

---

---

# **Utilizing Reinforcement Learning to Optimize Non-invasive BCIs for Robotic Rehabilitation**

- Master Thesis -

---

---

Master Thesis  
Group 1071 - Christian Grønborg Madsen

Aalborg University  
Robotics



**Robotics**  
Aalborg University  
<http://www.aau.dk>

## **AALBORG UNIVERSITY**

### STUDENT REPORT

**Title:**

Utilizing Reinforcement Learning to Optimize Non-invasive BCIs for Robotic Rehabilitation

**Theme:**

Scientific Theme

**Project Period:**

Spring Semester 2025

**Project Group:**

Group 1071

**Participant(s):**

Christian Grønborg Madsen

**Supervisor(s):**

Rasmus Leck Kæseler

**Page Numbers:** 52**Date of Completion:**

June 4, 2025

**Abstract:**

This project implements a novel RL and CSP FFNN pipeline to try and create a BCI system that can adapt to new users without a calibration session before use. Specifically, the system can optimize pretrained CSP weights so that they fit to new, unknown data, thereby enhancing and adapting the spatial filters. The results showed that it achieved accuracies of 45% for hands-, 40% for feet-, and 14% for rest class. For comparison, an LDA classifier was used on the same data. This project lays out the groundwork for future work and research using this pipeline.

*The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Initial Problem Statement . . . . .	2
<b>2</b>	<b>Problem Analysis</b>	<b>3</b>
2.1	EEG BCI . . . . .	3
2.2	Current Methods . . . . .	7
2.3	Application . . . . .	8
2.4	Considerations for BCI Feasibility . . . . .	10
2.5	Final Problem Statement . . . . .	11
<b>3</b>	<b>Requirements</b>	<b>12</b>
<b>4</b>	<b>Methods</b>	<b>14</b>
4.1	System Overview . . . . .	14
4.1.1	Classical CSP vs CSP FFNN . . . . .	15
4.2	Dataset and Preprocessing . . . . .	16
4.2.1	Dataset . . . . .	16
4.2.2	Preprocessing . . . . .	17
4.3	Common Spatial Pattern . . . . .	17
4.4	Linear Discriminant Analysis Classifier . . . . .	18
4.5	Reinforcement Learning Framework . . . . .	19
4.5.1	Proximal Policy Optimization . . . . .	20
4.5.2	CSP FFNN . . . . .	23
4.5.3	Environment . . . . .	26
4.5.4	System Architecture . . . . .	27
4.5.5	Implementation . . . . .	27
4.6	Robotic Implementation . . . . .	28
<b>5</b>	<b>Testing</b>	<b>31</b>
5.1	Data . . . . .	31
5.2	Testing Process . . . . .	32
5.2.1	Individual Subject Testing . . . . .	32

Contents	iii
5.2.2 Across Subject testing . . . . .	32
<b>6 Results</b>	<b>33</b>
6.1 Confusion Matrices . . . . .	35
<b>7 Discussion</b>	<b>40</b>
<b>8 Future Work</b>	<b>44</b>
<b>9 Conclusion</b>	<b>46</b>
<b>Bibliography</b>	<b>47</b>
<b>A Appendix</b>	<b>a</b>
A.1 Theory of CSP . . . . .	a

# Chapter 1

## Introduction

The number of strokes annually is increasing, related to an growing age problem in many countries[1]. Most of these victims require some sort of rehabilitation to recover lost motor functions in limbs and regain independence. The general priority for most victims of stroke and spinal cord injuries (SCI) when going into rehabilitation is regaining movement in the upper extremity, specifically regaining arm and hand movement[2][3][4]. The main reason is that arm and hand movement are essential for everyday tasks[5] and therefore it will be the focus point of this project. There are different ways that the therapy can be carried out, such as Constraint Induced Movement Therapy (CIMT), repeated tasks, and electrical stimulation[6]. Newer solutions include robot-assisted therapy, a growing topic for rehabilitation[7]. This growing interest in using robotic solutions could be related to the increasing age problem as mentioned earlier, and the lack of therapists to carry out the rehabilitation. Many of the current robotic solutions are concerned with creating a system that can actively assist in movement, such as moving the patient's arm with varying assistance, but the patient is not in direct control of the exoskeleton or robot arm. Recent research has shown that employing Brain Computer Interface (BCI) alongside a robotic device could further improve the quality of the outcome of the rehabilitation therapy as compared to traditional therapy with a therapist and passive assistance[8]. This is also supported and justified by the theory that Motor Imagery (MI) is part of a shared network of neurons that are also activated by simulation of action by the observation of action, which is an essential part of motor control and learning.[9].

The increased interest in robotic solutions for rehabilitation purposes also demands systems that can both perform well and have high usability. One of the main complications of non-invasive BCIs, and other systems that rely on signals from the brain, is the diverse nature of the signals from person to person and even from day to day. This requires a long calibration session before each usage. While a system calibrated for one person might work well for that person, it would struggle to generalize to signals from a different person and therefore both the performance and usability go down. A system created for rehabilitation purposes should strive to be able to perform as well as if it were a therapist

carrying out the exercises. The main difference also lies in the fact that a therapist can not see when a person is trying to move their limb, but a rehabilitation system could be able to know. For that reason, it is paramount to create a system that can generalize to a broader spectrum of EEG signals. It is, however, difficult to make a system that has high performance for every person immediately upon first use[10]. Especially if the system was not trained on data from the person using it. This calls for a system that can be trained online and calibrated for the specific person using it to increase performance over time.

## **1.1 Initial Problem Statement**

With the motivation as described above, the following initial problem statement has been formulated:

**How can a non-invasive EEG BCI system be created for rehabilitation using a robotic device?**

## Chapter 2

# Problem Analysis

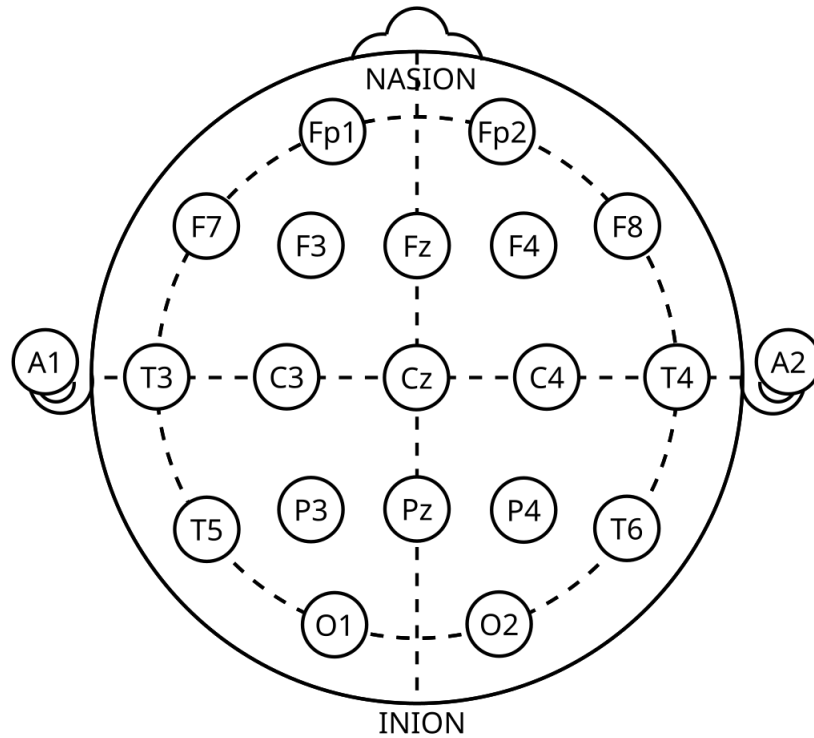
### 2.1 EEG BCI

EEG is the process of obtaining brain signals via electrodes either implanted in the brain, which is called invasive EEG, or by electrodes placed on the scalp of the head via a cap or similar equipment, or directly, which is called non-invasive EEG. This project will focus on non-invasive EEG. Electrode positioning on the scalp can vary quite a bit, but the international standard is the 10-20 system, which denotes the actual distance between the electrodes measured from front-back of the head and left-right of the head in percentage. That means that the 10-20 system would be either 10% or 20% of these distances[11]. The 10-20 system, along with 21 electrode placements, can be seen in figure 2.1. Other methods, such as the 10-10 or the 10-5 system, are also used. However, they use a lot of electrodes, but they also provide a higher spatial resolution. The signal is typically broken down into 5 bands called delta, theta, alpha, beta, and gamma. These bands range from 0 to above 30Hz, usually no more than 100Hz. See table 2.1 for the distribution of frequencies of the five bands.

Band	Frequency in Hz
Gamma	>30
Beta	13-30
Alpha	8-13
Theta	4-8
Delta	up to 4

**Table 2.1:** Overview of the five types of brain waves and their frequency range.

MI primarily lies within the alpha, mu, and beta bands, where mu is a subset of the alpha band. The EEG signal will usually contain artefacts from eye-blinking, heartbeat, muscle movement, and electrical interference from outside[12]. Eye-blinking artifacts are usually removed by having electrooculogram (EOG) electrodes placed on either side of the eyes, and that signal is subtracted from the overall EEG signal. Eye movement and blinking affect the frontal electrodes F and Fp. Heartbeat artifacts are more predominant on the left side, as that is where the heart is. Electrical interference is usually removed by applying a Notch filter at 50Hz. Typically, the signal will be filtered upon acquiring it to remove as many of these artifacts as possible before going into the preprocessing of the signal.

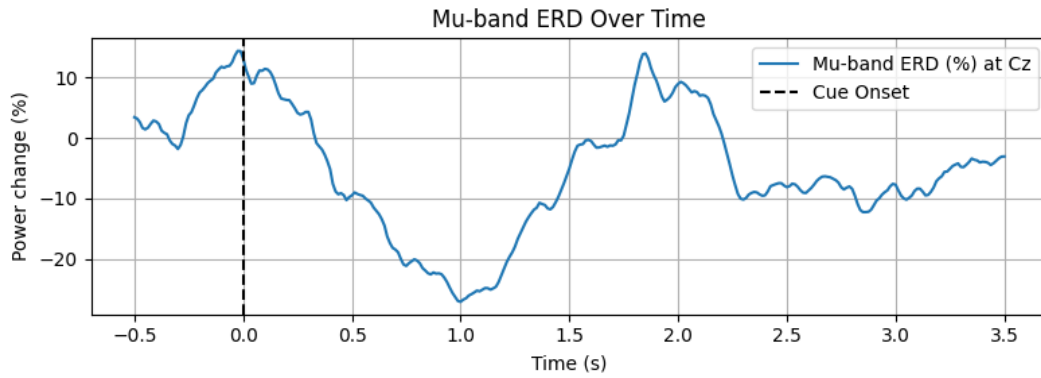


**Figure 2.1:** 10-20 system showing 21 electrode placements including pre-frontal (Fp), frontal (F), central (C), parietal (P), occipital (O), and mastoids (A).

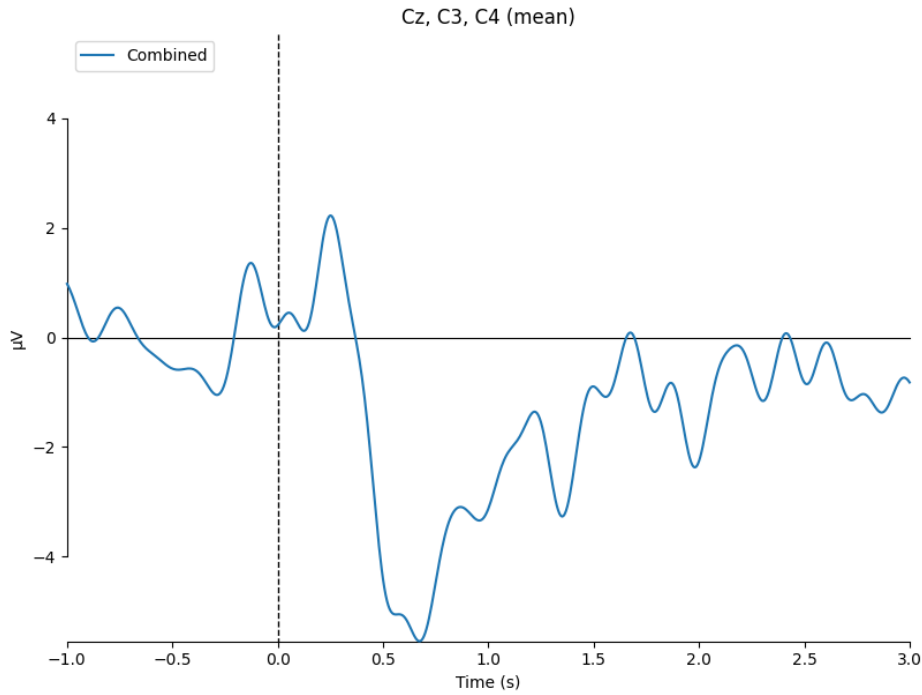
When working with EEG MI BCI, there are typically two types of signals that can be used: Sensory Motor Rhythm (SMR) and Movement Related Cortical Potential (MRCP). SMR and MRCP cover both imagined movement, MI, and executed movements, and are exhibited during or imagined movement. SMR lies within the alpha and beta bands, while MRCP lies within 0 to 5Hz. SMR is a fast modulating signal, only taking milliseconds between modulations, and the mechanism behind it is Event Related Desynchronization (ERD) and Event Related Synchronization (ERS). ERD is seen as a decrease in power of the signal, while ERS is seen as an increase in power and happens, respectively, when there



is an activation of a cortical area (ERD) and a deactivation of a cortical area (ERS). An example of SMR can be seen in figure 2.2 in which Time-Frequency Representation (TFR) has been calculated and averaged over 7 subjects, amounting to 140 trials of opening and closing both hands. The data is from a dataset created by Schalk et al.[13]. The relative power change in the signal in % was extracted over 4 seconds. It is calculated only from Cz and filtered to be within the mu band (8-14Hz). The plot shows a clear ERD from cue onset to 1 second after. MRCP's however, are low-frequency, slow modulating signals that occur before movement and have a gradual negative shift over typically a couple of seconds before movement. After the movement onset, the signal will exhibit a positive slope, and the MRCP will have peak negativity around this onset[14]. An example of MRCP can be seen in figure 2.3. This is a grand average MRCP over the same 7 subjects from dataset[13] using electrodes Cz, C3, and C4. MRCPs is a potentially good way of controlling a BCI system if the system is synchronous (cue-based), but could struggle with online asynchronous systems[15].



