user's concentration state and improve it by playing some games. However, most BCI games that are based on the MI paradigm provide support for 2 or 3 classes and that, can limit the control potential a user has on the game. Additionally, BCI games rely on supervised machine learning algorithms such as SVM and LDA for classification of MI trials. However, machine learning algorithms provide poor classification accuracy on EEG data derived from users with no experience in motor imagery-based BCIs. [44, 45, 46, 47]

Figure 10: BCI Games — BCI Game Jam 2022

In this thesis, we implemented a BCI game, where we tested the accuracy of the system for 2, 3 and 4 classes. Furthermore, we also utilized deep neural networks to improve the classification accuracy compared to that of traditional machine learning algorithms.

2.3 Background on Virtual Reality

2.3.1 History of VR

If we focus more strictly on the scope of virtual reality as a means of creating the illusion that we are present somewhere we are not, then the first attempts on virtual reality are the 360-degrees panoramic paintings that were first presented in the 19th century in Europe and the United States. In 1838 Charles Wheatstone's research demonstrated that when viewing side by side, two two-dimensional scenes, the brain processes them as a single three dimensional scene. This results in a sense of depth and immersion when viewing stereoscopic images through a stereoscope.

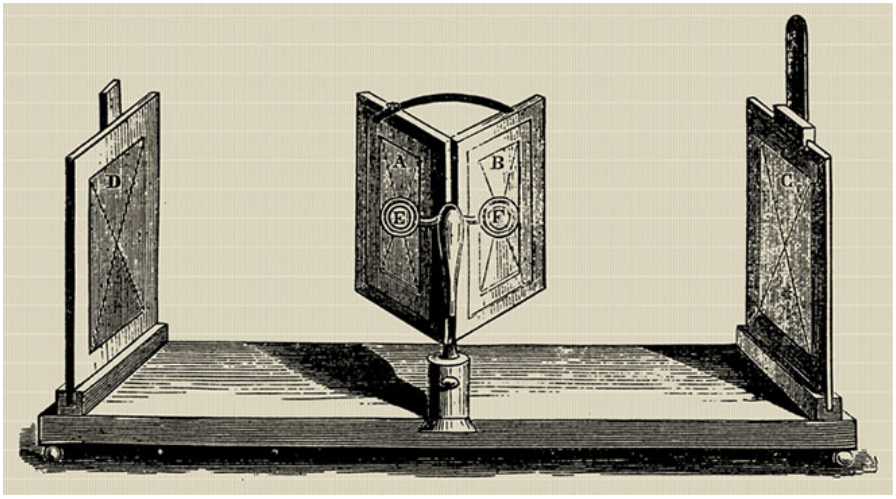

Figure 11: The Wheatstone mirror stereoscope

In the mid 1950s cinematographer Morton Heilig developed the Sensorama, a theatre cabinet that would stimulate all the senses and not just sound or sight. It featured full colour 3D video, audio, vibrations, smell and atmospheric effects, such as wind that would produce an immersive feeling.

Figure 12: The Sensorama VR machine

In 1960, Heilig developed the Telesphere Mask which was the first head-mounted display. This provided stereoscopic 3D images with wide vision and stereo sound. There was no motion tracking in the headset at this point. In 1968, Sutherland created the first virtual reality HMD called, the Sword of Damocles, where the user could see 3D virtual wire-frame images. The project did not get introduced to the general public because it was too heavy for users to comfortably wear. In the 1970s and 1980s came the boom of VR, with the most notables events being the first interactive VR platform called, VIDEOPLACE, which featured computer graphics, projectors, video cameras, video displays and position-sensing technology that would create computer-generated silhouettes; and the commercial distribution of VR goggles and gloves to the general public by VPL Research, Inc. In the last two decades there has been radical advancements in VR with the creation of many VR HMDs such as the, Oculus Rift, HTC Vive, Valve Index and recently the Apple Vision Pro. [48, 49, 50]

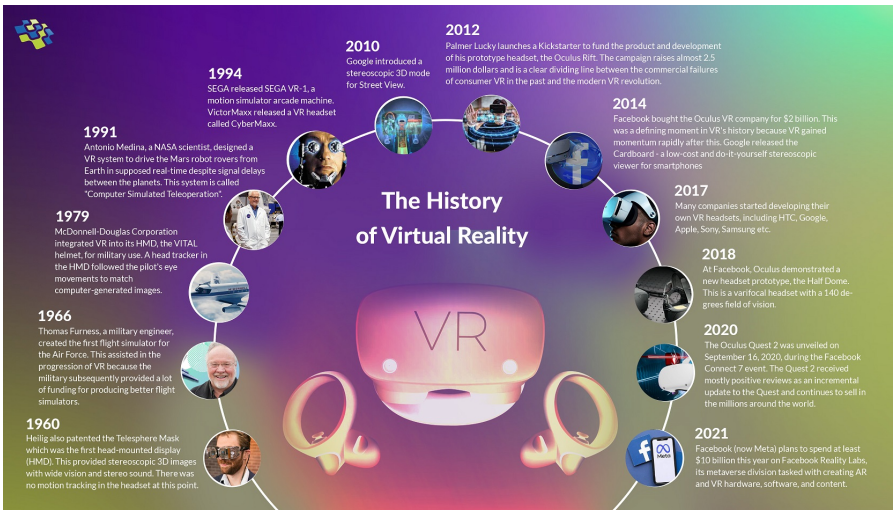

Figure 13: History of VR (BULB)

2.3.2 Immersion

Immersion refers to a vivid illusion that is generated through a computer system that aims to present the user with an extensive, diverse and accurate imitation of reality. Virtual Reality offers the user, the ability to explore other worlds, be part of them and interact with them, through a VR headset. It offers a unique and an immersive experience that cannot be replicated with standard computer screens. Virtual reality aims to stimulate all human senses through the use of visual, auditory and haptic stimulus and engage the user in a digital environment without being there. This allows for a level of immersion where the user actually feels as a part of an artificial world , offering an one of a kind experience.

With the use of VR hand-held controllers, the user can also interact with the digital world and have an active part in it. This equipment enables the user to interact with objects and analyze them. Furthermore, this experience can be further augmented with the use of BCIs where the user can control objects and movement without the need for VR controllers.

2.3.3 VR Headsets

A Virtual Reality Head Mounted Display (HMD) is a device that is meant to be worn on the head and is being used in gaming, engineering, medical applications, simulators and in personnel training. The most popular commercially available HMDs are the Oculus Rift S, Valve Index and the Apple Vision Pro, ranging from low to high price points.

Figure 14: Oculus Rift S

The Oculus Rift S is a tethered VR Headset that features a display with a resolution of 1440×1280 per eye and a screen refresh rate of 80hz. It reaches a field of view (FoV) of up to 115 degrees and it includes a variety of sensors such as accelerometers, gyroscopes and magnetometers. The touch controllers included with the headset offer accurate hand tracking and presence in a VR environment. The Rift S is headset that needs to be connected to an external PC.

Figure 15: Valve Index

The Valve Index headset is another PC-powered headset that offers a LCD display with a resolution of 1440×1600 with a 144hz refresh rate, achieving a 108-degree of horizontal FoV. It includes four base station for accurate positional tracking and two hand-held controllers for hand tracking.

Figure 16: Apple Vision Pro

The Apple Vision Pro is the most recent commercial VR headset, released in February 2024. It is a standalone VR headset powered by Apple's own M2 and R1 chips. It features a 3D Micro-LED display with a resolution 3660×3200 with 100 degrees FoV, and a supported refresh rate of 100hz. It does not include any hand-tracking controllers, however it relies on a set of external cameras and sensors, positioned on the headset, to track the movement of the hands. It also features voice and eye input to navigate its own visionOS operating system. Other important technologies of the headset is iris-based user authentication and spatial audio with dynamic head tracking.

2.4 Deep Learning

Machine Learning refers to the study of computer systems that learn from data and adapt automatically from experience without being explicitly programmed. They are the basis for many modern computer systems that rely on processing and labeling a large amount of data. Machine Learning is considered to be a subset of Artificial Intelligence. Machine learning algorithms take data as input, called training data and create models that can classify the data to an output or make predictions with no human intervention. In machine learning, the more data are fed to the training model the better the model becomes at classifying data.

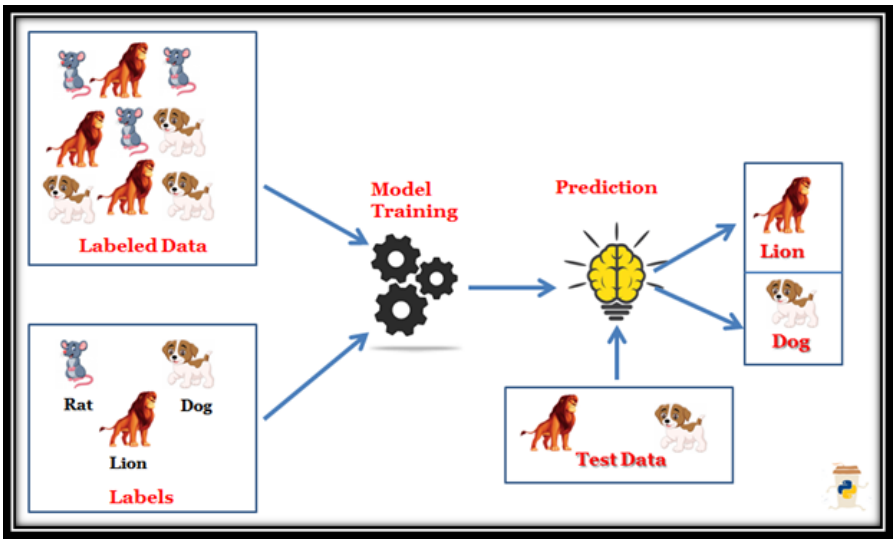

Figure 17: Supervised Machine Learning (Finxter)

Machine Learning is also used in BCIs, where the most used machine algorithms are SVM and LDA which we covered extensively in chapter 2.2.3. In the example of EEG-based BCIs using the MI paradigm, extracted EEG features, along with their labels indicating a specific imaginary movement, are fed to the machine learning algorithm for training, which then outputs a prediction class, trying to be as accurate as possible. But traditional machine learning algorithms come with limitations when processing EEG data. More specifically, when the data distribution of the EEG signal is more complex or too noisy, the machine learning algorithm cannot classify the signals and fail to deliver good accuracy. Furthermore, in the case of SVM, a BCI cannot function properly when there are more than two classes because SVM solves a binary classification problem. For this reason there has been a modified version of SVM that can handle more classes by dividing the multiclass problem into set of binary classification problems, that

SVM can solve, but this approach does not offer as good accuracy as we would expect. LDA, another binary classifier, tries to find a subspace that contains all of the class variability, but still suffers from the same problem of low accuracy when the data provided are too complex or too noisy. LDA also has a modified version to for multiclass scenarios.

Deep Learning aims to solve the problems that are present with traditional machine learning algorithms. Deep Learning is a machine learning technique that layers algorithms and computing units, or neurons, into what is called an artificial neural network. These deep neural networks take inspiration from the structure of the human brain. [51, 52]

2.4.1 Neural Networks

Artificial Neural Networks, as mentioned before, are networks with many layers of computing units called neurons that imitate the procedure of information processing in biological systems. Neural networks are a form of supervised learning, in which they are provided with input data and their respective labels, and try to learn from them in order to classify them into categories. In its most basic form a neural network is characterized by three layers: an input layer, a hidden layer and output layer. The data are inserted in the input layer which are then propagated to the hidden layer, which is responsible for learning a nonlinear mapping between input and output and then to the output layer to calculate the final prediction. Each computational layer consists of mathematically representations of processing units called neurons. The neurons in each layer are what holds the information of the data, which is then transmitted for additional processing in the subsequent layers. All neurons of each layer has a weight that amplifies or attenuates transmitted data to the next layer. Data are processed by subsequent neurons when a certain threshold is exceeded, which is determined by a function called activation function, that it is necessary for a neural network as it adds non linearity to the architecture. The most common activation function for a neural network is the ReLU (Rectified Linear Unit) which is characterized by the following equation:

$$(x)^+ = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

where $(x)^+$ is the input to the next layer. Other popular activation functions are the ELU, LeakyReLU and tahn. There are also a set of parameters, called hyperparameters, that optimize the training process of the neural network and must be set manually before training. These hyperparameters are the learning rate of the network, the number of epochs the network will be trained for, the loss function which compares the input and target values and measures the performance of the neural network in modelling the training data and the optimizer of the

network, which tunes the weights and learning rate to optimize the loss. In the binary classification problem the most basic function is the binary cross entropy, where it computes the cross entropy of the log probability distributions of the two classes; meaning that the function will return a high loss when the network classifies incorrectly a label with high probability and a low loss when the network classifies correctly a label with low probability. The binary cross entropy loss is given by the equation:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot (1 - p(y_i))$$

For the multiclass problem there is the categorical cross entropy loss function. For the optimizer of the network there is the stochastic gradient descent algorithm which is an iterative method for optimizing an objective function with suitable smoothness properties. It can be regarded as a stochastic approximation of gradient descent optimization, since it calculates the gradient on a randomly selected subset and not the entire dataset. There is also the Adam and RMSprop optimizers that are essentially the same as SGD but with the addition of momentum in the gradient which improves training speed.

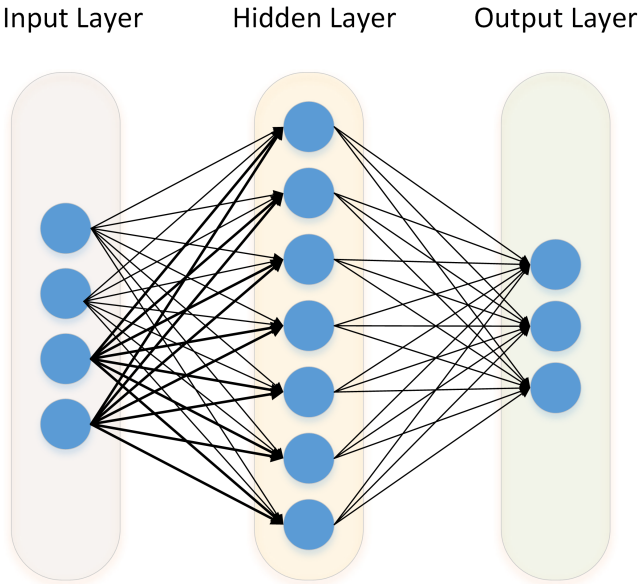

Figure 18: Multi Layer Perceptron

Deep Neural Networks consist of more than one hidden layer, structured in a nested architecture. They usually contain advanced neurons instead of simple

neurons found in ANNs. These advanced neurons allow for more complex calculations in one neuron instead of a single activation function. These characteristics allow DNNs to automatically discover an accurate representation of the input data needed for the corresponding learning task. There are many types of deep neural networks and each differs in the type of hidden layers and the proffered learning task in which they are used. The most basic type of a deep neural network is the artificial neural network that consists of more than one hidden layer, also called Multi-Layer Perceptron (MLP). Its hidden layers contains only simple neurons and the activation function. This type of hidden layer is also called a fully-connected layer as all outputs of one hidden layer are connected to all inputs of the next hidden layer. Another type of DNN is the Convolutional Neural Network (CNN), where the hidden layers consist mainly of convolutional layers that perform convolutions on the input data with a fixed kernel filter and the output is then fed to the activation function. Other parameters of the convolution are the padding which the kernel filter is applied before the start of the data sequence instead of the start of the sequence; and the stride which dictates the amount of movement between the applications of the filter. Lower stride means that the filter will be applied to all data while a bigger stride will skip some data points. [53] Frequently, in CNNs, max pooling layers are added between the convolution layer and the activation function, to reduce the dimensionality of the dataset. Batch normalization and Dropout layers are also added between the convolution layer and the activation function, to prevent overfitting and stabilize training. Other popular deep neural networks architectures are the recurrent neural networks (RNN), which are able to remember past iterations and apply this to future predictions; and auto encoders (AE) that learn efficient codings of unlabeled data. [52, 54, 55]

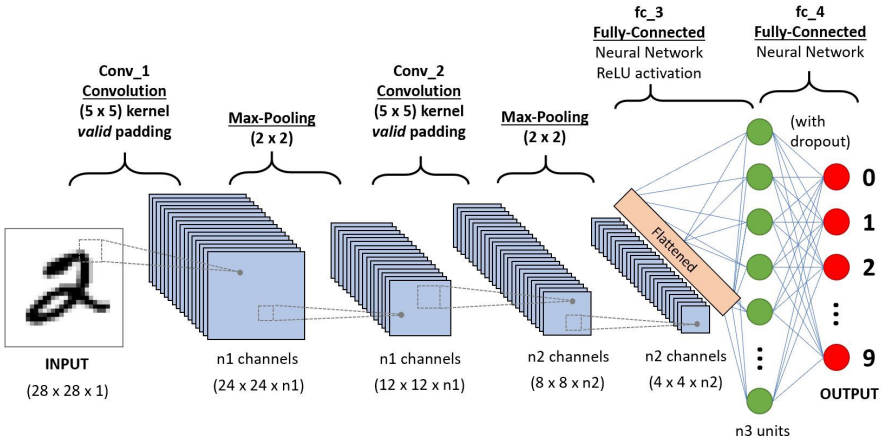

Figure 19: Convolutional Neural Network for hand-written numbers classification [6]

2.4.2 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a type of generative deep neural network, meaning that instead of learning from raw data to classify them, GANs learn from raw data to generate artificial data that imitate the input data. While deep learning has had success in many discriminatory artificial intelligence applications like language processing, image classification and speech recognition; there has been growing demand in generative models that generate close to realistic artificial data. Deep generative models have had less of an impact, due to the difficulty in approximating probabilistic computations. GANs aim to solve this problem by having the generative model in an adversarial environment. More specifically, a GAN is comprised of two models that are trained simultaneously. The generative model G that captures the data distribution, and the discriminatory model D that estimates the probability that a sample came from the data distribution rather than the generator. The goal of this adversarial environment is for the generator to maximize the loss of the discriminator. Essentially, is a min-max problem where we aim to minimise the loss of the generator while maximising the loss of the discriminator. GANs also feature a set of hyperparameters that optimize the training process, like in regular DNNs, however they differ in the loss function of the network. In the original GAN paper, the loss function was described by the following equation:

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

where $p_{data}(x)$ is the data distribution and $p_z(z)$ is the input noise distribution of the generator. From the loss function we can observe that there is no input for a label that characterizes the input data. That is because GANs are a framework that is trained unsupervised without the need for labeling the input data, and is capable of learn just from the probability distribution. The training process of a GAN is as follows. First, samples from the training data and generated data are taken and fed to the discriminator, which tries to identify the fake distribution, and the discriminator loss is extracted. Then, the generator generates a new sample of fake data and we calculate the loss. The generator is trained for every training iteration of the discriminator. [56]

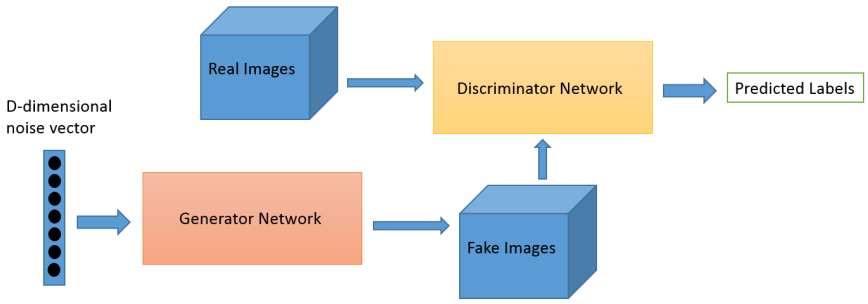

Figure 20: GAN for image generation (Medium)

While GANs are a major improvement from original generative networks, the loss function that describes the learning process, suffers from the problems of vanishing gradients in the generator and instability of generator gradient updates, which leads to the overall instability of the GAN. Arjovsky's and Bottou's paper [57] offer an extensive mathematical explanation on the instability of the GAN architecture. For the aforementioned problems, a new type of generative adversarial network was developed, the Wasserstein Generative Adversarial Network (WGAN), that keeps the framework of the original GAN but changes the loss function and the training process of the discriminator, to eliminate vanishing gradients and stabilise training.

The Wasserstein Generative Adversarial Network (WGAN) is an improved generative network that is based on the generator-discriminator adversarial environment of the original GAN architecture. The WGAN implements the $Earth - Mover(EM)$ distance, also called $Wasserstein - 1$, as the loss function of the network. The Wasserstein loss function is described by the equation:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbf{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

where $\Pi(P_r, P_g)$ is the set of joint distributions of $\gamma(x, y)$. $\gamma(x, y)$ indicates how much the probability distributions have to change in order to reach the optimal loss; and in the case of WGAN, P_r and P_g are the data distribution and generated distribution respectively. Compared to the original GAN, the WGAN also implements a different training process for the discriminator, which WGAN calls the critic. More specifically, the critic is trained for every iteration of the training process, while the generator is trained for every $2 - 5$ iterations. Every training iteration the weights of the critic are clipped as a way to enforce a Lipschitz constrain. This training process combined with the Wasserstein distance as the loss function eliminates vanishing gradients and the network is more stable. [58] Other types of improved GANs are the DCGAN, which introduces convolutional layers in the generator's model architecture; and the CGAN, where the model is provided with additional information such as labels or any modal data that improves the estimation of the conditional probability.

