FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Deep Reinforcement Learning for Myoelectric Control of Upper Limb Movement

Ana Mafalda Costa Santos

Master's Programme in Informatics and Computing Engineering

Supervisor: Rosaldo J. F. Rossetti

Co-Supervisor: Zaferis Kokkinogenis

July 30, 2022

Deep Reinforcement Learning for Myoelectric Control of Upper Limb Movement

Ana Mafalda Costa Santos

Master's Programme in Informatics and Computing Engineering

July 30, 2022

Abstract

Limb dysfunction and, in some cases, even amputation is a reality that several people worldwide have to face on a daily basis. Such limitations are frequently associated with traumatic causes, strokes, or cardiovascular diseases. With technological advances, these adversities can be softened with prosthetics or exoskeletons. However, access to these solutions often requires an expensive investment and is not readily available to everyone. Even though one can opt for ad-hoc low-cost alternatives, developing a functional and adaptable prototype is not a trivial task.

The application of this research is to implement a Deep Reinforcement Learning model capable of moving a virtual arm in response to a person's brain activity. It also aims at exploring and studying the advantages and limitations of using these approaches to solve control problems in prosthetic robotics. With this in mind, the implementation structure can be divided into three main components. Firstly, processing the data that consists of EMG signals from the bicep and tricep. Afterwards, the development and analysis of a supervised learning approach followed by different Reinforcement Learning scenarios. Finally, the work also addresses the generalisation problem by testing a trained model's reaction when facing a new subject's EMG signals.

The methodology used different metrics to evaluate the models' performance to provide a more comprehensive understanding of the system's behaviour. This assessment takes into account the arm movement's delay to direction changes and precision. Multiple graphs were made to compare the real and the predicted angles visually. According to this analysis, the algorithm that offered the best results was a Deep Deterministic Policy Gradient version optimised for sample efficiency. It showed great potential to be used in real-life scenarios. However, when approaching the generalisation, this same algorithm showed unsatisfactory results, suggesting that further research may be required.

Keywords: Deep Reinforcement Learning, Control Problem, Upper Limb Model

Resumo

A disfunção de membros e a amputação são realidades que várias pessoas enfrentam diariamente, em todo o mundo; limitações frequentemente associadas a causas traumáticas, acidentes vasculares cerebrais ou doenças cardiovasculares. Com os avanços tecnológicos, estas adversidades podem ser amenizadas com próteses ou exoesqueletos. No entanto, geralmente, o acesso a essas soluções requer um investimento muito elevado, que não está prontamente disponível para todos – há ainda a possibilidade de alternativas ad-hoc e de baixo custo; no entanto, desenvolver um protótipo funcional e adaptável não é tarefa fácil.

O principal objetivo desta dissertação é avaliar a implementação de um modelo de *Deep Reinforcement Learning* capaz de fazer mover um braço virtual em resposta à atividade cerebral do seu portador, bem como a exploração e o estudo das vantagens e limitações destas abordagens (DRL) na resolução de problemas de controlo em próteses robóticas. Nesta perspetiva, o estudo estrutura-se em três níveis. Primeiro, o processamento dos sinais enviados continuamente do cérebro para o músculo e registados na eletromiografia. Posteriormente, a implementação e análise de uma abordagem de *supervised learning*, seguida de diferentes algoritmos de *Reinforcement Learning*. Finalmente, uma experiência de generalização, testando a adaptação de um modelo já treinado a um novo sujeito.

Diferentes métricas foram usadas para avaliar o desempenho dos modelos, por exemplo, a latência do movimento quando ocorre mudança de direção (quanto tempo demora a reagir ao sinal) e a precisão (se a posição do braço artificial é a esperada). Vários gráficos foram criados para facilitar a comparação visual entre o ângulo real e o previsto pelo modelo. Após a análise dos resultados, o algoritmo que mostrou o melhor desempenho foi uma versão de *Deep Deterministic Policy Gradient* otimizada para eficiência amostral. O modelo apresentou grande potencial para ser usado num cenário real; no entanto, quando usado para testar a generalização, os resultados não corresponderam às expectativas, o que sugere a necessidade de mais investigação.

Palavras-chave: Deep Reinforcement Learning, Problema de Controlo, Upper Limb Model

Acknowledgements

I would like to start by showing appreciation for my supervisors, Rosaldo Rossetti and Zafeiris Kokkinogenis. To Professor Rosaldo Rossetti for helping me find a project that would allow me to contribute to the health field. Merging biology and computer science has always been one of my aspirations. His patience and dedication helped me look at the bright side when faced with seemingly impossible challenges or deadends. On the other hand, I thank Professor Zafeiris Kokkinogenis for always encouraging me to improve, think outside the box and explore new subjects in DRL that could lead to great results. I am also thankful for his guidance in finding the appropriate and helpful literature that cleared the path for this project.

I would like to mention Pedro Fonseca from LABIOMEP, for his help was undoubtfully essential for the first stages of this dissertation. Although my signal processing knowledge was limited, I could always rely on his expertise to advise me on what steps to take.

For all the help in understanding the different reinforcement algorithms and grammar checks for this report, I thank my colleague, João Luz, an Informatics Master's student at FEUP.

Finally, I would like to thank my family, teachers and colleagues. As they have consistently contributed to my growth as a person and future engineer, I am very grateful to have you by my side.

Ana Mafalda Costa Santos

"The way positive reinforcement is carried out is more important than the amount."

B. F. Skinner

Contents

List of Figures

List of Tables

Abbreviations

A3C	Asynchronous Advantage Actor-Critic
ANN	Artificial Neural Network
CDE	Curiosity Driven Exploration
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EMG	Electromyography
HER	Hindsight Experience Replay
MAE	Mean Absolute Error
MC	Monte Carlo
MDP	Markov Decision Process
MSE	Mean Squared Error
NN	Neural Network
PG	Policy Gradient
PPO	Proximal Policy Optimization
PSD	Power Spectral Density
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
SAC	Soft-Actor-Critic
STFT	Short Term Fourier Transform
SVR	Support Vector Regressor
WT	Wavelet Transform

Chapter 1

Introduction

Powered limbs have been on the spotlight in the last few years. In this chapter, a brief explanation on the context, motivation and goals of this project will be given. A small guide on the document structure is also presented.

1.1 Context and Motivation

Limb dysfunction and, in some cases, even amputation, is a reality that several people worldwide have to face on a daily basis. Such limitations are frequently associated with traumatic causes, strokes, or cardiovascular diseases [49, 37]. These health problems are more probable the older a person is; therefore, with the increase of average life expectancy they are tending to rise [10].

With technological advances, these adversities can be softened with prosthetics or exoskeletons. These are defined as mechanical devices that copy human form and are worn by a user to work in accordance to her movements. The term prosthesis refers to an artificial substitute to a missing body part, whereas exoskeletons refers to a mechanical device used to boost the performance of the user [10].

However, access to these solutions often requires an expensive investment and is not readily available to everyone. Even though one can opt for ad-hoc, low-cost alternatives, developing a functional and adaptable prototype is not a trivial task.

The interest in artificial powered limbs is increasing, and new technologies are arising. These vary in price, intrusiveness degree, functionality, and so forth. However, there is always room for improvement and new experiments. The proven value of Deep Reinforcement Learning approaches in real-life challenges suggests that it might be a good solution for continuous control problems. As discussed in the chapter about the state of the art, these solutions are known to be advantageous in situations requiring sequential and even dynamic decision-making. With this in mind, this study will focus on how to bring these benefits to improve myoelectric prosthesis control.

1.2 Objectives

The end result of this dissertation is a DRL model that can move a virtual prosthetic or exoskeleton. Specifically, a prototype of an artificial upper limb with one degree of freedom, making it able to contract and extend the arm. Using patients' previously acquired brain signals, the model should be able to mimic the real upper limp movements. For this purpose, the electromyography technique will be used as input for the model. It measures the electrical muscle activity in response to a nerve's stimulation.

The processing of continuous signals to get the most appropriate reaction is a problem that can benefit from a Deep Reinforcement Learning approach. For this very reason, the main goal of this work is to determine the applicability of DRL in this context. With this in mind, a direct comparison between two different models will be made, and different metrics will be used to evaluate the models' performance. Arm movement's delay to direction changes and precision will be taken into account.

For the sake of comparison and evaluation of the proposed approach, another important goal is having a working supervised learning model. This will allow having a baseline with which the Reinforcement Learning results can be compared. It is also helpful to test the dataset and different model inputs.

Finally, it would be attractive to test the model generalisation capacity. For this, the best model will be trained with multiple subjects and then tested with an unseen one. EMG vary from person to person, analysing if it is possible, with data from only three subjects to predict a fourth is not a priority, but it is still a meaningful objective.

1.3 Document Structure

This dissertation is divided in five chapters. The remaining of the document is organised as follows.

Chapter 2 presents the literature review. It is described in five parts: prosthetics overview - how prosthetics evolved and how they are today - EMG signals - aquisition types and preprocessing - technologies background - specifically, supervised learning, reinforcement learning and related algorithms - problem-related work and lastly, existing tools.

Chapter 3 presents the problem and proposed solution, also where the methodology steps are carefully explained.

In chapter 4, there will be the result analyses from each methodology step: supervised learning, Reinforcement Learning and generalisation.

Finally, in the last chapter, chapter 5, the conclusion and future work are defined.

Chapter 2

Literature Review

This chapter will be divided into four different topics. Firstly, an overview of prosthetics, how they were and how they are now - followed by some valuable information on EMG signals and Machine Learning approaches, such as supervised learning, Reinforcement Learning and Deep Reinforcement Learning. Then a section with the prior work related to the topic and lastly, the existing tools in the field.

2.1 Prosthetics Overview

Prosthetics have been around for many centuries, not with today's capabilities but with the same purpose, to replace a missing part of the body. The first known was neither an arm nor a leg. It was a toe found in Egypt dated between 950-710 B.C.E [17]. From this date until the 16th century, prosthetics didn't change much. They were still motionless [17].

After this period, people custom made prototypes relying on various combinations of wood, metal, leather etc. These could be controlled by cables or gears and could rotate and bend [27].

Advances in this field are often related to times of war, soldiers that lost limbs and needed a replacement. As expected, the first and second World Wars contributed to the increase in demand for such devices. These events led us to where we are today.

Modern prosthetics are now made using advanced plastic and carbon fibre composites. We can classify these into three significant categories: cosmetic, body-powered, and myoelectric externally powered.

Cosmetic: Have limited movements and can usually grasp and catch lightweight objects. Their main goal is to provide a life-like feel to the user and its surroundings. It is the cheapest option in the market, not considering homemade solutions.

Body-powered: Muscles are able to control the prothetic with cables. Even though it allows the user to have control over the limb, it can only handle one movement simultaneously. Due to being a more invasive approach, the user may experience fatigue.

Myoelectric externally powered: Reads the signals received by the muscle from the brain and uses it as input for the movement. It gives the user a lot of freedom without performing frequent muscle contractions (like the previous case). These are the most advanced options available and also the most expensive [36].

2.2 Eletromyography Measurement

Electromyography signals are biomedical signals that evaluate the electrical activity in a muscle. An electromyograph is used to measure and record the EMG signals. It detects the electrical currents generated by the muscle fibres whenever the brain activates the muscles.

Our nervous system controls muscle activity, making the signals complicated and dependent on the muscles' anatomical and physiological properties [38].

The signal we get from the sensor is usually full of noise, either from the electrical current passing through different tissues, the device quality or even both. There are two different approaches when it comes to acquiring EMG signals: invasive methods usually resort to needle electrodes, while non-invasive techniques only require electrodes on the skin surface to provide surface EMGs (sEMG). Figure 2.1 shows how EMGs can be acquired.

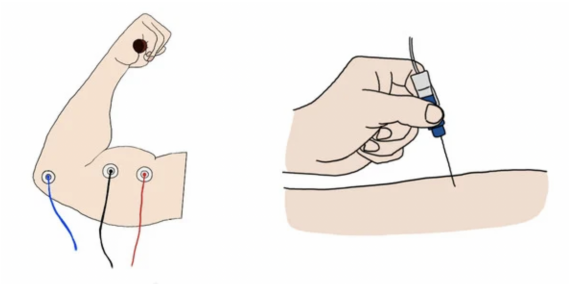

Figure 2.1: non-invasive and invasive EMG acquisition.
[33]

This dissertation will focus on sEMG signals acquired from the bicep and tricep.. As stated in [2]: "sEMG can be recorded from various parts of the body, but electrode placement is critical in ensuring reliable and repeatable results". In Figure 2.2 we can see what a normal EMG signal should look like.

2.2.1 Noise Sources

As mentioned before, EMG readings usually suffer from noise, mainly when using a non-invasive approach. This noise can have multiple sources, which will have a direct impact on the signal quality [4], for example:

- **Inherent noise:** It cannot be eliminated. All electrical types of equipment have it. It can be reduced by using high-quality components.