**Figure 2.2:** Averaged ERD plot over time for 7 subjects using Cz channel.



**Figure 2.3:** Grand average MRCP for 7 subjects in dataset[13] over electrodes Cz, C3, and C4.

The signal from the brain is fairly weak, being in the range of microvolts, so the signal is run through an amplifier to make the signal stronger. Gamma band brain signals are usually too weak for non-invasive electrodes to pick up anything useful from them[16]. Different types of electrodes can be used, including active and passive electrodes and wet and dry ones. The gold standard for non-invasive EEG acquisition is wet active electrodes[17]. Once the signal has been acquired and filtered to remove artefacts, it goes into the preprocessing stage. There could still be artefacts contained in the signal, so methods such as Independent Component Analysis (ICA) can be used as an artefact remover. One of the more prevalent methods of preprocessing is using a filter bank to break down the signal into the relevant frequency ranges and discard the irrelevant bands, such as gamma, theta, and delta, for MI SMR usage and creation of spatial filters.

After preprocessing, the next step would be to extract relevant features from the signal. There are many methods such as Wavelet Transform (WT)[18], Fourier transform (FT)[19], Empirical Mode Decomposition (EMD)[20], Power Spectral Density (PSD)[21], and one of the most common methods is Common Spatial Patterns (CSP)[22] which could work as both a feature extractor and also a preprocessing tool. The last step is for the signal to enter the classification stage, in which some methods have already been mentioned in chapter 1. Once the EEG has been acquired, processed, and classified, one could now use this pipeline for BCI purposes to, for instance, facilitate rehabilitation using an external device.

## 2.2 Current Methods

Machine learning (ML) has seen increased usage in BCI scenarios because of its ability to generalize well to unseen data. From simple classifiers such as k-Nearest Neighbour (k-NN)[23][24] and Linear Discriminant Analysis (LDA)[25][26] to more complex pipelines such as Artificial Neural Networks (ANN)[23] and even more complex Deep Learning (DL)[27]. Another type of ML is Reinforcement Learning (RL), which differs from other ML approaches in that the system learns what to do by trial and error. Essentially, it is a trade-off between exploration of unknown territory and exploitation of current knowledge to maximize a reward[28]. Exploration in a BCI setting would be taking new or uncertain actions when given an EEG signal, be it processed or otherwise raw, while exploitation would be taking actions and doing direct commands that the systems know have worked well in the past. This trade-off is essential for the system to generalize to the varying EEG data. Not only would it help increase performance between users, but also allow it to adapt to a single user's signal if they change due to, e.g., fatigue.

Using RL also opens up avenues in which the system would directly control the robotic rehabilitation device without having to send classification results or similar to a separate control system. While this, in theory, could be done with other ML approaches, the advantage of using RL is having the complex nature of actions, which are optimized within the system itself. Furthermore, it would cut out processing time which in turn should streamline the system and make it faster. While repetition and high accuracy are important in rehabilitation purposes and others, it is also important to keep the processing time to the least amount possible to ensure that the rehabilitation takes full advantage of the neuroplasticity of the brain[8].

While ML approaches such as ANNs and DL are extensively used within the BCI field [29][30][31][32], RL has not seen the same amount of research. While RL is a method that has been used in a lot of different areas of autonomous decision-making, such as autonomous driving, its utilization in a BCI scenario has not been extensively explored. RL focuses on the trade-off between exploration and exploitation, as mentioned earlier. The purpose of an RL system is to perform actions that generate the highest reward to find the optimal or near-optimal policy. RL differs from most ML approaches because it is mainly unsupervised learning, meaning it does not necessarily need labelled input-output data pairs. However, in some cases, it can also be semi-supervised, in which one has a smaller subset of labelled data. It will learn the optimal policy, meaning the "correct" action associated with a certain state the system may be in, for example, receiving an EEG signal and associating the correct class output of that signal. The system requires feedback, such as negative rewards from choosing the incorrect class, and one such way this has been done in BCI is using Error-related Potentials (ErrP's)[33].

ErrPs are signals from the brain that occur when a person recognizes a mistake. This

signal can then be fed into the system to ensure that the RL framework receives some sort of penalty to correct its mistake and improve its performance. The way this would be implemented could be in an actor-critic RL framework. In such a system, the actor, defined by  $\pi(a|s)$ , takes an action,  $a$ , based on the current state,  $s$ , defined by the policy  $\pi$ . The critic evaluates these actions and estimates their value or quality by providing feedback to the actor. As such, the critic will guide the actor towards actions that provide the greatest reward[28].

A different approach could be using a multi-agent framework such as the one created by Shin et al.[34]. In this, multiple agents act in a shared environment in which they interact with each other and execute a series of actions to maximize the cumulative reward. In this paper, they work together to determine the most appropriate features to later feed into a DL classifier. This RL framework also uses a critic to give feedback to the system. Both of these methods fall under the actor-critic method for RL. The main point about this method is that it combines both policy-based methods and value-based methods. Policy-based methods, such as Proximal Policy Optimization (PPO) used by Vukelić et al. [35], focus on learning and optimizing the policy directly without estimating the value function. The value function is an estimation of how good it is for the agent to be in a certain state or how good it is to take a certain action based on a state. This is what value-based methods try to achieve. These methods, such as Deep Q-Learning (DQN) used by Nallani et al.[36], employ various techniques to estimate the value-function, which in turn optimizes the policy by picking the actions that maximize this value-function.

The presented methods are just a few of the plethora of methods used within RL. There are a lot of branches of interesting ways of implementing RL, but these implementations also come with certain requirements.

## 2.3 Application

As stated earlier, there is a positive effect on rehabilitation when using BCI systems over traditional therapy and passively assisting systems. There are different approaches to accomplish this. There is, to the best of the author's knowledge, not a particularly large commercial area for BCI rehabilitation devices. However, one such commercially available solution is the IpsiHand[37]. This device connects to a non-invasive multi-electrode cap and is used to rehabilitate wrist and hand motions by detecting ipsilateral motor cortex signals. The number of electrodes they use is 6[38], however, a typical requirement for such a system would be at least up to 16 electrodes to give adequate spatial information[39].

Another similar device, which also included arm movement, was created by Ramos-Murguialdy et al.[40]. In this, a hand and arm orthosis was used to open and close the hand and move the arm when the patient tried to perform a movement. This study did not use pure MI for the control, instead, they asked the patient to try and perform a move-

ment even if the arm did not follow. They acquired SMR during the intend to perform a movement. They controlled it continuously, meaning that as long as the SMR was correct, the movement was executed in the orthosis, only stopping when they got a rest SMR state. This type of continuous control of an external device is important to make the system intuitive and have a natural control rather than simply executing a predefined movement upon getting a single correct signal from the brain. And, as mentioned previously, to get the most out of the neuroplasticity of the brain, the experience should be as natural as possible. In their continuous classification, they created zones associated with either motion or rest, defined by the power of the SMR signal. If the classification remained in either zone for 200 milliseconds, it would either stop moving or execute the motion. The system would continue the last state it was in until the classification stays in one of the zones in the allotted time, regardless of it switching between states.

There are numerous other solutions, including orthosis, robotic manipulators, and exoskeletons[41][42][43][44]. In the application of a BCI system, several key factors need to be accounted for to ensure proper usage and the best outcome. Two of these factors have already been mentioned in this section, being the number of electrodes and the classification of continuous control commands. These key factors cover both system and hardware requirements. Typically, a sampling frequency of a minimum of 256Hz for the acquisition of the EEG signal[17], and studies show that exceeding a delay from intent from the user to action of the external device over 0.5 seconds causes noticeable disruption and diminishes the user experience[45]. In this delay, there are two parts to it: the system delay and the hardware delay. Effort is mainly put into minimizing the system delay by making sure that the time it takes to acquire, process, and classify the signal is as fast as possible. The other part concerns hardware delay, which in most cases is limited by the hardware available and can not necessarily be optimized. The actuation delay is limited by the actuators so the only real delay that could be adjusted would be the delay from sending the command from the system to whatever is controlling the external device.

Many of the hardware requirements are usually focused on safety engineering of the device. Safety features should include emergency stop and excessive force fail-safes and preventions[17]. The safety requirements of the robotic device are important when developing a control system for a rehabilitation device. Considerations such as the force of the actuators resulting in faster or slower movement and adjustments to the execution speed are also important in the design. However, it is a tremendously difficult task to associate the desired speed and intensity of the MI task. All these requirement, along with classification accuracy, has to be considered throughout the design process of the EEG BCI system.

While the solutions mentioned so far have BCI usage, it is also worth looking at current rehabilitation robots that do not use BCI to carry out the training, but could have the potential to do so. One of these solutions is Robert rehabilitation robot developed by Life

Science Robotics[46]. This system has both a lower- and upper extremity attachment. It is an assistive device that helps the patient perform an exercise after the therapist has "shown" the movement to the robot. This means that the therapist records the motion using the robot, and afterwards the robot can keep performing the same movement. They also include an upgrade with Electromyographic (EMG)-triggered Functional Electrical Stimulation (FES). They currently do not use any BCI elements, but could potentially utilize this in their current system. One way would be to still maintain how the therapist records the motion, but then let the patient rely on activating the motion using MI BCI. This would enable them to take advantage of the notion that the outcome of the rehabilitation process is improved when the patient actively uses their motor cortex to engage with the motion.

As stated earlier, the priority for most victims of stroke or SCI when going into rehabilitation is regaining movement in specifically arm and hand. A lot of current research is focusing on specific areas, such as the ones mentioned above, concerned with hand movement, arm movement, finger movement, or other specific areas. There are also systems that focus on more than one area, such as the hand and arm orthosis mentioned previously, but even they had a focus on hand movements, particularly, and did not include wrist rotation, but only elbow flex/extension. For that reason, it is interesting to look into creating a system that could encompass the entirety of the arm to carry out a broader spectrum of exercises.

## 2.4 Considerations for BCI Feasibility

If one were to try and create a system that encompasses hand, wrist, and elbow movement, then several factors need to be considered before developing a control system for it. The major consideration lies within the multi-class paradigm of the classifier in that both hand open/close, wrist pronation/supination, and elbow flexion/extension would be part of it. The distinguishability between two types of movement of the same joint could be difficult to achieve a satisfactory accuracy due to the possibly low spatial difference of the signals.

However, current research in this subject shows that creating such a system in which it can distinguish between, e.g., wrist pronation/supination is feasible within a satisfactory accuracy[47][48][49]. In these articles, the focus was on hand and wrist motion, which are important motions in terms of upper limb rehabilitation[2]. Using RL for this kind of multi-class paradigm could potentially yield higher accuracy results as the system would be more susceptible to changes and online learning. This means that throughout the usage of the system, the individual subject's performance should be increasing as the system learns the patterns of the subject's EEG. Having the ability to control the system intuitively would also help the subject ease the usage and perhaps lower the potential fatigue. This would be instead of a multi-level control system in which the subject could switch control modes (up/down, open/close, supination/pronation) by "activating" a different muscle

movement, such as thinking about moving the left hand. This could put more strain on the subject, however, it could also be used to perform more complex movements without the difficulties of distinguishing, e.g., both wrist supination and pronation in the same control level by locking out one of the classes.

## 2.5 Final Problem Statement

Using BCI for rehabilitation improves the outcome for the patient, as mentioned earlier. Several systems have already been created both commercially and on a research level, and newer research has begun to use RL as a machine learning approach. RL heavily relies on learning by trial and error, and it is, therefore, necessary to create a model that works at a satisfactory level in a simulated environment before deploying it to control an external device. The benefit of using RL is the ability of the system to learn and improve while it is being used. ErrPs have been used for feedback to the system to enable it to correct misclassification and improve its accuracy. There are different approaches to how the system gets feedback, but it is important to implement a concise and accurate one to facilitate negative rewards for the system to correctly optimize towards an optimal or near optimal policy. With this and the remaining considerations discussed in the problem analysis, the following final problem statement has been formulated:

**How can an RL framework be created to improve non-invasive EEG BCI systems for upper limb rehabilitation purposes?**

## Chapter 3

# Requirements

There are several requirements when it comes to an EEG MI BCI system. Section 2.3 discusses a few of these. However, the priority for each requirement differs in how important they are to consider during development and deployment. Therefore, a MoSCoW model will be used to identify the priority of each requirement.

A system used to control a rehabilitation robot has some natural requirements that would need to be met to achieve the highest possible safety and usability. The system would need to be accurate enough to classify the correct movement continuously. This requirement would be set by saying that the system would need to classify  $x$  amount of correct movement intent within a set time frame. These classifications would need to exceed some minimum confidence before a command is sent to the equipment. The minimum confidence will be set as the upper confidence bound of the chance level as explained by Müller-Putz et al. in [50]. A deeper explanation of this will be given later. This also puts a requirement on the delay of the system, as this would need to be as low as possible to ensure a smooth user experience.

From the problem analysis, several key requirements can be pointed out. First of which are the number of electrodes, which typically range between 8 to 16 for adequate spatial information. However, using fewer electrodes can significantly reduce the time it takes to set up the electrodes and speed up preprocessing time, as the system would have a reduced data stream. The sampling frequency is typically set to 256Hz for EEG signals, but the actual lower limit is set by the Nyquist theorem, in which the sampling rate should be twice that of the highest filter used.

When creating a rehabilitation system, there will be a minimum number of movement classes that the system should be able to classify to perform a movement exercise. If, for instance, elbow movement is used as an example, you would want to be able to classify one movement for flexing the elbow joint and another for extending. It would then also be useful, for any movement exercise, to be able to classify when the subject does not intend



to move, i.e, a rest class. Therefore, the minimum number of classes is three.