In the context of BCIs, research papers have proposed implementations of GANs to generate artificial EEG signals to augment the recorded dataset [59, 60, 61]. However these implementations need many epochs of training and each channel needs to be trained separately leading to a long training time. In this thesis, we implemented a WGAN for augmentation of our recorded dataset by training the WGAN only on the EEG features and not on the signal and generating all channels simultaneously, thus, needing less epochs for training. Furthermore, a CNN was designed to learn from the augmented dataset and provide better accuracy for our BCI than machine learning methods.

3 Requirements and Technological Background

3.1 Introduction

In this chapter of the thesis we will cover the requirements for implementing a BCI in virtual environment with improved performance, by expanding the toolset of OpenVibe with the use of Deep Neural Networks. Furthermore, we will provide a layout of the general system design, the hardware that was used as well as the software tools to implement the BCI and the VR game. More explicitly, we will present the EEG headset that was used for recording EEG signals, the VR headset for them game and the technical specifications of the PC that was tested on. We will analyse the OpenVibe software platform and its capabilities; present the PyTorch platform used for the development of our neural networks; and the Unity 3D game engine along with its tools.

3.2 Requirements

This thesis aims to integrate an EEG-based BCI system with a VR game as a more direct method of controlling movement, while also improving the general usability and accuracy of the system. To achieve this, the BCI system and the VR game must satisfy some requirements, in order to not hinder player movement; and integrate with all the other software tools. More specifically, the requirements to ensure a reliable and performant system are specified below.

- The EEG signal that is recorded with the EEG headset must be as noise-free and artifact-free as possible. It is imperative to adhere to the general recommendations of EEG recording procedures and use the EEG headset per the EEG headset's company instructions. This will ensure that good quality data will be transmitted to the OpenVibe platform.

- It must be ensured that the OpenVibe platform receives the data from the EEG headset in a secure manner where no data loss or latency of the signal is occurred. BCI systems rely on the integrity of the signal both in training and regular use of the system.

- In the processing stage, the signal must be denoised and possible artifacts from the external environment or from minor muscle movements, must be cleaned. Then the redundant information of the signal must discarded keeping only information related to the imagined movements from the MI paradigm.

- Data must be transmitted from OpenVibe to our neural networks in an intuitive way securing the accuracy of data flow.

- The implementation of the neural networks must be lightweight without compromising on accuracy and performance. This will ensure the ability of the system to run in computer systems with low processing power or weaker GPUs. Moreover, the training time of our neural networks will be kept at a minimum. Finally, by keeping the footprint of the neural networks architecture small will guarantee that the BCI system will run optimally in conjunction with the VR game as it relies on the host's PC for computational power.

- When it comes to the training process of the BCI, the training scene implemented in the VR environment of the game, must be presented to the user with appropriate colors as to not tire the user's eyes. The environment lighting must also be dim as this will ensure the user will follow the training procedure to the best of his ability. Furthermore, the instructions dictated

by the MI BCI paradigm must be descriptive so as the user will follow them accurately. Adequate time between instructions must be given, for the user to be able to process the visual cues that indicate the action that must be imagined. Finally, the stimulation/visual cue codes that were presented to the user must be transmitted to OpenVibe with accurate timestamps.

- The VR game must have an intuitive User Interface that leaves no question to its functionality. VR controller use must be kept at a minimum and only until the BCI is trained.

3.3 System Components

In this subsection we will present the hardware components used for the VR game, recording the EEG signal, and training the neural networks. We will also analyse the software platforms that were used for processing the EEG signal and developing the VR game, along with some frameworks and software packages.

3.3.1 Hardware

The device we used to record the EEG signal was the Enobio 8 EEG headset made by the company Neuroelectrics. It features a neoprene headcap for comfortable application to the head, with 64 tES/EEG electrode placeholders with electrode annotations in the international 10/20 system. It features 8 electrode channels with a bandwidth of 0 to 125 Hz (DC-coupled) and a sampling rate of 500hz. Its on cap controller can be connected to an external PC wirelessly via WiFi or Bluetooth; or wired via a USB cable. With a USB connection its rechargeable battery can last up to 24 hours of usage. It can offer real-time EEG monitoring and can export raw EEG data to other software platforms, such as OpenVibe, via the Lab Streaming Layer (LSL) protocol. Finally, the Enobio's packaging includes Ag/AgCl wet electrodes, Ag/AgCl dry electrodes and ECoG electrodes for semi-invasive BCI systems. The Enobio 8 can be used for the development of BCIs, ERP experiments, mobile brain imaging and hyperscanning. [62]

Figure 21: Enobio 8 EEG headset (Neuroelectrics)

For the VR headset we used the Meta Quest 2. It is a lightweight standalone VR headset that features a Fast-Switch LCD Display with a resolution of 1832×1920

per eye and a supported refresh rate of up to 90hz. It is also capable of 3D positional audio for an immersive audio experience. Finally, the meta quest 2 can connect to an external PC wirelessly via WiFi or wired via a USB 3.0 cable. [63]

Figure 22: Meta Quest 2 Standalone VR headset

Finally, the training of the neural networks and the testing of both the BCI system and the VR game, was conducted on a PC with an Intel i7 8700 CPU, 16GB of RAM memory and a Nvidia RTX 2060 graphics card with 8GB VRAM.

3.3.2 Software

NIC2 Controller

The Neuroelectrics Instrument Controller (NIC2) is a specialized software tool developed by Neuroelectrics for the Enobio 8 EEG headset, and is used to transmit and record the raw EEG signal from the Enobio 8 to a connected PC. It is an intuitive application equipped with basic and advanced modes to design and monitor any experiment involving electroencephalography and/or non-invasive brain stimulation with transcranial current stimulation (tCS). It allows for real-time Power Spectrum Density (PSD) plots, spectogram visualization and representation of scalp maps with power distribution display. Finally, it also provides line noise filtering at 50 or 60 Hz, to eliminate the noise generated by the powerline. [64]

Figure 23: NIC2 Controller (Neuroelectrics)

OpenVibe

OpenVibe is a software platform dedicated to designing, testing and using brain computer interfaces. It offers a huge library of implemented algorithms to process EEG signals in real-time. It consists of its Acquisition Server application from where the brain signals can be acquired either directly from a supported EEG headset or via the Lab Streaming Layer protocol (LSL) from a headset that supports it; and its Designer Platform where the user can design a BCI system architecture by choosing boxes, that represent implemented algorithms; dragging them into the scenario and connecting the outputs of one box to the inputs of another box. This software platform is capable of acquiring, filtering, processing, classifying and visualizing brain signals in real time. OpenVibe is supported for Windows, Ubuntu Linux and Fedora Linux operating systems.

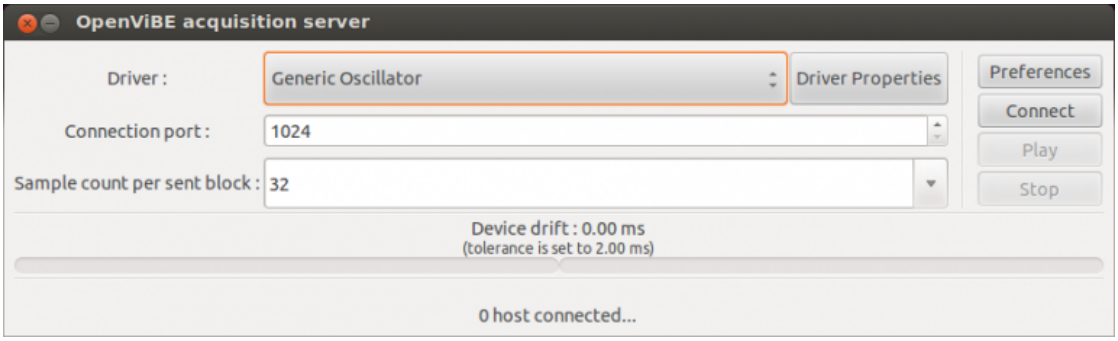

Figure 24: OpenVibe Acquisition Server

In the Acquisition Server the user can select the appropriate driver that the EEG headset supports and edit its properties such as, the connection port, sample count per sent block, sampling frequency and fallback sampling frequency. The Server also supports TCP tagging for transmitting stimulations from an external application to the server. When the driver properties are configured the server can start acquiring the raw EEG signal.

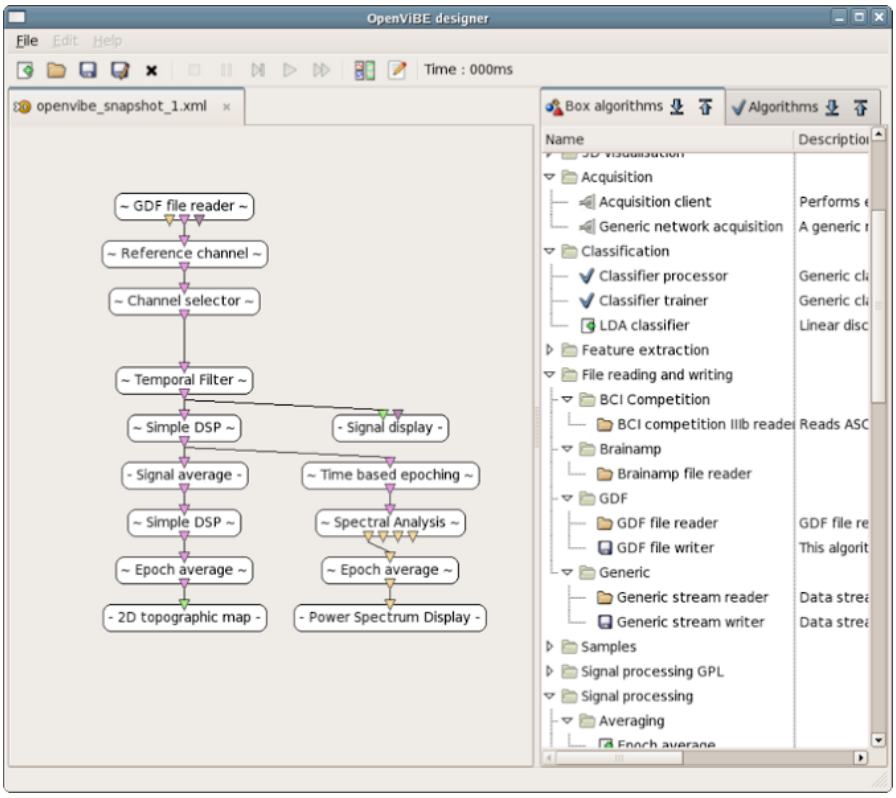

Figure 25: OpenVibe Designer

In the Designer Application the user can design his desired scenario for the BCI, by dragging the appropriate boxes into the scenario scene. OpenVibe already implements algorithms for temporal filtering, frequency analysis, time-frequency analysis, feature extraction, signal epoching, classification with machine learning algorithms such as SVM and LDA, file reading/writing and signal visualization. OpenVibe also supports external processing boxes, where the user can implement custom algorithms to integrate with the platform. [65]

Unity Game Engine

Unity is a cross-platform game engine that provides developers with a set of tools to enable the creation of 2D and 3D games on both standard computer screens and virtual/augmented reality games. It features an integrated development platform, physics engine, animation designer, support for scripting and the asset store; where the user can discover a vast library of assets and plugins that integrate with the engine. It is based on Microsoft's C# programming language for creating scripts that define the behavior of the game. [66]

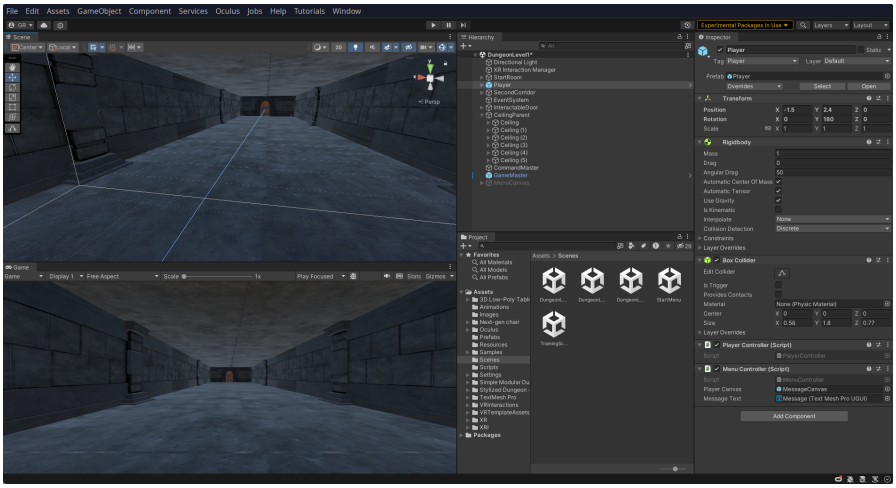

Figure 26: Unity Game Engine

Oculus Integration Plugin

The Oculus Integration is a plugin made for Unity by Oculus developers to offer out of the box integration for Oculus/Meta headsets. It supports character movement control from the VR controllers, VR head tracking and a set of scripts for assisting VR physics integration into a game. In this thesis, we used the Oculus integration plugin to integrate the Meta Quest 2 VR headset into Unity.

PyTorch

PyTorch is a powerful python machine learning framework used for designing neural networks for computer vision, large language models (LLMs) and data classification. It is based on the Torch library, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is a free and open source software with a modified BSD license. PyTorch also has a C++ interface. We used PyTorch for this thesis, to develop our generative and convolutional neural networks.

4 Brain Computer Interface Implementation

4.1 Introduction

In this chapter, a thorough analysis of the BCI implementation will be provided, based on previous research and tools that were presented in chapter 2 and chapter 3, respectively. Screenshots of the implementation of both OpenVibe scenarios and the neural networks architecture, will also be shared.

The general procedure of EEG recording and acquisition, that was followed to obtain accurate and good quality EEG samples, as well as the training protocol for the MI paradigm to generate unique brain signal patterns, will be presented in full detail. The algorithmic pipeline that was implemented to clean the EEG samples, extract the features related to the MI paradigm and classify those features into actions, will be explained.

Finally, an analysis will be provided for the classification stage of the BCI where we tested a traditional machine learning approach with the implemented algorithms of OpenVibe and a custom Deep Learning approach with the implementation of a WGAN with a CNN. The architecture of the implemented WGAN and CNN and their respective training process will be delineated in this chapter.

4.2 Training Procedure

In this subsection we will describe the training procedure we followed for the user's MI training. The MI paradigm, as explained in chapter 2.2.3 relies on spontaneous visual cues to indicate the action the user needs to imagine. Since we are testing this system for 2, 3 and 4 classes, the actions the user needs to rehearse is the rest state where the user is relaxing, feet movement, right and left hand movement. The rest state and the imagined feet movement are present on all classes. In 3 classes we add right hand movement and in 4 classes we add left hand movement. For the visual cues we used arrows pointing to a direction to indicate the movement. The arrow pointing upwards indicates feet movement; the arrow pointing left and right indicate left and right hand movement, respectively; and for the rest state there is no visual cue where the user relaxes.

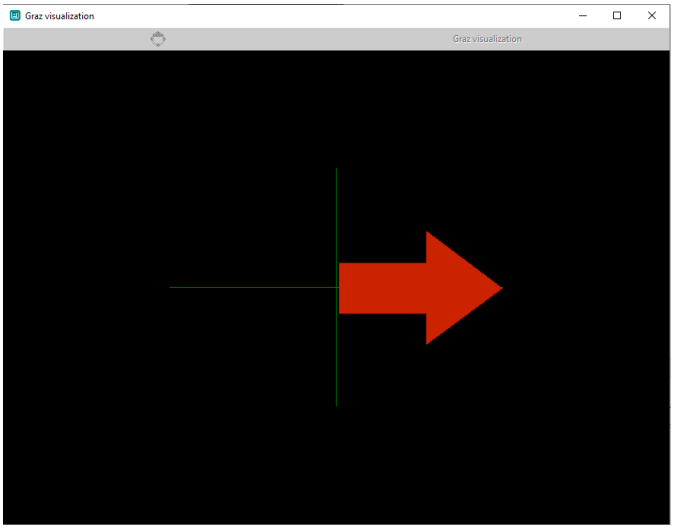

Figure 27: Example MI trial in OpenVibe

First we give the user a 20 second time window to relax. After 20 seconds the first trial begins, where an audio beep is played to prepare the user for the trial followed by a 1 second waiting time window. After the window passes, the visual cue is displayed on the screen for 1.75 seconds. Finally, when the visual cue disappears a 3.75 second time window is given to the user to imagine the appropriate movement, and the trial procedure is repeated for 20 trials for each class. When a visual cue for a trial is displayed in the screen an internal stimulation code of OpenVibe is appended to the EEG signal stream data to indicate the time point when the trial has happened. A visual representation of the time windows of the trials is given in the following figure.

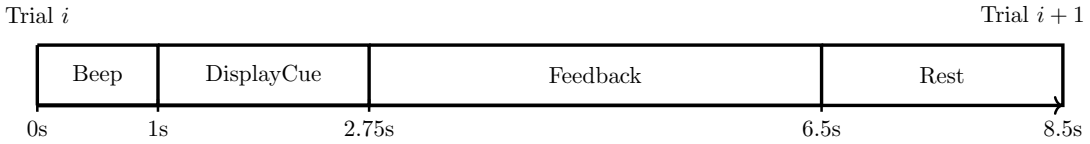

Figure 28: Single MI trial time windows

This procedure was implemented in our VR game as a standalone scene, to give the user a better environment to focus on the MI training. The VR training scene will be explained in the chapter 5.2.

4.3 Acquisition Stage

In this stage of the BCI system we will describe the policy we followed for acquiring and recording the raw EEG signal for offline training. First, comes the preparation of the headset. The Enobio 8 comes with 8 wet electrodes that consist of the electrode and its base. The bottom part of the base of the electrode is inserted in the holes of the EEG headset according to the 10-20 international system. During a trial of the MI paradigm, the main areas of the brain that are actively engaged in imagining body limb movement are the premotor cortex, primary motor cortex, somatosensory cortex and parts of the parietal lobe; as shown in the figures below. [67]

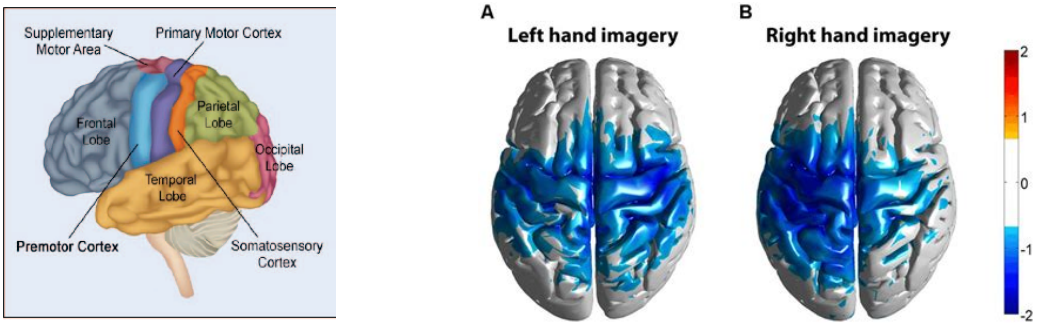

Figure 29: Areas of the brain(left) [7]. Engaged areas during MI trial(right, Motor/Imagery Task Classification ConvNET)

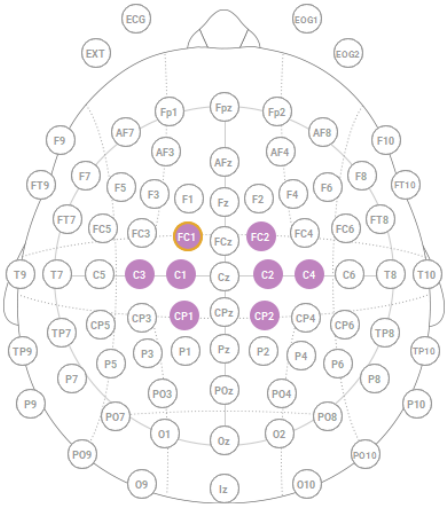

Figure 30: Electrode position of our EEG Headset

For this reason we chose to insert the electrodes base in the positions FC1, C1,

CP1, C3, FC2, C2, CP2 and C4. This ensures that we cover all engaged areas of the brain, and that we do not have missing information. Then the neoprene headcap is positioned on the head of the user. It is important to align the CZ electrode position to be in the midpoint from ear to ear and that the front of the headcap is positioned 1 to 2 cm above the eyebrows, to secure the correct alignment of the electrode annotations with the areas of the brain they indicate. The next step is to expose the scalp of the user by moving hair away from the electrode position and insert highly conductive electrode gel into the bottom part of the base, since we are using Ag/AgCl wet electrodes. This is to achieve a reliable electrical signal transmission between the scalp and the electrode. The top part of the electrode base are then screwed in. Then, we attach the electrode cables on the base along with 2 extra electrodes on the ear lobe of the user with an earclip, that will work as ground, and connect all the electrodes to the adapter. Finally, the adapter is connected to the PC via a USB connection. A USB connection was preferred to reduce the amount of latency in the recording.