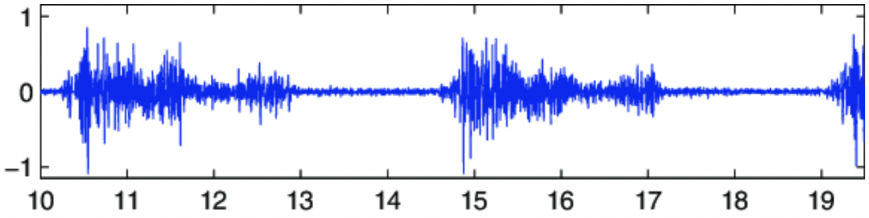

Figure 2.2: Bicep EMG example recording.
[54]

- **Electrical noise from powerlines or ambient noise:** It is almost impossible to eliminate the electromagnetic radiation the human body is exposed to. Powerline noise, the ambient noise caused by the radiation of power sources, distorts biomedical signals. It is characterized by a chronic sinusoidal 50/60 Hz element and can be eliminated/reduced [1].
- **Motion artefact:** It occurs when electrodes move, changing their relative position. This happens when there is muscle movement underneath the skin or when a force impulse causes a motion at the electrode-skin interface [26].
- **Cross-Talk:** It happens when the EMG captures signals from muscles that aren't the target. This can be mitigated by changing the electrode location and size [30].

2.2.2 Noise Reduction Methods

Different recommendations can be followed in order to acquire a high-quality EMG signal. The most important and independent of the device are:

- **Sensor Placement:** Position the sensor in the "belly" of the muscle and away from tendons and innervation zones.
- **Skin Preparation:** Clean the skin using alcool and remove excessive hair.
- **Sensor Contact:** Ensure that the electrodes are well attached to the skin.
- **Filtering:** After having the EMG signal, filter the unwanted frequencies.

The focus will be on the latter since we will explore different ways to preprocess the EMG signal so it can be used as an input model.

2.2.3 Signal Preprocessing

"Raw EMG offers us valuable information in a particularly useless form." [38] When using an EMG as a placeholder to get information about a body part movement, various methods need to be applied so we can finish with a useful EMG signal. This section will focus on some of the previously used and proven to work solutions to date.

2.2.3.1 Signal Rectification

There are several recommended steps one should go through to work with EMG signals. The first one would be Full-Wave Rectification. This type of rectifier converts both parts (positive and negative) of each cycle from an alternating current into a continuous current [48]. It is necessary to convert all signal amplitudes to their absolute value, leaving no negative amplitudes in the resulting signal. According to [24], since raw EMG signals have a mean value of zero, this rectification makes them easier to read as well as study standard amplitude metrics such as the max value and mean and the arc beneath the curve.

2.2.3.2 Signal Smoothing

Given the high sensitivity to the acquisition conditions, EMGs are often very unique and challenging to reproduce with high precision. A way to address this is by minimising the very distinct characteristics of the signal through smoothing operations. Examples of these operations are the Root Mean Squared Error and the Moving Average [39]. The latter requires the use of a sliding window mechanism to limit the scope of the calculation. It is crucial to take into account that one should adapt the size of this window to the activity taking place upon data acquisition. Although bigger windows are recommended for slow-paced experiments (around 500 ms) and smaller sized windows (around 20ms) are advised for rapidly changing signals related to abrupt movements, there are no rules to limit this decision.

Choosing windows with larger dimensions increases the chances of considering very contrasting signals associated with significantly distinct activities. On the other hand, applying windows with minimal time intervals may lead to a complete loss of the benefits of smoothing in the first place. One should seek a balance in order to get the best results. As a rule of thumb, windows with sizes between 50ms and 100ms tend to do well in most situations.

2.2.3.3 Signal Normalization

Besides noise, another essential aspect of EMG acquisition is that the amplitude measurement can be very unpredictable, even under the same set of conditions. This unpredictability means that scale of the voltage values captured may differ between two consecutive samples, making it harder to compare and study [8].

A possible solution to this issue is to use signal normalisation according to an arbitrary reference. After such normalisation, the data is interpreted not according to its voltage but to its percentual relation to the criteria stipulated previously. This criteria may be the mean value of the sample or the maximum value of a reference activity. Another widespread approach is the use of the maximum voluntary contraction.

2.2.3.4 Time-Frequency Analysis

One of the ways to approach EMG analysis is to represent the signal as a function of time. However, it is important to consider that it is common for myoelectric signals, especially the ones acquired through non-invasive methods,such as sEMGs, to present a continuously changing frequency. This nonstationarity occurs due to the contractions of the muscles highly affecting the content of the signal acquired. The fact that the frequency changes throughout each sample combined with other artifacts such as noise, pose a serious obstacle to some time-frequency approaches.

The Cohen's class is a class of time-frequency quadratic energy distributions which are covariant by translations in time and in frequency.

2.2.3.5 Wavelet transform

Wavelet transform is used in the time-frequency analyses of non-stationary signals, like EMG, in order to decompose the signal. It breaks down the signal into a set of wavelets (basic functions). This transformation is similar to the Fourier transform.

Fourier transform is used to analyse signals in the frequency domain, not taking time into account. Therefore it should only be used for stationary signals - signals that do not change over time. To try and solve this handicap, the short term Fourier transform (STFT) was created. It allows dividing the signal in windows where it keeps the same format (is stationary) and then applying the Fourier transform to each window. The only difference in their equations is the teak of the window function. In figure 2.3 we can see an example of a signal with three different windows.

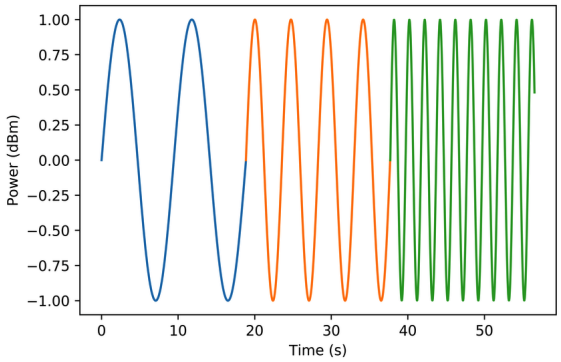

Figure 2.3: Sin(x) function with different frequencies. Each colour represents a different window. [22]

Even though it may seem this is the solution to the problem, the STFT method has limitations and practical challenges. It works with the assumption that the signal is stationary in the finite intervals (windows), which isn't true for EMG and most real signals [45].

Unlike the STFT, the Wavelet transform allows for variable frequency and window width. This is what allows the use of this method in signals that vary in time and frequency. With it, we can

analyse a signal at different resolutions, the so-called multiresolution analyses. The WT equation is 2.1.

$$F(\tau,s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} f(t)\psi^*(\frac{t-\tau}{s})\,dt \tag{2.1}$$

Where τ represents the signal localisation parameter used to shift the wavelet through the signal, and s is the inverse frequency (observing parameter) used to scale the wavelet to better adjust to the signal. By changing the width of the wavelet (scaling), it becomes useful for different scenarios. For example, expanded wavelets are better at resolving low-frequency components, while shrunken wavelets are better for high-frequency components s [12]. Figure 2.4 visually shows what shrunken and expanded wavelets are.

Figure 2.4: Shrunken and Expanded wavelet, respectively.
[33]

2.3 Machine Learning Methods

To better understand the technologies used in this paper, it is essential to give some background. Machine Learning is a subfield of artificial intelligence that uses input data and experience to improve algorithms automatically. This data allows building models that can predict outcomes without being explicitly programmed to do so.

There are three major approaches: Supervised learning, Unsupervised learning and Reinforcement Learning. We will focus on Supervised Learning first, and then, Reinforcement Learning.

2.3.1 Supervised Learning

Supervised learning consists of learning a function that is able to map an input to the desired output. This is done by iteratively feeding examples to a predictive model and performing minor corrections to improve the predictions. Each example used for training has an input (a single entry or a vector) and the desired output/label (what the model should predict). The labelled data used for training creates a mapping function that will enable the model to label testing data. Testing data consists of examples that still weren't seen by the model. In an optimal scenario, the model determines its label correctly. For this to be possible, the algorithm must measure its accuracy while training using a loss function. The loss function computes the distance between the correct output (given label) and the model result. The goal of training with different and meaningful

examples is for this error to be minimised so the algorithm can generalise and perform well with unseen data.

Supervised learning can be separated into two big categories. Depending on the problem and the type of data at hand, we may choose to use classification or regression. Figure 2.5 shows the supervised learning taxonomy.

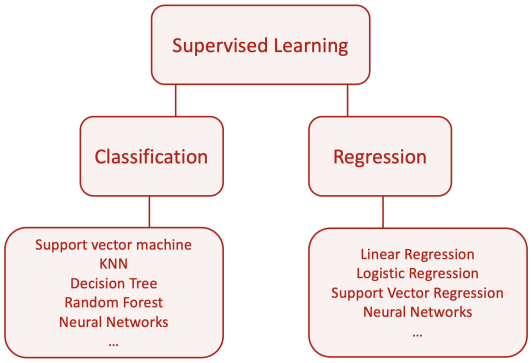

Figure 2.5: Supervised learning algorithm examples.

2.3.1.1 Classification

Classification uses algorithms to assign examples into preset categories (sub-populations). It recognises similarities and relations between the instances to attempt to label the data correctly. The algorithm uses the learning from the training data to predict the probability of that instance belonging to one of the predetermined classes. For this, it uses an approximate function to map the input variables to a discrete output variable. Some of the most used algorithms for classification are Support Vector Machines, Decision Trees, K-Nearest Neighbour and Neural Networks.

2.3.1.2 Regression

Regression is used to investigate relationships between independent variables and the target variable. This technique allows for predicting continuous outcomes. It aims at finding the best line or curve that fits the data in order to minimise error. For this, it should make the distance between each instance point and the line/curve as slight as possible.

It is possible to evaluate the trained regression model with three metrics: error, bias, variance.

- **Error:** There are different metrics to calculate the error, only the most used will be presented below. They all consider the average distance between every predicted entry and the true value.

  The mean squared error (MSE) and root mean squared error (RMSE) are used to evaluate how well the model fits the problem at hand. They measure the quality of the estimator. The

difference between them is that RMSE is the square root of MSE [7]. The equations 2.2 and 2.3 show how to compute MSE and RMSE, respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.3}$$

The mean absolute error (MAE), described in 2.4, is a metric used when one does not want outliers to influence the performance evaluation of the model. It consists of a loss function for regression. To estimate it, one can sum the errors (the difference between the predicted and the actual value) and calculate its mean.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.4}$$

The coefficient of determination or R^2, whose equation can be found at 2.5, determines the variance ratio in the target variable that the independent variable can explain. It measures the quality of the data for the model [53].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.5}$$

- **Bias:** Can be described as the assumptions the model makes so the target function can be learned more easily. It can be measured considering how different the model prediction and the actual value being estimated is. We may want to choose one model over another depending on the bias.

  A high-level bias is when the average predicted values are very different from the target values. This suggests that the approximation to the target function was made with more simplification. Linear algorithms try to approximate the correlation between the input and output to a straight line. Even though it makes learning more accessible, these simple models can't always pick on the tendencies of a real-life problem. On the other hand, a low bias relates to fewer simplifications of the target function. This is achieved using more complex algorithms, nonlinear [28].

- **Variance:** It is a measure of how different a model behaves between previously seen data and testing data. In other words, how much a target function changes depending on the data used for training. High variance values mean that the testing data's target variable is very different from the one the model predicted. It can mean overfitting. Overfitting happens when the model learns noise and the detail in the training data, fitting precisely against it. This impacts the model performance on unseen data, making it unable to generalise and defeating its purpose. It is often connected with noise and too many unneeded variables

in the dataset. The goal is to minimise the variance. The lower it is, the better the model predicts and generalises [42].

The main goal of any regression model is to achieve good predictions for the data. That's where the Bias-Variance Trade-Off kicks in. We want to achieve a low bias and a low variance.

Regression Algorithms This section will discuss three regression algorithms, focusing mainly on Linear Regression, Support Vector Regressor (SVR) and Artificial Neural Networks (ANN).

- **Linear Regression:** Analyses a linear relationship between the independent variable (x) and the dependent variable (y), there can only be one y, but there may be multiple x. The model finds how the value of the dependant variable changes along with the value of the independent variable [29]. The mathematical representation of the linear regression can be given as the following 2.6:

$$y = \beta_0 + \beta_1 X + \varepsilon \tag{2.6}$$

  where a0 is the line translation (where it intercepts the y axis), a1 is the linear regression algorithm (a scaling factor for each input), and ε is the random error. The linear line that shows the relationship between the variables is called the regression line, and it can be of two types, as illustrated in Figure 2.6:

  - Positive linear relationship: is when both the dependent and independent variables increase in their axis, y and x, respectively.
  - Negative linear relationship: is when the dependent variable decreases on the y-axis while the independent variable increases on the x-axis.