1. The system should not exceed a delay of over 0.5 seconds. **M**
2. The classification exceeds a minimum accuracy of the upper confidence bound of chance level as explained by Müller-Putz[50] **M**
3. The system should, at max, use 16 electrodes while still achieving the required accuracy. **S**
4. Should use a minimum sampling frequency that satisfies the Nyquist theorem. **M**
5. Should be able to sustain continuous classification of the same intent for 200ms before changing movement execution. **S**
6. The system should be able to classify at least 3 classes of movement **M**

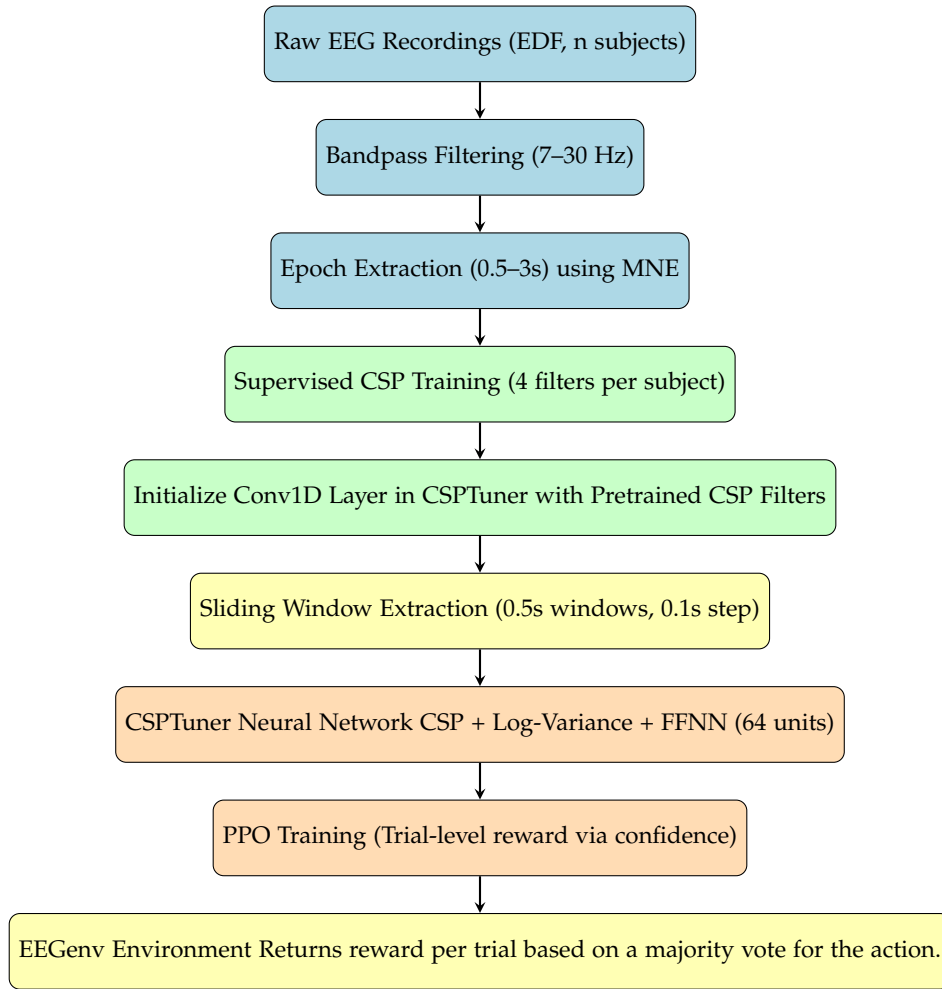
## Chapter 4

# Methods

This section will present the methods applied in creating the system, as well as the control of a robotic arm using the output of the RL framework. Firstly, a system overview will be presented. This will explain the system in its entirety, from the acquisition of the EEG signal to the control of the robotic arm in a broad overview. Then, the individual parts of the system will be explained in greater detail following the flow of figure 4.1. Lastly, a method for implementing a robotic device into the system will be presented.

### 4.1 System Overview

An overview of the training and validation pipeline for the CSP RL framework can be seen in figure 4.1. The color coding corresponds to the following: blue is data acquisition and filtering, green is CSP, orange is PPO, and yellow is the environment and output. Initially, data from  $n$  subjects are loaded in. The data is then bandpass filtered in the range of 7-30Hz which covers alpha and beta ranges. This filtering is common within EEG preprocessing and used for example in Lun et al.[51] in which they used the same dataset as this project. Epochs are then created and cropped to be between 0.5 and 3 seconds. Afterwards, the CSP is fitted to the training data, which is determined by the split and is different for individual subject testing and across subject testing. Afterwards, the filters are extracted, which the 1D convolutional layer is initialized with. The epochs are split up into windows, which are sent into the CSP Feed-Forward Neural Network (CSP FFNN), which applies the pretrained CSP filters to the incoming data via the convolutional layer. Log variance is then extracted, and the PPO training paradigm is started using the custom environment to evaluate the actions and provide feedback.



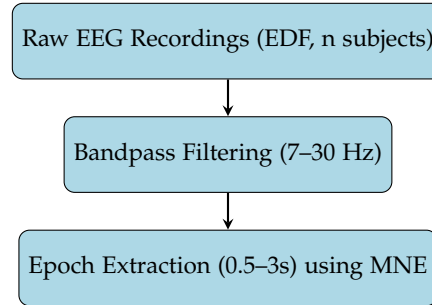
**Figure 4.1:** System overview of the CSP-based RL training pipeline using CSPTuner and PPO

#### 4.1.1 Classical CSP vs CSP FFNN

Some parts of the system deviate heavily from the classical CSP pipeline, while others stay true. The filtering is used in both and epoch extraction with an offset of 1 second from cue to avoid evoked responses are normal. Then, the CSP is trained on the epochs, and this is where the main deviations is. The filters are extracted by taking the largest and smallest eigenvalues, also the same as the classical pipeline, but then initialize a learnable convolutional layer with the pretrained filters. This layer would then be able to adjust the weights of the CSP to optimize the spatial filtering on a subject level. While typically the CSP filters would at this point be frozen and incoming data would have the filters applied, the filters become a learnable parameter of the system. The next deviation is the sliding window, where the classical pipeline would take the entire epoch and train a classifier such as LDA; instead, this breaks the epoch down into windows, which are fed one by one

into the classifier.

## 4.2 Dataset and Preprocessing



**Figure 4.2:** The data input, preprocessing, and epoch extraction of the system.

### 4.2.1 Dataset

The dataset used in this project is one created by Schalk et al. [13]. It includes over 1500 two- and one-minute EEG recordings from 109 subjects. It has four classes of MI: left and right hand MI, both hands, and both feet. The main reason for choosing this dataset is that it has been used countless times in BCI articles[52][53], it is a well-recorded dataset, and has a high amount of subjects. It is recorded using 64 EEG channels with a sampling rate of 160Hz. Previous studies [54][55] have noted that subjects 38, 88, 89, 82, 100, and 104 have wrong annotations so these will be omitted from testing and validation.

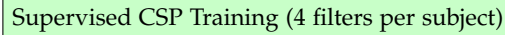
Another dataset is also worth noting by Ofner et al. in 2017[56]. It contains data from 15 subjects, in which 10 runs of data were collected from each subject. The dataset consists of six movement classes and a rest class. The six movement classes are: hand open & close, wrist supination & pronation, and elbow flexion & extension. Each class has 60 trials per subject which equates to 900 trials per class in total. The dataset was recorded with 61 channels covering parietal, central, frontal, and temporal areas using active electrodes. A Notch filter at 50Hz and an 8th-order Chebyshev bandpass filter from 0.01Hz to 200Hz were applied. The sample rate was 512Hz.

The advantages of using the first dataset presented is the quality of the recordings and the amount of subjects, which could help create a more general model. However, using Ofner et al. means that the classes are meaningful in terms of rehabilitation of the arm and to follow the notion of the neuroplasticity of the brain, which may be lost using Schalk et al. Since this is a proof of concept of whether or not RL can be used to classify EEG signals, the aforementioned will be ignored and would require further research.

### 4.2.2 Preprocessing

The preprocessing of the data starts with loading the EDF files and concatenating the runs into one raw file. The timing of the data is set to be between -1 second before cue onset to 3 seconds after. This captures both a baseline of EEG data in a resting state and the full MI execution. The data is first filtered with a Notch filter at 50Hz to filter out electrical interference, then band-pass filtered between 7-30Hz as this captures the entire range of both alpha and beta. After filtering, epochs are extracted for the relevant events, which in this case are both hands, both feet, and rest. The epochs are cropped to be between 0.5 and 3 seconds to only include the MI part of the run and to not classify evoked responses. Bad runs are rejected, and the total number of trials per subject is 63 for hands, feet movement, and rest. The class balance is originally skewed, with typically having 24 hand trials, 21 feet trials, and roughly 60 rest classes. A balancing function is used to correctly adjust for the skewness by dropping all trials to the lowest denominator, which is 21. The dropping of trials is done randomly.

## 4.3 Common Spatial Pattern



Supervised CSP Training (4 filters per subject)

**Figure 4.3:** CSP component of the training flow.

CSP is used as the feature extraction method of the EEG signal. Once the signal has been preprocessed, it is fed into the CSP to train it and extract filters that are applied to the test data. This is done in the training phase of the model. For an online adaptation, the CSP will not be fitted to new data but would instead rely on the pretrained filters. The CSP is trained on subject-specific band-passed training data. While the CSP is processing the test data on each window of the EEG data, the pretrained filters are fitted on the entire epoch data from 0.5 to 3 seconds. The CSP calculates a filter for each channel present, meaning that if all 64 sensors are used from the dataset, then the CSP calculates 64 eigenvectors containing the weights to linearly combine the channels. After calculating the vectors, they are ordered in terms of magnitude, meaning the first filters maximizes variance for one class while the last filters maximizes it for the other class. For this reason, the first  $m$  and last  $m$  filters are chosen to ensure maximum separability between the two classes. The number of  $m$  filters chosen satisfies the following  $CSP_{comp} = 2m$ , meaning that if 4 components are chosen, then it uses the first 2 and the last 2 filters. For the final model, the CSP is trained on the entire dataset created by Schalk et al. to ensure it starts with filters that are as general as possible to create a stable base for future learning. The training of the CSP can be seen in figure 4.3. For the mathematical theory of CSP, see appendix A. For this project, 4 components were chosen for the CSP. This was chosen by testing different

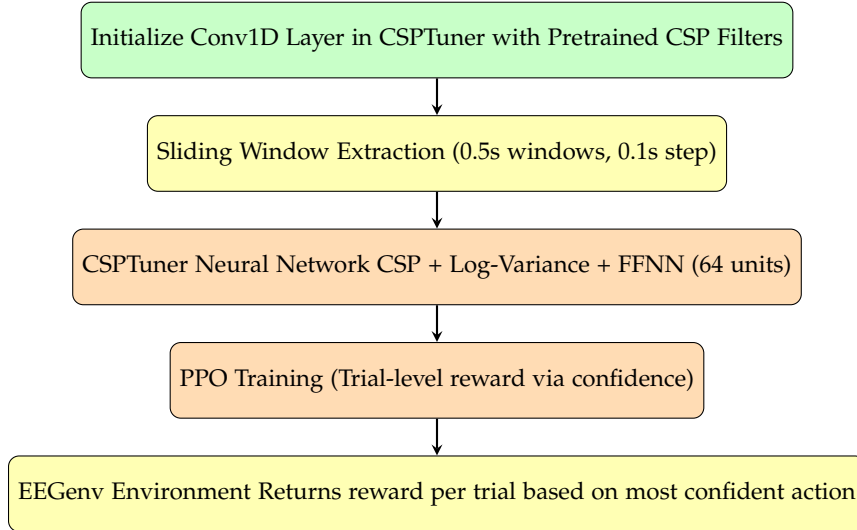
amount of components using an LDA classifier, which will be explained later. The highest accuracy was achieved by using 4 components.

#### 4.4 Linear Discriminant Analysis Classifier

To have a measure for comparison of the RL framework, the more simple LDA classifier was used. This uses the same preprocessing as the RL framework, along with a sliding window approach for classification. It utilizes the same CSP as the RL framework and with 4 components as well taking the first 2 filters and the last 2 filters. This follows the same logic as explained previously, with  $CSP_{comp} = 2m$  where  $m$  is the number of filters to use. The training and testing of the LDA is done both subject wise and across subjects by Leave-One-Out (LOO), which means it is tested on a single subject's data while trained on the remaining subjects' data.

The sliding window classification is done on the cropped epoch time between 0.5 and 3 seconds. Windows of length 0.5 seconds are used with steps of 0.1 seconds. This allows for the determination of peak accuracy for each subject, which varies in time, where some have peak accuracy between 0.7 and 1.5 seconds, and others may have between 1 and 2 seconds. For this reason, doing a sliding window approach helps locate the time at which individual subjects have the highest peak accuracy. This could mean that the classifier would need some sort of calibration to know when to trust the classification. The way that the classification rate was done in this project is to take the majority vote of the total classification of all windows. A confidence measure was also experimented with in which it would classify the data based on the most confident window prediction. Both approaches gave similar results, and the choice to go with the majority was made. This accuracy then represents the LDA's actual performance rather than having it diluted by utilizing non-informative periods of MI. This approach is also what the RL utilizes to better help learn individual patterns for subjects.

## 4.5 Reinforcement Learning Framework



**Figure 4.4:** The RL part of the pipeline including CSP, agent, training, and environment.

The RL framework has several blocks to it. These include the CSP FFNN block, training block, and environment. This part of the pipeline can be seen in figure 4.4. Firstly, PPO will be introduced and explained, focusing on the core operation: the clipped objective function. Then, the CSP FFNN will be described, how it is constructed, and how it acts as the agent of the system. Lastly, the environment, along with the sliding windowing of the EEG data and the reward feedback, will be explained.

The RL agent is validated in the same way that the LDA was trained and tested: LOO. It goes through each single subject as the test data while training on the remaining subjects' data. This is done to simulate giving the agent unknown data to evaluate the generalizability of the model. An overview of the system architecture can be seen in figure 4.5. An in-depth explanation of the architecture will be given, but key elements of PPO will be explained prior.