In the NIC2 controller software we configure the connection with the headset so that it can transmit the EEG signal to OpenVibe via LSL. After the configuration is done and the recording is started, we get a first visualization of the signal and an indication whether the electrode connection is healthy.

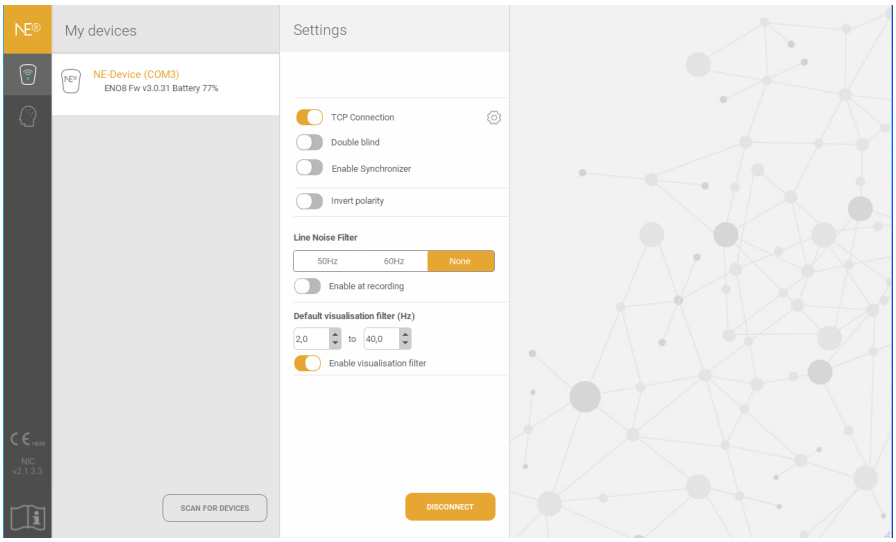

Figure 31: Enobio 8 Configuration (Neuroelectrics)

The next step is to configure OpenVibe's Acquisition server to receive the signal and marker stream from the EEG headset. We choose the LSL protocol for the input stream in the driver properties, select the signal and marker EEG stream of the headset, set the number of samples per block sent to 64 while also enabling the

TCP tagging mode to permit external stimulations to be received by the server. We chose to set the number of EEG samples to 64 as it is the best option for our, 500hz capable sampling rate, Enobio. We start the acquisition server and load the acquisition scenario in OpenVibe's designer. The acquisition scenario consists of the acquisition server that is responsible for receiving the EEG signal along with the MI stimulation codes, from the acquisition server; a signal display box that will visualize the raw EEG signal; and a Generic stream writer box that will write the EEG stream to an .ov file, which is OpenVibe's own implementation of a data file to offer the best accuracy for EEG timestamps. During the Acquisition stage is when the MI training of the user is performed, in the VR game, to record a local file of the EEG data with the stimulations of the MI trial. Finally, the recorded file will be used for offline processing and training of the classifiers for the BCI.

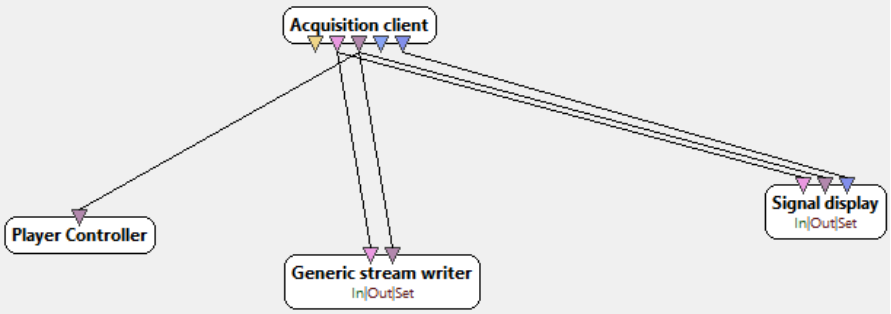

Figure 32: Acquisition Scenario

4.4 Preprocessing Stage

In the preprocessing stage, the filtering of the signal is occurred, where we remove from the EEG signal, artifacts and noise. The most common cases where noise and artifacting is present in the EEG signal are in jaw clenching, ocular activity, cardiac activity, perspiration and respiration. These artifacts are found in low frequencies that overlap with the theta and delta frequency waves, in the range of 0.1-5hz. In the range of 40-100hz the EEG signal displays artifacts that come from electromagnetic waves generated by the AC powerline, due to the insufficient or lack of wire shielding in EEG electrodes. [68]

For the above reasons, the filtering method we chose to denoise the EEG signal is temporal filtering. More specifically, we apply a band pass Butterworth filter of 4th order, with low cut-off frequency of 8hz, a high cut-off frequency of 24hz and a pass band ripple of 0dB. The band pass filter will remove from the EEG signal all frequencies where noise and artifacts are present. Furthermore, we keep the frequency band of 8-24hz where the alpha and beta bands are active during a MI trial. A Butterworth filter was chosen due to its maximally flat characteristic in the passband, and the 4th order of the filter allows for better amplitude characteristics. A Butterworth filter also provides better efficiency since it needs only 4 past outputs to calculate the current output making a suitable choice for online classification. [69]

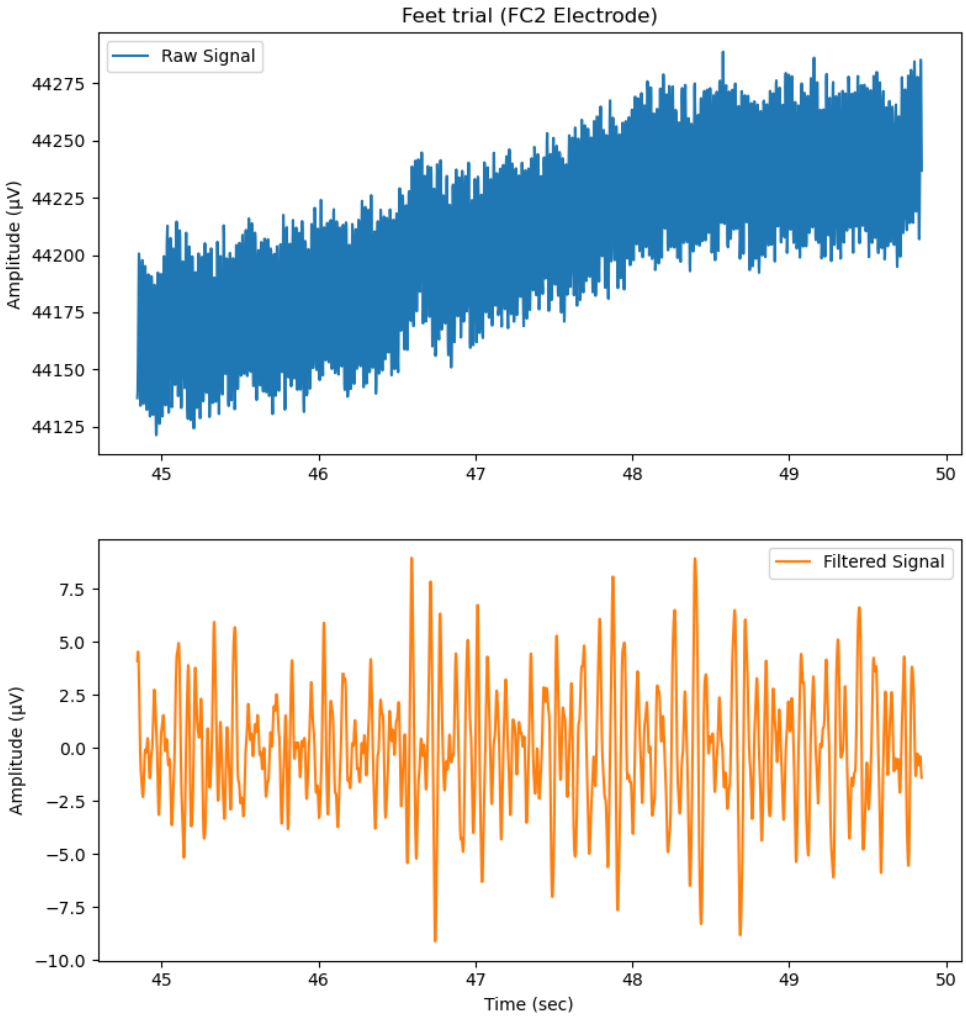

Figure 33: Raw and Filtered EEG Signal for Feet trial

4.5 Feature Extraction Stage

In this stage of the BCI, the filtered EEG signal is epoched and segmented into small time series keeping only the signal that is related to the MI trials. With feature extraction techniques we transform the filtered signal into statistical features and aggregate them into a feature vector. In this thesis, we initially epoch our EEG signal into 4 seconds time windows based on the start of a stimulation MI trial that has occurred with a 0.5 second offset, using OpenVibe's stimulation based epoching box. This epoching will result in separate signal streams, one for each MI class. More specifically, in the case of 2 classes, we will have one EEG stream comprised of all 4 second time windows where imagined feet movement has occurred, and one similar stream for the rest state.

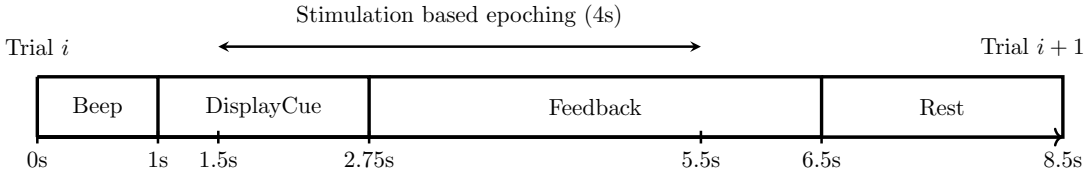

Figure 34: Stimulation based epoching

Then the 4 second epoched signal is segmented into 1 second signal chunks every 0.0625 seconds using OpenVibe's time based epoching box, to create samples from where we will extract the features of the signal. The final step is to perform statistical analysis on the extracted signal streams and aggregate the results in a feature vector. On each of the three extracted signal streams we calculate the logarithmic band power of the signal, referred as Signal Power Log (SPL) in OpenVibe, which indicates a signal's power content per unit time. This metric provides the squared amplitude distribution of the signal over the selected time windows. The signal power log metric is given by the equation:

$$Signal\ Power\ Log = \log(1 + \frac{1}{N}\sum_{i=0}^{N-1} x_i^2), \quad N = length(x)$$

Where x is every signal point in 1 second of signal.

Finally, the three metric vectors are aggregated into a feature vector resulting in 980 features($49\ SPL\ features \times 20\ trials$) from 8 feature channels ($8\ electrodes$) for each class; and are pipelined to the classifiers for classification.

Figure 35: Feature distribution

4.6 Classification Stage

In the final processing stage of the EEG signal, the aggregated vector is fed as input to the classifiers along with the stimulation codes of the MI trial as labels. In this thesis, we developed a Wasserstein Generative Adversarial Network (WGAN) to augment the extracted dataset of features and a Convolutional Neural Network (CNN) to classify the features into actions. We tested our system's accuracy against OpenVibe's own approach to classification with machine learning algorithms.

4.6.1 OpenVibe

In OpenVibe the available machine learning algorithms for classification are Support Vector Machine (SVM), Linear Discriminant Analysis (LDA) and a Multi-Layer Perceptron with one hidden layer. In this tesis, we chose the SVM and LDA algorithms to classify the feature vectors. For SVM we chose a C-SVC SVM type with a radial basis kernel function, as it provided better performance, and a degree of 3. The epsilon parameter was set to 0.1, the epsilon tolerance was set to 0.001, the cost factor to 1 and the cache size to 100. LDA was trained with no shrinkage. The hyperparameters' values for both algorithms were chosen based on the recommended defaults of OpenVibe and based on experimental results. Since both algorithms are binary classifiers we chose for the case of the multiclass problem the OneVsAll strategy. With this strategy a set of binary classifiers are trained, one for each class that will classify features against the rest of classes, resulting in N classifiers for N classes. Compared to OneVsOne strategy, where a different classifier is trained for each different pair of classes leading to $\frac{N(N-1)}{2}$ classifiers, the OneVsAll strategy is less computationally expensive. Finally, the classifiers are validated with the k-fold validation strategy for 5 folds where the training dataset is the $\frac{4}{5}$ of the original and the validation dataset is the remaining $\frac{1}{5}$. This validation procedure is repeated for five times.

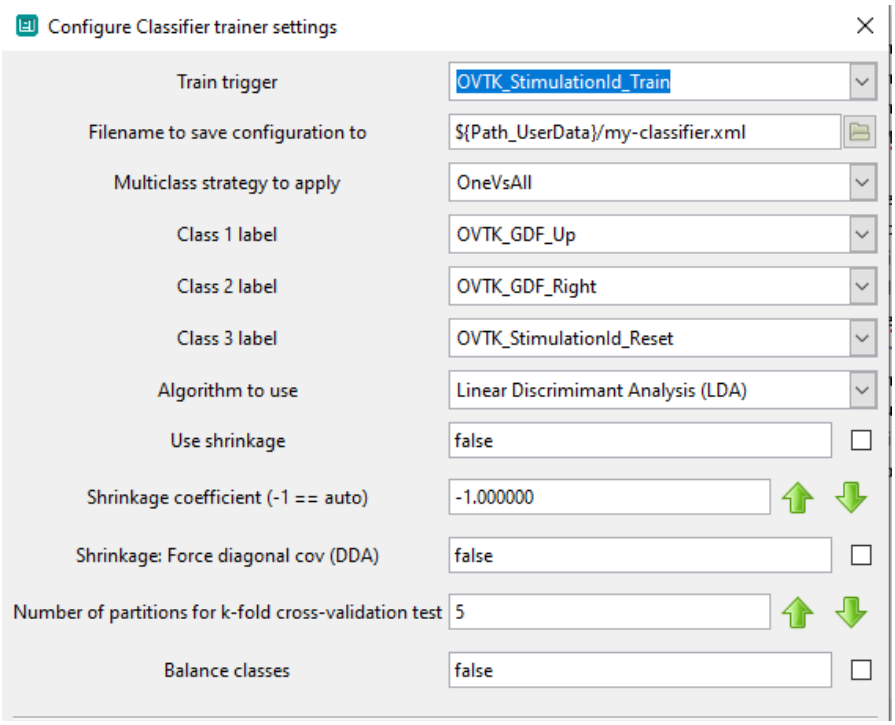

Figure 36: LDA Configuration for three classes

The classifiers are implemented in OpenVibe with the Classifier Trainer box where the feature vectors for each class along with the corresponding labels are fed as inputs. From the configuration window of the box the user can choose the type of desired algorithm and configure its hyperparameters. When the training is completed the box saves the trained classifiers into a single configuration file which can be used for online classification and outputs a stimulation code indicating that the training is finished.

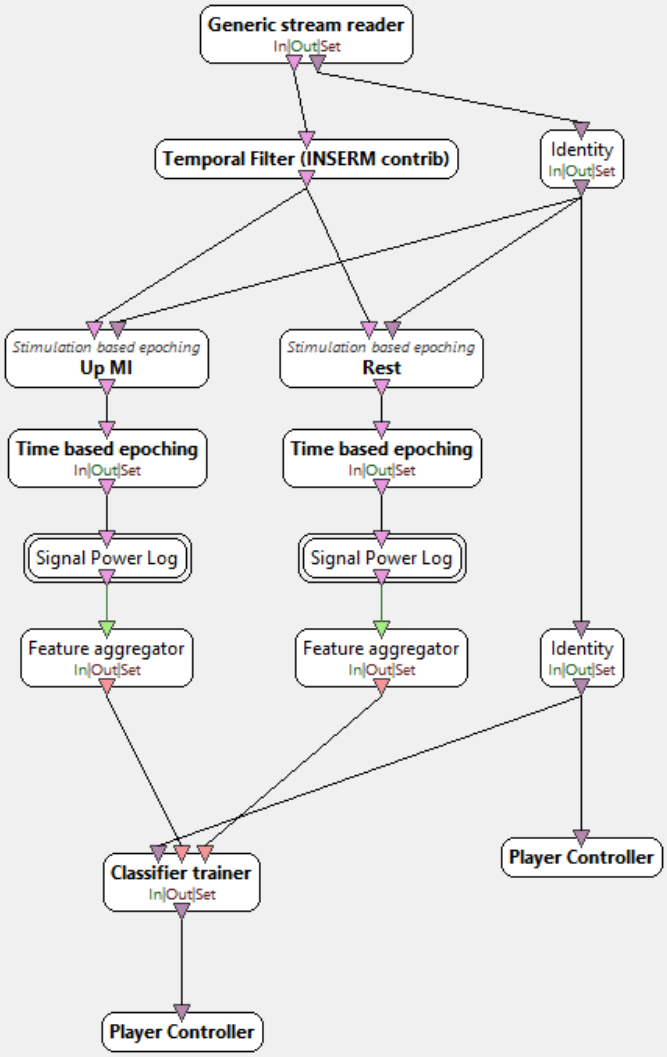

Figure 37: Final OpenVibe scenario for two classes

4.6.2 Deep Learning

For our own implementation of the classification stage we developed a WGAN for data augmentation and a CNN for classification, in the python framework PyTorch. Before feeding the feature vectors into our neural networks for processing, some data preparation must be done. First, at the end of our processing scenario we removed the classifier box and instead the outputs of the feature aggregators boxes are connected to a writer box that writes the feature vectors to one csv file per class.

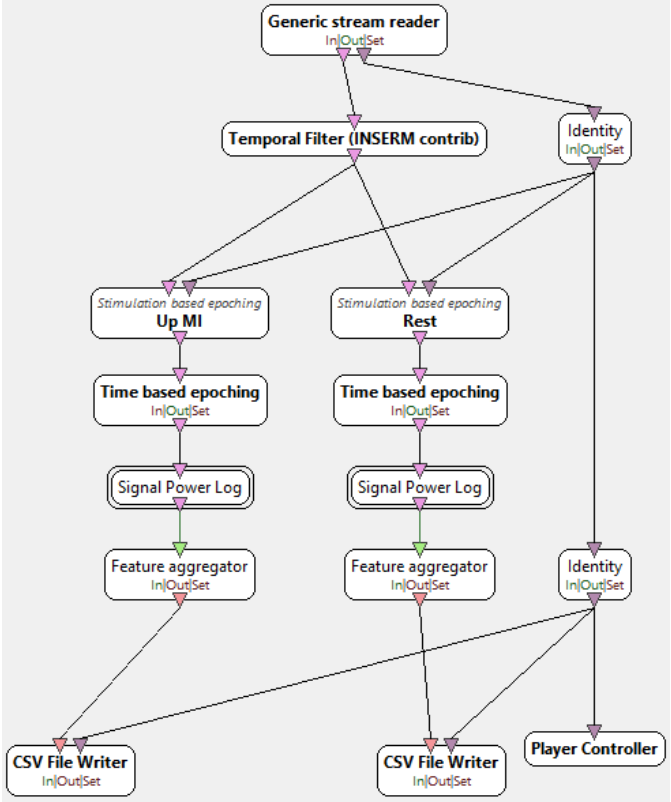

Figure 38: Final OpenVibe scenario for two classes (For Neural Networks)

Feature file								
Start Time	End Time	Feature 1	Feature 2	...	Feature 8	Event ID	Event date	Event Duration
Timestamp 1	Timestamp 1 + 1 second	SPL value 1	SPL value 2	...	SPL value 8	Stimulation code 1	Event timestamp 1	duration 1
Timestamp 2	Timestamp 2 + 1 second	SPL value 1-2	SPL value 2-2	...	SPL value 8-2	Stimulation code 2	Event timestamp 2	duration 2
⋮	⋮	⋮	⋮	...	⋮	⋮	⋮	⋮
Timestamp 980	Timestamp 980 + 1 second	SPL value 1-980	SPL value 2-980	...	SPL value 8-980	Stimulation code 980	Event timestamp 980	duration 980

Figure 39: Structure of recorded EEG feature files

Then the csv files are pipelined through a python script that clears redundant information from the file, removes all timestamps, embeds a column with our own internal label and aggregate all features into a single feature file. The final dataset contains 49 SPL features with 8 feature channels for 20 MI trials for each class, resulting in a feature vector with a total shape $[\#classes \times 980, 8]$. The way we handle labels in our implementation is that we assign each class with an integer for its label. The rest class is assigned to 0, the feet movement class to 1, right and left hand class to 2 and 3, respectively. This was done to simplify label parsing in our scripts. The final feature file is now ready to be fed into the WGAN.

In this thesis we implemented a WGAN to avoid the vanishing gradients problem present in the original GAN and stabilize the training process. The goal of the WGAN is to learn the data distribution for each class from the feature file and augment the dataset to improve the classification accuracy of the CNN. The WGAN relies on two neural networks, as explained in chapter 2.4, where the critic is trying to distinguish between artificial and real EEG features and the generator is trying to produce realistic EEG features to increase the loss of the critic.

For the architecture of the WGAN, we developed a Convolutional neural network with fully connected layers, for both the generator and the critic. More specifically, the model for the generator consists of one input layer of noise sampled from a normal distribution, a fully connected layer which projects the noise to a feature vector and five convolutional transpose layers that take the feature vector and perform pointwise transpose convolutions with a one dimensional kernel of size 1x1, to generate an output of higher dimension to resemble actual EEG features. For each convolutional layer the ReLU was used as the activation function. The critic's model is based on three convolutional layers with a one dimensional kernel of size 1x5, combined with three max pooling layers of size 1x2 to decrease the dimensionality of the input feature vector and a fully connected layer as the output layer. The ReLU was also used as the activation function for the convolutional layers. The critic and generator were based on deep convolutional layers because they generate less parameters compared to just fully connected layers resulting in a lightweight and fast architecture while also providing more realistic EEG samples. Below a more detailed diagram of the architecture is given.