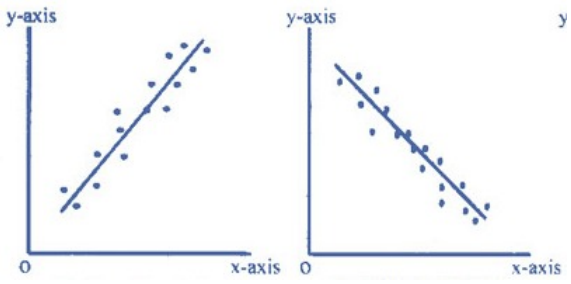

Figure 2.6: Positive and Negative linear relationship, respectively.
[11]

- **Support Vector Regressor:**

  To address classification problems where data is not linearly separable, one can resort to Support Vector Machines (SVMs). The adaptation of this method to regression problems is a Support Vector Regression. As the problem suggests a different goal, the solution also

changes: instead of trying to find the best hyperplane that will divide data into different categories, SVR shapes its hyperplane to better fit as many points as possible.

Using this technique, it is possible to parameterise a threshold value ε for the distance between a boundary line and the hyperplane. This threshold sets the criteria for the model updates as opposed to common approaches such as the Ordinary Least Squares, which try to minimise the squared error. With this in mind, SVR analyses the data points, letting those falling outside this threshold produce a correction in the hyperplane.

Depending on the input line the model is trying to estimate, SVR may resort to different kernels: linear, polynomial and radial basis functions are the most used.

- **Artificial Neural Network (ANN):** It consists of a group of interconnected nodes that try to mimic the human brain's neurons. An Artificial Neural Network comprises an input, hidden, and output layer. Each layer has n nodes representing an artificial neuron. The arrows between nodes connect the output of a node and the other's input. In Figure 2.7 one can see an example of a neural network.

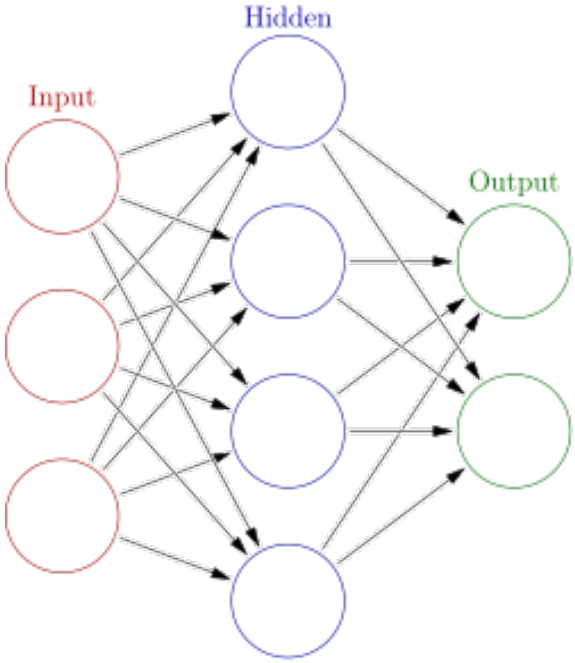

Figure 2.7: Artificial Neural Network layers.
[51]

An artificial neuron has an input, weights, a bias and an output. Each node receives input from other nodes, multiplies them by the weights, sums them along with the bias and passes the result to the node on the next layer. In the output, a nonlinear function may also be applied. It is called the activation function. It is what decides whether the node is activated or not and introduces nonlinearity in the output. This sequence is called the forward

propagation phase. The backward propagation phase consists in updating the weights in the model.

Unlike linear regression, ANNs can learn a complex nonlinear relationship between the features and the target variable. Different types of Artificial Neural Networks exist, like Vanilla, Recurrent and Convolution Neural Networks. The main difference is that Vanilla Neural Networks can only handle structured data while the others handle unstructured data.

2.3.2 Reinforcement Learning

Reinforcement Learning consists of training models to make a sequence of decisions. In each situation, the model's goal is maximizing the reward, which was defined beforehand. It is about finding the best possible behaviour it can take.

The RL system is composed of four components. An agent and the environment it interacts with. It can be static or dynamic. The policy learned to take actions, and lastly, the Reward Function, which the agent uses to tell apart what is correct (reward) and what's wrong (punishment).

The main difference between this approach and supervised learning is that in the second, the input given to the model for training already has the solution - it is labelled. The model needs to understand which situation corresponds to each label and use this knowledge in the testing set (unlabelled). While there is no answer in RL, the model chooses the next step based on the experience of previous rewards, or even randomly.

Since the environment dynamics are not explicitly given to the agent, it needs to extrapolate most of the information through trial and error. Furthermore, to achieve the ideal sequence of actions to solve the problem at hand, the model needs to explore the options in the action space and profit from the ones with maximum reward (exploration-exploitation trade-off).

Positive and negative are two types of reinforcement learning.

Positive: revolves around stimulating specific types of behaviour by giving it a positive outcome. This will make this action more likely of repeating itself in the future.

Negative: is used to teach a behaviour to prevent something undesired. For example, putting sunscreen not to get sunburned. The goal is that, over time, the unpleasant thing won't happen again (getting sunburned).

2.3.2.1 Markov decision Process

Before diving into some of the most relevant RL algorithms, it is pertinent to mention that these problems are often described as Markov decision processes (MDPs). This formalization provides the groundwork to solve stochastic sequential decision-making problems. To define an MDP, one needs to specify the tuple < S, A, P, R >. The state-space S is the set of states where the problem can be found. For each state, the action space A defines the available actions to be taken. The probability function P is the probability of reaching a state s' from the state s with action a. Finally, the immediate reward R is the immediate reward of the same transition described above [3].

2.3.2.2 Reinforcement Learning Algorithms

One can rely on multiple algorithms when dealing with RL problems. The choice between them depends on the inherent properties of each issue. An essential factor in this decision is the agent's information within the environment. For this reason, it is possible to distinguish between model-free and model-based approaches.

In comparison, the first ones try to learn an optimal policy or value function using experience exclusively; in the latter, the agent endeavours to learn while creating a model of its surroundings - this model weighs in on the decisions made. If this is the case, the agent can plan ahead and deduce the consequences of each action. These methods usually result in a better sample efficiency (requiring less data) than model-free approaches [15]. The most common algorithm used when the model is fully comprehended is **AlphaZero** [43].

However, the model can often be unavailable to the agent, hence the need for model-free approaches. Q-learning and Sarsa are model-free value-based algorithms. Both can handle problems with stochastic transitions and rewards. The major difference is that Q-learning is an off-policy technique while Sarsa is on policy. In the first, the learning agent learns the value function of the action based on another policy while the other is based on the current policy.

Q-learning: Being Q[S,A] the current estimate of the expected value, the agent keeps a Q table of Q[S,A]. S corresponds to a set of states space and A to a set of actions. The Q-function uses the Bellman equation 2.7 and takes "s" (state) and "a" (action) as an input:

$$NewQ(S_t,A_t) = Q(S_t,A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1},a) - Q(S_t,A_t)] \tag{2.7}$$

The first step of the algorithm is starting to build the Q-table. From a set of options, it chooses the following action. The table is then updated using the new reward value with the Bellman equation. All of the steps, with the exception of the first one, are repeated until the training finishes. In order to choose a new action, the epsilon rate needs to be taken into account. In the beginning, the epsilon rate will be higher, and it will decrease over time. This way, the model starts by exploring the environment choosing actions randomly and then exploits the environment [20].

Sarsa: This algorithm derives from Q-learning. The main change is that the Bellman equation considers the action to be taken in the newly changed state 2.8. This choice also involves a greedy algorithm.

$$NewQ(S_t,A_t) = Q(S_t,A_t) + \alpha[R_{t+1} + \gamma Q(S_t+1,A_{t+1}) - Q(S_t,A_t)] \tag{2.8}$$

Besides these temporal difference approaches, another relevant strategy is using **Monte Carlo** Methods [50]. MC processes attempt to estimate the expected return by simulating the various outcomes of each possible action in each state in the state space. This method is considered computationally costly and inefficient. The methods explained so far work for discrete action and state spaces. Other algorithms used in continuous problems will be described in the next section.

2.3.2.3 Deep Reinforcement Learning Algorithms

In many practical problems, traditional Reinforcement Learning techniques may not be enough. Decisions may become too complex for tabular methods. For these situations it is preferable to use Deep Reinforcement Learning.

Deep Learning is a branch of Machine Learning that tries to mimic the human brain by using neural networks with multiple hidden layers. DRL takes advantage of these to solve RL problems.

Deep Reinforcement Learning algorithms have the same structure described above: model-free (value-based and policy-based) and model-based.

Deep-Q-Network: is a value-based approach that addresses ideas from tabular Q-Leaning using neural networks. Similarly to Q-Learning, DQN maintains and iteratively improves an approximation of the Q value function, the Q-network. The use of neural networks opens the doors to more intricate problems, especially regarding state spaces. Instead of addressing each state individually and assessing the value of each action, DQN allows the agent to embrace a more significant number of states by feeding them into a NN. Depending on the architecture, the agent might be able to detect similarities between states or even extrapolate meaningful information from the limited data that is given. These capabilities make this algorithm a powerful tool for addressing RL problems. However, it is relevant to mention that this new approach also introduced a few unfamiliar challenges that were later addressed.

Firstly, the Q function update depends on the estimation of the rewards that the agent expects to get in future states and the reward for the current state. As the state of the Q-function provides the values used in this computation at the time of the update, this introduces some instability in the learning process. In later improvements made to DQN, a target network was introduced to mitigate such problems. This target network is initialised as a copy of the Q-network, but as these parameters are improved, the target network gets periodically updated every N steps.

Another important aspect of using NNs in this context is that they may become overfit to specific sequences of states in the environment. For instance, this could translate to an agent learning to navigate a particular part of the environment but failing to do so when faced with slight variations. To break this correlation between consecutive states, DQN employs a technique called experience replay, which consists of storing all the transitions the agent goes through and randomly sampling them to perform the updates [32].

Policy gradient algorithms try to improve the policy $\pi_\theta(s,a)$ that dictates the probability of choosing each action a in a particular state s, as seen in 2.9. It consists of repeatedly optimizing the parameters θ of the neural network defining π_θ to get the maximal reward. This way, the neural network will output higher probabilities for the optimal action a^* in each state [9]. As stated by equation 2.10, ideally, the policy outputs a probability close to 1 for the optimal action to take in s.

$$\pi_\theta(s,a) = P[a|s,\theta] \tag{2.9}$$

$$\pi_\theta(s,a^*) \approx 1 \tag{2.10}$$

Proximal Policy Optimization: as a policy gradient algorithm, the current policy is updated in each iteration, allowing parameters improvement. It uses two deep neural networks, one with the policy parameters (actor) and the other with the rewards (critic). This Actor-Critic architecture is the base of several Policy-Gradient Methods. PPO tries to battle hypersensitivity to hyperparameter tuning (stepsize, learning rate, etc.) that is present in vanilla policy gradient. With this in mind, it ensures a slight variance between the new and old policy in training, which is accomplished by clipping the updated region to a very narrow range. Its mathematical equation is shown below 2.11 :

$$L^{CLIP}(\theta) = \bar{E}_t[min(r_t(\theta)\bar{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\bar{A}_t)] \tag{2.11}$$

Other advantages of PPO include sample efficiency and ease of implementation and tuning [41].

Some approaches are a mix of value-based and policy-based. DDPG and A3C are some examples.

Deep Deterministic Policy Gradient [25] is an off-policy actor-critic method that relies on the innovative techniques of DQN to tackle problems with continuous action spaces. However, DDPG can become slightly more computationally expensive since it combines the use of target networks (introduced in DQN) with the Actor-Critic architecture. As such, training an agent requires maintaining four neural networks: an actor-network, a critic-network, a target actor-network and a target-critic network.

With respect to its actor network, it is relevant to mention that, as the name suggests, DDPG develop a deterministic policy. This means that the agent will always choose the same action a for each specific state s. For this reason, unlike other standard Actor-Critic methods, the policy returns a continuous value (the action) rather than the probability of selecting the actions available. At the same time, the critic network approximates the action-value function to improve its estimations of the expected reward for each action chosen by the actor.