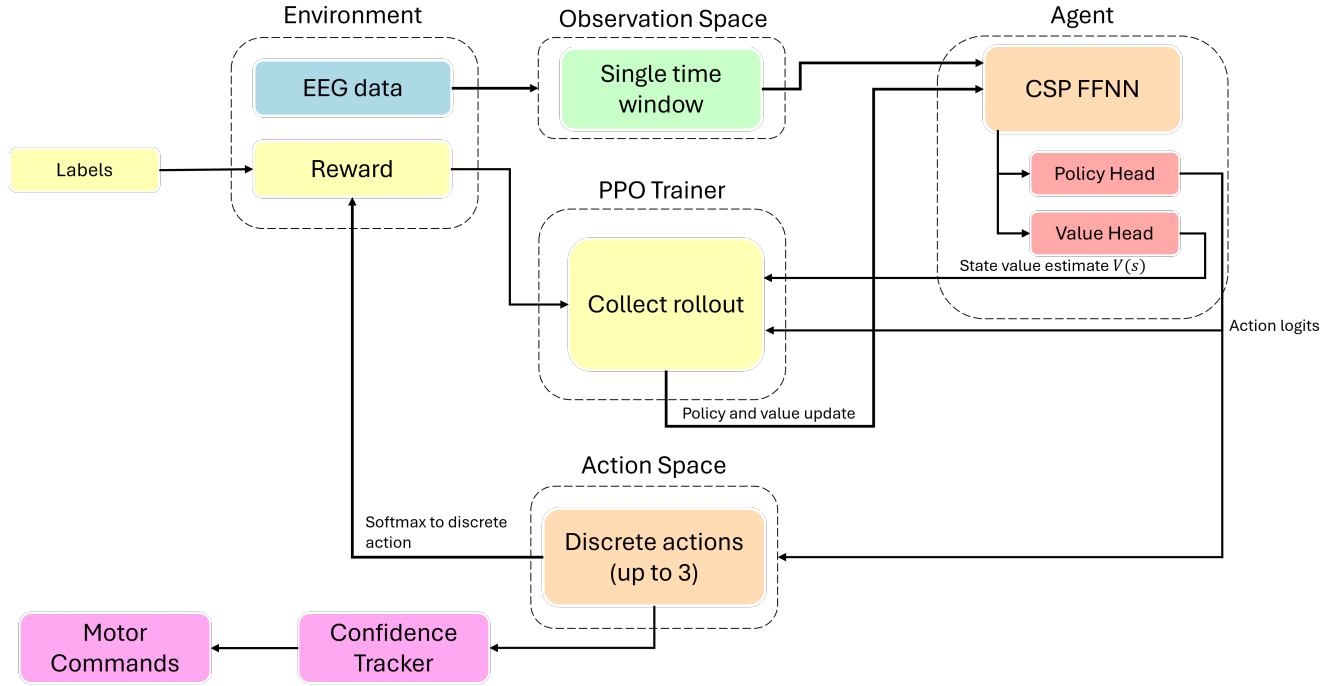


Figure 4.5: System architecture of the PPO reinforcement learning framework.

#### 4.5.1 Proximal Policy Optimization

Reinforcement learning, as stated earlier, is concerned with training an agent to choose appropriate actions that optimize or near-optimizes a policy,  $\pi(s, a)$ , which in turn ensures maximum cumulative rewards. The policy is what determines which action,  $a$ , should be chosen in regards to the current state,  $s$ . The main dilemma of RL is the exploration/-exploitation ratio. How much should the agent explore, which in this case is choosing uncertain actions based on the state, versus exploiting current knowledge, which would be choosing confident actions based on the current state. The agents actions are confined to the action space  $A$  and the states are confined to the state space  $S$ . [28]

There are three major categories of RL [57]: Value-based methods, Policy-based methods, and Actor-Critic based methods. In value-based methods, the goal is to optimize the value function such that it gives the highest cumulative reward. The value function represents a measure of how good it is for the agent to be in a given state or perform a specific action. In policy-based methods, the goal is to optimize the policy that maps states to probabilities of selecting a certain action. The actor-critic approach combines this such that the system, the actor, tries to optimize the policy while getting feedback from the critic that evaluates the value function and thereby gives an estimate of how good the chosen action was. PPO typically falls under the category of policy-based methods, but its structure is more dynamic and can therefore also be applied in an actor-critic scenario.



PPO tries to take the biggest possible improvement step on a policy without causing it to collapse by not stepping too far. It tries to achieve this by clipping in the objective function so that the new policy does not stray too far from the old policy[58]. Before explaining the objective function and the main purpose of PPO, several other key aspects needs to be explained before. The main reason for choosing PPO is exactly this. When trying to achieve adaptability of the CSP filters by allowing the model to optimize the weights, it needs to be stable enough so that the CSP filters are not altered too much. This is achieved since the policy won't take massive leaps but is clipped such that the updates are not too far from the old policy.

### Actor-Critic and Advantage

The agent in RL is the acting force of the system. This is where the policy output and the value function reside. The agent has two parts to it: the actor and the critic. The actor is the policy that determines actions based on the current state. The critic evaluates this choice and tells the agent how good that choice was. Both the actor and the critic has a shared neural network that determines the outputs. The policy network outputs are logits, which are unnormalized scores associated with each action. Softmax is applied to these logits to give probabilities which then directly translates into what action it chooses. The value network outputs are values that are a scalar estimate of the expected return from the current state. It is not probabilities or a classification but numbers such as +10 if it expects a good outcome, -10 if it expects failure, or even 0 if it is neutral or uncertain. These values are then used to compute advantage,  $A^\pi(s, a)$ , by way of[59]:

$$A^\pi(s, a) = G_t(s, a) - V(s) \quad (4.1)$$

where returns,  $G_t(s, a)$ , are

$$G_t(s, a) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (4.2)$$

In this,  $r_t$  is the reward, +1 if correct or -1 if incorrect, at time step  $t$ , and  $\gamma$  is the discount factor, which is typically 0.99. This is the discounted total future rewards an agent can expect. This advantage tells the system how much better or worse the chosen action was compared to what the value function expected, following the logic of[58]

$$if \begin{cases} A > 0 & : \text{increase probability of action } a \\ A < 0 & : \text{decrease probability of action } a \\ A \approx 0 & : \text{no significant change} \end{cases}$$

### Objective Function

There are two types of PPO: PPO-penalty and PPO-clipped[58]. PPO-penalty is not used in this project and will not be explained since the focus will be on PPO-clipped. The core

equation of PPO-clipped is how it updates the policy, which is:

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s,a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \quad (4.3)$$

In this,  $\theta$  and  $\theta_k$  are the policy parameters that are being optimized. In this project, the initial parameters are a set of different weights and biases, including the pretrained CSP filters and weights and biases for the policy head and value head. The biases are initialised as all zeroes, and the weights for the policy and value head are a uniform normal distribution using Kaiming:

$$W_{ij} \sim \mathcal{N} \left( 0, \sqrt{\frac{2}{64}} \right) \quad (4.4)$$

where  $\sqrt{\frac{2}{64}}$  is the standard deviation where 64 represents the amount of nodes[60]. This function tries to maximize the expected clipped advantage-weighted policy using data collected from the old policy,  $\pi_{\theta_k}$ . This is typically done over several mini epochs using stochastic gradient descent as an optimizer.  $s, a \sim \pi_k$  are the states and actions sampled from the old policy.  $L$  is the loss function described by

$$L(s, a, \theta_k, \theta) = \min \left( \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}(s, a)}, \text{clip} \left( \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}(s, a)} \right) \quad (4.5)$$

which can be simplified to:

$$L(s, a, \theta_k, \theta) = \min \left( \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}(s, a)}, g(\epsilon, A^{\pi_{\theta_k}(s, a)}) \right) \quad (4.6)$$

This equation can be broken down into its different parts. The first part:

$$\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}(s, a)} \quad (4.7)$$

is the ratio of the new policy versus the old policy, weighted by the advantage of the current policy. The term:

$$g(\epsilon, A^{\pi_{\theta_k}(s, a)}) \quad (4.8)$$

is a simplified version of the clip function in equation 4.5. It follows the logic of:

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & \text{if } A \geq 0 \\ (1 - \epsilon)A & \text{if } A < 0. \end{cases} \quad (4.9)$$

in which  $\epsilon$  is a clip variable that is set to 0.2, as this was the optimal value found by Schulman et al. [61], who were the ones who proposed the PPO algorithm. This means that if the advantage is positive, it is a minimum operation that limits how much the

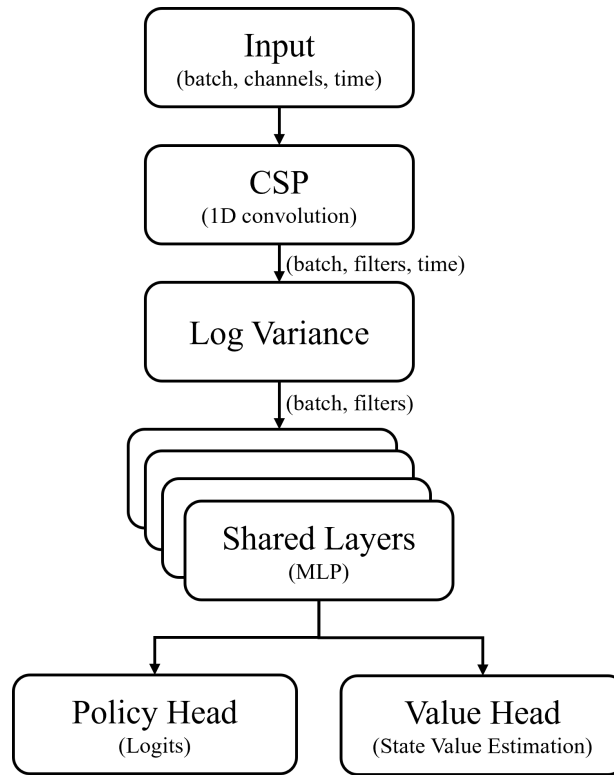
objective can increase. Since the advantage is positive, the objective function will increase if the action becomes more likely. This increase is capped by  $(1 + \epsilon)A^{\pi_{\theta_k}}(s, a)$ , which means that the policy will not benefit from straying too far from the old policy. If the advantage is negative, then the minimum operation becomes a maximum operation, and the objective will increase if the action becomes less likely. This is limited by the max operation and is capped by  $(1 - \epsilon)A^{\pi_{\theta_k}}(s, a)$ , which, just as if the advantage is positive, means that the policy will not benefit from moving too far from the old one. [58]

By using this clipped objective function, the PPO policy becomes more stable by removing incentives for the policy to change dramatically. This is the core function of PPO, which provides stable training and throughput. [58]

#### 4.5.2 CSP FFNN

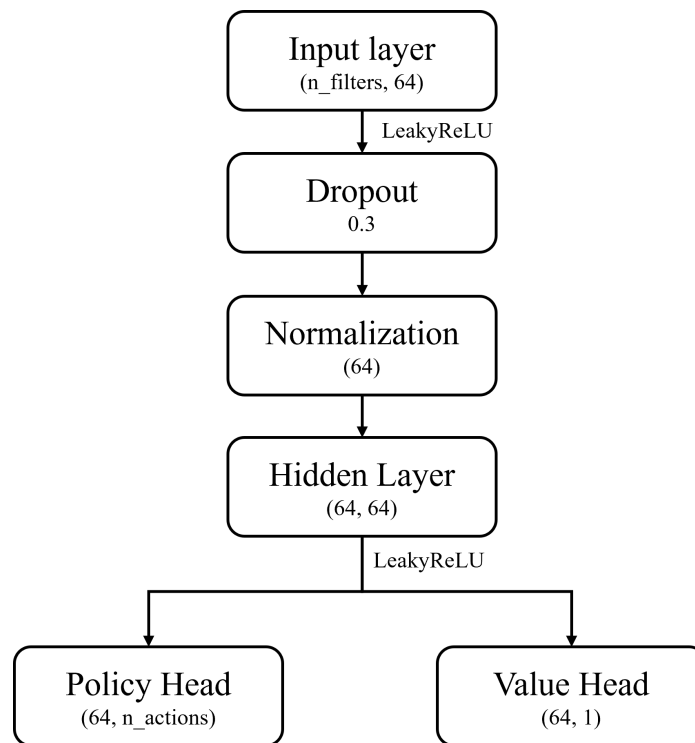
The CSP Feed-Forward Neural Network is the agent in the PPO framework. It contains the core feature extraction and classification pipeline by outputting the policy head and the value head. The CSP FFNN structure can be seen in figure 4.6. Firstly, it is initialized with pre-trained CSP filters from the test subjects. The input to the agent is windowed EEG data of size (batch, channels, time). This represents a single window of the band-passed EEG data. That is, one observation from the environment is sent as the input to the agent. It then uses a 1D convolutional layer to project the data onto the filters.

The pre-trained filters are created using  $m$  as the first and last components of the CSP filters. The number for  $m$  satisfies that  $csp_{comp} = 2m$ . In this case, the components are set to four and therefore  $m = 2$ . Section 4.3 explains the CSP used in this project. These filters can be further tuned during training by using gradient descent based on trial-level reward feedback, which is incorporated into the PPO optimizer. After the data has been spatially projected, the log of the variance over time is calculated. This is then fed into the Multilayer Perceptron shared neural network, which can be seen in figure 4.7. After the data has passed through the MLP, it is then fed into both the Policy head, which outputs logits, and the Value head, which outputs a state value estimation.



**Figure 4.6:** The entire structure of the CSP FFNN.

The MLP architecture starts with an input layer of size  $(n\_filters, 64)$ . With the CSP components set to 4, this means that the input is  $(4, 64)$ . The input of 4 is expanded to 64 dimensions so that the network can extract more abstract representations. The activation function used between the hidden layers is LeakyReLU. It is then passed through a dropout layer of 0.3 to help prevent overfitting. After dropout, it is sent into a normalization layer, which helps stabilize the hidden representation so the activations do not grow too large or too small across training. After normalization, it moves into the last hidden layer of size  $(64, 64)$ . The size of 64 is kept throughout the network to allow the network to perform deeper transformations without compressing or expanding any further. This network is shared between the policy head and the value head, which means that the input is passed into both of these layers. The policy head is of size  $(64, n\_actions)$ , compressing the representations into logits associated with each action. The value head is of size  $(64, 1)$ , compressing the representations to a single state value estimation. This entire structure can be seen in figure 4.7 as stated earlier.