Layer	**Filters**	**Kernel size**	**Stride**	**Output**	**Activation function**
Input	-	-	-	[1, 100]	-
FC Layer 1	6272	-	-	[1, 6272]	-
Reshape	-	-	-	[128, 49]	-
ConvTranspose 1	128	[1, 1]	1	[128, 49]	ReLU
ConvTranspose 2	128	[1, 1]	1	[128, 49]	ReLU
ConvTranspose 3	128	[1, 1]	1	[128, 49]	ReLU
ConvTranspose 4	128	[1, 1]	1	[128, 49]	ReLU
ConvTranspose 5	8	[1, 1]	1	[8, 49]	-

Table 1: Generator

Layer	Filters	Kernel size	Stride	Output	Activation function
Input	-	-	-	[8, 49]	-
Conv 1	32	[1, 5]	1	[32, 45]	ReLU
MaxPool 1	32	[1, 2]	1	[32, 22]	-
Conv 2	32	[1, 5]	1	[32, 18]	ReLU
MaxPool 2	32	[1, 2]	1	[32, 9]	-
Conv 3	32	[1, 5]	1	[32, 5]	ReLU
MaxPool 3	32	[1, 2]	1	[32, 2]	-
Flatten	-	-	-	[1, 64]	-
FC Layer 1	1	-	-	[1, 1]	-

Table 2: Critic

We used the Adam algorithm as the optimizer, a learning rate of $5e-5$, a weight clip factor of 0.01 and a batch size of 4, since our EEG dataset is small. The generator was trained once for every 5 training iterations of the critic. The WGAN was trained initially for 2000 epochs to observe the loss curve of the critic and generator. We set the training epochs to 1000 as the network performed the best around that mark. A different WGAN was trained for each class resulting into N WGAN models for N classes. The training process of the WGAN is described in the following pseudo code.

Algorithm 1 Algorithm for WGAN training

Require: clipping parameter *clip*, batch size *bs*, number of iterations of the discriminator per generator iteration *n_critic*

Initialization θ_c *critic's paramaters*, θ_g *generator's parameters*, $clip = 0.01$, $bs = 4$, $n_{critic} = 5$

for $class = 1, 2, \ldots, \#num\_classes$ **do**
  for $epoch = 1, 2, \ldots, 1000$ **do**
    for all $\{x^{(i)}\}_{i=1}^{bs} \sim p(x)$ **do**
      Sample noise $\{z^{(i)}\}_{i=1}^{bs} \sim p(z)$ from normal distribution
      Updating the discriminator by ascending its stochastic gradient $\bigtriangledown_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\mathbf{E}[Critic(Generator(z)] - \mathbf{E}[Critic(x)]]$
      Clip parameters $\theta_d \longleftarrow clip(\theta_d, -c, c)$
      if Critic trained n_{critic} iterations **then**
        Sample noise $\{z^{(i)}\}_{i=1}^{bs} \sim p(z)$ from normal distribution
        Updating the generator by ascending its stochastic gradient $\bigtriangledown_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -\mathbf{E}[Critic(Generator(z))]$
      end if
    end for
  end for
  Save generator for current class
end for

With the trained WGANs we augmented our EEG feature dataset to double the size of the original resulting in 40 trial features per class from the original 20. We also tested to increase the original dataset by 50% and 200% but this offered poor classification results compared to a 100% increase.

For the classification of the EEG features we designed a Convolutional Neural Network. The CNN model consists of three convolutional layers with a 1 dimensional kernel of size 1x3 and the ReLU for the activation function. For every convolutional layer, a batch normalization layer was used to stabilize training and a max pooling layer of size 1x2 was appended to reduce the dimensionality of the features. Then the output is flattened and passed to two fully connected layers,

with one Dropout layer appended to the first fully connected layer with a dropout rate of 0.5 to add regularization to the model and reduce overfitting. The fully connected layers will decode the convolved features to produce one logits vector of #classes size. The input dataset has a shape of $[\#classes \times 40, 49, 8]$. The architecture of the CNN is displayed below.

Layer	**Filters**	**Kernel size**	**Stride**	**Output**	**Activation function**
Input	-	-	-	[8, 49]	-
Conv 1	64	[1, 3]	1	[64, 47]	-
BatchNorm 1	64	-	-	[64, 47]	ReLU
MaxPool 1	64	[1, 2]	1	[64, 23]	-
Conv 2	64	[1, 3]	1	[64, 21]	-
BatchNorm 2	64	-	-	[64, 21]	ReLU
MaxPool 2	64	[1, 2]	1	[64, 10]	-
Conv 3	64	[1, 3]	1	[64, 8]	-
BatchNorm 3	64	-	-	[64, 8]	ReLU
MaxPool 3	64	[1, 2]	1	[64, 4]	-
Flatten	-	-	-	[1, 256]	-
FC Layer 1	32	-	-	[1, 32]	ReLU
Dropout	-	-	-	[1, 32]	-
FC Layer 2	2	-	-	[1, 2]	-

Table 3: CNN architecture

For the training of the CNN we used the Adam optimizer with a learning rate of $5e-5$ and a batch size of 16. We also added L2 regularization to the model with a weight decay of $1e-5$ to reduce overfitting. The categorical cross entropy was chosen as the loss function since we have up to 4 classes. Note that in the output layer of the network we did not add a softmax layer to produce a probability distribution of the predicted classes as it is handled internally in the loss function [70]. The logits vector from the final fully connected layer is then passed as the predicted class logits vector in the loss function along with the target class label. The network was first trained with the k-fold strategy to obtain a robust estimate of the network's performance, and then trained on the whole dataset to produce the final parameters of the network. In the case of the augmented dataset, the dataset was shuffled to create a random data distribution to stabilize training; and then separated into a training set and a validation set with an analogy of 70/30 of the original, where no artificial features were present in the validation set. Finally, the CNN was trained for 500 epochs. The pseudo code for the training process is given below along with the loss curve of the training.

Algorithm 2 Algorithm for CNN training

Require: learning rate $lr = 5e - 5$, batch size 16.

Initialization CNN Network net, Adam optimizer $optimizer$, Cross Entropy Loss $criterion$

for $epoch = 1, 2, \ldots, 500$ **do**
  for all $(x, label) \in (X_{train}, Y_{train})$ **do**
    $Y_{pred} = net(x)$
    $loss = criterion(Y_{pred}, label)$
    Ascend gradient of $criterion$
    Optimizer step
  end for
end for

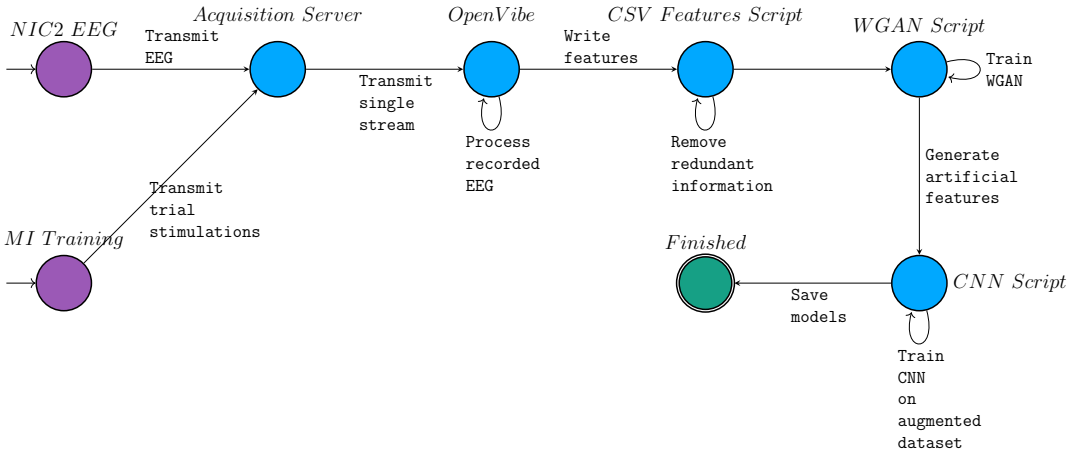

Figure 40: Offline training

4.6.3 Online Classification

At this stage of the BCI the neural networks are already trained on the recorded dataset from the offline scenario. To be able to receive online EEG recordings from our headset and classify them in real time for the VR game we designed an online acquisition scenario in OpenVibe.

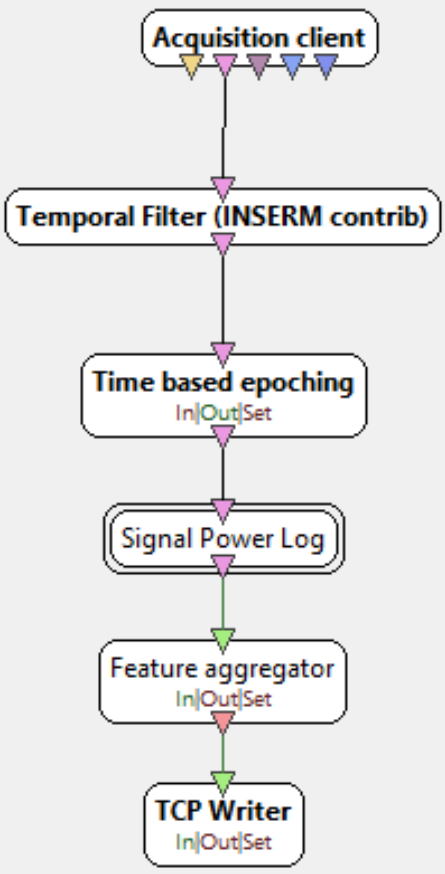

Figure 41: Online acquisition scenario

This scenario receives the raw EEG signal from the Acquisition Server and processes it with the same steps applied as in the offline scenario without the stimulation based epoching box. The TCP writer box acts as a TCP server where a client can connect to receive the data in real time. To receive the EEG data we implemented a simple TCP client in python that saves the EEG features until we have enough features to input to our CNN. The TCP client keeps receiving EEG features from OpenVibe, classifies them and the prediction is sent to the VR game via another TCP socket until the user exits the game.

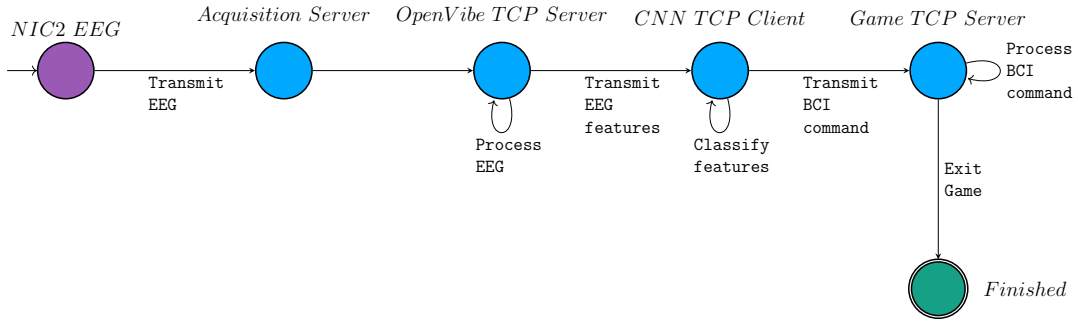

Figure 42: Online classification

5 VR Game Implementation

5.1 Introduction

The VR game we designed to integrate with the BCI, is a simple maze game with a medieval theme. The goal of the game is to navigate through the maze by controlling the movement of the character with BCI commands and find the treasure chest to reach the end of the game. The user can also interact with in-game props and User Interface (UI) menus using BCI commands.

In this chapter of the thesis, the technical implementation of the VR game will be analysed, with all the scripts that define the game's behavior. Screenshots of each scene in the game will be presented.

The VR training scene for the BCI will be explained along with the method that transmits stimulation codes to OpenVibe for the MI trials. Each level of the VR game will be explained along with the differentiations in each level in movement, interactions with in-game props and User Interface menus. Furthermore, a detailed description on the integration of the VR game with the BCI system will be offered.

This chapter will conclude with a list of 3rd Party Assets that were used for the game, downloaded from the Unity Asset Store.

5.2 Start Menu

The Start Menu Scene is the first scene the user encounters. The user is in a room in front of a table with two VR buttons the player can select, the start game button and the training button. In order to press one button, the user needs to make the index finger point motion on the VR controller and press the button with his index finger. The user can select the training button to enter the training scene where he can perform the MI training of the BCI. To press the start button, the user needs to have completed a training session at least once, where the game will wait until the trained neural network is loaded and will load automatically the level corresponding to the number of classes the network was trained with.

Figure 43: Start Menu

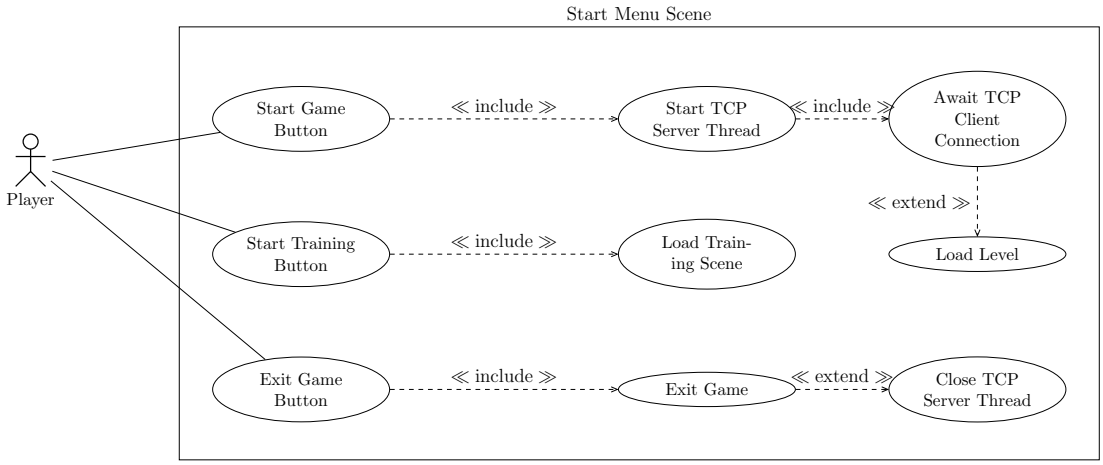

Figure 44: Use Case Diagram of Start Menu

The Start Scene and the Training Scene of the game is the only scenes where the user can interact with the VR controllers.

5.3 Training Scene

In this scene of the VR game, the user performs the training of the BCI for the MI paradigm. The training procedure that was used, is the same as explained in chapter 4.2 and it was implemented using OpenVibe's example of the Graz Motor Imagery Visualization. In this example, for every section of a MI trial, an appropriate stimulation code is sent to OpenVibe to indicate the timestamp for the start of the section. In our training procedure we have four sections, the section where an audio beep is played; the section where the visual cue is displayed; the feedback section; and the rest section. Stimulation codes for each section is sent to OpenVibe's Acquisition Server via a TCP socket.

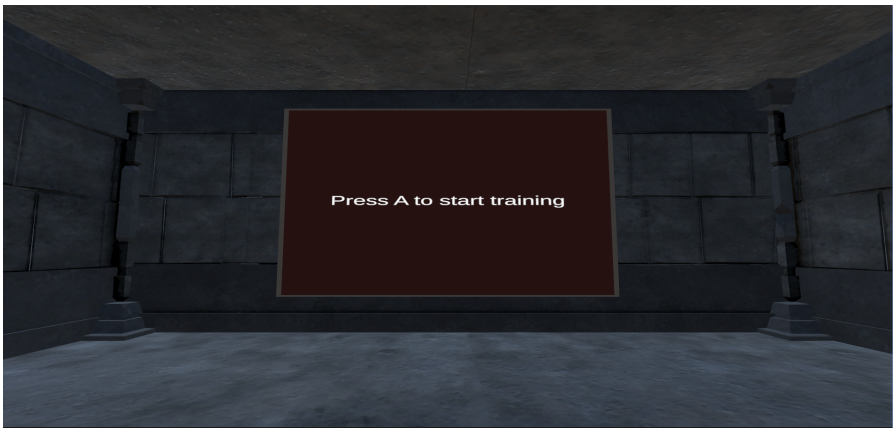

Figure 45: VR training scene

In Figure 45, a visual representation of the training scene is depicted, where the user does not move and concentrates on the central part of the screen. A randomized sequence of visual cues indicating the class movement, is displayed, according to the training procedure. Each visual cue amounts to a trial and the user is trained for 20 trials per class. For the color palette of the training scene we chose dim colors to limit the eye exhaustion factor and avoid any training problems such as rehearsal of wrong body limb movement. The training scene can be configured via a JSON file where the user can tune the hyper parameters of the MI training such as the number of classes; the number of trials per class; the base rest window, beep, cue and feedback time windows. After the MI training is finished, the user is instructed to wait unit the completion of OpenVibe's processing scenario and our neural networks training.

5.4 BCI Integration

In this subsection the integration of our VR game with the BCI system will be analyzed. The interaction of our training scene with OpenVibe for stimulation transmission of trials, was discussed in the previous chapter. For the rest of the game the movement of the character, interaction with props and UI control relies on a TCP server that is running in the background in a separate thread from the main game. This TCP server will listen for connections from the TCP client that receives EEG data from OpenVibe.

When the client connects to the server, it sends a header first that contains the number of classes the CNN is trained with. This header is used to load the level corresponding to the number of classes. All subsequent packets that are received from the client contain the prediction of the CNN. The available predictions the CNN sends are the $FORWRAD$ prediction for feet movement, $RIGHT$ prediction for right hand movement, $LEFT$ prediction for left hand movement and $REST$ prediction for the relaxed state. These predictions are treated as the BCI commands for the game. A new prediction packet is sent every four seconds since we need four seconds of EEG signal for feature extraction. The predictions are used to control the behavior of player movement, prop interactions and UI. When the TCP server receives a new prediction it sends a message with the prediction to all objects of the game that require a BCI command to operate such as the player game object, props and UI menus. The prediction is then handled appropriately by each object.

All the game mechanics such as loading a level and controlling movement, that rely on a BCI command from the game's TCP server, need to be handled in Unity's main thread, for them to work properly. Since our TCP server is operating in a background thread, we need to sent BCI commands to the game's objects in a thread-safe way to avoid any race conditions on the priority in which the BCI commands are executed by the game. For this reason, we utilized mutexes, where we lock the mutex during the acquisition of a BCI command and schedule the messages that need to be sent to the game's objects, in a Queue. Then in Unity's main thread when a new frame is displayed, the messages in the Queue is extracted one by one and are sent to all appropriate game objects. The queued messages are in a form of a custom function call that is implemented in all game objects and is the function responsible for receiving the BCI command and handle it according to the object's properties. This procedure is repeated until the user exits the game.

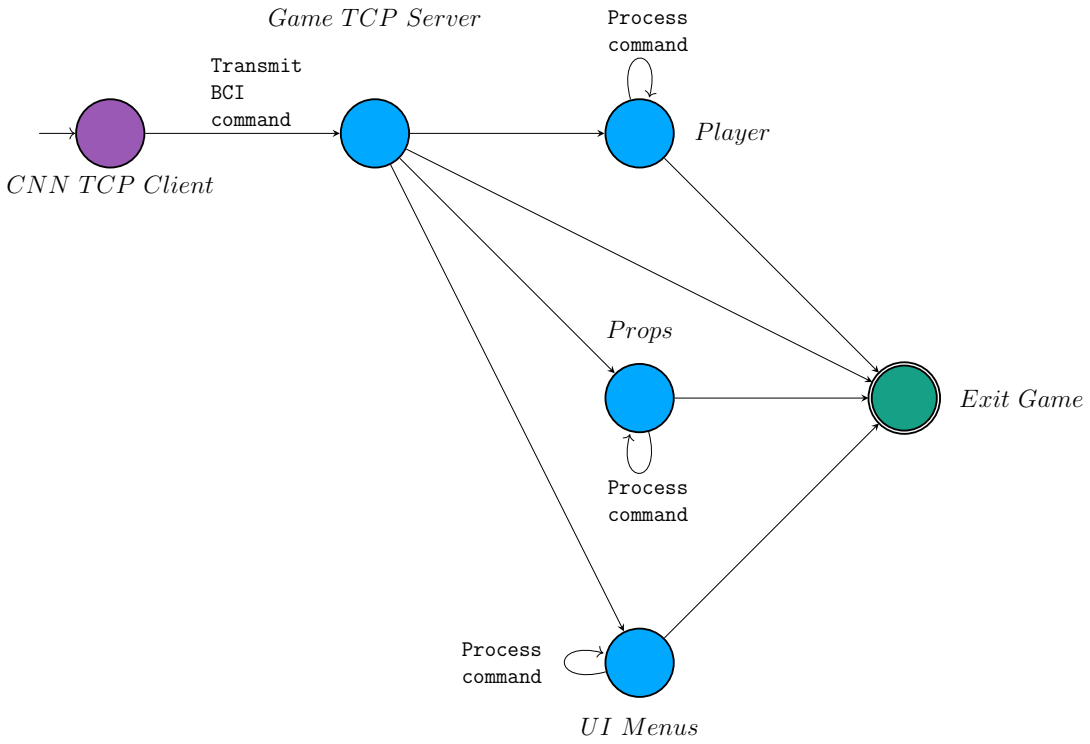

Figure 46: Transmission of BCI command to game objects

5.5 Movement

In this subsection, we will describe how the movement mechanic of the game is implemented. We will also analyze the differences between levels when changing the number of classes, and thus, the number of available BCI commands.