Just like the value-based approach, DDPG resorts to using a replay buffer to perform experience replay. The transitions executed by the agent are stored in memory so that they can later be randomly sampled. This randomness helps to break the correlation between transitions of states, reducing the chances of the model overfitting. As the agent interacts with its surroundings, it collects the experience that will later be used to update the networks. While DQN employs an ε-greedy strategy to explore the action space, DDPQ does so by adding noise to the actor predictions. Since it is a policy gradient method, to update the actor network μ one must compute the gradient of an objective function J with respect to the actor parameters θ^μ (2.12). The J function returns the expected sum of rewards obtained by following a policy π. This gradient calculated also takes into account the value estimations given by the critic Q and its parameters θ^Q.

$$\nabla_{\theta^\mu} J \approx E_{s_t \rho^\beta}[\nabla_{\theta^\mu} Q(s,a|\theta^Q)|s=s_t, a=\mu(s|\theta^\mu)] \tag{2.12}$$

The update on the critic network is aims to minimize the mean squared error loss described in 2.13. As shown in the equations this definition of loss considered the estimations of the target actor μ' and the target critic Q'.

$$\begin{aligned} L &= \frac{1}{N} \cdot \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2 \\ y_i &= r_i + \gamma \cdot Q'(s_{i+1}, \mu'(s_{i+1} | \theta^{\mu'}) | \theta^{Q'}) \end{aligned} \tag{2.13}$$

Contrary to that is done in DQN, the target updates can be described as a *soft-update*. Instead of overwriting the target parameter $\theta^{\mu'}$ and $\theta^{Q'}$ with the values of θ^{μ} and θ^{Q}, the targets are drawn closer to the original networks using the variable τ (2.14). This value is a hyperparameter restricted between 0 and 1.

$$\begin{aligned} \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \\ \theta^{Q'} &=\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \end{aligned} \tag{2.14}$$

Asynchronous Advantage Actor-Critic: This is the asynchronous variant of the algorithm A2C, used for single agents. It is a recent addition to the field (2017) and was developed by Google's DeepMind. Independent from each other, multiple agents have a copy of the environment. When each agent has more knowledge to share, it asynchronously updates the global shared network. A3C also uses advantage metrics, which uses the difference between its expectations and the actual reward to learn [31]. This metric improves the learning process 2.15 :

$$A^{\pi}(s,a) \quad = \quad Q^{\pi}(s,a) - V^{\pi}(s) \tag{2.15}$$

2.3.2.4 Algorithms Optimization

In many Reinforcement Learning problems, the agent must interact with the environment aimlessly until an action eventually produces a state with a reward. Scenarios like the one described are characterised by a sparse reward signal. By this, it is meant that the number of states that produces feedback is very scarce compared to the size of the state space. Environments with very sparse reward functions make it difficult for the agent to understand the dynamics involved, potentially harming the learning process overall. To mitigate these effects, several techniques have been suggested producing promising results. These techniques revolve around augmenting the feedback given by the interactions with the agent's surroundings. The goal is to provide a denser signal that motivates intermediate goals thought to be highly correlated with the primary objective.

It is usual for agents to employ several procedures, such as feature extractions, to get meaningful information from raw data before the actual inference of the actions to be taken. The limitation of these pipelines is that, although they prosper in supervised-like settings, with minimal rewards, the agent might never extract helpful information. Some researchers have used smaller auxiliary tasks to guide the training process [19]. While trying to reach the main objective, the agent simultaneously optimises policies related to parallel functions that have specific rewards attached.

On the one hand, the secondary tasks can be associated with understanding the general dynamics of the environment. However, they can also be related to reward prediction. Finally, it is also possible to use tasks such as value-function replay - forecast the expected future rewards for a given state as done in Deep Q-Learning. All these other goals are added to optimise the feature extraction pipeline in the hopes that it also aids the general purpose of the problem. It is also worth mentioning that using auxiliary tasks makes the learning process significantly more sample efficient.

Another issue related to sparse rewards is that the models may get stuck in local optima, never unveiling their full potential in the environment. With this in mind, Curiosity Driven Exploration (CDE) methods attempt to motivate exploration during training [35]. When using an epsilon-greedy strategy, the exploration is dependent on a diminishing probability. In CDE, exploration depends on how well the agent understands what will happen in the future. Using a Forward Model, the agent tries to predict the next state of the environment at each step of the episode. If the agent is repeatedly facing similar situations, these predictions will become gradually better. The idea is to incorporate the error between the predictions and the actual outcomes in the reward signal to incentivise the agent to encounter unforeseen circumstances.

A simple and effective solution to maximise the sample efficiency when dealing with sparse rewards is Hindsight Experience Replay (HER) [5]. The technique allows the model to learn even from unsuccessful actions. The method requires that the agent acts for every state they encounter. Regardless of the action that is chosen, an ideal trajectory between states is computed. Besides storing the original transition along with its reward, an ideal transition where the best action would be the chosen one is also stored in a buffer. All these transitions collected are later used to update the agent's intelligence.

2.4 Related Work

There is increasingly more research in the prosthetics field, with different technologies and targets. The solutions mentioned in this literature review all focus on processing user inputs to replicate or somehow aid the movement of a specific part of the body. Some examples of this include controlling a hand, elbow or shoulder, or even a combination of those. After a careful analysis of relevant articles, it is possible to divide solutions into two main branches: those not based on pattern recognition and those based on pattern recognition.

The first group contains approaches that focus on analysing the function that describes the signal. These solutions do not require any memory of past actions and their effectiveness. With this goal in mind, different approaches can be employed. Some looked for local and global extremes in the signal with methods like zero-crossing [46] or window amplitude-based [47] while others checked whether the signal voltage went over a specified threshold. These options stand out since they require less computational power without compromising real-time results.

On the other hand, approaches based on pattern recognition often rely on Machine Learning techniques. The main advantage of these strategies is that they allow processing significant amounts of information from the input signal to draw relevant conclusions.

In the subfield of supervised learning, the dissertation of a past mechanical engineering student, Francisco Sampaio, used random forest and Bayes optimisation to predict the users' intentions. The results attained were favourable. The mean of the sum squared error in the different experiences (online, offline, with and without Bayes optimisation) was 0,6 [40].

There have been more advances and experiences with this subfield of Machine Learning. Some authors explored more conventional algorithms like Support vector machines [34] and light gradient boosting machines [44]. On the other hand, neural networks have also proven effective—satisfactory results have been achieved with linear neural networks [52], fully connected Artificial Neural Networks [44] and adaptive network-based fuzzy inference systems (ANFIS) [21].

Pattern-based approaches also include solutions that resort to Reinforcement Learning and Deep Reinforcement Learning methods. For instance, researchers have implemented deterministic policy gradient models to learn how to position the fingers of a hand to mimic specified poses and grab objects. The authors added strength sensors to the fingertips that allowed the agent to infer meaningful information such as the object's position and shape [16]. Another interesting aspect is that, since the model was learning using sparse rewards, the authors implemented hindsight experience replay (HER) [5] to take advantage of all the experience gathered.

Another project making use of this technology uses a soft-actor-critic algorithm (SAC) to create an agent capable of controlling an arm (shoulder rotation, elbow and hand) in order to reach for random points in space. The main goal of the model was to discover the most efficient trajectory to get to the point [13].

2.5 Existing Tools

"More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments." [6]

That's when OpenAI gym comes in handy. It is a toolkit used to develop Reinforcement Learning algorithms. Since it provides standardization, it helps when trying to reproduce experiments and compare the results obtained. This tool offers a vast set of premade environments to choose from depending on the type of problem. Additionally, it provides an interface so that developers can create their own (personalized) environments to suit their projects better. This way, it is easier to define the state and actions spaces as well as how the system reacts and how rewards are affected. Another advantage is that it allows using implementations of algorithms from libraries like TensorFlow and PyTorch. These include multiple tools that are also useful for Reinforcement Learning.

There are, however, a few alternatives to OpenAi Gym. KerasRL is used to ease the RL algorithms programming, DeepMind Control Suite - for physics-based simulations -, ReAgent -

an open-source platform that uses PyTorch algorithms - and Unity ML-agent - easily allows to train agents in 2D and 3D environments. Besides these alternatives, it is relevant to mention the OpenSim software.

OpenSim is an open-source tool for modelling and simulating human movement, "bridges biomechanics, neuroscience and computer science communities" [23]. Together with the osim-rl package, it is possible to develop control strategies based on Deep Reinforcement Learning algorithms. Besides taking care of the physics and dynamics related to the human body, it provides visualization capabilities. This is a clear advantage since it allows us to see the results even in the absence of a physical system. Since this visualization feature is available to anyone, it also facilitates the reproduction of the experiments in the future. In Figure 2.8 one can find an example of the visualization for an existing model.

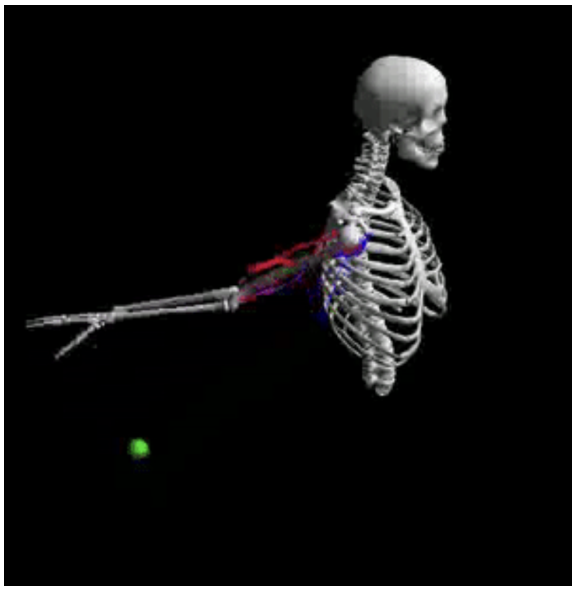

Figure 2.8: Existing osim-rl model.

Chapter 3

Methodological Approach

This chapter aims to explain the problem at hand and how it was solved. With this in mind, a detailed description of the methodology is provided to describe the process behind the solution so as the experiments may be replicated.

3.1 Problem Description

As technology evolves, new ways of doing things, constructing materials and even thinking also change, often for the better. The health sector is no exception. In order to improve what current solutions have to offer, it is necessary to spend time and money on exploring new possibilities and upcoming technologies. However, sometimes existing technologies can be applied to several different fields, as it is the case of Deep Reinforcement Learning.

In this dissertation, the problem of prosthetic control functions as the subject of a DRL study. This research attempts to provide a formalisation of a DRL problem so that an agent can learn how to respond appropriately when faced with a subject's EMG signals. This formalisation includes the definition of what information is present in the agent's observations. This data will be the input to the decision-making process that will dictate the model's reaction. Additionally, it is necessary to stipulate the action space from which the agent may select a response. Another critical element is the specification of the reward function. This feedback signal is the only implicit data from which the agent can extrapolate its purpose. Finally, to test this formulation and compare different methods to tackle it, the implementation should provide an environment that combines all of these aspects to allow the agent's training.

3.2 Proposed Solution

In order to achieve the desired results, this dissertation will go through different stages, each with distinct goals in mind.

The first step would be to acquire EMG data from different subjects, specifically from the bicep and tricep muscles. These are the muscles that receive the signals to move the upper limb. This can be done manually, having access to the proper devices and subjects, or by a previously acquired open-source dataset. The latter option was chosen using the following dataset [18]. After having the required data, there is a need for signal processing, and only after that the data can be considered useful. This allows to mitigate inherent noise common in measurement equipment, and ambient noise [38].

Then, an early definition of metrics to allow the quantisation of the model's evolution and learning progress is essential. Similarly, performance indicators are necessary to understand how well the technology behaves in the current problem. These metrics will constitute the criteria for evaluating the final solution.

The first predictions should be from a supervised learning model, specifically an Artificial Neural Network. This allows for validating and determining the most successful data formats to be fed as model input.

After this, we have all the information to start working on a Deep Reinforcement Learning system. This includes the environment, reward definition, model architecture, and so forth.

The next step is to evaluate the model predictions. The output will be in degrees. Based on this, we can compare the output with the optimal movement by analysing their degree difference. Movement delay in direction change is also an interesting metric. It determines how long the robotic arm takes to react to a given signal. Figure 3.1 shows the information flow, both for the supervised learning model and Deep Reinforcement Learning.

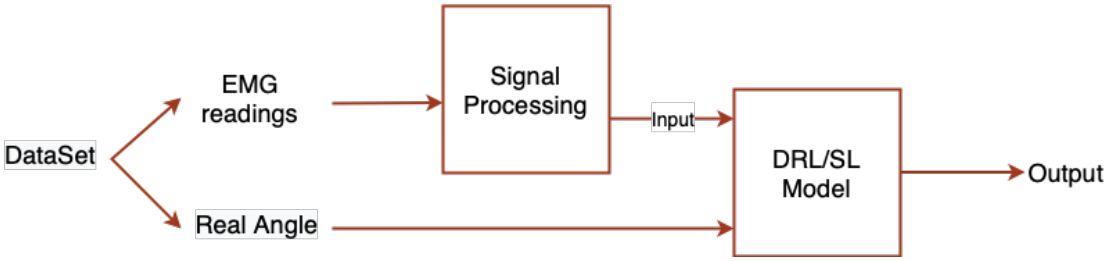

Figure 3.1: Data Flow from input to output.

Finally, the best model can be used for empirical generalisation. This will evaluate the model's ability to predict movements from different subjects.