**Figure 4.7:** The Multilayer Perceptron structure.

The second part of the CSP FFNN utilises the output of the policy head to acquire four distinct values essential to the training of the PPO:

- **The action:** Firstly, the function takes an input of a state. It then passes this state through the shared MLP network and then into the policy head layer. This will acquire the logits, which are turned into probabilities by applying Softmax. These probabilities are then turned into a categorical distribution. From this distribution, the four distinct values are extracted. The first of the values is the action which is directly sampled from the distribution.
- **Log probability:** Represents how confident the agent was in choosing that action in log space. This is a key value used in calculating the ratio between the old policy and the new policy used in the objective function
- **Confidence:** The raw softmax score of the chosen action and represents how confident the agent was in choosing said action. This value is a key value used in the confidence tracker that allows motor commands to be sent and will be explained further in section 4.6.
- **Entropy:** Represents the uncertainty or randomness of the action distribution. A high entropy means the agent is uncertain and is exploring, while a low entropy means

that the agent is confident and deterministic. The theoretical limit to how high the entropy can be is defined by the natural log of the number of actions, as such:

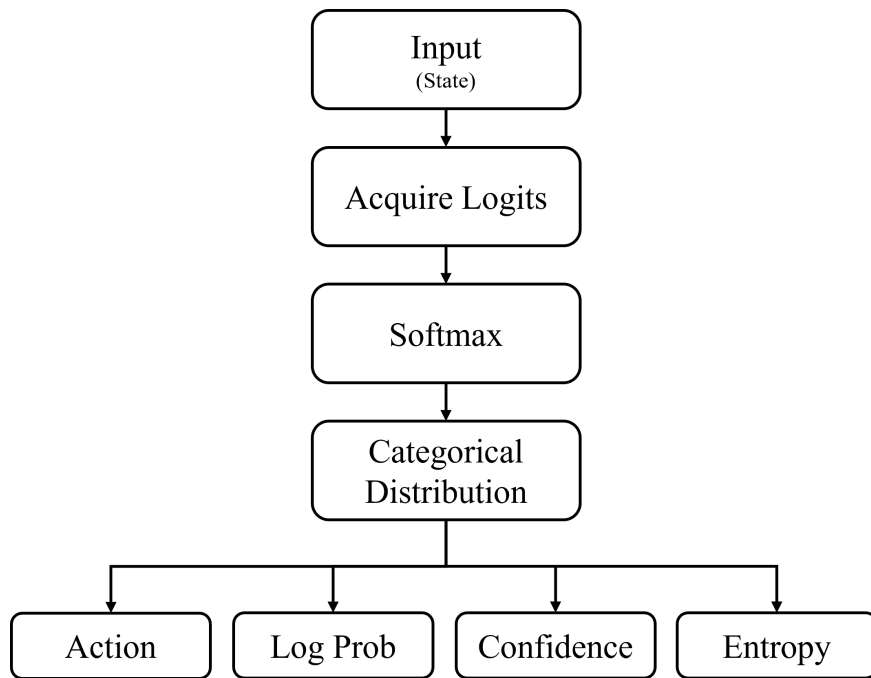
$$\text{Max entropy} = \ln(n_{\text{actions}}) = \ln(3) \approx 1.09 \quad (4.10)$$

The entropy in this system is calculated using Shannon entropy as such[62]:

$$H(\pi(a|s)) = - \sum_a p_a \log p_a \quad (4.11)$$

in which  $p_a$  is the probability of the action, and the entropy is computed per state.

The flow of how these four values are calculated can be seen in figure 4.8.



**Figure 4.8:** The flow of the sampling of action, log probability, confidence, and entropy within the CSP FFNN.

### 4.5.3 Environment

The environment is what provides the agent with observations and attaches rewards based on the chosen action of the agent. It takes an input of band-passed EEG data of shape (n\_trials, n\_channels, time\_samples) along with the associated labels of the trials. The EEG data is then broken down into windows of shape (channels, window\_size) which span 0.5s of the data with steps of 0.1s. These windows are what define the observation space of the environment. One observation is sent to the agent to be processed which corresponds to a single time window of the EEG data. The agent then performs an action

on this window. After all windows of the entire trial have been processed, the environment will give a reward based on the majority prediction across all windows. The action space is simply defined by the actions available to the agent, which in this case are hands, feet, and rest.

#### 4.5.4 System Architecture

The architecture of the PPO RL system can be seen in figure 4.5. The system consists of several components: Environment, observation- & action space, PPO agent, PPO trainer, and confidence tracker with motor commands. The general relationship between these components will be explained in this section. The colors of the boxes represent the different stages shown in figure 4.1. Initially, the environment gets the EEG data either from a dataset or online. This data is then processed and sent into the observation space, where it is divided into windows. A single time window is sent into the CSP FFNN, where the pretrained CSP filters are applied. The agent will then classify the incoming EEG data and output a policy head and a value head. The policy head is the actor of the system and will determine the logits which are sent to the rollout collection and the action space. The action space will determine the appropriate action based on softmax on the logits. The action determined is then sent to the reward function, which attaches a reward based on whether the predicted action corresponds to the true label of the action. This is heavily based on supervised learning as a way of providing negative feedback to the system. The reward system in an online deployment would not necessarily be able to rely on labels and hence would be subject to change. Methods for this will be discussed later. Furthermore, the action is also sent into a confidence tracker, which tracks the confidence of the chosen action. If the tracker gets multiple actions that surpass a threshold for confidence over a set amount of time, then the action is translated into a motor command that would control an external device.

The value head given by the CSP FFNN, which acts as the critic, returns a state value estimate. This estimate is collected in the rollout along with the reward, states, entropies, logits, and actions. This rollout is then used to calculate loss using the outputs of the policy and value head and optimize the weights of the CSP FFNN by performing back-propagation. Which means it is providing the CSP FFNN with a policy and value update to optimize the parameters of the agent using stochastic gradient descent (SGD).

#### 4.5.5 Implementation

The implementation of the PPO algorithm and CSP FFNN was done using PyTorch. It uses a custom-made environment created with gymnasium. The PPO agent has three specific actions: hands, feet, and rest. The action space of the environment defines these. They correspond to a binary classification that the agent learns to associate with each observation. The agent never sees the labels attached to each trial, which means it is

learning to choose the correct action based on the reward feedback. There are a plethora of ways to design the reward feedback. A simple way to design this is by doing a hardcoded positive and negative reward for correct and wrong actions. This is what has been utilized in this project. Specifically, the system will get a +1 reward for correctly chosen actions and a -1 for incorrectly chosen actions. A different approach could be using the confidence to shape the reward, meaning that it will achieve a higher reward for high-confidence correct guesses and an equally large negative reward for high-confidence wrong guesses. Both of these systems were tried, but no apparent change was seen between them. Setting the reward is one of the most important parts of creating a stable RL system. A wrongly adjusted reward feedback could mean that the agent learns to optimize poorly by, for example, choosing the same action every time as this still gives the highest cumulative reward despite choosing the wrong action around 50% of the time.

The reward shaping is also what encourages the exploration versus exploitation dilemma by incentivizing exploration to achieve higher confidence. This, in turn, should achieve the highest possible cumulative reward once the exploitation of current knowledge achieves confident guesses. The main difference between regular machine learning, e.g., LDA, Support Vector Machine, neural networks, etc, is exactly this: the system learns to classify the data unsupervised by optimizing its weights and parameters to achieve higher accuracy and thereby maximize the cumulative reward. Not only does the system optimize its weights and parameters, but the CSP FFNN also tries to optimize the CSP weights in the 1D convolutional layer. This is done by backpropagation of the policy loss function, in which it flows back through the policy head into the network and back into the convolutional layer. By doing this, the system optimizes the weights so that the CSP not only separates the classes better, but is optimized to achieve higher rewards and become more robust to noise.

## 4.6 Robotic Implementation

To start with the robotic implementation, the output of the RL framework has to be translated into commands to control the robotic device. In this case, as mentioned earlier, it will be commands to a robotic manipulator. Since this is a proof of concept of whether or not RL can be used as a classification pipeline, the dataset contains MI for tasks somewhat unrelated to the actual upper limb rehabilitation. That being both hands open/close and both feet. This would more than likely not be in line with the previous findings that rehabilitation is affected positively by utilizing BCI, as this engages the motor cortex and thereby enhances the ability for the brain to regain the lost motion of the affected limb. However, as a proof of concept, these MI tasks will still be translated into commands that a proper system using appropriate MI tasks would use. The targeted joint for the rehabilitation is the elbow joint, and therefore the tasks will be translated as follows:



$$if \begin{cases} out = "Hands" & : \text{Elbow flexion} \\ out = "Feet" & : \text{Elbow extension} \\ out = "Rest" & : \text{Do nothing} \end{cases}$$

Elbow flexion/extension is the abstracted version of the motor commands sent to the manipulator. The classification of the continuous signal will mimic the same logic as presented in [40]. This means that the classification will have to stay above a certain threshold. As a starting point, this threshold will be set following how Müller-Putz et al.[50] described the calculation of chance level in different class paradigms of BCI classification. They argued that the chance level is not a theoretical 50%, 33.3%, 25%, and so on for a 2-, 3-, 4-, and so on class paradigm. Instead, they have an upper bound of confidence that correlates to the number of trials used, which defines the chance level given a certain class paradigm with a certain number of trials. In this case, it will be calculated from the number of trials that a single subject contains with a perfect distribution of classes. For a given subject, they will have 21 of each class, resulting in a combined total of 63 trials. With a significance level set to  $\alpha = 0.05$ , meaning the confidence level is 95%, the upper confidence bound is 42.9%. While Müller-Putz et al. described a way to discern whether or not a BCI system performs better than randomly guessing based on its accuracy, this calculation will purely be used as a way to set the initial threshold. Since it does not directly correlate to an online scenario, the meaning of what they described is lost. However, by using it to calculate a threshold of when an equivalent offline system performs better than random, the choice of threshold has some degree of merit instead of purely going off the theoretical chance level of 33.33%. It is, however, important to note the difference between confidence and accuracy. Confidence, as explained earlier, is the raw probability that the agent associates with a certain action. It is therefore not the same as accuracy but simply a representation of how confident the agent is that that specific action is the correct one to choose given the data.

The upper confidence bound now represents the threshold for the system. Mimicking the logic of [40], the system would have to contentiously classify the input with a confidence above the threshold for 200ms for the system to recognize the MI task and send the motor command to the manipulator. The rest class, while still being a class that the RL system can classify, would in this case simply be whenever the confidence is below the threshold. The system will only perform another movement if it recognized a different output for this amount of time. This also assumes the system operates at a speed that can accommodate this. That puts a requirement on the system to perform the classification fast enough, and to have the signal sampled fast enough to get useful information extraction. The sampling frequency is difficult to determine since the quality of the signal is paramount for high accuracy. A poor signal sampled at a high rate is not going to improve the accuracy; however, if the input is windows of 1 second, then the sampling frequency would need to be sufficient to determine the correct class. The dataset used in training and validation

of the RL model as well as the LDA, uses a sampling frequency of 160Hz. Therefore, this will be used as a baseline in an online model as well. With the current setup using the sliding window technique as explained previously, the model would be able to give a prediction every 100 ms because of the step size of the windows. This means that the first prediction would be after the first window, 0-0.5 seconds, then the next prediction would be 100 ms after using the window of 0.1-0.6 seconds and so on. Assuming the classification is done within this time step, it would take two consecutive correct classifications before the motor command is sent to the manipulator. The initial latency of the system would therefore be 0.7 seconds since its taking in windows of 0.5 seconds and waiting for correct classification for 0.2 seconds. This could be noticeable in the beginning, but should not be felt with continuous data flow.

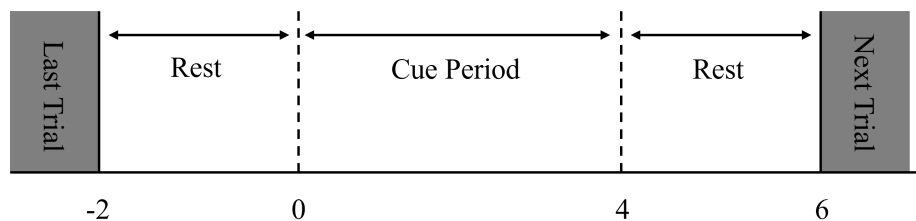
Safety constraints would also need to be considered since the rehabilitation robot should not be able to either over-extend or over-flex the elbow, for example. If, for some reason, the system classifies two or more consecutive elbow flex signals but the subject is trying to extend, then at some point, the system should stop moving to not cause harm to the subject. The workspace of the robot would more than likely have to be tailored to each subject.

## Chapter 5

# Testing

### 5.1 Data

The data used for testing is the first 14 subjects from the dataset created by Schalk et al.[13]. Each subject did three runs of MI with roughly 7 trials per class, and the total number of trials per class is downsampled to have 21 trials of each of the three used MI classes. This equates to 63 trials per subject, meaning a total of 854 trials across all 14 subjects. The reason for choosing the first 14 instead of using more was to get an idea of the individual subject performance without having too many subjects and to have a non-biased ground truth for performance. This means that the 14 subjects were not selected because of the quality or any other reason than simply being the first 14 subjects. This was simply carried over into across subject testing, and because of time limitations, an extended test was not done. Each trial is constructed in the way that is shown in figure 5.1. It starts at -2 seconds with a rest period until the cue is shown at 0 seconds. From 0 to 4 seconds, the subject performs an MI task as instructed by the cue, after which, at 4 seconds, another rest period begins for 2 seconds. Rest is not a specific task that the subjects performed; however, because of the paradigm they used, there are essentially 4 seconds of rest before the MI task. The rest class was therefore extracted from these 4 seconds and matched in length to MI task epochs of 2.5 seconds.



**Figure 5.1:** Timeline of a single trial.

## 5.2 Testing Process

The testing process is split up into two sections: Individual and across-subject testing. The testing is both for validation of the model and to see how the model classifies unknown but limited data. The intra-subject data from the dataset[13] amounts to roughly 21 trials of each class, which may not be enough to notice a change in the model i.e., there is not enough data for the model to calibrate/learn and adjust the CSP filters to unknown data. Both the LDA and the RL are tested subject-wise and across subjects to get a comparison between them.

### 5.2.1 Individual Subject Testing

The method used for individual subject testing is to split up the 3 available runs for each subject into a test set and a train set. 2 runs were selected as training and 1 was testing. This goes through each subject 3 times as a run-wise cross-validation. The mean of this accuracy then represents the accuracy achieved by the model. By doing this, it provides a more general accuracy of the model that better prevents the odds of bias towards runs that has better quality recordings. At first, a shuffle split with 10 folds was used, but such a split did not work well with the RL model implementation. Therefore, the change was made to use run-wise cross-validation.

### 5.2.2 Across Subject testing

The across-subject testing and validation were done by subject-wise cross-validation using the leave-one-out (LOO) method. This means that the model is trained on all available data (13 out of the 14 subjects for reasons stated above), and the last subject is used for testing. This gives an idea of how the model handles unknown data, and for the RL specifically, showcases how it would potentially handle a new user using the system. The LOO goes through every subject as a test set and utilizes the remaining as training. The benefit of doing an across-subject testing is to showcase a more accurate representation of the model's performance. Relying only on intra-subject testing reflects a model that is purposefully built for each subject which does not reflect a deployed version of that model. Across-subject testing simulates a more real deployment in which the model is trained on known data and then gets introduced to unknown data to reflect the generalizability of the model.