The player is able to navigate the character in the maze by trying to invoke BCI commands depending on the action the player wants to take, as it was described in the previous chapter. The number of actions that the user is able to perform is depended on the number of classes the BCI was trained with. The player object receives a BCI command every two seconds from the game's TCP server in the form of an asynchronous function call. This function call will check the received BCI command and will invoke a movement animation based on the command.

Due to the periodic nature of BCI command transmission, the movement animations are designed to move the player in continuous, smooth movement to a certain distance, by invoking a single command, instead of continuously pressing a keyboard key for continuous movement that is found in conventional computer games. This smooth animation was achieved by applying a linear interpolation filter on the movement that uses the current time and duration of the movement as its parameters. More specifically, in the case of forward movement, the player moves forward for a distance of 5 unit meters. The animation starts aggressively on the start of movement and smooths out during the end. The linear interpolation filter described by the following algorithm.

Algorithm 3 Linear Interpolated Forward Movement Algorithm

Require: forward offset in unit meters $offset$, duration in seconds $duration$, time passed from Unity's last frame $interval$.

Initialization $offset = 5,\ duration = 1,\ time = 0,\ FinalPos = CurrentPos + CurrentPos.forward \cdot offset.$

1: **while** $time < duration$ **do**
2: $t = \frac{time}{duration}$
3: $t = t^2 \cdot (3 - 2t)$ ▷ t describes the interpolation point
4: $CurrentPos = CurrentPos + (FinalPos - CurrentPos) \cdot t$
5: $time = time + interval$
6: **end while**
7: $CurrentPos = FinalPos$ ▷ This ensures that we are exactly on $FinalPos$

The same interpolation algorithm is applied on all other movements.

When the player is navigating in the maze he is considered to be in a playing state. While the player is in the playing state BCI commands that are sent to other game objects such as props and UI menus, are not handled and are discarded.

5.5.1 Two Class level

In this level of the game the medieval maze is a straight corridor where the user can navigate the character through the maze to reach the end and find the treasure chest to finish the game. The player can invoke a BCI command by rehearsing an MI movement. The *FORWARD* command that is received by the server is invoked by rehearsing feet movement and enables the character to move in the forward direction by a distance of 5 unit meters. The *REST* command is invoked by relaxing and the character does not move during this state.

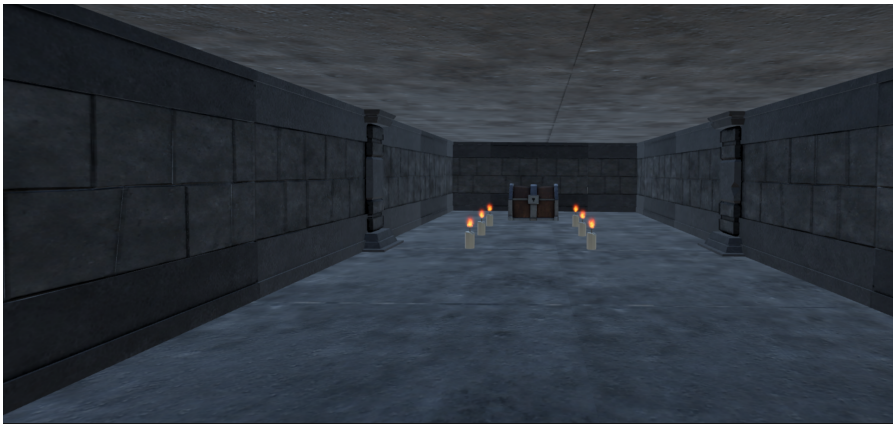

Figure 47: Level 1 of the VR game

5.5.2 Three Class level

In this level of the game, the player is able to invoke the *FORWARD* and *REST* command as in the previous level. The third command that is implemented is the *RIGHT* command and can be invoked by rehearsing right hand movement. The *RIGHT* command enables right rotation movement from the current axis of the character and rotates the character by an offset of 15 degrees to the right. By enabling right rotation the player is now able to rotate a full 360 degrees and go to any direction of the axis.

Since rotation of the character is enabled, the 3 class level of the maze introduces corners where the user can turn. More explicitly, the 3 class level is a closed loop maze, where the user turns right until a certain point of the loop is reached and the treasure chest is found.

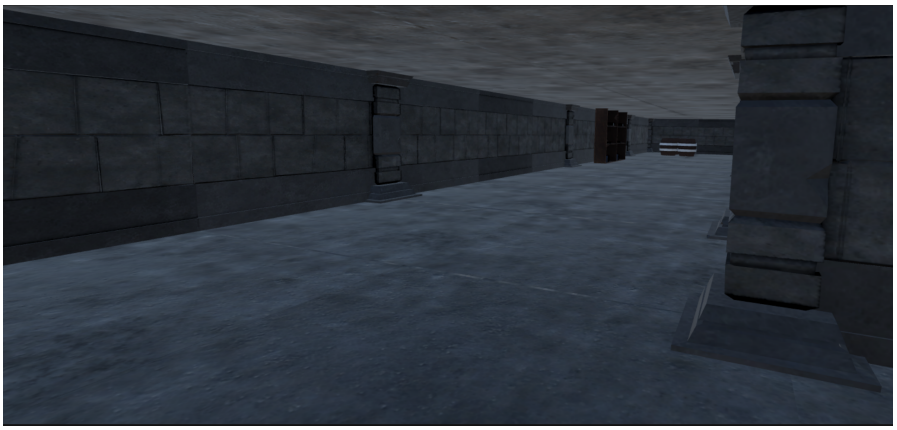

Figure 48: Level 2 of the VR game

5.5.3 Four Class level

The 4 class level of the game introduces the *LEFT* command where the user is rehearsing left hand movement, and enables left axis rotation by an offset of 15 degrees. By enabling left rotation, the control of movement is now more intuitive where all axis directions are easily accessible by the player.

For this level, the game now offers a complete maze environment with forward movement, right and left turns. The goal of this level is the same as previous levels where the player needs to find the treasure chest. A screenshot of the level is presented below.

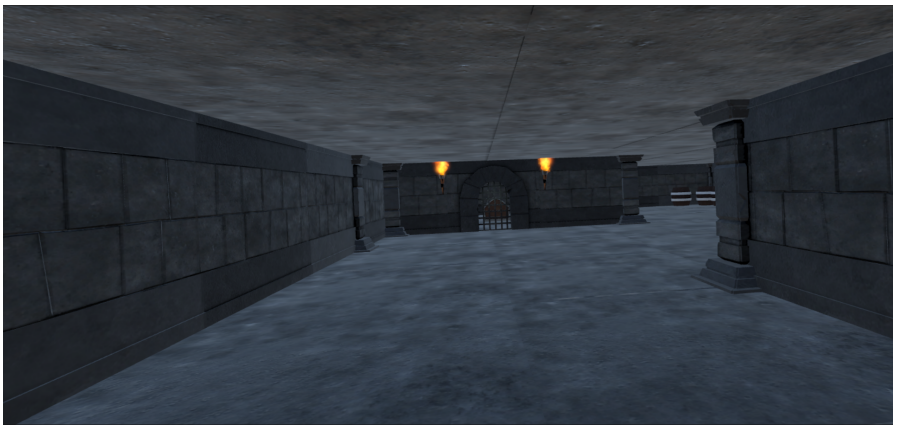

Figure 49: Level 3 of the VR game

5.6 Interaction with Props

In this subsection a description on the interaction of the player with in-game props and the impact it has on the game, will be offered.

In this game the player will encounter obstacles, when navigating in the maze. The player needs to interact with the obstacles in order to advance further into the game and finish it. There is only one type of obstacle in the game, a door which blocks the player's path. In order to open the door and advance further, the player needs to interact with the door by invoking a BCI command. The BCI command that was chosen for interactions with props, is the $FORWARD$ command as it is present in all classes, and thus, all levels.

When the player approaches the door and is in a certain distance from the door, character movement will cease and the player will be unable to move. A message window will then appear in the player's view to prompt him to invoke the $FORWARD$ command and open the door. The message window will stay for an interval of one second and then will disappear. When the player invokes the appropriate command an animation will be played of the door opening, and player movement will be enabled again.

This state, where the player is interacting with a door, is called interacting state. During this state all BCI commands that are sent to the player's game object for movement, and UI menus will not be handled and are discarded.

Figure 50: Door interaction

5.7 UI Menus

In this VR game the player can interact with User Interface (UI) menus. A UI menu appears at the end of a level, when the player has reached the treasure chest. The UI menu consists of an array of buttons where each button corresponds to an action. The number of buttons in each level is different as it depends on the number of classes. The currently selected button in the array of button will be highlighted in a gray color, while all the other inactive buttons will be highlighted in a white color. The behavior behind the player interaction with the button array will be analyzed in the following subsections.

5.7.1 Two Class level

In the 2 class level the UI menu only has one button since the only available command we have for selection is the $FORWARD$ command while the $REST$ command is used for the inactive state. Thus the button array consists of only one button where the player can invoke it by invoking the $FORWARD$ command. By invoking the button the player will exit the level and the game will be finished.

Figure 51: Level 1 UI

5.7.2 Three Class level

In the 3 class level the UI menu has two buttons for the player to select. One button will exit the level and the other button will restart the current level for the player to play again. The player by invoking the $FORWARD$ command will select the currently highlighted button. If the player invokes the $RIGHT$ command the menu will highlight the next button in the button array with a grey color and the previously selected button will be highlighted with a white color. Since the button array has two buttons in this level if the player has highlighted the last button

and invokes the *RIGHT* command again, the menu will highlight the first button of the area while the last button will become inactive.

Figure 52: Level 2 UI

5.7.3 Four Class level

The 4 class level also features a two button UI menu. As in the other menus the player can invoke the *FORWARD* command to select the current button, the *RIGHT* command to move to the next button and the *REST* command to do nothing. With 4 classes and the addition of the *LEFT* command the player can move to the previous button. When the player is in the last button and invokes the *RIGHT* command the currently highlighted button is the first button. When the player is in the first button and invokes the *LEFT* command the currently highlighted button is the last button in the array. In the 4 class level the player can have full control of a UI menu that consists of only buttons.

Figure 53: Level 3 UI

5.8 Unity Assets

All of the assets used in the game were downloaded from Unity's Asset store. More specifically, the *Simple Modular Dungeon* asset was used to design and decorate the medieval maze. The *Low-Poly Tables* asset was used for the table in the Start Menu, and the *VR Buttons and Levers* asset was used for the two VR buttons in the Start Menu.

6 Evaluation & Testing

6.1 Introduction

After the BCI and VR game was implemented, we evaluated the system on the accuracy it offered, its performance and how well the users could interact with the VR game. For the evaluation we tested both the BCI and VR game on two participants. For both participants, this experience was their first interaction with a BCI and VR gaming. We evaluated the BCI and VR game separately, where for the BCI we focused on the evaluation of accuracy and performance on the hardware we had at our disposal; whereas for the VR game we focused on the evaluation on general usability, and how well the users were able to interact with the game.

6.2 BCI Evaluation

For our BCI evaluation, we employed a series of metrics and validation methods to measure the accuracy of the system. More specifically, OpenVibe's machine learning classification method was evaluated with k-fold validation for 5 folds where the training and validation dataset are split into $\frac{4}{5}$ and $\frac{1}{5}$ chunks of the original to validate the accuracy and stability of the algorithm. Our WGAN was evaluated by comparing the generated features against the real features to observe any differences. Moreover, the Mean Squared Error was calculated to estimate the deviation in amplitude between the two distributions. Finally, the CNN was evaluated on both the recorded and augmented dataset.

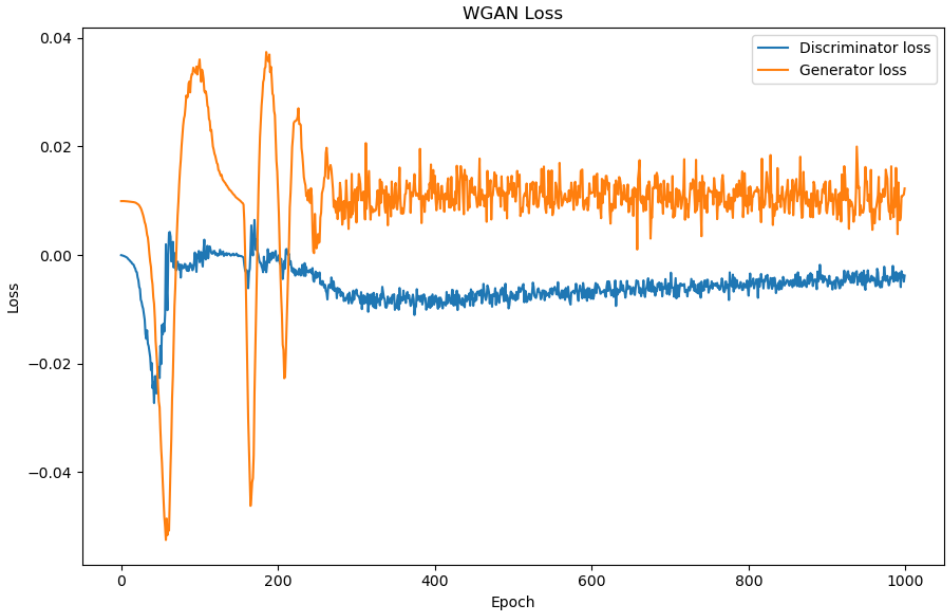

Figure 54: WGAN Loss for Rest MI trial

Training our WGAN was proven to be challenging. Figure 54 displays the loss curve of both the discriminator loss and the generator loss. We can observe from the loss curve that the generator and discriminator converges early and there is minimal change in the loss at later epochs. Even though the model converges early, it was observed that the generated features at later epochs, had better overlap in the SPL values compared to real features.

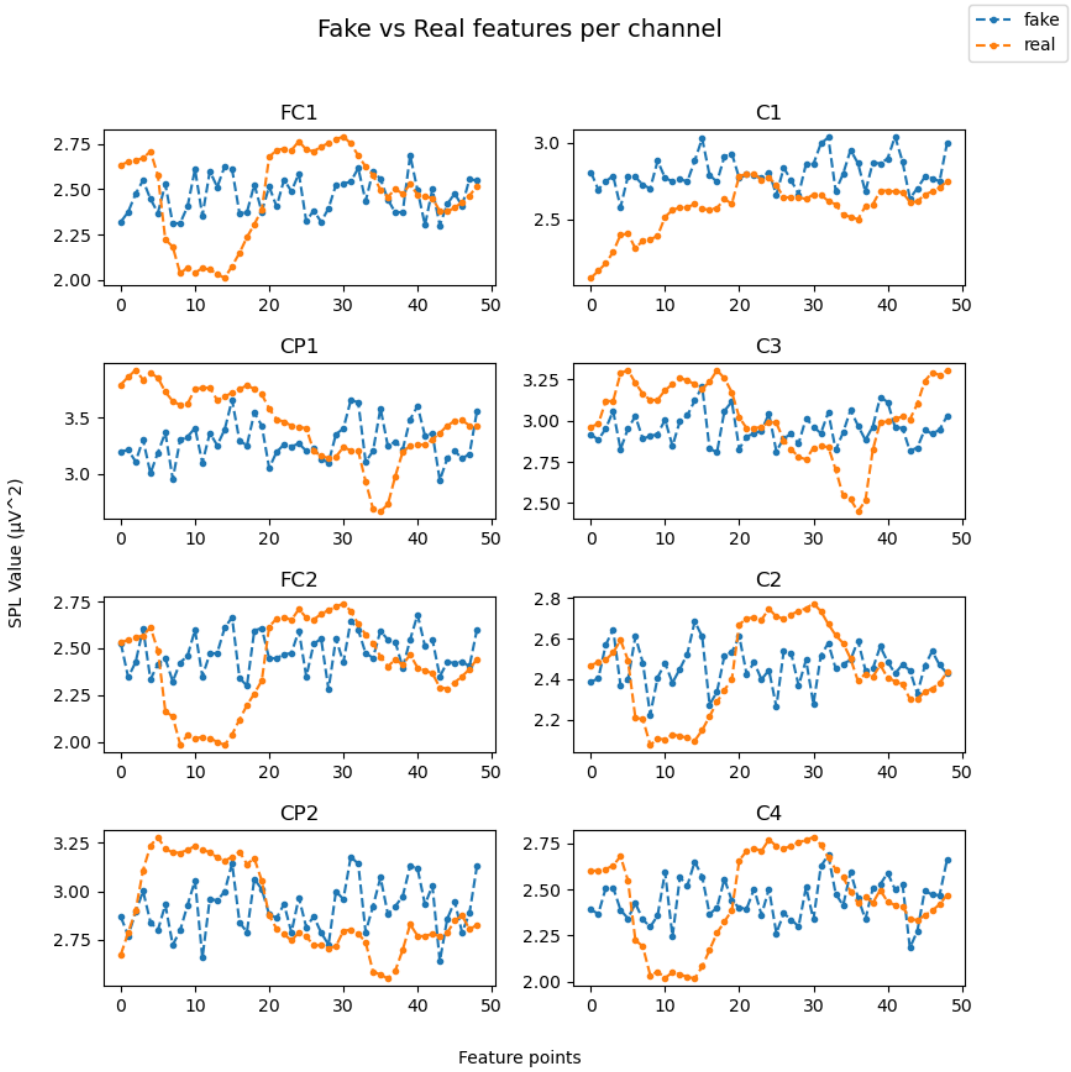

Figure 55: Generated vs Real Features for Rest MI trial

Figure 55 shows a comparison of the artificial SPL features and a sample from the real features for all 8 feature channels for the Rest MI trial. To further validate the quality of the artificial features we calculated the Mean Squared Error (MSE) for all 8 channels and averaged the results for a total estimate.

$$eg. \quad MSE = 0.08, \quad Rest\ trial\ (Fig.\ 55)$$

By observing Figure 55 and given the low MSE, we can deduce that the SPL values of the artificial features are close to the real features but the WGAN fails

to learn the SPL features distribution correctly. This can be attributed to bad feature samples present in our EEG dataset that could impact training or the architecture of the WGAN where the critic does not provide a meaningful loss for the generator to learn the distribution of features.

Our implementation of CNN was validated using the stratified k-fold validation method. Furthermore a confusion matrix was calculated with the predicted and actual labels to ensure that our CNN is able to learn from the data distribution and not randomly predict the data. The AUC score of the ROC curve and F1 score were calculated, using the scikit-learn library, to further validate the accuracy of the network. A histogram of the loss values as well as a plot of the training and validation loss function are calculated to ensure that no overfitting is present in our network. Finally, a comparison between the accuracy of OpenVibe's machine learning algorithms and our deep neural networks is offered.

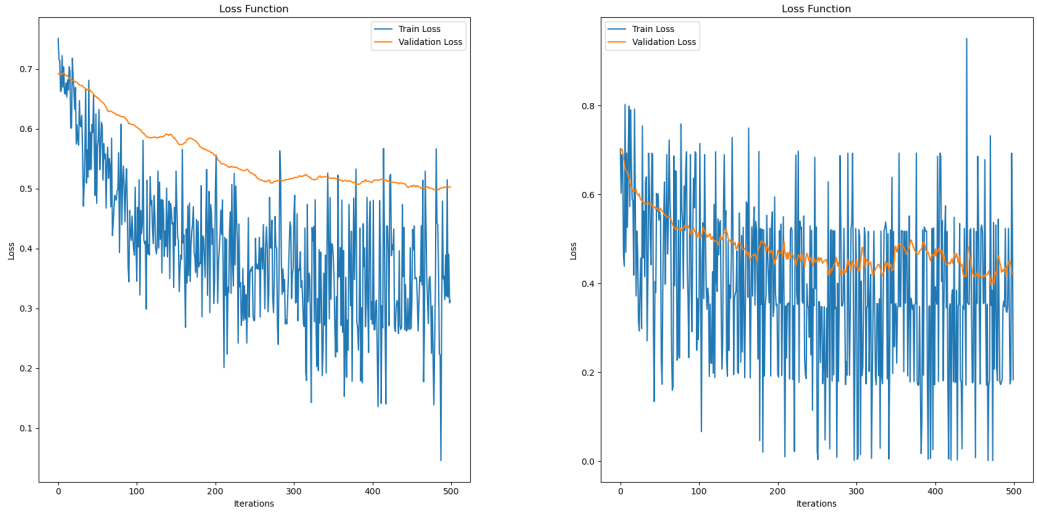

Figure 56: Training/Validation Loss on recorded(left) and augmented(right) dataset of P2 for two classes

From Figure 56, we observe that out neural network presents a small amount of overfitting on the recorded EEG dataset. However, data augmentation has improved our model and displays close to zero overfitting. Furthermore, the training loss curve on the augmented dataset becomes noisier due to the noisier artificial data present in the training samples.