3.3 Dataset

The data used was found online. It consisted of surface EMG recordings of five patients. An overall of 37 recording files, since each patient went through, more or less, six trials. The dataset contains three columns. The two first columns include EMG signal recordings of the right arm. Specifically, the bicep in the first column and the tricep in the second. The third and last column has the right elbow flexo-extention angle. When the arm is fully extended, the angle takes the

value 0; when it is fully flexed, the angle varies according to the arm's size but is usually between 120 and 160 degrees. The frequency used for the recording was 2kHz. In Figure 3.2, it is possible to see the file tree of the dataset files:

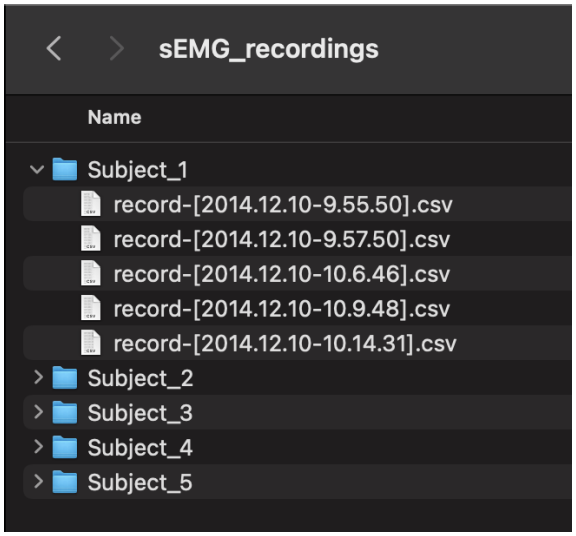

Figure 3.2: Dataset file tree.

In order to better visualise the dataset, a new column was added containing the recording time. This was calculated using the frequency of the recording (2kHz). Each entry was captured with a 0,0005 interval. The surface EMG values were in scientific notation; after dividing by 10000, the values were in the range of -10 and 10. This change was to ease the interpretation of the dataset values and their comparison between different people's recordings. Finally, a heading with the label names emg_bicep, emg_tricep, angle and time was added. This allows manipulating the data better using python's library, Pandas. In Figure 3.3 one can see the dataset after some tweaks.

	emg_bicep	emg_tricep	angle	time
0	-3.51	4.72	0.0	0.0005
1	-3.60	4.78	0.0	0.0010
2	-3.65	4.81	0.0	0.0015
3	-3.68	4.83	0.0	0.0020
4	-3.74	4.85	0.0	0.0025
...	...	...	...	...

Figure 3.3: Initial entries of the dataset after tweaks.

3.4 Signal Processing

The programming language used for all preprocessing steps was python, with NumPy and pandas libraries to work with the data more smoothly. After plotting the EMG signals, it is evident that it needs processing because of the noise. The following image 3.4 shows the sEMG captured in the bicep/tricep (y) and the time (x).

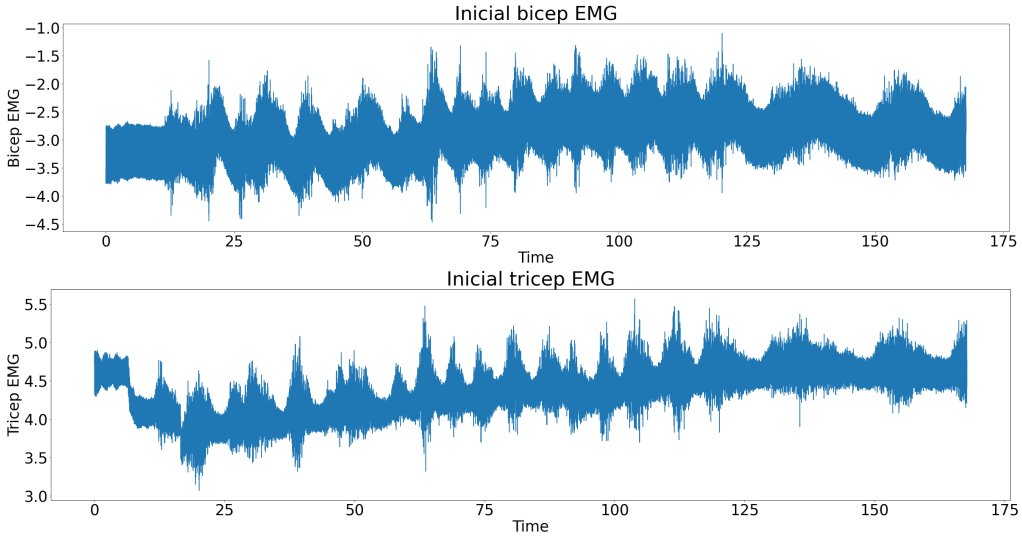

Figure 3.4: EMG's plotted over time from the bicep and tricep, respectively.

A low and high pass filter was used to eliminate high and low frequency noises in the signal. After research and practical experiments, the conclusion is that the best cut-off for EMG signals is between 20 and 450 Hz [14]. The high pass keeps frequencies above 20Hz the same, and the frequencies below are smoothed while the low pass does the opposite. Frequencies below 450Hz are kept the same, and above are smoothed. This signal gets substantially better after the filters are applied, as can be seen in 3.5.

After analysing the resulting signal, it is still very noisy, and it is hard to see the signal tendency. It is not giving helpful information about the arm movement. Another way of eliminating noise is filtering a specific frequency band. This is useful when talking about electrical noise from powerlines, usually between 50-70Hz. So that fundamental EMG frequencies are not eliminated, the best way to find the interval to use the filter is by plotting the power spectral density (PSD). PSD is the measure of the signal's power versus its frequency. The resulting plot can be found in Figure 3.6.

The plot shows a peak near 60Hz, which is not usual for EMG signals. In order to solve this problem, a notch filter was applied. This filter attenuates signals in a specific band (stop band) and leaves the frequencies below and above the same. The frequencies filtered were between 57 and 63 Hz. The resulting signal is in Figure 3.7.

When analysing the signal after the notch filter and comparing it to the previous signal, it is easy to evaluate the improvement. By plotting the PSD 3.8 again, it is possible to see the action of

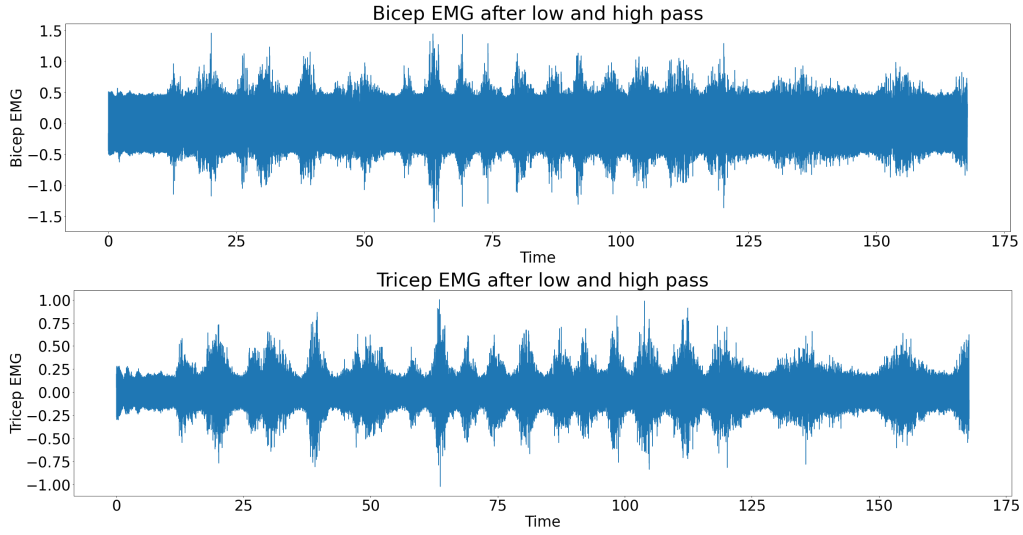

Figure 3.5: EMG's after a low and a high pass filter from the bicep and tricep, respectively.

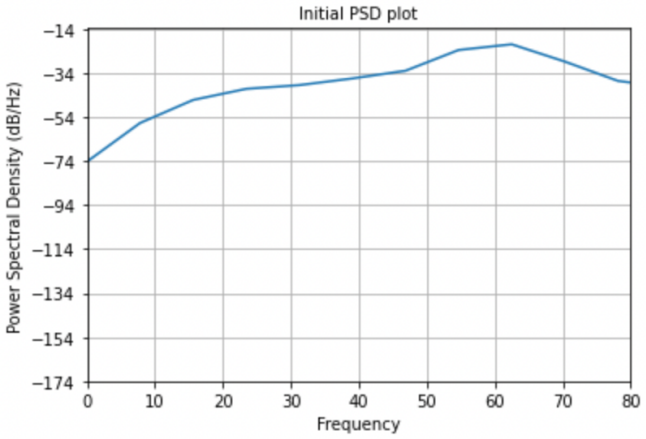

Figure 3.6: PSD plotted for the bicep EMG.

the notch filter.

When studying the signal, having a negative and a positive division does not bring any advantages and may even make it harder to read. For this reason, it is a common practice to calculate the absolute of the negative signs, so the result is an all-positive signal as can be found in 3.9.

The last step of the signal processing was to eliminate unnecessary abrupt changes. In order to achieve a thinner line, which is called an "envelope", a low pass filter on 2Hz was applied, and all the values were normalised between 0 and 1. The final EMG signal can be found in Figure 3.10.

With all the updates to the EMG signals, it is now easier to understand how the angle changes relate to the signal variations. Before saving the new and improved result to a file, a resample was applied. The initial dataset was vast, making it impossible to train the different models in a feasible time. To solve this problem, a resample of 60T using the mean of the values was used.

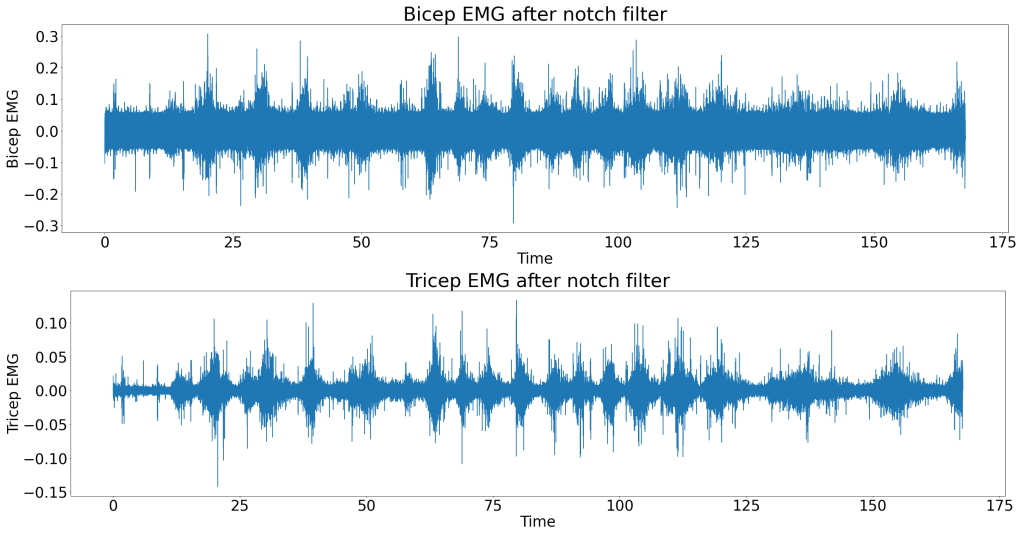

Figure 3.7: EMG's after notch filtering from the bicep and tricep, respectively.

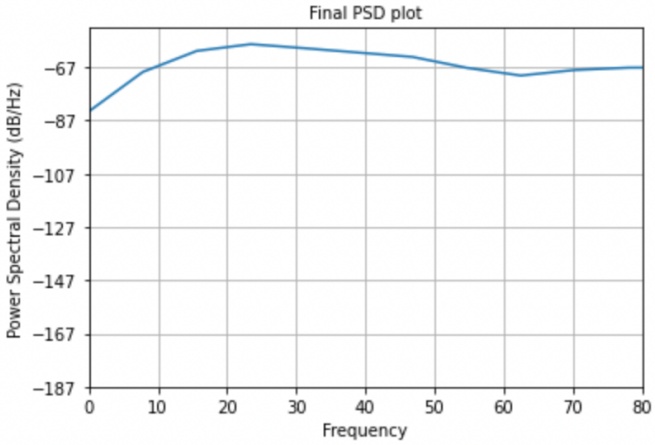

Figure 3.8: PSD plotted for the bicep EMG after the notch filter.

3.5 Supervised Learning

The starting point was to use supervised learning to study different options to solve the problem and work as a baseline for the Reinforcement Learning algorithms. The algorithm chosen was a simple ANN regression due to being the most easily compared to Deep Reinforcement Learning since they both take advantage of neural networks. The programming language used was python with the library Keras for the development of the networks models. The data used was from a trial of subject 1.