## Chapter 6

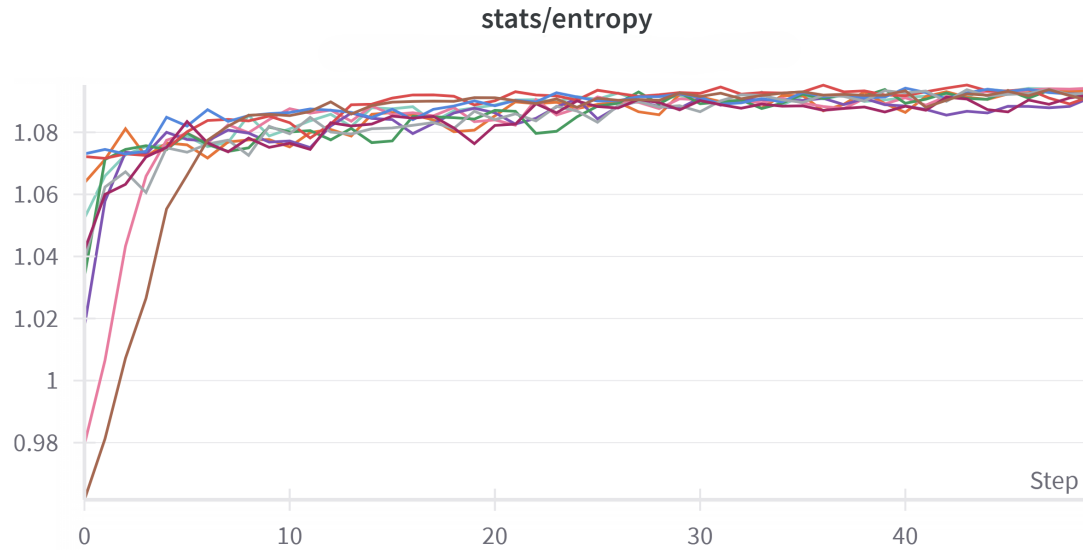
# Results

The final results of both subject-wise and across-subject training and testing are shown in figure 6.1. The LDA achieved peak accuracies of 79% for single subject and 52% for LOO. It had a mean accuracy of  $51\% \pm 13\%$  for single subject and  $38\% \pm 7\%$  for LOO. The RL model achieved a peak accuracy of 41% for single subject and 41% for LOO. Mean accuracies are  $32\% \pm 9\%$  for single subject and  $33\% \pm 5\%$  for LOO. The theoretical chance level for all of these tests is 33%, however, using Müller-Putz et al.[50] as explained earlier gives an upper confidence bound of 42.9%.

The entropy of the system was logged for all subjects and can be seen in figure 6.1. It is clear to see that throughout the 50 epochs/steps that the RL takes, the entropy increases to the theoretical maximum.

Subject	LDA Individual Accuracy	RL Individual Accuracy	LDA LOO	RL LOO
1	68%	37%	48%	32%
2	54%	30%	43%	35%
3	48%	27%	27%	38%
4	68%	32%	45%	32%
5	43%	27%	33%	33%
6	41%	30%	32%	30%
7	79%	38%	52%	38%
8	46%	25%	42%	36%
9	27%	40%	38%	41%
10	44%	29%	39%	29%
11	35%	27%	33%	32%
12	54%	41%	38%	20%
13	48%	38%	38%	38%
14	52%	32%	25%	32%
Mean	51% $\pm$ 13%	32% $\pm$ 9%	38% $\pm$ 7%	33% $\pm$ 5%

**Table 6.1:** Results of the testing of LDA and RL as a 3-class (hands open/close, feet, and rest) classifier for EEG MI BCI. Subjects 1 through 14 were used with a run-wise cross-validation for individual accuracy and leave-one-out for across-subject accuracy.



**Figure 6.1:** Entropy for each of the 14 individual subjects over 50 epochs(steps). Each line represents a single subject.

## 6.1 Confusion Matrices

Below are the confusion matrices from the two types of testing for both the LDA and the RL model.

### Single Subject Testing

The LDA confusion matrix, which can be seen in figure 6.2, shows a clear diagonal of correct predictions. In total, there are 294 trials of each class, meaning it correctly classified the following: 61% of hands-, 46% of feet-, and 45% of rest class. The RL confusion matrix for single subject testing can be seen in figure 6.3. The RL correctly classified the following: 40% of hands-, 28% of feet-, and 29% of rest class. The RL matrix shows a clear bias towards the hands class and an almost 50/50 split between the feet and the rest class.

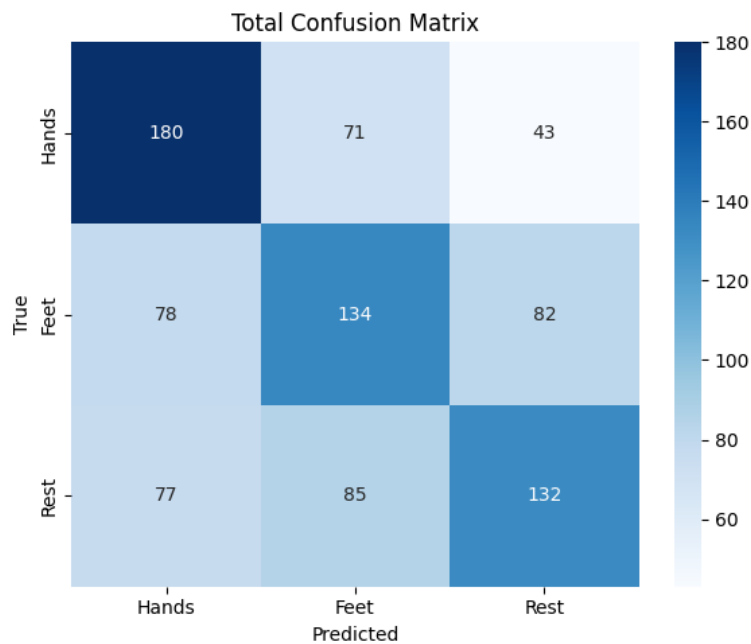


Figure 6.2: Total confusion matrix for single subject testing of the LDA.

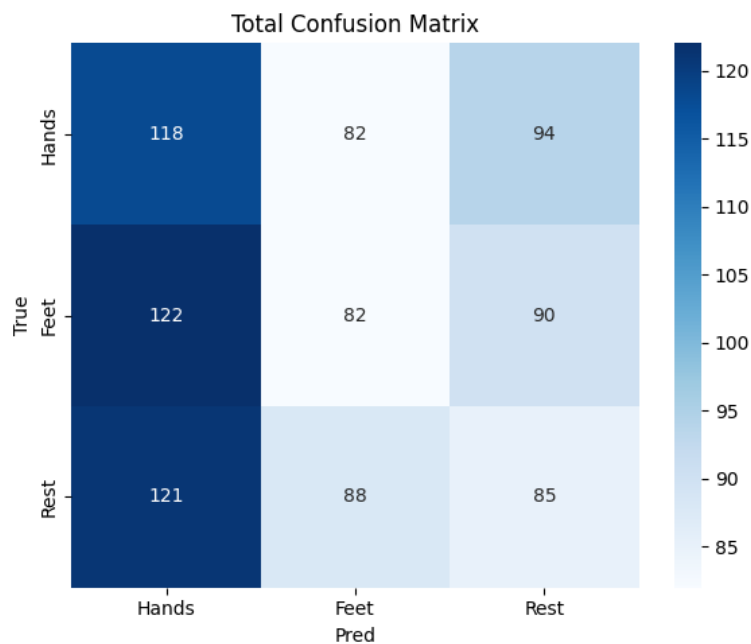


Figure 6.3: Total Confusion Matrix for single subject testing of the RL.



LOO testing

The confusion matrix for the LDA LOO test can be seen in figure 6.4. There are a total of 294 trials for each class which means it correctly classified the following: 58% of hands-, 25% of feet-, and 35% of rest class. The RL LOO test confusion matrix can be seen in figure 6.5. It shows a bias towards the hands and feet class while rest have a substantially lower amount of predictions. The RL correctly classified the following: 47% of hands-, 41% of feet-, and 15% of the rest class.

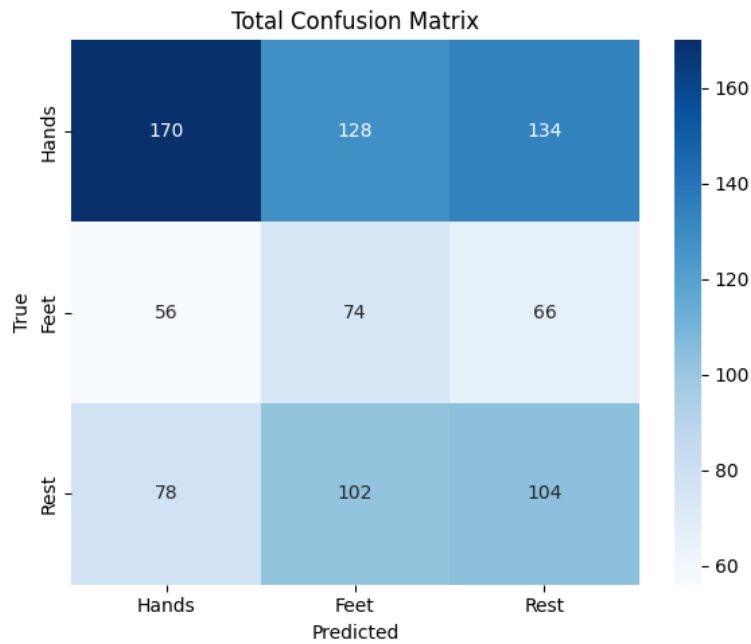
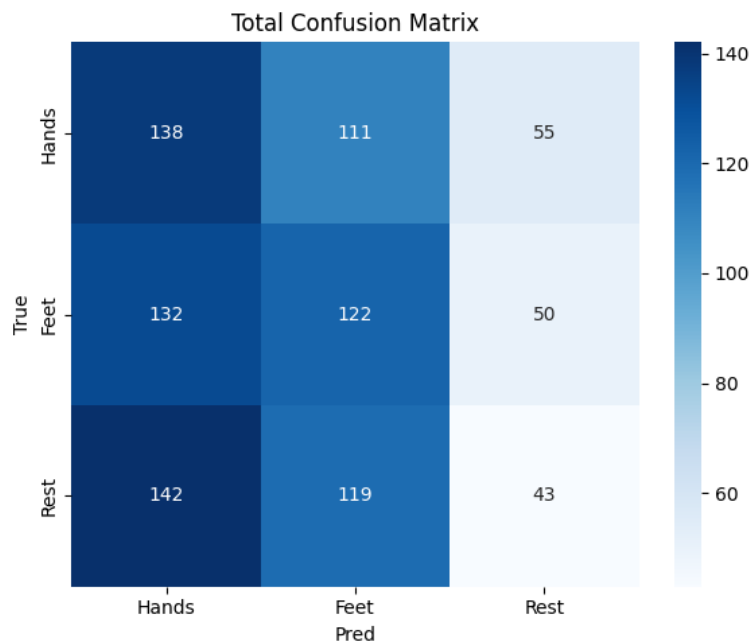


Figure 6.4: The total confusion matrix for LDA LOO.



**Figure 6.5:** The total confusion matrix for the RL model with LOO.

### Summary of Confusion Matrices

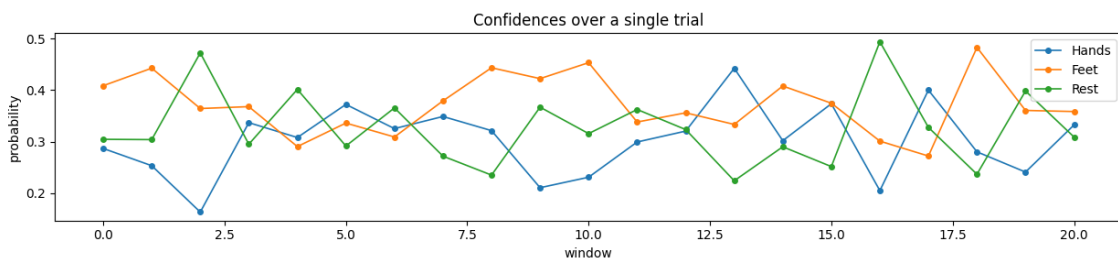
The summary for all correctly classified classes can be seen in table 6.2. The table shows the discrepancy between the three classes and the two cross-validation techniques.

Class	LDA Single Subject	RL Single Subject	LDA LOO	RL LOO
Hands	61%	40%	58%	47%
Feet	46%	28%	25%	41%
Rest	45%	29%	35%	15%

**Table 6.2:** Summary of the amount of correctly classified classes for the LDA and RL for both single subject testing and LOO testing

## Predictions Over Time

A visual graph of the probabilities of the three classes over each window of a random trial in single-subject evaluation can be seen in figure 6.6. This shows the varying nature of the probabilities that the RL agent outputs at each window of a trial. The highest probability is around 0.5, while the lowest is around 0.15.



**Figure 6.6:** A random evaluation trial in which the probabilities of each of the three classes are plotted over time, showcasing the varying nature of predictions in a single trial. Each marker represents a window prediction of the three classes.

## Chapter 7

# Discussion

From the results, specifically table 6.1, it is clear that the LDA outperforms the RL in terms of pure accuracy for both cross-validation tests, with the smallest difference being between the LOO tests. The RL model lies on the theoretical chance level of 33% in both single subject and LOO. This was more than likely to do with the high entropy through every test as seen in figure 6.1. This means that the model essentially collapsed into a random guessing model by only doing exploration and not exploitation. The confusion matrix for the RL shows a clear bias towards hands in single subject and both hands and feet in the LOO test. While the model is more evenly distributed in the single-subject test, it becomes more deterministic in the LOO test. This means that for the single subject test, it acts more like a random guess classifier, while in the LOO test, it has more than likely over-fitted to the hands and feet class.

To prevent over-fitting, both balancing of the classes and shuffling were introduced, but had minimal impact on the model besides preventing over-fitting early. Different entropy coefficients were tested in the loss function to try and sway the model into becoming deterministic and push it into exploitation rather than exploration. However, no matter the coefficient value, the model kept being random. Even removing the incentive to explore had no effect. This more than likely means that the policy could not learn enough from the data to distinguish between the classes.

The main feature extraction relied on CSP, and it enables the PPO agent to optimize the CSP weights under training to further enhance the spatial filters depending on the data. This, in theory, should allow the agent to further optimize the variance between the classes. However, it seemed not to affect either keeping the entropy low or ensuring that the agent learns to distinguish the classes better. By making use of windows as the observation to the RL agent, there was a possibility that the agent could wrongly optimize the CSP filters at each step, since the most discriminant window/windows could lie anywhere within the trial. To circumvent this, the agent only optimizes its parameters after a full trial.