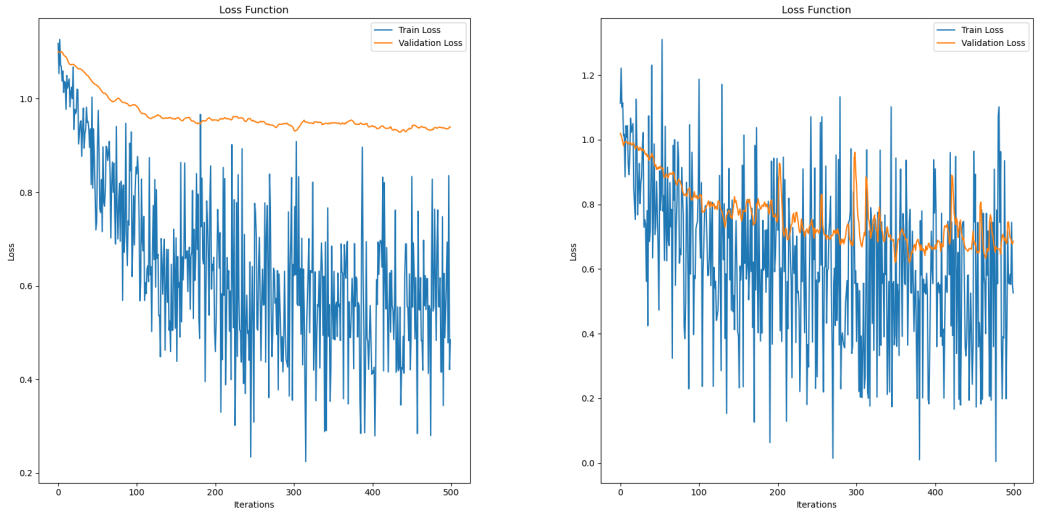

Figure 57: Loss on recorded(left) and augmented(right) dataset of P2 for three classes

The same trend is present for three classes, where the augmented dataset improves the fitting of our neural network.

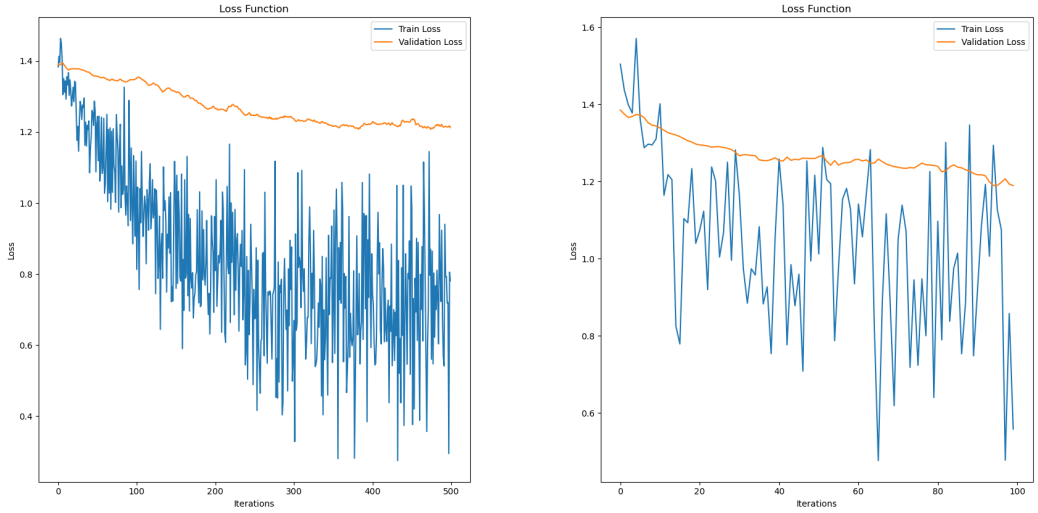

Figure 58: Loss on recorded(left) and augmented(right) dataset of P1 for four classes

For four classes, the model was always overfitting on the augmented dataset, thus, early stopping was used to stop training around the 100 epoch mark for a better model.

The BCI system was tested with two male subjects in the age of 22 and 23. A single MI training for each subject was performed for four classes with 20 trials per class. The BCI was also evaluated on a dataset provided by the research team that developed OpenVibe, Inria, which features left and right hand motor imagery with 20 trial features each, recorded with a 512hz sampling rate from a 11 channel EEG Headset. Our implemented CNN was evaluated on both the recorded and the augmented dataset, generated from the WGAN. The stratified k-fold validation method was employed to avoid the class imbalance problem and provide a more accurate evaluation of the network. Furthermore, a confusion matrix was calculated to provide a visual representation of the predicted label distribution compared to the target labels. The AUC and F1 score were also calculated to measure the probability of true positives/negatives and the harmonic mean of precision and recall of the confusion matrix. The AUC score is calculated as the area under the roc curve via integral calculus. Whereas the F1 score is calculated with the following equation.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

Where TP is the True Positives, FP is the False Positives and FN the False Negatives. From (4),(5) we have:

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

The accuracy between the different methods that were used to classify the EEG signal is compared in the following tables.

Accuracy				
Participants	**LDA**	**SVM**	**CNN**	**WGAN+CNN**
P1	62.65%	61.38%	75%	**77.5%**
P2	53.03%	52.11%	71.25%	**72.5%**
Inria	66.32%	65.36%	**86.43%**	81.82%
Mean	60.66% (+/- 5.6%)	59.61% (+/- 5.55%)	**77.56% (+/- 6.45%)**	77.27% (+/- 3.8%)

Table 4: Mean Accuracy of classification methods for two classes.

For two classes, there is a significant increase in accuracy when using a convolutional neural network for classification compared to traditional machine learning

algorithms. By augmenting the recorded EEG dataset the CNN is able to increase the accuracy by an additional $1.25 - 1.5\%$. However, there is a decrease in accuracy when classifying the augmented dataset from Inria. Since the neural network and OpenVibe's classifiers performed well on the original dataset from Inria, the augmentation with artificial data resulted in noisier EEG features and made the classification harder for our neural network. Thus, the augmented dataset resulted in worse classification performance than the original.

Accuracy				
Participants	**LDA**	**SVM**	**CNN**	**WGAN+CNN**
P1	39.81%	34.30%	**46%**	45.55%
P2	36.87%	37.20%	48.33%	**49.44%**
Mean	38.34% (+/- 3.94%)	35.75% (+/- 2.38%)	47.16% (+/- 1.16%)	**47.49% (+/- 1.94)**

Table 5: Mean Accuracy of classification methods for three classes.

For three classes, our neural network performed better than OpenVibe's classifiers with an increase of about $8 - 10\%$. By augmenting the recorded dataset, there is a decrease in accuracy for the first participant, while there is an increase of around 1% for the second participant.

Accuracy				
Participants	**LDA**	**SVM**	**CNN**	**WGAN+CNN**
P1	30.70%	26.70%	**39%**	38.75%
P2	27.87%	27.89%	36.27%	**37.34%**
Mean	29.28% (+/- 2.50%)	27.29% (+/- 2.43%)	37.63% (+/- 1.36%)	**38.04% (+/- 0.7%)**

Table 6: Mean Accuracy of classification methods for four classes.

There is also an increase in accuracy, for four classes, when using a CNN compared to LDA and SVM. When augmenting the original dataset, our CNN achieves an increase of 1% in accuracy for the second participant while there is a decrease for the first participant. Even with the early stopping technique that was used to train the neural network for four classes, the model was proven to present instability in training and the results should be considered optimistic.

AUC score		
Participants	**CNN**	**WGAN+CNN**
P1	0.75	0.77
P2	0.75	0.73
Inria	0.88	0.83

Table 7: AUC score for two classes.

F1 score		
Participants	**CNN**	**WGAN+CNN**
P1	0.77	0.78
P2	0.71	0.8
Inria	0.86	0.8

Table 8: F1 score for two classes.

6.3 VR Game Evaluation

For evaluating our VR Game, the players were asked to rate each scene they experienced in the game, on the Likert-5 scale, based on categories that focus on user experience, general usability of the system and performance of the game. More specifically, the categories are as followed:

- Training Scene
  - Were you accurate in MI training
  - Were the time intervals adequate to complete every trial stage
  - Was the environment tiring
  - Was the training procedure clear
- Level 1, Level 2, Level 3 Scenes
  - Was the control of the movement accurate
  - Were the commands difficult to invoke
  - Was the interaction with props enjoyable
  - Was the control of the UI accurate
  - Overall, was the level enjoyable

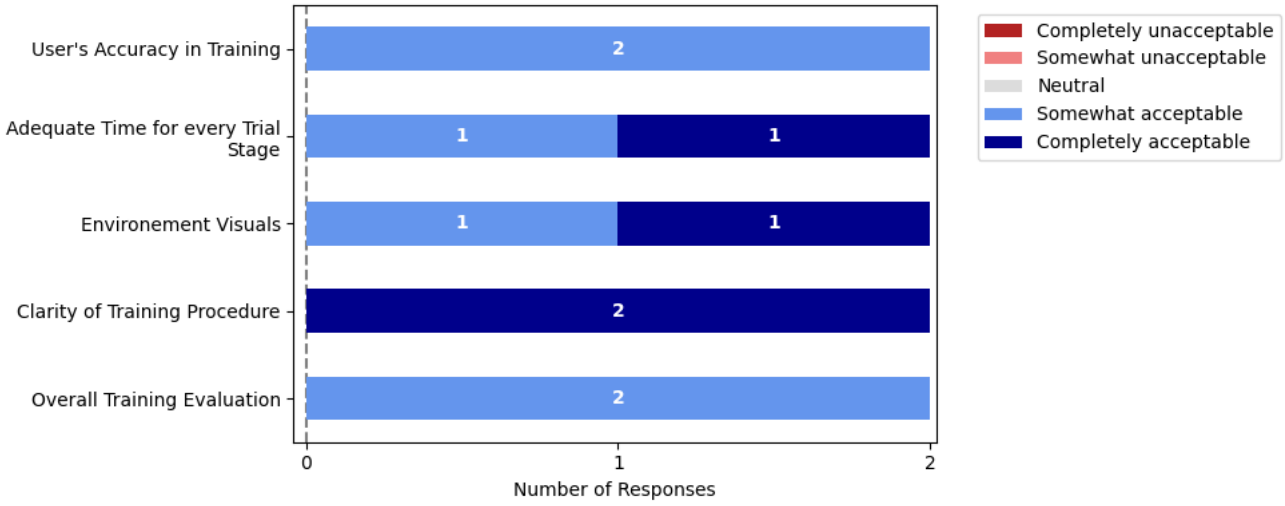

Figure 59: Training Scene Evaluation

Overall, the participants were satisfied with the training scene of the game. The training procedure was understood by both participants and there was adequate

time to complete every stage of a MI trial. Furthermore, the training environment was not found tiring although one participant noted that the training procedure was quite long.

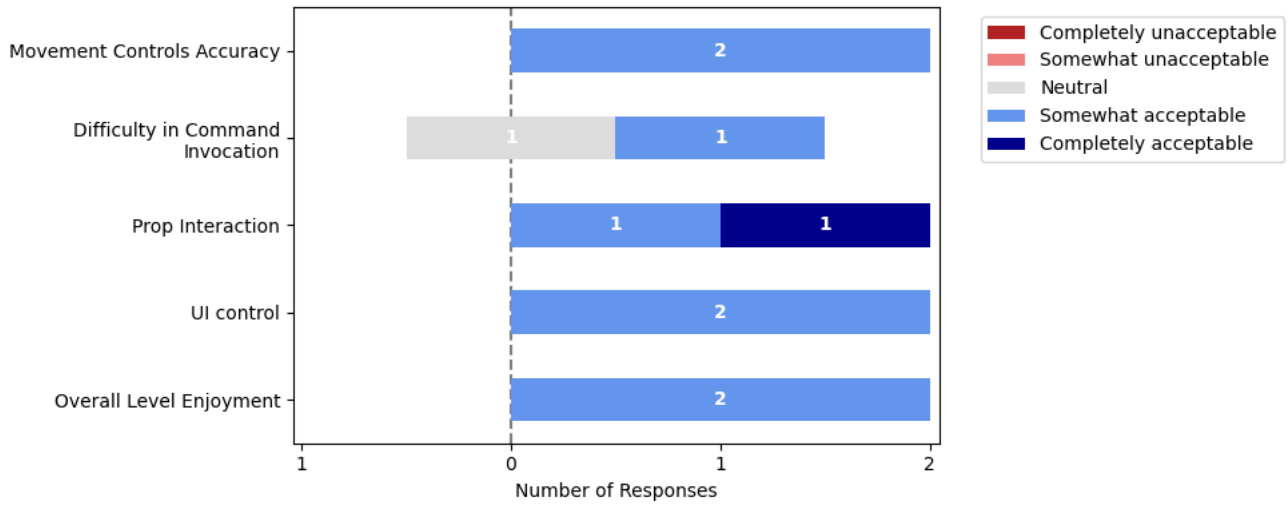

Figure 60: Level 1 Evaluation

The first level of the game, where two BCI commands were available, worked really well and was found quite enjoyable by the participants. Both participants were accurate with the movement control except from two instances where the character moved without the participant thinking of moving forward. Interaction with props was accurate and was regarded by the participants as a nice addition to the game's mechanics.

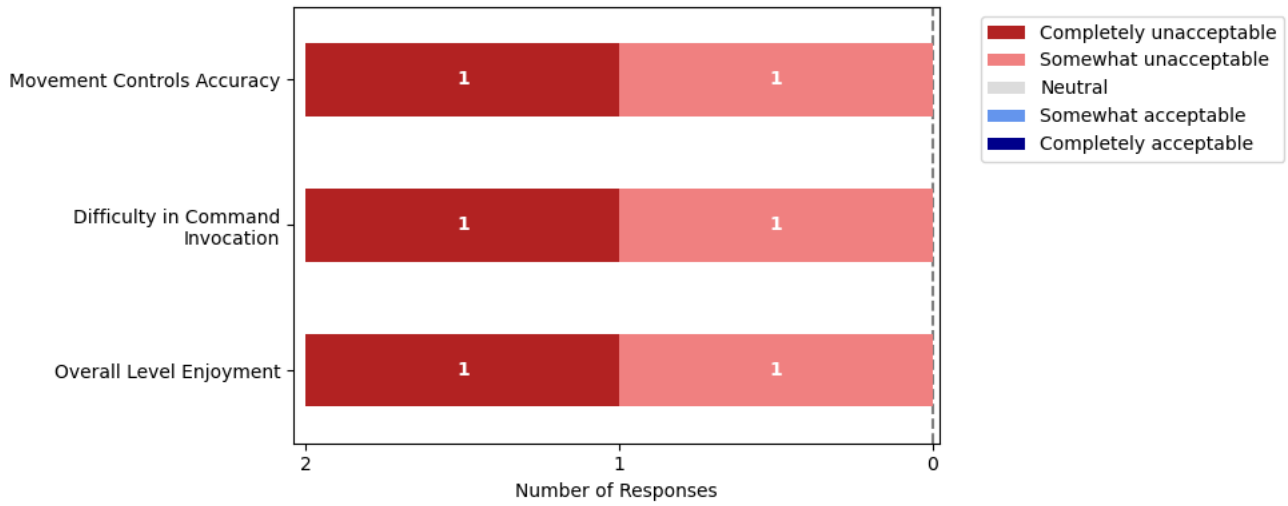

Figure 61: Level 2 Evaluation

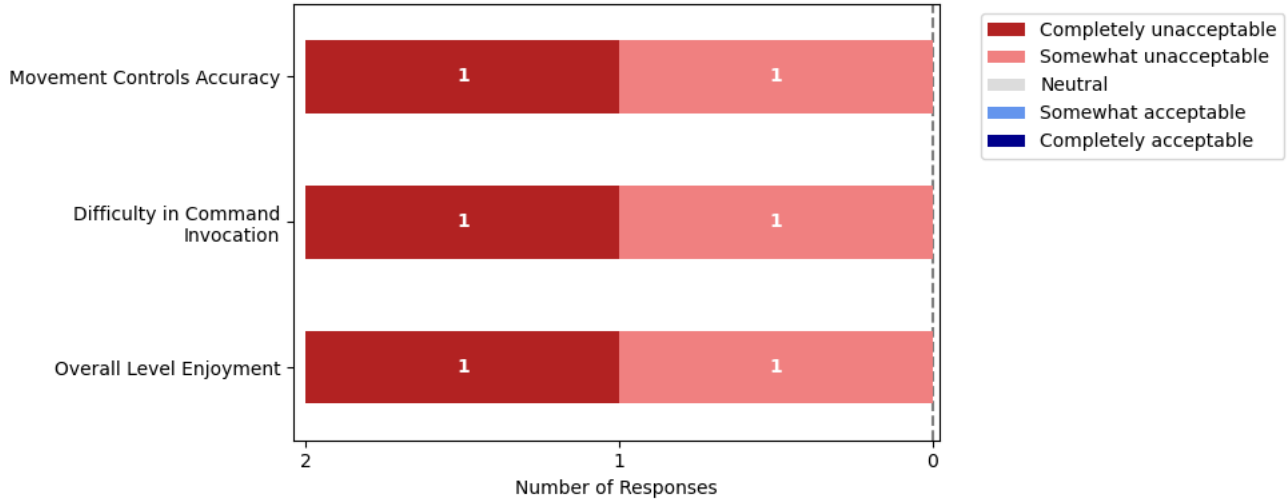

Figure 62: Level 3 Evaluation

On the other hand, the poor classification performance of our neural network for three and four classes, significantly degrades the experience and both levels were found unacceptable. There were many instances where the BCI predicted the wrong movement and it felt like the participants were fighting with the BCI system to perform a correct move.

7 Conclusions & Future Work

7.1 Conclusions

In this thesis, we created an EEG-based BCI to control movement in a VR Game. The user can drive movement for a character, interact with in-game props and control User Interface menus by invoking mental commands with his brain. For the classification of the EEG signal a combination of a convolutional neural network and a wasserstein generative adversarial network was used to further improve the accuracy and performance of the widely used software OpenVibe. The BCI system was tested on two participants for two, three and four available mental commands.

Our implemented neural networks are able to augment and classify EEG features with a fast training procedure. Our WGAN is able to generate EEG features that increases the average accuracy in classification, however, the model failed to learn the actual feature distribution, therefore, improvements could be made to the architecture. For two classes, the user is able to control the character, interact with props and UI menus with acceptable accuracy, even with minimal training. Furthermore, when increasing the available mental commands to three and four classes, the deep learning-based classification system still improves the accuracy of the BCI system compared to OpenVibe, however, the performance drops significantly compared to two classes, making the game unplayable.

Furthermore, the use of a BCI system in a VR game offers a more immersive experience for the user by combining the realistic environments of VR ,with the hands-free and brain-controlled movement of a BCI system. Though, more research needs to be conducted on improving performance for three and four classes to further increase the capabilities of BCI systems.

During MI training, the user is asked to rehearse MI mental commands as accurately as possible. However, there were problems when recording the EEG signal, during training sessions, where minor head movements with the VR headset altered the placement of the conductive gel and degraded the quality of the recorded EEG signal, resulting in worse classification performance. For this reason, more research on the performance of dry electrodes needs to be conducted.

Finally, the implemented neural networks were severely limited by the small amount of data that were recorded and they presented overfitting, especially for three and four classes. For more robust results more data could be recorded by different participants and for multiple training sessions.

Overall, the deep learning approach for the classification of EEG signals significantly improves the performance of BCI systems compared to the widely used software tool, OpenVibe. However, there is still room for improvements in the research field of BCIs, mainly, on the processing and classification performance for more than two classes, and the accuracy of non-invasive EEG electrodes.

7.2 Future Work

The main limitations of this thesis is the poor classification accuracy of the BCI system for three and four classes and the lacking performance of the WGAN. However, there are some improvements and ideas that could be done to surpass those limitations and even improve the overall performance of BCIs.

One action that could be taken to improve the performance of the BCI, is the acquisition of more electrodes. Currently, the eight wet electrodes of the Enobio device can cover a small area of the head, thus, losing important EEG information that could be recorded by a larger brain coverage. Furthermore, with the acquisition of more electrodes we will be able to implement channel selection algorithms that could dynamically decide which electrodes has the best quality of EEG signal and discard information of lesser quality. This can ensure that the recorded EEG signal contains only information relevant to the experiments we perform and discard dead or noisy electrodes.

In reality, EEG electrodes do not only record brain signal derived from the position that were placed, but also from nearby areas of the brain. This nature of the EEG electrode can cause degradation in classification performance when many electrodes are placed at positions close to each other, due to the overlap of EEG information captured by more than one electrode. For this reason, spatial filters can be utilized such as a Surface Laplacian filter where we can combine the information of nearby electrodes to a single output resulting in less overlap of information.

Currently, the BCI game requires a second person to manually start the acquisition of the EEG signal, the processing scenarios of OpenVibe and the training of the neural networks. For future work, a back-end system could be developed that automates all of the training procedure of the BCI and that could automatically begin the acquisition of the EEG signal when the user begins the training session.

Furthermore, more research could be done on improving Motor Imagery by testing various mental commands that could generate more distinguishable brain electrical patterns, and mental commands that could be easier for the user to rehearse.

In this thesis, we explored the capabilities of using a BCI for controlling UI menus and props in game. More research could be done to explore more use cases for BCIs that could not only expand the capabilities of BCIs in neurogaming but also in other research areas of BCIs.