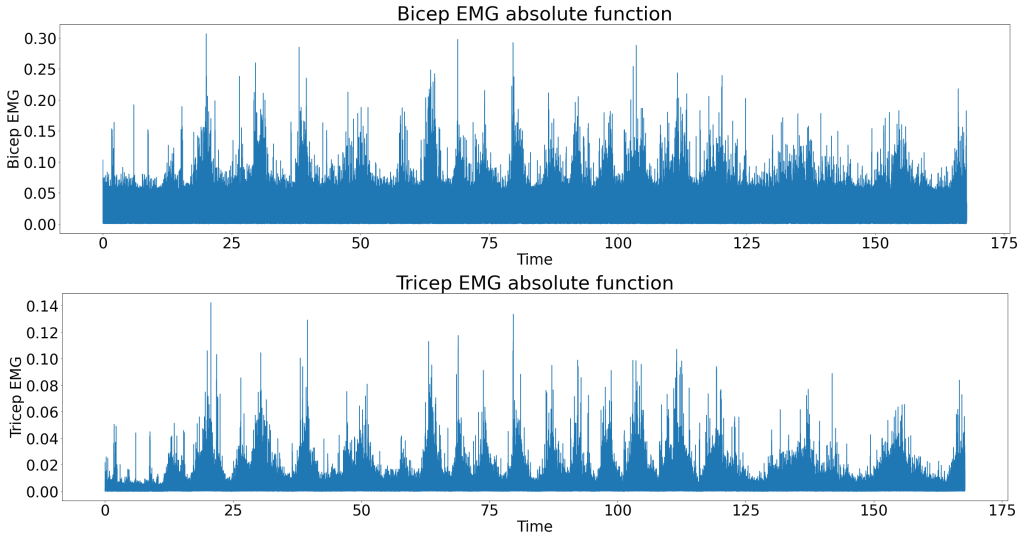

Figure 3.9: EMG's after absolute function from the bicep and tricep, respectively.

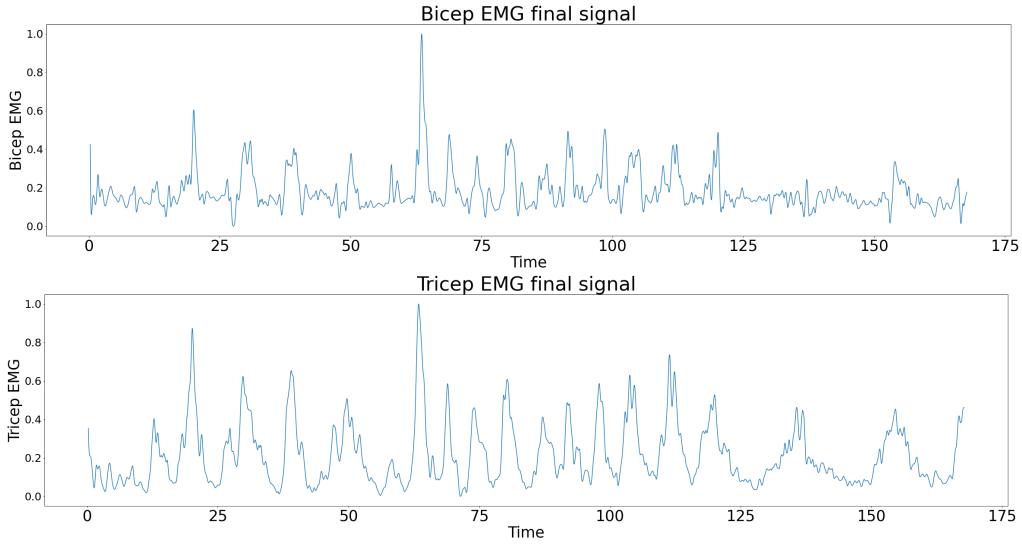

Figure 3.10: EMG's after processing from the bicep and tricep, respectively.

3.5.1 Training Data

After some consideration and proven examples, the dataset needed some changes. Up until this point, it had four columns: bicep's EMG, tricep's EMG, angle and time. However, this amount of data wasn't enough for the model to learn. It needed more information on the previous EMG reading to use them as knowledge for the prediction. The use of "windows" is not new, especially when talking about sequential learning. To the dataset were added new columns with the previous EMG readings of the bicep and tricep, as can be attested in 3.11. The window size differs from one experience to another, making the dataset column number change as well. Since this modification creates rows with Nan values and the datasets are very big, the first n (window size) rows were

eliminated.

	angle	emg_bicep	emg_tricep	bicep-1	tricep-1	bicep-2	tricep-2	bicep-3	tricep-3	bicep-4	...
0	8.0	0.042696	0.255287	0.045911	0.254297	0.049184	0.253148	0.052489	0.251831	0.055803	...
1	8.0	0.039565	0.256130	0.042696	0.255287	0.045911	0.254297	0.049184	0.253148	0.052489	...
2	8.0	0.036550	0.256837	0.039565	0.256130	0.042696	0.255287	0.045911	0.254297	0.049184	...
3	8.0	0.033676	0.257418	0.036550	0.256837	0.039565	0.256130	0.042696	0.255287	0.045911	...
4	8.0	0.030969	0.257886	0.033676	0.257418	0.036550	0.256837	0.039565	0.256130	0.042696	...

Figure 3.11: Dataset after the introduction of windows.

The model uses the EMG readings as input, and the target variable is the arm's angle. Of the overall data, 20% is used as testing data and 80% to train the model.

3.5.2 Scenarios

Two different components were evaluated in the different scenarios: the window_size and the neural network structure. The window size varies between 20 and 50, changing the number of past readings being considered. The structure can be distinguished between a deeper tree and a wider tree. The deeper network has one input layer, two hidden layers, three dropout layers and an output layer. The wider network has similar composition but with only one hidden layer. The Table 3.1 summarises the hyperparameters chosen for the ANN.

Table 3.1: ANN Parameters.

Parameter	Value
Test Size	0.20
Batch Size	64
Epochs	64
Window	20/50

Four scenarios were analysed: the combinations between the different window sizes and the tree structure.

First Scenario: This consists of a wider neural network with a window size of 20. The input layer is a dense layer with input dimension 42. This includes the current EMG reading recording of the bicep and tricep and the 20 past recordings of each. The activation function is a ReLU. Between the input and hidden layer there is a dropout layer. The hidden layer used has 22288 units corresponding to the number of observations (entries in the dataset). A dense layer with one unit and a linear activation function was created to provide an output.

Second Scenario: This consists of a deep neural network with a window size of 20. The input and output layers are the same as in the previous scenario. However, there are two hidden layers. The first is a dense layer with 216 units. This value was chosen by dividing the number of observations (entries in the dataset, which are 22288) by the input (102) summed with the output (1). The second dense hidden layer has 108 units (approximately half the number of units of the

first hidden layer). Both used ReLU as an activation function; between them, there is a dropout layer.

Third Scenario: For this experience, a wider neural network was used with the same architecture as the first scenario, but the window size was increased to 50.

Fourth Scenario: For this experience, a deep neural network was used with the same architecture as the first scenario, but the window size was increased to 50.

3.6 Reinforcement Learning

As the main focus of this work was to approach the problem described using Reinforcement Learning, it required the definition of the environment, the choice of algorithms used to train the model and possible optimisation strategies. The tools used to ease this experiment were open AI Gym (a python library) to develop the environment and TensorFlow for the Machine Learning models. The following section aims to describe the decisions made along the process.

3.6.1 Environment

In order to migrate the problem into the Reinforcement Learning paradigm, one first needs to address the Markov Attributes previously established. With this in mind, implementing the environment required the definition of the agent's states and observations for each time step of the episode. Moreover, it needed to react accordingly to the actions chosen by the learner, providing both a new state and an insightful reward. Concerning the information available to the agent, the observation was similar to the input given to the Regression model. Each observation contained 102 features, the current EMG readings of the bicep and tricep and the 50 previous readings from each muscle. Additionally, the current angle was also included.

For this experiment to succeed, the agent's arm must move between an angle of 0 and 180 degrees with a step of up to 5 degrees. A relevant difference between the Reinforcement Learning and the supervised learning methods explored in this work is that for RL, the agent did not need to predict the angle for the EMG readings. Instead, it was required for the agent to understand the necessary correction to be applied to its current state (angle). The agent was able to contract or distend the arm up to 5 degrees. Therefore, the actions consisted of drawing values within the interval [-5,5]. As will be explained further, in the case of DQN, there are 11 possible actions corresponding to all integers in the interval. This constraint is due to the discretisation of the action space inherent to the algorithm.

One of the most important aspects of the environment is the reward. This function provides the most impactful feedback to the learner. A challenging aspect of designing a reward signal is that it should clearly illustrate the goal of the problem, as the rules and dynamics are rarely explicit to the agent. For this specific environment, the agent's actions were rewarded with a score of 1 if the arm's angle was within a threshold of 5 degrees. A less significant reward of 0.5 was given in case the arm moved in the correct direction. Although the intention behind this small incentive was to make the reward slightly denser, some optimisations were still needed. Otherwise, in situations

where the target angle and the current angle were too far apart, the agent was too limited to make the proper correction. Finally, the agent would get penalised if none of the above was achieved, meaning it stayed outside of the interval without an attempt to reduce the difference. All this information is summed in 3.1.

$$R_t = \begin{cases} 1 & \text{if the angle is within the threshold} \\ 0.5 & \text{if the correction reduced the difference to the target angle} \\ -0.5 & \text{otherwise} \end{cases} \tag{3.1}$$

The training consisted of several episodes, each one requiring a complete reading of the dataset. In the specific case of training for subject 1, the trial has 22288 entries. This could imply that the best possible reward at the end of an episode would be of 22288. However, the agent can only move 5 degrees at a time. For instance, this limitation implies that if the real arm changes from 5 to 20 degrees, the agent will not receive a reward of 1, even with the most significant step of 5. Having this in mind depending on the steps between angles, the reward interval can vary. In Figure 3.12 it is represented the cycle for each time step of the episode.

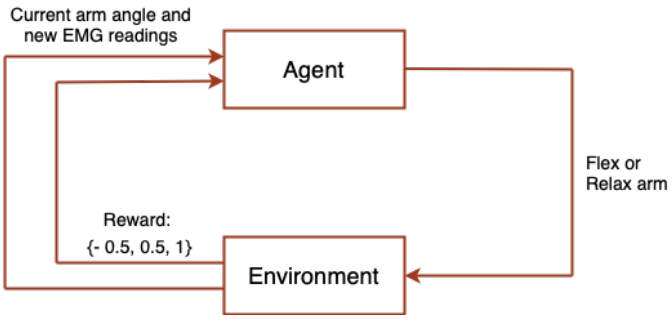

Figure 3.12: DRL agent-environment interaction cycle.

This environment implementation was used for the different experiments, and the algorithms chosen for the problem were the Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG).

3.6.2 Algorithms

As established by the literature review, there are several techniques that one can use to tackle such a problem. One of the main approaches used in this implementation was DQN since it was relatively straightforward to develop and yielded promising results in several specific environments. As previously mentioned, since DQN is a deep learning approach to Q-learning, it demands the use of a discrete action space. This requires the discretization of the continuous space of A into 11 discrete actions as in 3.2.

$$A_{discrete} = \{a_1, a_2, ..., a_i, ..., a_{11}\} \tag{3.2}$$

Where, a_1 is the action of subtracting -5 degrees out of the current angle, a_2 subtracts -4 degrees, and so on and so forth, up to a_{11}, which adds 5 degrees to the current angle.

On the other hand, Deep Deterministic Policy Gradient was chosen as the policy gradient counterpart. The decision to compare these two methods was influenced by the fact that they employ similar techniques (target networks and experience replay) while attempting to approximate different functions, optimising very distinct policies. On the other hand, DDPG allows the agent to act withing a continuous action space, and pick any value in $A_{continuous}$, as defined in 3.3.

$$A_{continuous} = [-5.0, 5.0] \tag{3.3}$$

Where any $a \in A_{continuous} : -5.0 \leq a \leq 5.0$.

Both DQN and DDPG follow a similar flow. The agent starts by collecting experiences by repeatedly making observations and choosing what correction to apply to the current angle drawn by the arm. Each interaction produced a transition between states and a reward. All of the transitions are stored in the Replay Buffer. Once the Replay Buffer has a number of transitions greater or equal to the batch size, the experience replay can occur. This happens by randomly sampling transitions to be used to update the networks. In the case of DQN, after each update, the exploration factor epsilon (ε) is decremented, which means that the mini-batch size dictates how fast this value drops to its minimum.

The Table 3.2 summarises the hyperparameters chosen for DQN.

Table 3.2: DQN Parameters.

Parameter	Value
Replay buffer size	1000000
Batch Size	64
Initial Epsilon	1.0
Epsilon Decay	0.0001
Minimum Epsilon	0.01
Gamma (Discount Factor)	0.99
Alpha (Learning rate)	0.001
Target Update Rate	1000

Furthermore, the network has the following composition: two fully connected layers with 256 neurons and a ReLU activation function. Finally, the dense output layer has eleven neurons (one for every possible discrete action) and no activation function.

Regarding DDPG, it is worth noting that its training is computationally more expensive when compared to DQN. Instead of training a network to approximate Q(s,a) and maintaining a second target network, DDPG introduces two networks, each with a target correspondent. Although only the actor and critic networks are constantly updated, this process requires predictions from all networks to compute the gradient.

The Table 3.3 summarises the hyperparameters chosen for DDPG.