The heavy focus in this project is on the adaptable CSP layer of the agent. The main reason for focusing on CSP is the vast amount of research that has gone into it. It is a proven method used countless times in BCI research. However, most applications of CSP have the weights frozen after training. One article in which the CSP weights are not frozen and can be optimized with gradient descent is Jiang et al. [63], in which they found that a trainable CSP layer improved the accuracy of their CNN. However, to the best of the authors' knowledge, combining a trainable CSP layer with RL is a novel approach. Having a CSP layer that can optimize to each subject that uses the system could mean higher accuracy as the filters would be calibrated to that specific person.

If the results of this project are compared to other research using the same dataset but different approaches, it is clear to see that the model is underperforming. Dose et al.[52] built a CNN to classify the data in both 2-, 3-, and 4 classes. They achieved, respectively, 80.38%, 69.82%, and 58.58% on a 5-fold cross-validation. Another article by Wang et al.[53] built an improved EEGNet-based model, which was the one that Dose et al. created, that achieved accuracies of 82.49%, 75.07%, and 65.07% respectively as well. A slightly different approach combining CNN with Gated Recurrent Units (GRU) was made by Bouchane et al.[64] in which they achieved an overall average accuracy of 98.69% for 4 classes, in which they also included a baseline class. This was done intra-subject using a 10-fold cross-validation. In general for studies that uses this dataset including the ones mentioned here and others[54][65][66], do not include a rest class except Wang et al. which includes a "baseline" class as well. This could be because the rest class is not a specific task they include in the dataset, but is only defined by the subject relaxing between tasks. This may mean that the rest class is not a particularly suited class to use for classification since there may not be as much control as in the MI tasks. That could result in the rest classes containing a lot of noise and or artifacts if the subject is only instructed to "relax" as the dataset description says.

One of the reasons for the RL having a difficult time learning the signals could be the difference in quality between the classes. Both the LDA and the RL show that they can distinguish the hands class better than the other two, as seen in table 6.2, which summarizes the confusion matrix accuracies for all four tests. This could lead to over-fitting and the model becoming overconfident about a certain action, as it may believe that it can achieve the highest cumulative reward by only choosing hands, or at least mainly choosing hands.

While PPO works on both continuous and discrete action spaces, it may not be the most appropriate algorithm to use when working with continuous time-series data such as EEG in a classification setting. Other algorithms, such as Deep Q Network (DQN)[67], may be a better fit. However, there is promise in using PPO as a classifier when looking at the confusion matrices, but it needs more hyperparameter tuning so that the accuracy can increase. Therefore, further research is needed to validate if PPO is a viable method to use

when working with MI EEG signals in a BCI scenario.

The sliding window approach posed several problems. While it is a method that closely resembles how an online system would classify continuous signals, using it as a way of training the PPO caused issues with how the training and optimization happened. Designing the system around this idea was therefore a difficult task, which could explain the performance of the RL model. Several different ways of designing the reward scaling were tried including basing it off confidence of the model, awarding rewards on a window scale instead of trial, terminal rewards i.e. a final set score when the trial is finished, majority voting of all the windows, averaging the highest confident guess of each window and choosing the highest as the action, but they all fell short of improving the model. The underlying issue is more than likely how the optimization of the PPO is done in conjunction with action at every step. PPO, and RL algorithms in general, expect a reward for each chosen action and then optimizes its parameters based on the reward gotten from that action. However, by splitting the EEG signals into windows, this paradigm gets broken since either: the PPO agent takes an action in each window but only gets a reward at the end based off the final estimated action of that trial or the PPO agent receives a reward at each action taken within a trial but may then end up optimizing poorly if the reward scaling is based on a boolean function i.e. true or false action in terms of correct label. Both reward strategies have limitations. If the PPO agent receives a reward only at the end of a trial, it cannot identify which individual actions were effective. The agent learns only that at least one action during the trial contributed positively, but not precisely when or how this occurred within the signal. Conversely, if the agent receives rewards for each action, it might optimize inefficiently. This is because it only learns that a particular action was incorrect, without deeper context about why the action failed or how it impacts subsequent decisions. This lack of detailed context can negatively affect CSP filter optimization. Specifically, the agent might update filters based solely on the immediate incorrect action without a clear indication of how exactly these filters should be optimized. For example, if the trial's correct label is "hands," but the agent incorrectly classifies it as "rest," the CSP filters might update in a suboptimal manner. In this scenario, filters intended for the "rest" state could inadvertently become overly discriminative against other classes, rather than specifically enhancing discriminability for the correct class, "hands."

When looking at the requirements presented in chapter 3, it is clear that some of these requirements are not met. The first requirement states that the delay should not be more than 0.5 seconds. Because of the design, an immediate delay of 0.5 seconds in the very start of acquiring the data is introduced, after which at least 200 ms passes before a motor command would be able to be sent. The current computation time of a single trial during evaluation is on average 0.04 seconds. It has 100 ms to output an action, and it is therefore feasible that this requirement is upheld. The next requirement is to achieve a classification accuracy above the upper confidence bound of the chance level. This requirement has not been met by the RL in both subject specific testing and LOO. Testing and validation was

done with all 64 electrodes for both LDA and the RL. Testing was also done with fewer electrodes, but accuracy scaled down with the number of electrodes used, and therefore, for proof of concept, it was chosen to use all 64 electrodes. The sample rate of the data set was 160Hz, and the highest filter used was a 50Hz Notch filter and therefore, the Nyquist theorem is upheld. Online testing was not done, meaning that requirement number 5 could not be tested. The last requirement is upheld in that the model can classify, to varying degrees, three different classes of MI movement.

## Chapter 8

# Future Work

The main part of the future work on this model is enhancing it further and performing online testing to see the validity and usability of the system. Currently, the model is underperforming in terms of the calculated upper bound for chance level, meaning it does not perform better than randomly guessing. More work needs to be put into expanding the agent and refining the structure of the system to ensure higher accuracy. The feature extraction is more than likely too sparse for the agent to learn anything useful from the signals, therefore, it could be expanded such that the feature extraction becomes more complex and exhaustive to maximize the distinguishability between classes. This could include having more than a single convolutional layer to extract even deeper and more complex spatial patterns. Including more features such as Power Spectral Density (PSD) and statistical features such as kurtosis, skewness, and root mean square. Providing more features for the agent to learn from could enhance its ability to correctly classify EEG signals. With extra features, it could then be useful to also introduce dimensionality reduction to enhance dominant features by using layers such as max and average pooling. However, creating a complex model also means that the computation time increases and could be very hardware dependent, as there could be more parameters to optimize during online usage.

While improving the accuracy of the model by reformatting the agent and the network would be an essential part of further research, online testing would need to be done to evaluate how effective such a system could be. For online testing, there are some key considerations to take into account: Filtering of the incoming signal, pre-processing, and handling how the optimization of the model is being done. However, firstly, the data needs to be acquired. For the data acquisition, an OpenBCI 16-electrode cap with a Cyton + Daisy board would be used. The data would be filtered in real time as the EEG signal gets acquired and would then follow the time windows as explained in section 4.6. The data would then go into the model for classification, and at this point, there is an important consideration to be made. The CSP filters are being optimized, and the whole point of having the agent be able to optimize them is to make sure they fit the new unknown



subject. However, there has to be a method to prevent the CSP filters from being optimized when the model is not receiving specific class data. That means that the CSP filters have to be frozen unless the subject is trying to imagine a movement. Otherwise, they may drift out of control from random noise and idleness in the signal. Furthermore, a feedback design would have to be made to ensure that the model receives appropriate feedback on whether or not it is classified correctly. One way this could be done is simply by having a button that the user can press when the system misclassifies something. Another, arguably more difficult, method could be to use Error-related Potentials, ErrP's, which are neurophysiological signals that the brain can exhibit when noticing an error[68].

The reward scheme and the optimization as discussed in chapter 7, would also require further enhancement. Either by moving away from the sliding window when training the RL, or by enhancing how the system works and optimizes based on the sliding windows. Another approach could be to simply use the entire epoch of a trial as one observation instead of breaking it down into windows. However, if this approach is done, it is uncertain how the model would react in online mode where continuous data would flow into the system.

There are a lot of room for improvements, and as this is a novel approach, there has not been a lot of research done using this method. Therefore, the current system provides a proof of concept and lays out the ground works for further research.

## Chapter 9

# Conclusion

This project tried to answer the problem statement, which read the following: How can an RL framework be created to improve non-invasive EEG BCI systems for upper limb rehabilitation purposes? To answer that, a novel reinforcement learning framework using PPO and a CSP FFNN was created. This system had the capability of classifying three MI tasks. The reason for developing such a system was to enhance the usability of an online BCI system, in which the system can adapt to the user while they are using the system, instead of having long calibration sessions before use. The model achieved accuracies of 40% for hands-, 28% for feet-, and 29% for the rest class while performing single subject testing and training. For across-subject training, it achieved 45% for hands-, 40% for feet-, and 14% for rest. This is above the theoretical 33% chance level but not above the upper confidence bound of 42.9% for a 3-class paradigm. Several issues arose concerning the structure of the system, and further research into both how the optimization of the system is done and the data structure presented to the model needs to be conducted. In conclusion, this novel approach showcases a new way of adapting BCI systems to individual users and lays the groundwork for future improvements and research.

# Bibliography

- [1] WHO. *Stroke, Cerebrovascular accident*. [https://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html?utm\\_source=chatgpt.com](https://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html?utm_source=chatgpt.com). Accessed: 27/11/2024.
- [2] Charles Lo et al. "Functional priorities in persons with spinal cord injury: using discrete choice experiments to determine preferences". In: *Journal of neurotrauma* 33.21 (2016), pp. 1958–1968.
- [3] Alex Pollock et al. "Interventions for improving upper limb function after stroke". In: *Cochrane Database of Systematic Reviews* 11 (2014).
- [4] Lisa A Simpson et al. "The health and life priorities of individuals with spinal cord injury: a systematic review". In: *Journal of neurotrauma* 29.8 (2012), pp. 1548–1555.
- [5] Emily L. Lawrence et al. "Quantification of Dexterity as the Dynamical Regulation of Instabilities: Comparisons Across Gender, Age, and Disease". In: *Frontiers in Neurology* Volume 5 - 2014 (2014). ISSN: 1664-2295. DOI: 10.3389/fneur.2014.00053. URL: <https://www.frontiersin.org/journals/neurology/articles/10.3389/fneur.2014.00053>.
- [6] Jing Huang et al. "Effects of physical therapy-based rehabilitation on recovery of upper limb motor function after stroke in adults: a systematic review and meta-analysis of randomized controlled trials". In: *Annals of palliative medicine* 11.2 (2022), pp. 52131–52531.
- [7] Alexa B Keeling et al. "Robot enhanced stroke therapy optimizes rehabilitation (RESTORE): a pilot study". In: *Journal of neuroengineering and rehabilitation* 18 (2021), pp. 1–16.
- [8] Paul Dominick E. Baniqued et al. "Brain-computer interface robotics for hand rehabilitation after stroke: a systematic review". In: *Journal of neuroengineering and rehabilitation* 18.1 (2021), pp. 15–15. ISSN: 1743-0003.
- [9] Marc Jeannerod. "Neural Simulation of Action: A Unifying Mechanism for Motor Cognition". In: *NeuroImage* 14.1 (2001), S103–S109. ISSN: 1053-8119. DOI: <https://doi.org/10.1006/nimg.2001.0832>. URL: <https://www.sciencedirect.com/science/article/pii/S1053811901908328>.

- [10] Kaido Värbu, Naveed Muhammad, and Yar Muhammad. "Past, present, and future of EEG-based BCI applications". In: *Sensors* 22.9 (2022), p. 3331.
- [11] Learning EEG. *Montage and Technicalities*. <https://www.learningeeg.com/montages-and-technical-componentsm>. Accessed: 30-05-2025.
- [12] Learning EEG. *Artifacts*. <https://www.learningeeg.com/artifacts>. Accessed: 03-06-2025.
- [13] Gerwin Schalk et al. "BCI2000: a general-purpose brain-computer interface (BCI) system". In: *IEEE Transactions on biomedical engineering* 51.6 (2004), pp. 1034–1043.
- [14] S Olsen et al. *Electroencephalographic recording of the movement-related cortical potential in ecologically valid movements: a scoping review*, *Front. Neurosci.* 15 (2021) 721387. 2021.
- [15] Andrej Savić et al. "Movement related cortical potentials and sensory motor rhythms during self initiated and cued movements". In: *Replace, Repair, Restore, Relieve—Bridging Clinical and Engineering Solutions in Neurorehabilitation: Proceedings of the 2nd International Conference on NeuroRehabilitation (ICNR2014), Aalborg, 24-26 June, 2014*. Springer. 2014, pp. 701–707.
- [16] Natasha M. J. Padfield et al. "EEG-Based Brain-Computer Interfaces Using Motor-Imagery: Techniques and Challenges". In: *Sensors (Basel, Switzerland)* 19 (2019). URL: <https://api.semanticscholar.org/CorpusID:85508474>.
- [17] Paul Dominick E Baniqued et al. "Brain–computer interface robotics for hand rehabilitation after stroke: a systematic review". In: *Journal of neuroengineering and rehabilitation* 18 (2021), pp. 1–25.
- [18] P Pavithara et al. "Implementation of EEG Signal Decomposition and Feature Extraction Through Efficient Wavelet Transforms". In: *2024 International Conference on Communication, Computing and Internet of Things (IC3IoT)*. 2024, pp. 1–6. DOI: 10.1109/IC3IoT60841.2024.10550231.
- [19] Mera Kartika Delimayanti et al. "Classification of brainwaves for sleep stages by high-dimensional FFT features from EEG signals". In: *Applied Sciences* 10.5 (2020), p. 1797.
- [20] Jian Shen et al. "Exploring the Intrinsic Features of EEG Signals via Empirical Mode Decomposition for Depression Recognition". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 31 (2023), pp. 356–365. DOI: 10.1109/TNSRE.2022.3221962.
- [21] Chungsong Kim et al. "An effective feature extraction method by power spectral density of EEG signal for 2-class motor imagery-based BCI". In: *Medical & biological engineering & computing* 56 (2018), pp. 1645–1658.