Overall, Brain Computer Interfaces are a highly developing field and there is still room for research that could improve the overall performance and usability of BCIs, especially non-invasive BCIs.

Appendices

Additional Results and Diagrams

In the following figures the feature distribution of three and four classes are displayed from the recorded EEG dataset from participant 1.

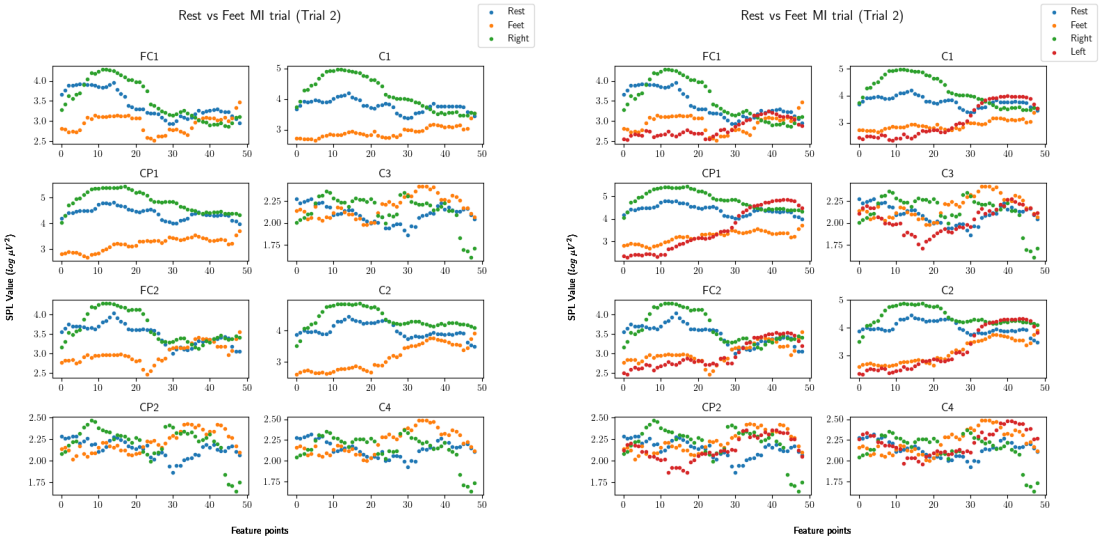

Figure 63: Three class(left) and four class(right) features from P1

From figure 63 we observe that in a lot of channels there is a lot of overlap in SPL feature values. This causes a major drop in accuracy for our neural network and OpenVibe's classifiers.

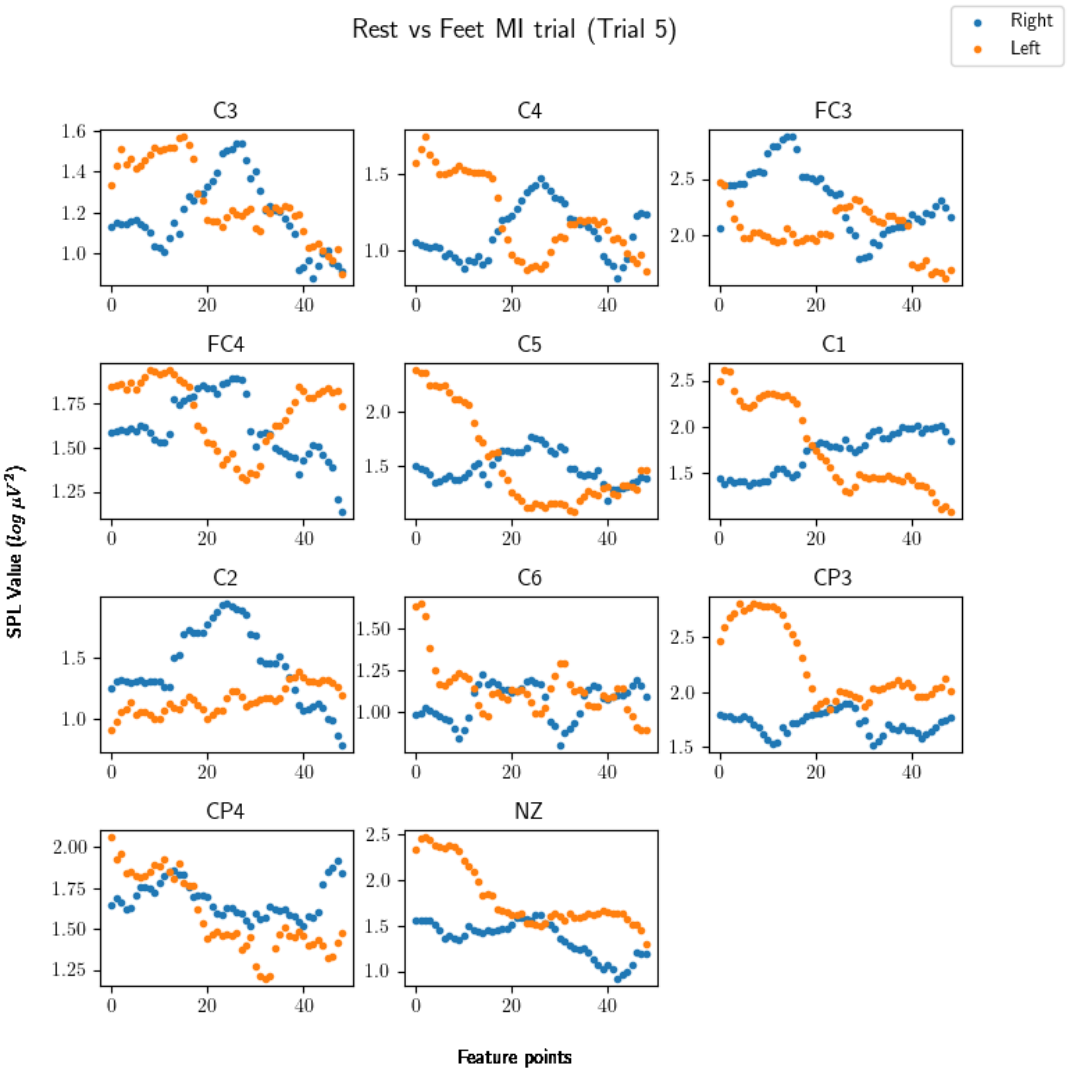

Figure 64: Feature distribution for Inria's dataset

Figure 64 shows the feature distribution of Inria's dataset for the Right and Left MI mental commands produced by the same processing scenario as our own dataset in OpenVibe. Since, the distribution of EEG features is different for each class our neural network was able to learn the unique temporal patterns from each feature class.

The following figures provide a visual representation of BCI command handling based on the player's state in the game.

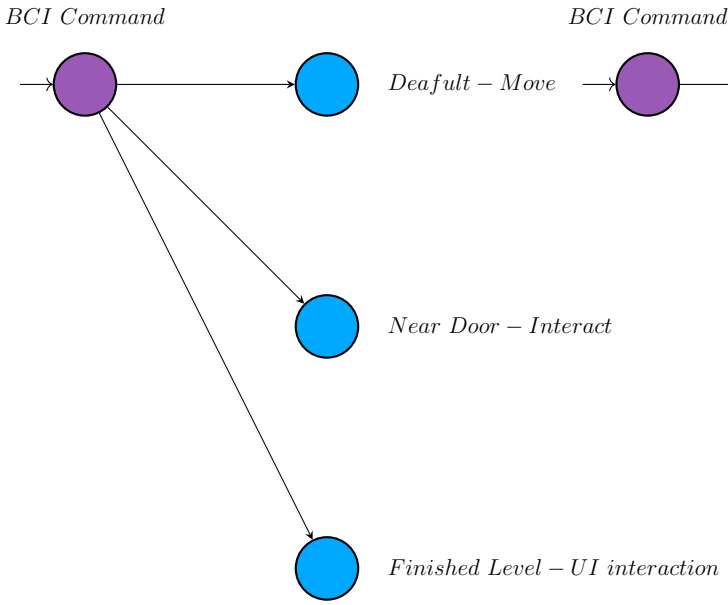

Figure 65: Initial State

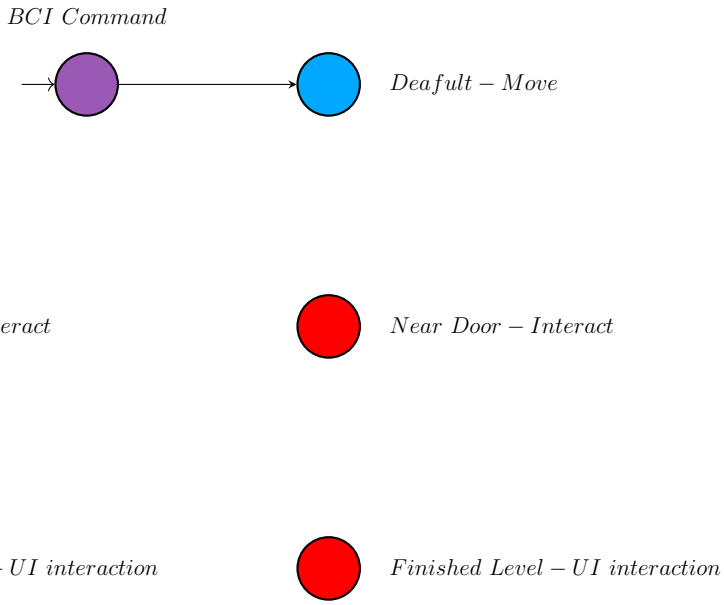

Figure 66: Playing State

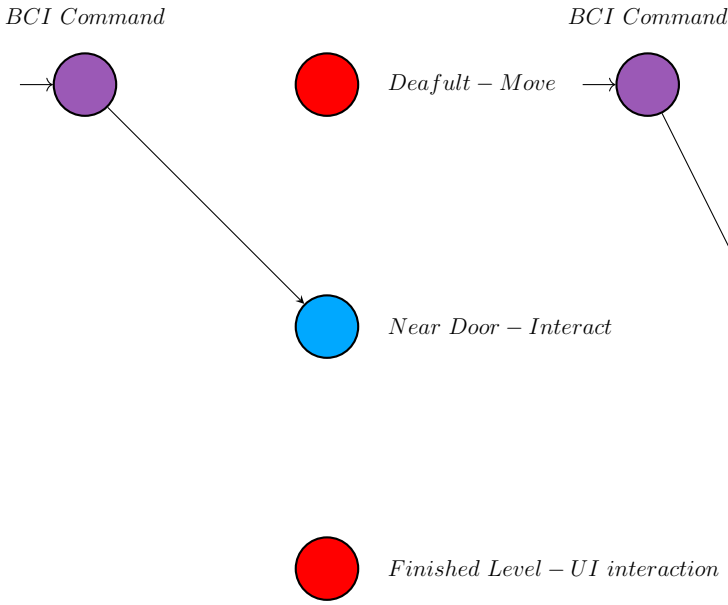

Figure 67: Interaction State

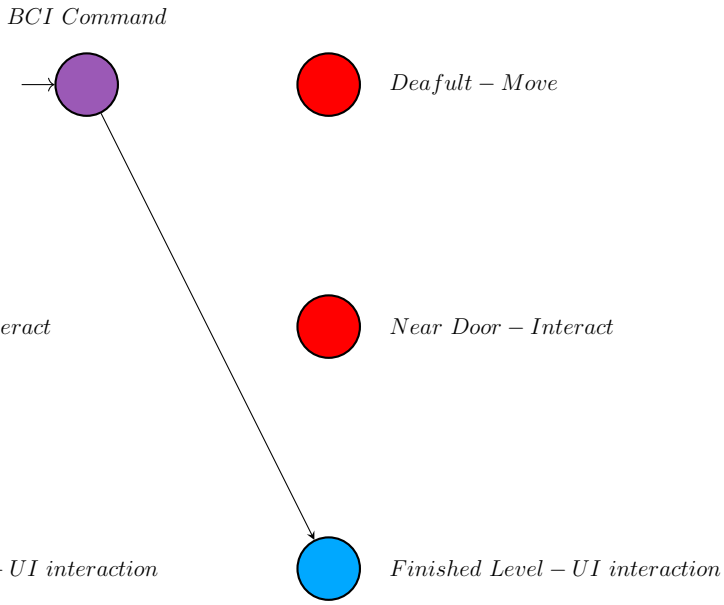

Figure 68: UI State

Additional results from the MI training sessions of each participant are presented in the following figures to compare the classification performance of the CNN on the recorded and augmented dataset. More specifically, histogram plots of the loss, confusion matrices and training accuracy plots for three and four classes, are provided in the following figures.

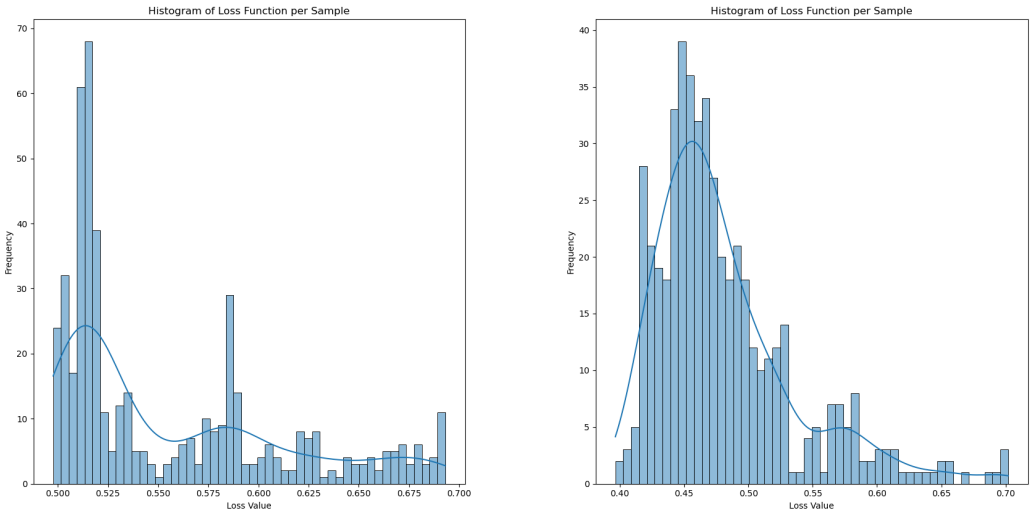

Figure 69: Loss histogram of two classes for recorded(left) and augmented(right) EEG dataset

The loss histogram further validates that close to zero overfitting is present in the trained model as the majority of loss samples are present in the low loss values.

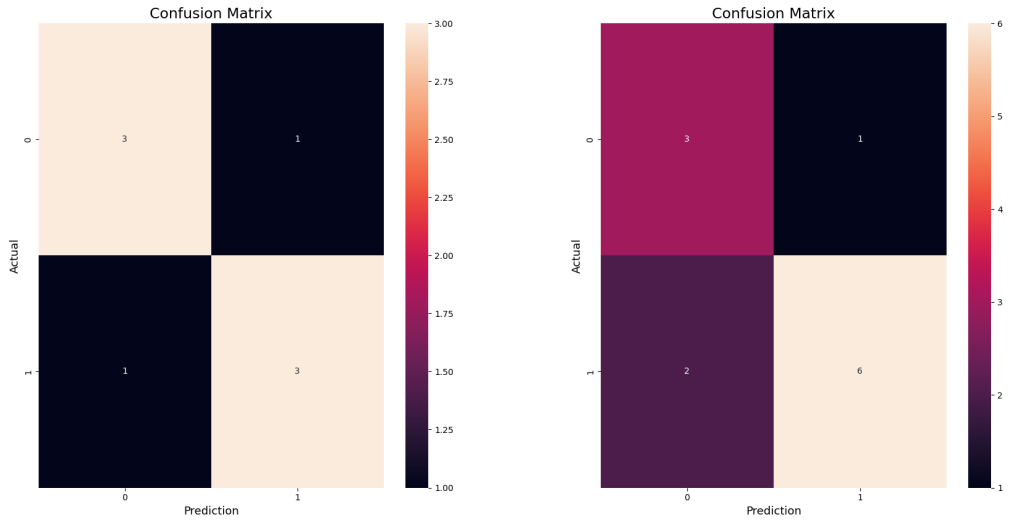

Figure 70: Confusion matrix of two classes for recorded(left) and augmented(right) EEG dataset

From the confusion matrix we observe that the model indeed learns to classify the EEG features and not randomly predicting a class or predicting only a single class.

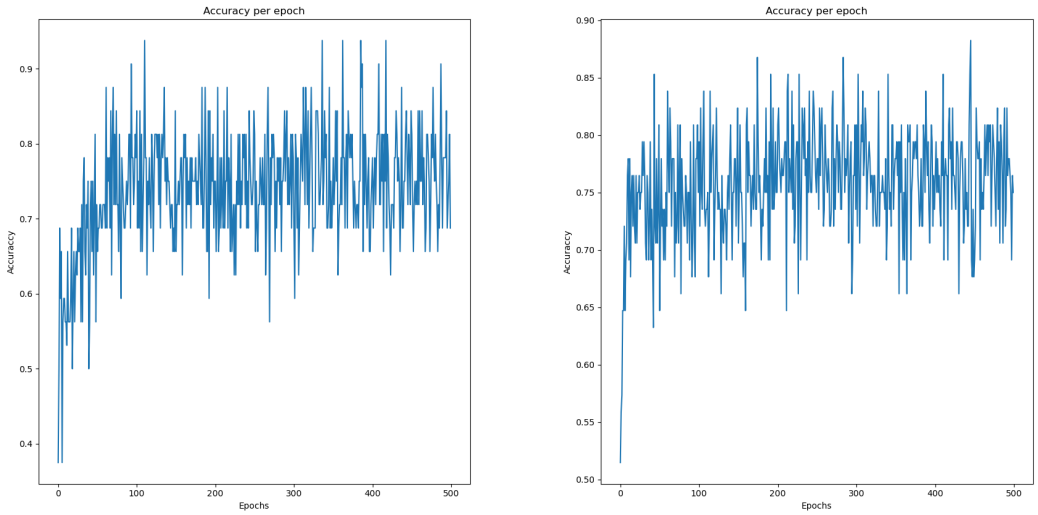

Figure 71: Train accuracy plot of two classes for recorded(left) and augmented(right) EEG dataset

Increasing the number of classes to three, we observe that some amount of overfitting is present, since more samples now are close to the mean loss value.

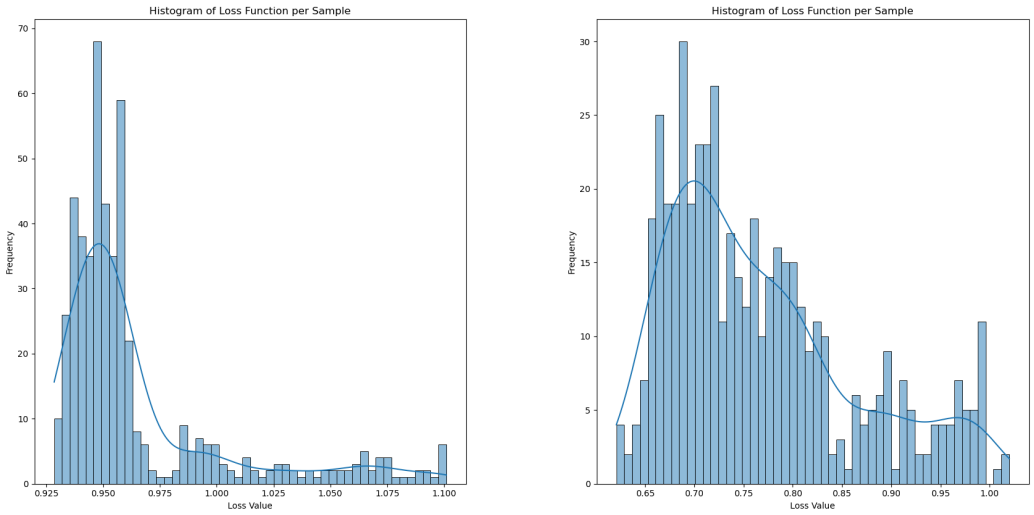

Figure 72: Loss histogram of three classes for recorded(left) and augmented(right) EEG dataset

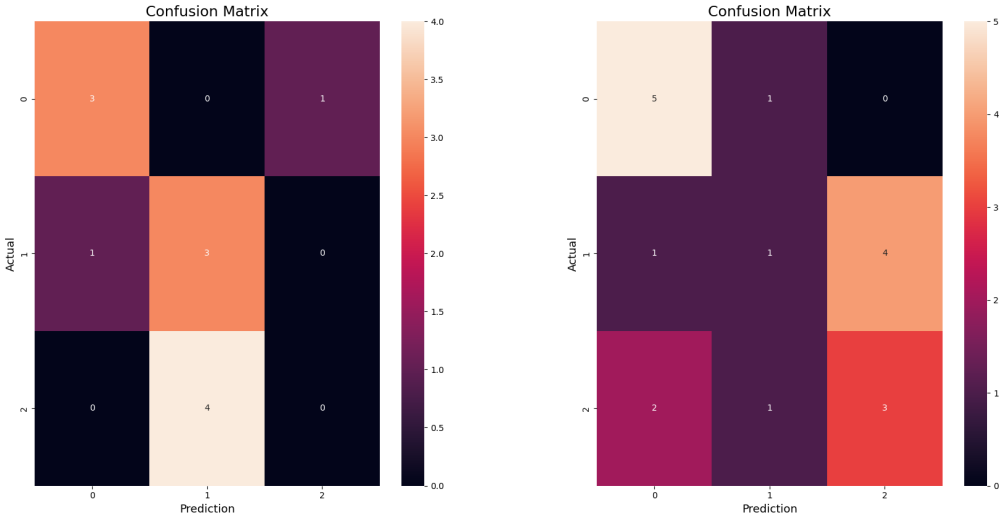

Figure 73: Confusion matrix of three classes for recorded(left) and augmented(right) EEG dataset

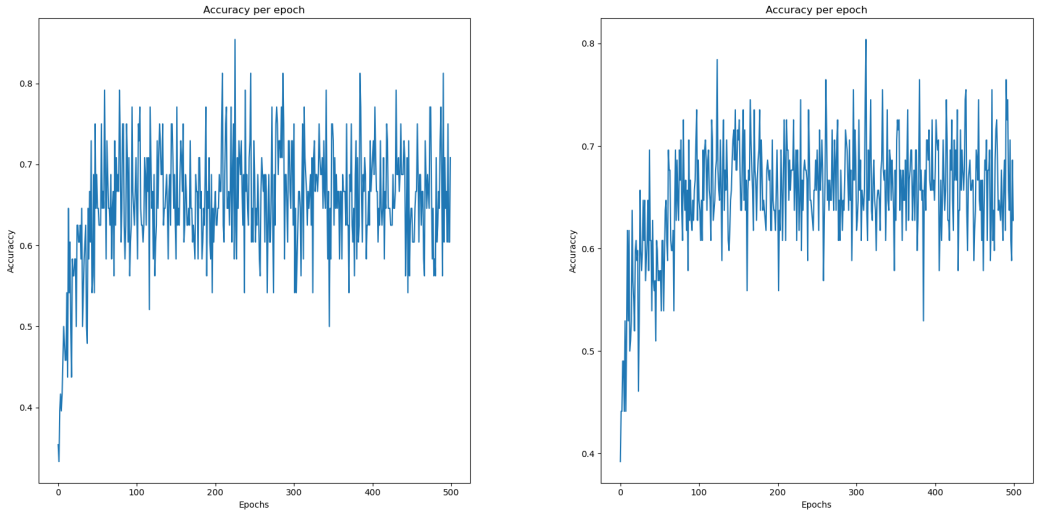

Figure 74: Train accuracy plot of three classes for recorded(left) and augmented(right) EEG dataset

In the case of four classes now more loss samples are close to the mean loss value in the histogram and that indicates that the four class CNN model presents some amount of overfitting.