Table 3.3: DDPG Parameters.

Parameter	Value
Replay buffer size	1000000
Batch Size	64
Gamma (Discount Factor)	0.99
Alpha (Target Learning rate)	0.001
Beta (Critic Learning rate)	0.002
Target Update Rate	1000
Tau (Target Update Factor)	0.005

The actor and critic networks used have similar architectures, the only difference being the output layer. They are composed of two fully connected layers with 512 neurons each and a ReLU activation function. The actor's output layer has a single neuron and a *Tahn* activation function. Similarly, the critic's output layer has a single neuron but no activation function.

3.6.3 Optimisations

Driven by the need to make the agent's observations as valuable as possible and thus reduce the experience necessary, two minor adjustments were made to the models' training.

The first was to make the feedback signal slightly denser. This was done by giving a reward of 0.5 if the agent moved the arm in the correct direction, regardless of how much.

Moreover, the implementation employed the Hindsight Experience Replay technique. Each time the agent corrected the arm angle by a particular measure, the environment also computed the best possible action that could be taken, given the current state and EMG reading. This ideal transition took into account that sometimes, the difference between the desired target angle and the current angle was more significant than the correction allowed by the action space.

This work will discuss the impact of these optimisations in the results chapter.

3.6.4 Visualization

To better understand the agent behaviour, a visualisation was implemented. This was a simple interface using pygame. Pygame library is an open source module for the python language. It is intended for games or other applications. For this work, the better approach was to be able to see the arm movement in real-time while the Reinforcement Learning algorithm predicted it. It would be even better to show the difference between the predicted and real angles. To fulfil all this, two robotic arms are being displayed, as shown in the following image 3.13.

In its true colour, the robotic arm is copying a real-life arm using the angle predicted by the model, while the red arm reproduces the angle that the actual subject was performing at the time.

Having a tool such as this allows the demonstration of this prototype to an audience that does not understand the domain of the application. As the implementation encapsulated this simple visualisation from the inner workings of the environment, this module could be improved or even changed in the future.

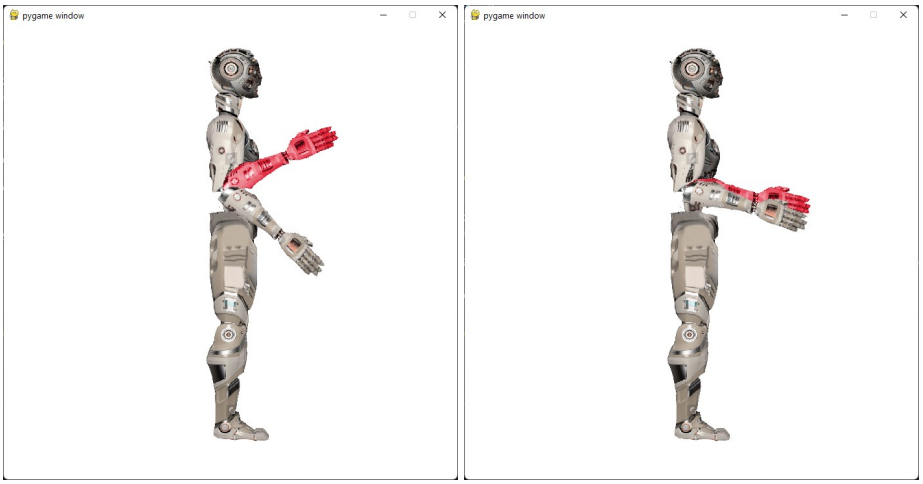

Figure 3.13: Predicted vs real arm angle in different stages of the episode.

3.7 EMG Generalization

Generalisation in Reinforcement Learning is still a demanding challenge to solve. As in supervised learning, one must train the model with different but still meaningful data for the agent to learn and perform generalisation. Only after a thorough training process it is possible to move on to the next steps. For this specific problem, generalisation meant that the agent could react appropriately to unseen EMG readings. Therefore, the goal was to predict the arm angle for a new subject whose EMG signals had never been seen.

The strategy used consisted in training the model with multiple EMG readings provided by four different subjects. The use of EMGs from others is essential since the shape and magnitude of the signals vary from person to person. Although it would be preferable to make the most out of the entirety of the dataset available, only four of the five subjects had viable trials that were significantly long. For this reason, three recordings were used to train the model, and a fourth was selected for testing the generalisation. For the robustness of this experiment, different combinations were used for training, but always three to train and a different one to test.

Chapter 4

Result Analysis

This chapter's purpose is to present and discuss the results obtained. This includes results from the different supervised learning scenarios, two Reinforcement Learning methods (DQN and DDPG) and the generalisation attempt. Finally, there will be a discussion of the outcomes of the overall techniques and how apt this solution would be to tackle real-world scenarios.

4.1 Supervised Learning

Before attempting to use Reinforcement Learning techniques, the implementation included a supervised learning approach. This stage consisted of using an Artificial Neural Network to evaluate the impact of different window sizes as well as various network layer architectures. The results achieved in the four scenarios described in the methodology are presented next.

4.1.1 Scenario 1

The first scenario is characterised by using a wider network combined with a window size of 20.

As shown in Figure 4.1, the model is able to minimise the loss over time, especially between 0 and 10 epochs. Moreover, the R2 metric (coefficient of determination) reaches 0.6 after the first ten epochs. The value does not change much afterwards, resulting in a final coefficient of determination of 0,639. When looking at the model's performance, it is possible to see that the model understands the motion of the arm as the tendency of predictions follows the changes in the actual angles. Another metric used to evaluate the model was the mean absolute error (MAE), which was assessed at 17,67.

4.1.2 Scenario 2

Moving on to the second scenario, employing a deeper network while maintaining the window size of 20 resulted in a decline in performance. This is demonstrated in Figure 4.2.

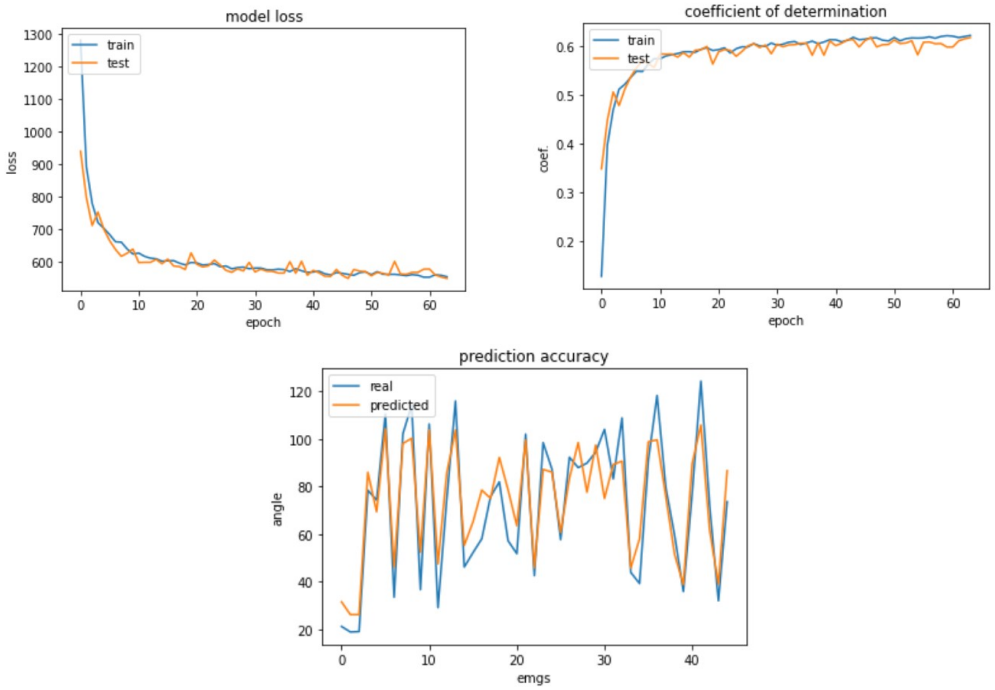

Figure 4.1: Loss, coefficient of determination and predicted angle plotted for scenario 1.

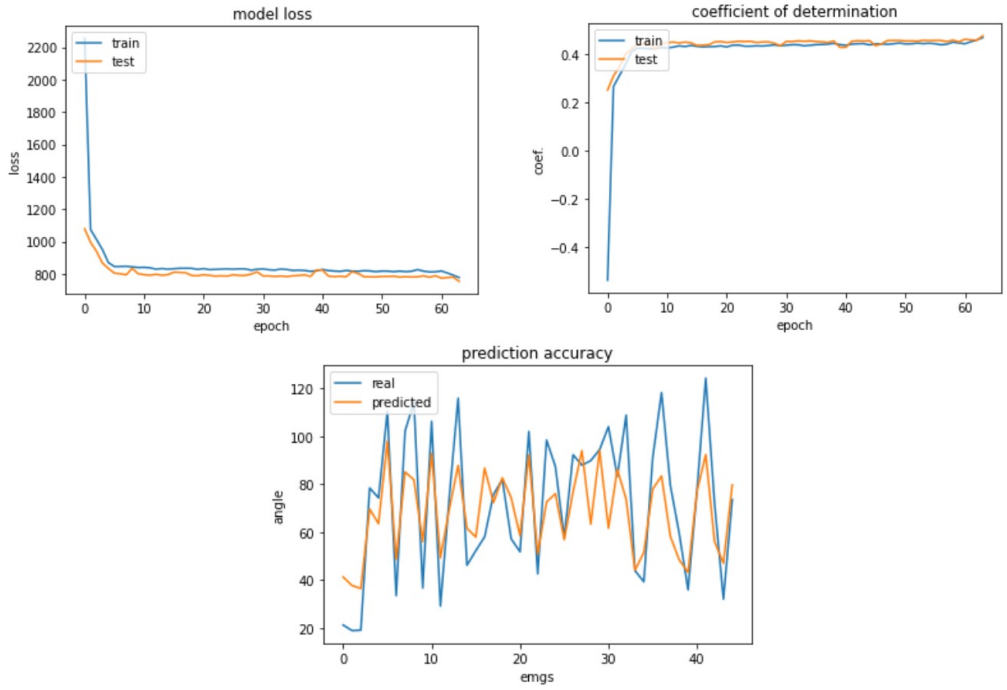

Figure 4.2: Loss, coefficient of determination and predicted angle plotted for scenario 2.

Although the loss is minimised, and the R2 metric improves, they only reach 800 and 0.5, respectively. These results are unsatisfying when compared to the previous configuration. More-

over, as seen by the testing plot, even though the model understands the overall motion of the arm, it lacks the understanding of how high or low the angle should be. For this scenario, the MAE computed was 22,33.

4.1.3 Scenario 3

Augmenting the window size brought a new set of promising results. In the third scenario, besides having a wider neural network, the model counts with an input containing a window size of 50.

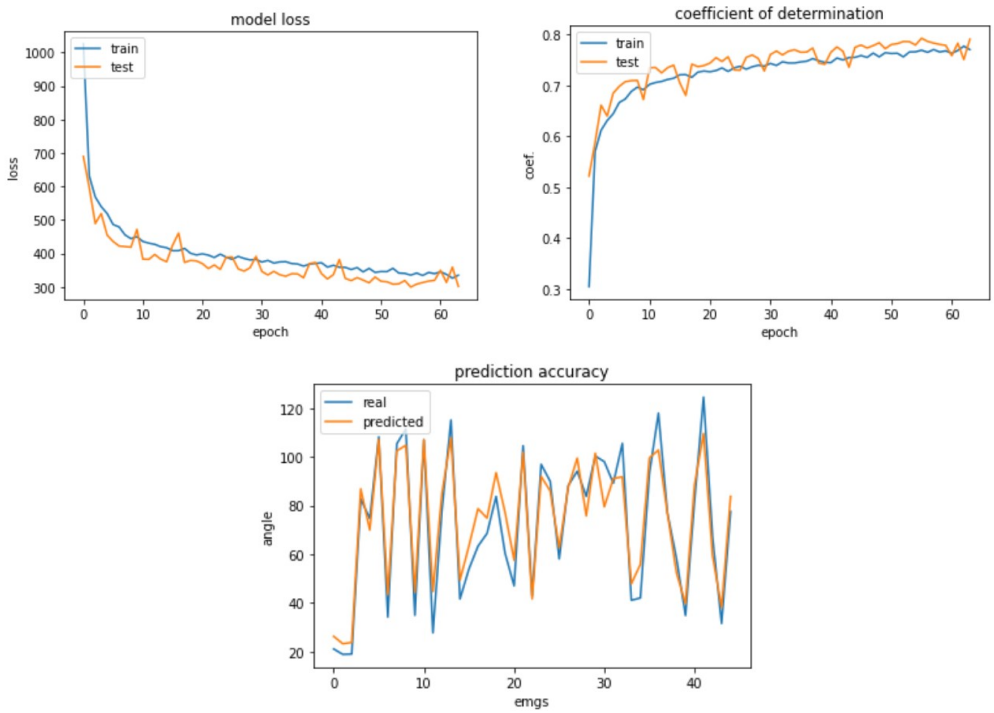

Figure 4.3: Loss, coefficient of determination and predicted angle plotted for scenario 3.