- [22] Pawan and Rohtash Dhiman. "Machine learning techniques for electroencephalogram based brain-computer interface: A systematic literature review". In: *Measurement: Sensors* 28 (2023), p. 100823. ISSN: 2665-9174. DOI: <https://doi.org/10.1016/j.measen.2023.100823>. URL: <https://www.sciencedirect.com/science/article/pii/S2665917423001599>.
- [23] Murside Degirmenci et al. "EEG channel and feature investigation in binary and multiple motor imagery task predictions". In: *Frontiers in Human Neuroscience* Volume 18 - 2024 (2024). ISSN: 1662-5161. DOI: 10.3389/fnhum.2024.1525139. URL: <https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2024.1525139>.
- [24] Juan José Escobar et al. "A distributed and energy-efficient KNN for EEG classification with dynamic money-saving policy in heterogeneous clusters". In: *Computing* 105.11 (2023), pp. 2487–2510.
- [25] Shizhe Wu et al. "Adaptive LDA classifier enhances real-time control of an EEG brain-computer interface for decoding imagined syllables". In: *Brain Sciences* 14.3 (2024), p. 196.
- [26] Eliana M Dos Santos, Rodrigo San-Martin, and Francisco J Fraga. "Comparison of subject-independent and subject-specific EEG-based BCI using LDA and SVM classifiers". In: *Medical & biological engineering & computing* 61.3 (2023), pp. 835–845.
- [27] Anupam Yadav et al. "Enhancing convolutional neural networks in electroencephalogram driver drowsiness detection using human inspired optimizers". In: *Scientific Reports* 15.1 (2025), p. 10842.
- [28] Zhiqing Xiao. *Reinforcement Learning : Theory and Python Implementation*. eng. First edition. Singapore: Springer, 2024. ISBN: 981-19-4933-6.
- [29] Sajjad Afrakhteh et al. "Accurate classification of EEG signals using neural networks trained by hybrid population-physic-based algorithm". In: *International Journal of Automation and Computing* 17.1 (2020), pp. 108–122.
- [30] G. S. Sagee and S. Hema. "EEG feature extraction and classification in multiclass multiuser motor imagery brain computer interface using Bayesian Network and ANN". In: *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICT)*. 2017, pp. 938–943. DOI: 10.1109/ICICT1.2017.8342691.
- [31] Navneet Tibrewal, Nikki Leeuwis, and Maryam Alimardani. "Classification of Motor Imagery EEG Using Deep Learning Increases Performance in Inefficient BCI Users". In: *PLOS ONE* 17.7 (2022), e0268880. DOI: 10.1371/journal.pone.0268880.
- [32] SKS Ferreira, AS Silveira, and A Pereira. "Eeg-based motor imagery classification using multilayer perceptron neural network". In: *Brazilian Congress on Biomedical Engineering*. Springer. 2020, pp. 1873–1878.

- [33] Scott A Roset, Hernán F Gonzalez, and Justin C Sanchez. "Development of an EEG based reinforcement learning brain-computer interface system for rehabilitation". In: *Conf Proc IEEE Eng Med Biol Soc.* 2013, pp. 1563–6.
- [34] Dong-Hee Shin et al. "MARS: Multiagent Reinforcement Learning for Spatial—Spectral and Temporal Feature Selection in EEG-Based BCI". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54.5 (2024), pp. 3084–3096. DOI: 10.1109/TSMC.2024.3355101.
- [35] Mathias Vukelić et al. "Combining brain-computer interfaces with deep reinforcement learning for robot training: a feasibility study in a simulation environment". In: *Frontiers in neuroergonomics* 4 (2023), p. 1274730.
- [36] Sriram VC Nallani and Gautham Ramachandran. "Rleegnet: Integrating brain-computer interfaces with adaptive ai for intuitive responsiveness and high-accuracy motor imagery classification". In: *arXiv preprint arXiv:2402.09465* (2024).
- [37] Neurolutions. *IpsiHand*. <https://www.neurolutions.com>. Accessed: 04-03-2025. 2024.
- [38] Nabi Rustamov et al. "IpsiHand brain-computer interface therapy induces broad upper extremity motor recovery in chronic stroke". In: *medRxiv* (2023), pp. 2023–08.
- [39] Jessica Cantillo-Negrete et al. "A comprehensive guide to BCI-based stroke neurorehabilitation interventions". In: *MethodsX* 11 (2023), p. 102452.
- [40] Ander Ramos-Murguialday et al. "Brain-machine interface in chronic stroke rehabilitation: a controlled study". In: *Annals of neurology* 74.1 (2013), pp. 100–108.
- [41] Akim Kapsalyamov et al. "Brain-computer interface and assist-as-needed model for upper limb robotic arm". In: *Advances in Mechanical Engineering* 11.9 (2019), p. 1687814019875537. DOI: 10.1177/1687814019875537.
- [42] Anthony Casey et al. "BCI controlled robotic arm as assistance to the rehabilitation of neurologically disabled patients". In: *Disability and Rehabilitation: Assistive Technology* 16.5 (2021), pp. 525–537.
- [43] Alexander A Frolov et al. "Post-stroke rehabilitation training with a motor-imagery-based brain-computer interface (BCI)-controlled hand exoskeleton: a randomized controlled multicenter trial". In: *Frontiers in neuroscience* 11 (2017), p. 400.
- [44] Michele Barsotti et al. "A full upper limb robotic exoskeleton for reaching and grasping rehabilitation triggered by MI-BCI". In: *2015 IEEE international conference on rehabilitation robotics (ICORR)*. IEEE. 2015, pp. 49–54.
- [45] Aleksandar Miladinović et al. "Optimizing Real-Time MI-BCI Performance in Post-Stroke Patients: Impact of Time Window Duration on Classification Accuracy and Responsiveness". In: *Sensors* 24.18 (2024). DOI: 10.3390/s24186125.
- [46] Life Science Robotics. *Life Science Robotics*. <https://www.lifescience-robotics.com>. Accessed: 22-05-2025.

- [47] Arpa Suwannarat, Setha Pan-Ngum, and Pasin Israsena. "Comparison of EEG measurement of upper limb movement in motor imagery training system". In: *Biomedical engineering online* 17 (2018), pp. 1–22.
- [48] Brad Edelman, Bryan Baxter, and Bin He. "Decoding and mapping of right hand motor imagery tasks using EEG source imaging". In: *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*. 2015, pp. 194–197. doi: 10.1109/NER.2015.7146593.
- [49] Bradley Jay Edelman, Bryan S. Baxter, and Bin He. "EEG Source Imaging Enhances the Decoding of Complex Right-Hand Motor Imagery Tasks". In: *IEEE Transactions on Biomedical Engineering* 63 (2016), pp. 4–14. URL: <https://api.semanticscholar.org/CorpusID:20199935>.
- [50] Gernot Müller-Putz et al. "Better than Random? A closer look on BCI results". In: *International Journal of Bioelektromagnetism* 10 (Jan. 2008), pp. 52–55.
- [51] Xiangmin Lun et al. "A Motor Imagery Signals Classification Method via the Difference of EEG Signals Between Left and Right Hemispheric Electrodes". In: *Frontiers in Neuroscience* Volume 16 - 2022 (2022). ISSN: 1662-453X. doi: 10.3389/fnins.2022.865594. URL: <https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2022.865594>.
- [52] Hauke Dose et al. "An end-to-end deep learning approach to MI-EEG signal classification for BCIs". In: *Expert Systems with Applications* 114 (2018), pp. 532–542. ISSN: 0957-4174. doi: <https://doi.org/10.1016/j.eswa.2018.08.031>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417418305359>.
- [53] Xiaying Wang et al. "An accurate eegnet-based motor-imagery brain-computer interface for low-power edge computing". In: *2020 IEEE international symposium on medical measurements and applications (MeMeA)*. IEEE. 2020, pp. 1–6.
- [54] Radia Rayan Chowdhury, Yar Muhammad, and Usman Adeel. "Enhancing Cross-Subject Motor Imagery Classification in EEG-Based Brain-Computer Interfaces by Using Multi-Branch CNN". In: *Sensors* 23.18 (2023). ISSN: 1424-8220. doi: 10.3390/s23187908. URL: <https://www.mdpi.com/1424-8220/23/18/7908>.
- [55] Karel Roots, Yar Muhammad, and Naveed Muhammad. "Fusion convolutional neural network for cross-subject EEG motor imagery classification". In: *Computers* 9.3 (2020), p. 72.
- [56] Patrick Ofner et al. "Upper limb movements can be decoded from the time-domain of low-frequency EEG". eng. In: *PloS one* 12.8 (2017), e0182578–e0182578. ISSN: 1932-6203.
- [57] Geeks for Geeks. *Types of Reinforcement Learning*. <https://www.geeksforgeeks.org/types-of-reinforcement-learning/>. Accessed: 30-05-2025.

- [58] OpenAI. *Proximal Policy Optimization*. <https://spinningup.openai.com/en/latest/algorithms/ppo.html>. Accessed: 30-05-2025.
- [59] OpenAI. *Key Concepts in RL*. [https://spinningup.openai.com/en/latest/spinningup/rl\\_intro.html#advantage-functions](https://spinningup.openai.com/en/latest/spinningup/rl_intro.html#advantage-functions). Accessed: 30-05-2025.
- [60] Geeks for Geeks. *Kaiming Initialization in Deep Learning*. <https://www.geeksforgeeks.org/kaiming-initialization-in-deep-learning/>. Accessed: 30-05-2025.
- [61] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: <https://arxiv.org/abs/1707.06347>.
- [62] Luis Serrano. *Shannon Entropy, Information Gain, and Picking Balls from Buckets*. <https://medium.com/udacity/shannon-entropy-information-gain-and-picking-balls-from-buckets-5810d35d54b4>. Accessed: 02-06-2025.
- [63] Xue Jiang et al. "CSP-Net: Common spatial pattern empowered neural networks for EEG-based motor imagery classification". In: *Knowledge-Based Systems* 305 (2024), p. 112668. ISSN: 0950-7051. DOI: <https://doi.org/10.1016/j.knosys.2024.112668>. URL: <https://www.sciencedirect.com/science/article/pii/S0950705124013029>.
- [64] Mouna Bouchane, Wei Guo, and Shuojin Yang. "Hybrid CNN-GRU Models for Improved EEG Motor Imagery Classification". In: *Sensors* 25.5 (2025). ISSN: 1424-8220. DOI: 10.3390/s25051399. URL: <https://www.mdpi.com/1424-8220/25/5/1399>.
- [65] Zaid Shuqfa, Abdelkader Nasreddine Belkacem, and Abderrahmane Lakas. "Decoding multi-class motor imagery and motor execution tasks using Riemannian geometry algorithms on large EEG datasets". In: *Sensors* 23.11 (2023), p. 5051.
- [66] Muhammad Ahsan Awais et al. "Effective Connectivity for Decoding Electroencephalographic Motor Imagery Using a Probabilistic Neural Network". In: *Sensors* 21.19 (2021). ISSN: 1424-8220. DOI: 10.3390/s21196570. URL: <https://www.mdpi.com/1424-8220/21/19/6570>.
- [67] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).
- [68] Ricardo Chavarriaga, Aleksander Sobolewski, and José del R Millán. "Errare machinale est: the use of error-related potentials in brain-machine interfaces". In: *Frontiers in neuroscience* 8 (2014), p. 208.
- [69] H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller. "Optimal spatial filtering of single trial EEG during imagined hand movement". eng. In: *IEEE transactions on neural systems and rehabilitation engineering* 8.4 (2000), pp. 441–446. ISSN: 1063-6528.



## Appendix A

# Appendix

### A.1 Theory of CSP

The general idea of CSP[69] is to create a spatial filter that separates two classes by maximizing the variance of one class and minimizing it for the other. The core equation that CSP is solving is a generalized eigenvalue problem defined by:

$$\Sigma_1 \mathbf{w} = \lambda \Sigma_2 \mathbf{w} \quad (\text{A.1})$$

where  $\mathbf{w}$  are eigenvectors which are the filters,  $\lambda$  are the eigenvalues, and  $\Sigma$  are the class covariance matrices. To solve this problem, first define a key set of values: Let  $C$  be the number of EEG channels,  $T$  be the number of time points in a trial,  $X^{(i)} \in \mathbb{R}^{C \times T}$  be the multi-channel EEG segment, and finally  $y^{(i)} \in \{0, 1\}$  be the class labels of trial  $i$ . Firstly, compute the normalized spatial covariance matrix:

$$\Sigma^{(i)} = \frac{X^{(i)} X^{(i)\top}}{\text{trace}(X^{(i)} X^{(i)\top})} \quad (\text{A.2})$$

Then compute mean covariance matrices for each class:

$$\Sigma_0 = \frac{1}{N_0} \sum_{i:y^{(i)}=0} \Sigma^{(i)}, \quad \Sigma_1 = \frac{1}{N_1} \sum_{i:y^{(i)}=1} \Sigma^{(i)} \quad (\text{A.3})$$

After computing these, the generalized eigenvalue problem can be solved where equation A.1 can equivalently be written as:

$$\Sigma_0 \mathbf{w} = \lambda (\Sigma_0 + \Sigma_1) \mathbf{w} \quad (\text{A.4})$$

This yields eigenvectors  $\mathbf{w}_i$  and eigenvalues  $\lambda_i$ . Sort the eigenvalues  $\lambda_i$  in descending order. Select the top  $m$  and bottom  $m$  eigenvectors corresponding to the largest and smallest eigenvalues:

$$\mathbf{W}_{\text{CSP}} = \begin{bmatrix} \mathbf{w}_1^\top \\ \vdots \\ \mathbf{w}_m^\top \\ \mathbf{w}_{C-m+1}^\top \\ \vdots \\ \mathbf{w}_C^\top \end{bmatrix} \quad (\text{A.5})$$

The spatially filtered signal is then:

$$\mathbf{Z} = \mathbf{W}_{\text{CSP}} \mathbf{X} \quad (\text{shape: } 2m \times T) \quad (\text{A.6})$$

Features are then extracted by computing the log variance of this spatially filtered signal as such:

$$f_j = \log \left( \frac{1}{T} \sum_{t=1}^T Z_j(t)^2 \right) \quad \text{for } j = 1, \dots, 2m \quad (\text{A.7})$$

which results in a feature vector:

$$\mathbf{f} = [f_1, f_2, \dots, f_{2m}]^\top \in \mathbb{R}^{2m} \quad (\text{A.8})$$

This feature vector is calculated within the CSP FFNN which takes in windowed EEG trials and applies the pretrained filters to the incoming data.