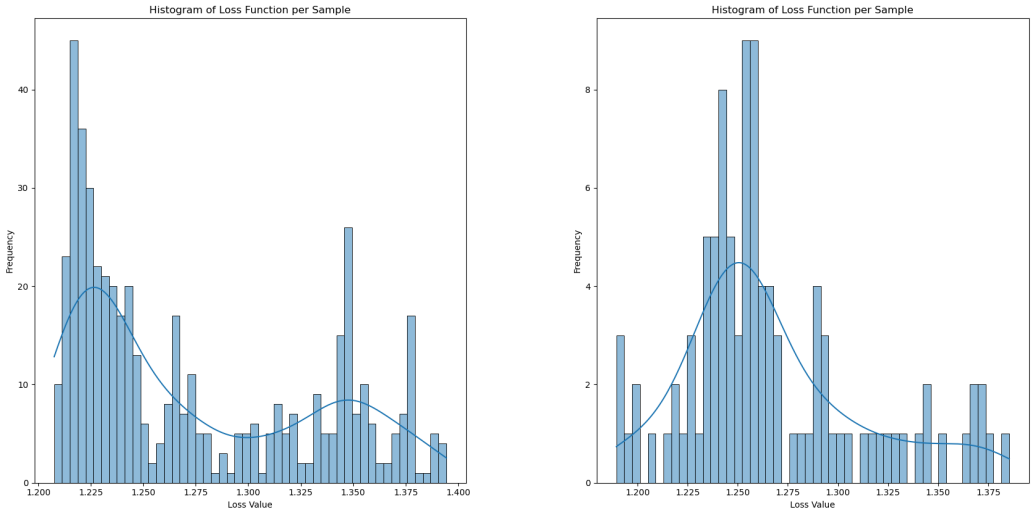

Figure 75: Loss histogram of four classes for recorded(left) and augmented(right) EEG dataset

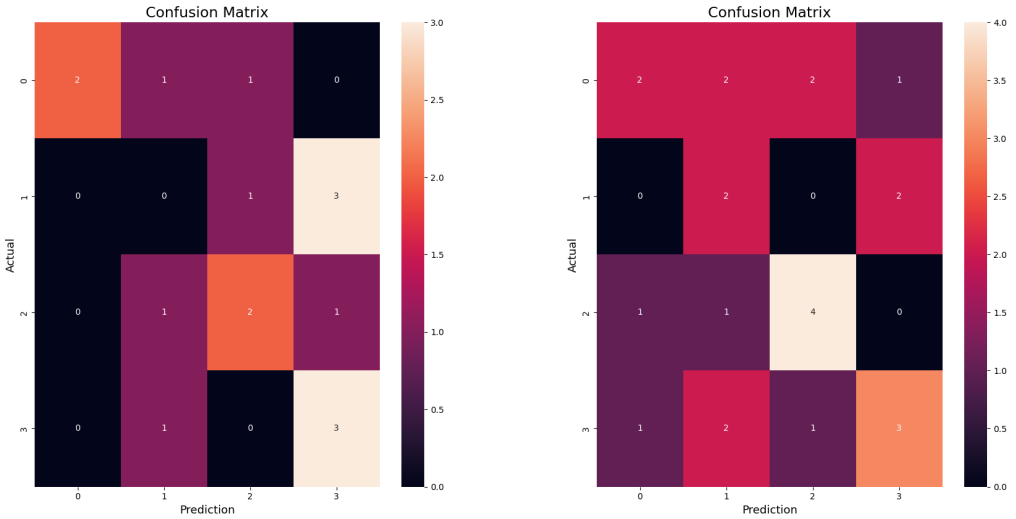

Figure 76: Confusion matrix of three classes for recorded(left) and augmented(right) EEG dataset

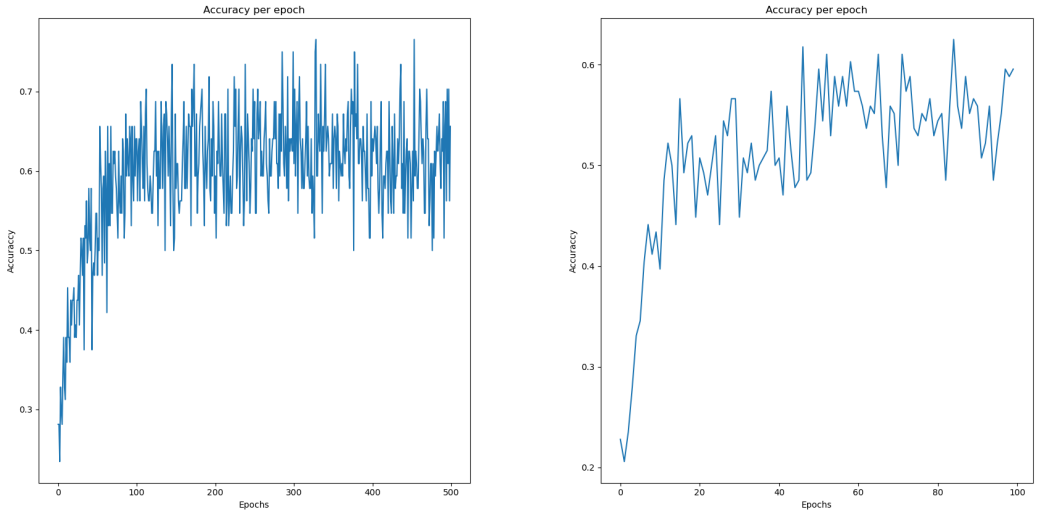

Figure 77: Train accuracy plot of three classes for recorded(left) and augmented(right) EEG dataset

References

[1] Nuraini Jamil et al. «Noninvasive Electroencephalography Equipment for Assistive, Adaptive, and Rehabilitative Brain–Computer Interfaces: A Systematic Literature Review». In: *Sensors* 21.14 (2021). ISSN: 1424-8220. DOI: 10.3390/s21144754. URL: https://www.mdpi.com/1424-8220/21/14/4754.

[2] Keng Peng Tee et al. «Augmenting cognitive processes in robot-assisted motor rehabilitation». In: Nov. 2008, pp. 698–703. DOI: 10.1109/BIOROB.2008.4762894.

[3] Lei Ren et al. «Fabrication of Flexible Microneedle Array Electrodes for Wearable Bio-Signal Recording». In: *Sensors* 18.4 (2018). ISSN: 1424-8220. DOI: 10.3390/s18041191. URL: https://www.mdpi.com/1424-8220/18/4/1191.

[4] Ling Guo et al. «Automatic epileptic seizure detection in EEGs based on line length feature and artificial neural networks». In: *Journal of Neuroscience Methods* 191.1 (2010), pp. 101–109. ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2010.05.020. URL: https://www.sciencedirect.com/science/article/pii/S0165027010002803.

[5] Eduardo López-Larraz et al. «Continuous decoding of motor attempt and motor imagery from EEG activity in spinal cord injury patients». In: vol. 2012. Aug. 2012, pp. 1798–801. DOI: 10.1109/EMBC.2012.6346299.

[6] Ismael Kazheen and Adnan Mohsin Abdulazeez. «Deep Learning Convolutional Neural Network for Speech Recognition: A Review». In: (Jan. 2021). DOI: 10.5281/zenodo.4475361.

[7] Audrey Aldridge and Cindy L. Bethel. «A Systematic Review of the Use of Art in Virtual Reality». In: *Electronics* 10.18 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10182314. URL: https://www.mdpi.com/2079-9292/10/18/2314.

[8] Wikipedia contributors. *Brain–computer interface — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Brain%E2%80%93computer_interface&oldid=1208835267.

[9] Aleksandra Kawala-Sterniuk et al. «Summary of over Fifty Years with Brain-Computer Interfaces—A Review». In: *Brain Sciences* 11.1 (2021). ISSN: 2076-3425. DOI: 10.3390/brainsci11010043. URL: https://www.mdpi.com/2076-3425/11/1/43.

[10] *The history of Brain-Computer Interfaces (BCIs) – Timeline*. https://roboticsbiz.com/the-history-of-brain-computer-interfaces-bcis-timeline/. Accessed: 2024-02-19.

[11] Léa Pillette et al. «Why we should systematically assess, control and report somatosensory impairments in BCI-based motor rehabilitation after stroke studies». In: *NeuroImage: Clinical* 28 (2020), p. 102417.

[12] Iolanda Pisotta, David Perruchoud, and Silvio Ionta. «Hand-in-hand advances in biomedical engineering and sensorimotor restoration». In: *Journal of neuroscience methods* 246 (2015), pp. 22–29.

[13] Danny Wee-Kiat Ng, Ying-Wei Soh, and Sing-Yau Goh. «Development of an autonomous BCI wheelchair». In: *2014 IEEE Symposium on Computational Intelligence in Brain Computer Interfaces (CIBCI)*. IEEE. 2014, pp. 1–4.

[14] Wikipedia contributors. *Electromyography — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Electromyography&oldid=1187000765.

[15] Wikipedia contributors. *Functional near-infrared spectroscopy — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Functional_near-infrared_spectroscopy&oldid=1195352358.

[16] Wikipedia contributors. *Electrocardiography — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Electrocardiography&oldid=1204996806.

[17] Wikipedia contributors. *Electroencephalography — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Electroencephalography&oldid=1200783236.

[18] Ashima Khosla, Padmavati Khandnor, and Trilok Chand. «A comparative analysis of signal processing and classification methods for different applications based on EEG signals». In: *Biocybernetics and Biomedical Engineering* 40.2 (2020), pp. 649–690.

[19] Reza Abiri et al. «A comprehensive review of EEG-based brain–computer interface paradigms». In: *Journal of Neural Engineering* 16.1 (Jan. 2019), p. 011001. DOI: 10.1088/1741-2552/aaf12e. URL: https://dx.doi.org/10.1088/1741-2552/aaf12e.

[20] Wikipedia contributors. *P300 (neuroscience) — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=P300_(neuroscience)&oldid=1203449623.

[21] John Polich. « Neuropsychology of P300». In: *The Oxford Handbook of Event-Related Potential Components*. Oxford University Press, Dec. 2011. ISBN: 9780195374148. DOI: 10.1093/oxfordhb/9780195374148.013.0089. eprint: https://academic.oup.com/book/0/chapter/293241580/chapter-ag-pdf/44513063/book_34558_section_293241580.ag.pdf. URL: https://doi.org/10.1093/oxfordhb/9780195374148.013.0089.

[22] Wikipedia contributors. *Steady state visually evoked potential — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Steady_state_visually_evoked_potential&oldid=1187358592.

[23] Wikipedia contributors. *Motor imagery — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Motor_imagery&oldid=1202571862.

[24] Young-Tak Kim et al. «Exploring the Number of Repetitions in Trials for the Performance Convergence of Classification in Motor Imagery Task with Hand-Grasping». In: *2019 7th International Winter Conference on Brain-Computer Interface (BCI)*. 2019, pp. 1–4. DOI: 10.1109/IWW-BCI.2019.8737308.

[25] Natasha Padfield et al. «EEG-based brain-computer interfaces using motor-imagery: Techniques and challenges». In: *Sensors* 19.6 (2019), p. 1423.

[26] Nelly Elsayed, Zaghloul Saad Zaghloul, and Magdy Bayoumi. «Brain computer interface: EEG signal preprocessing issues and solutions». In: *Int. J. Comput. Appl* 169.3 (2017), pp. 12–16.

[27] D Puthankattil Subha et al. «EEG signal analysis: a survey». In: *Journal of medical systems* 34 (2010), pp. 195–212.

[28] Duo Chen et al. «A high-performance seizure detection algorithm based on Discrete Wavelet Transform (DWT) and EEG». In: *PloS one* 12.3 (2017), e0173138.

[29] Maan M Shaker. «EEG waves classifier using wavelet transform and Fourier transform». In: *brain* 2.3 (2006), pp. 169–174.

[30] Brian J Roach and Daniel H Mathalon. «Event-related EEG time-frequency analysis: an overview of measures and an analysis of early gamma band phase locking in schizophrenia». In: *Schizophrenia bulletin* 34.5 (2008), pp. 907–926.

[31] Wikipedia contributors. *Discrete wavelet transform — Wikipedia, The Free Encyclopedia*. [Online; accessed 24-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Discrete_wavelet_transform&oldid=1186056197.

[32] Fahd A Alturki et al. «EEG signal analysis for diagnosing neurological disorders using discrete wavelet transform and intelligent techniques». In: *Sensors* 20.9 (2020), p. 2505.

[33] Jasmin Kevric and Abdulhamit Subasi. «Comparison of signal decomposition methods in classification of EEG signals for motor-imagery BCI system». In: *Biomedical Signal Processing and Control* 31 (2017), pp. 398–406.

[34] Vangelis P Oikonomou et al. «A comparison study on EEG signal processing techniques using motor imagery EEG data». In: *2017 IEEE 30th international symposium on computer-based medical systems (CBMS)*. IEEE. 2017, pp. 781–786.

[35] Wikipedia contributors. *Linear discriminant analysis — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Linear_discriminant_analysis&oldid=1188796231.

[36] Alois Schlögl et al. «Characterization of four-class motor imagery EEG data for the BCI-competition 2005». In: *Journal of neural engineering* 2.4 (2005), p. L14.

[37] Martin Steinisch, Maria Gabriella Tana, and Silvia Comani. «A post-stroke rehabilitation system integrating robotics, VR and high-resolution EEG imaging». In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 21.5 (2013), pp. 849–859.

[38] Daniela Muñoz et al. «EEG Evaluation in a neuropsychological intervention program based on virtual reality in adults with Parkinson's disease». In: *Biosensors* 12.9 (2022), p. 751.

[39] Christoph Guger et al. «Prosthetic control by an EEG-based brain-computer interface (BCI)». In: *Proc. aaate 5th european conference for the advancement of assistive technology*. Citeseer. 1999, pp. 3–6.

[40] Daniel Elstob and Emanuele Lindo Secco. «A low cost EEG based BCI prosthetic using motor imagery». In: *arXiv preprint arXiv:1603.02869* (2016).

[41] Marios Hadjiaros et al. «Virtual Reality Cognitive Gaming based on Brain Computer Interfacing: A narrative review». In: *IEEE Access* (2023).

[42] Szczepan Paszkiel. *Analysis and classification of EEG signals for brain-computer interfaces*. Springer, 2020.

[43] Qiang Wang, Olga Sourina, and Minh Khoa Nguyen. «Eeg-based" serious" games design for medical applications». In: *2010 international conference on cyberworlds*. IEEE. 2010, pp. 270–276.

[44] Ignas Martišius, Robertas Damaševičius, et al. «A prototype SSVEP based real time BCI gaming system». In: *Computational intelligence and neuroscience* 2016 (2016).

[45] Michael McMahon and Michael Schukat. «A low-Cost, Open-Source, BCI-VR Game Control Development Environment Prototype for Game Based Neurorehabilitation». In: *2018 IEEE Games, Entertainment, Media Conference (GEM)*. 2018, pp. 1–9. DOI: 10.1109/GEM.2018.8516468.

[46] Georgios Prapas et al. «Motor Imagery Approach for BCI Game Development». In: *2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. 2022, pp. 1–5. DOI: 10.1109/SEEDA-CECNSM57760.2022.9932937.

[47] Denis Delisle-Rodriguez et al. «Multi-channel EEG-based BCI using regression and classification methods for attention training by serious game». In: *Biomedical Signal Processing and Control* 85 (2023), p. 104937. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2023.104937. URL: https://www.sciencedirect.com/science/article/pii/S1746809423003701.

[48] Virtual Reality Society. *History Of Virtual Reality*. URL: https://www.vrs.org.uk/virtual-reality/history.html.

[49] Virtual Speech. *History of VR – Timeline of Events and Tech Development*. URL: https://virtualspeech.com/blog/history-of-vr.

[50] Wikipedia contributors. *Panoramic painting — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Panoramic_painting&oldid=1180908004.

[51] Le Cun Yan, B Yoshua, and H Geoffrey. «Deep learning». In: *nature* 521.7553 (2015), pp. 436–444.

[52] Christian Janiesch, Patrick Zschech, and Kai Heinrich. «Machine learning and deep learning». In: *Electronic Markets* 31.3 (2021), pp. 685–695.

[53] Jason Brownlee. *A Gentle Introduction to Padding and Stride for Convolutional Neural Networks*. 2019. URL: https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[55] Wikipedia contributors. *Autoencoder — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-February-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=1192294199.

[56] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[57] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. arXiv: 1701.04862 [stat.ML].

[58] Martin Arjovsky, Soumith Chintala, and Léon Bottou. «Wasserstein generative adversarial networks». In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[59] Yankun Xu, Jie Yang, and Mohamad Sawan. «Multichannel synthetic preictal EEG signals to enhance the prediction of epileptic seizures». In: *IEEE Transactions on Biomedical Engineering* 69.11 (2022), pp. 3516–3525.

[60] Kay Gregor Hartmann, Robin Tibor Schirrmeister, and Tonio Ball. «EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals». In: *arXiv preprint arXiv:1806.01875* (2018).

[61] Ghaith Bouallegue and Ridha Djemal. «EEG data augmentation using Wasserstein GAN». In: *2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE. 2020, pp. 40–45.

[62] Neuroelectrics. *Enobio 8 EEG Headset*. Accessed: 21-February-2024. URL: https://www.neuroelectrics.com/solutions/enobio#!.

[63] Meta. *Meta Quest 2 Standalone VR headset*. Accessed: 21-February-2024. URL: https://www.meta.com/quest/products/quest-2/#overview.

[64] Neuroelectrics. *Neuroelectrics website*. Accessed: 22-February-2024. URL: https://www.neuroelectrics.com/.

[65] Inria research team. *OpenVibe — Software for Brain Computer Interfaces and Real Time Neurosciences*. Accessed: 22-February-2024. URL: http://openvibe.inria.fr/.

[66] Unity. *Unity Game Engine*. Accessed: 22-February-2024. URL: https://unity.com/.

[67] Sreeja Sr et al. «Classification of Motor Imagery Based EEG Signals Using Sparsity Approach». In: Dec. 2017, pp. 47–59. ISBN: 978-3-319-72037-1. DOI: 10.1007/978-3-319-72038-8_5.

[68] Bitbrain. *All about EEG artifacts and filtering tools*. Accessed: 23-February-2024. URL: https://www.bitbrain.com/blog/eeg-artifacts.

[69] Brainclinics foundation. *Real-time Filtering in BioExplorer*. Accessed: 23-February-2024. URL: https://brainclinics.com/wp-content/uploads/Filtering_in_BioExplorer.pdf.

[70] PyTorch. *Categorical Cross Entropy*. Accessed: 26-February-2024. URL: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.