Out of all configurations, this was the one with the best results, as can be seen in figure 4.3. The loss presented a lower value of nearly 300. Unlike in the other scenarios analysed so far, this value did not converge early but kept decreasing over time. This decrease was more prominent in the early epochs and then slowed down but was never zero. The R2 method also reached a new record of almost 0,8. After only ten epochs, it was already 0,7, which was better than both the other scenarios. When looking at the comparison between the predicted and the real angles, the results are also favourable. One can observe that the model understands the tendencies around the motion shifts and most fundamental values. The mean absolute error also presented a great result of 13,03.

4.1.4 Scenario 4

Finally, the fourth and last scenario takes advantage of a large window (50) and a deeper neural network. The overall results represented in 4.4 are very satisfying.

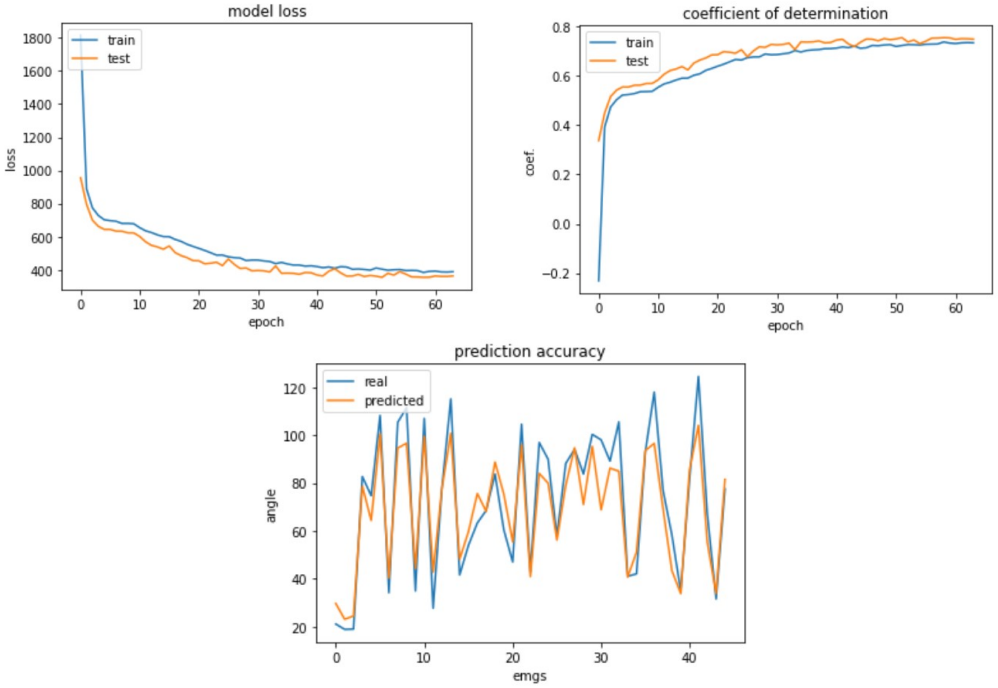

Figure 4.4: Loss, coefficient of determination and predicted angle plotted for scenario 3.

The loss did not decrease as much as in scenario three, but it did improve if we compared to scenarios one and two. Twenty epochs are when the loss starts stabilizing, only finishing at a loss of 400. The coefficient determination variation is very similar to the one in scenario 3, just a little lower at 0,75. The estimated angles are very close to the desired result. For this scenario, the mean absolute value was 14,45.

4.1.5 Discussion

After carefully examining the results obtained in each scenario, it is possible to draw some relevant conclusions. This experiment aimed to study the impact of the window size and different network configurations on the regression model performance. Using three different metrics, loss, coefficient determination and mean absolute error, makes it easier to compare how each variable affected the solution. The Table 4.1 sums up the obtained values for each metric.

Table 4.1: Supervised Learning Metrics.

Scenarios	Loss	R^2	Mae
Scenario 1	550	0,618	17,666
Scenario 2	755	0,477	22,34
Scenario 3	301	0,79	13,03
Scenario 4	363	0,749	14,46

Regarding the window size, all results pointed to the conclusion that providing a broader context to the neural network resulted in better model performance. Similarly, a wider network architecture achieved more satisfying results in every scenario. However, it is essential to note that both types of networks had very similar performance when operating with broader window sizes.

To sum up, we can conclude that the combination of factors used in scenario 3 (wider network and window size of 50) was the most successful. This information proved itself useful when approaching the problem from a Reinforcement Learning perspective that uses this data as the agent observation.

4.2 Reinforcement Learning

To better analyse the results obtained when using Reinforcement Learning, this section will approach each algorithm and technique individually before moving on to the final discussion.

4.2.1 DQN

Using Deep Q-Network was the most straightforward approach to tackle the problem with DRL. Figure 4.5 shows the difference between the angles corresponding to the EMG signals and the DQN model estimations. It displays this relation in two different stages of training, one in the beginning and one in the end. The horizontal axis in the graph represents time as the collection of the individual time steps in which the agent made a decision.

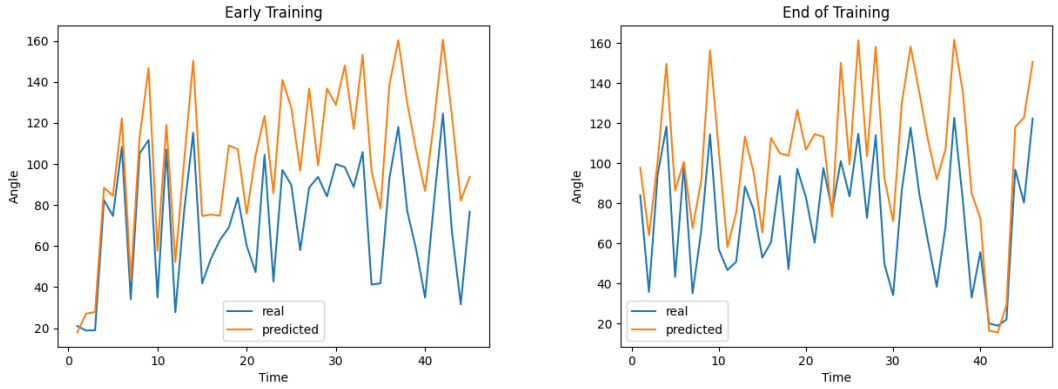

Figure 4.5: Plot of the beginning of the training and the end of the training for DQN, respectively.

By looking at said graphs, it is possible to see that the model frequently overestimates the desired angle. The agent performs a series of over corrections when relaxing the arm (increasing the amplitude) and under corrections when contracting. This results in a prediction line that is constantly above the target line, rarely overlapping or even intersecting. At some stages of training, this difference reached values between 50 to 60 degrees. This problem was recurrent as it was noticeable throughout the whole training process. In spite of this persistent error, it is evident that

the agent knows when to change the direction of the movement. That being said, the solution was flawed in solving the problem.

4.2.2 DDPG

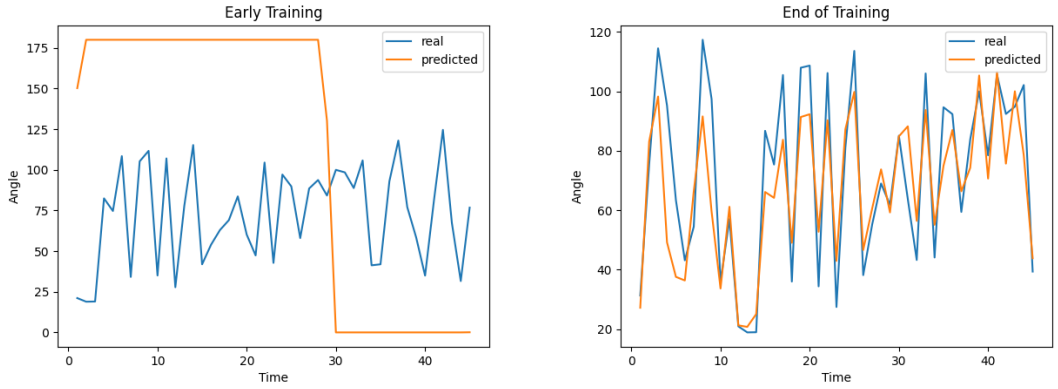

Figure 4.6: Plot of the beginning of the training and the end of the training for DDPG, respectively.

As seen in the graph 4.6, the DDPG model was able to learn the task. However, it is clear that this ability evolved over time. At the beginning of the training, the agent tended to fully relax the arm, insisting on maintaining an amplitude of 180 degrees. While still in the first episode, the agent made an aggressive over-correction to 0 degrees. As time went by, the agent started to correct its predictions more carefully. By looking into the comparison between the predicted angles and the actual recorded motion, it is possible to see that the model eventually learns how to follow the general gesture - relaxing or contracting the muscle. Shifts in direction (indicated by the local extremes in the graph) seem to throw off the model as the distance between the lines tends to increase after some peaks. However, the DDPG model appears much more capable when compared to DQN. The two lines in the graph overlap and cross multiple times by the end of training. Moreover, the most significant difference between the angles was around 25 degrees.

4.2.3 Optimised DDPG

The results obtained with the optimisation of DDPG for sample efficiency appear to be fruitful, as can be seen in 4.7. As the graph suggests, the model learns faster as it starts to deliver favourable results sooner. Even at the beginning of training, the predictions are more similar to the fully trained regular DDPG model than its early training version. By the end of the training, the optimised DDPG delivered the best results of all the approaches. The lines that represent the angles overlap frequently; both change directions at the same time step, and even though the value is not always the same, the most significant difference is only around 15 degrees. When testing the model, the results were still very satisfying. It understands the tendencies but sometimes evaluates the value wrongly.

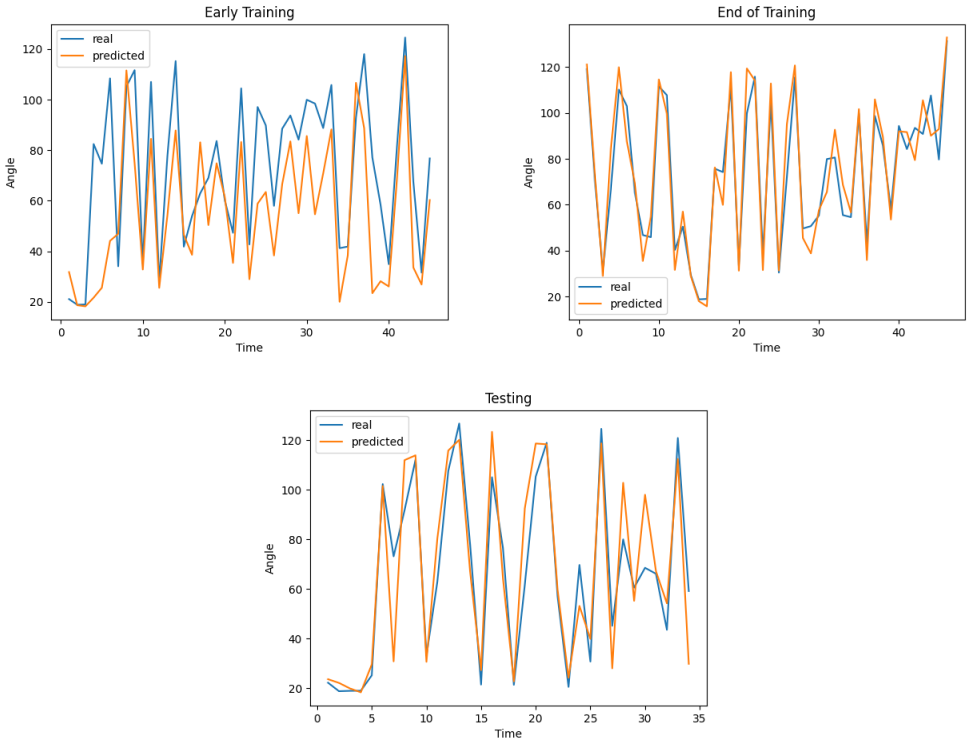

Figure 4.7: Plot of the beginning of the training, the end of the training and testing for optimised DDPG, respectively.

4.2.4 Discussion

Using a non-periodic trial for training, presenting irregular contractions and relaxations is an essential aspect since the agent depends on the EMG signals to predict the arm behaviour. This would not be the case if the angles in the trial could be described with a function similar to a sinusoidal function making it a lot easier to predict.

Table 4.2: RL Cumulative Rewards.

Models	Cumulative Reward
DQN	1650
DDPG	4729
DDPG Optimised	5780

The rewards obtained can be used to compare the performance between models. As it is possible to see from the results Table 4.2, using DDPG combined with the HER technique yielded the best results. The rewards can be used to compare the performance between models. This small modification proved itself extremely useful and provided much better results. For this reason, the generalisation attempts relied on this approach.

4.3 Generalisation

4.3.1 Results

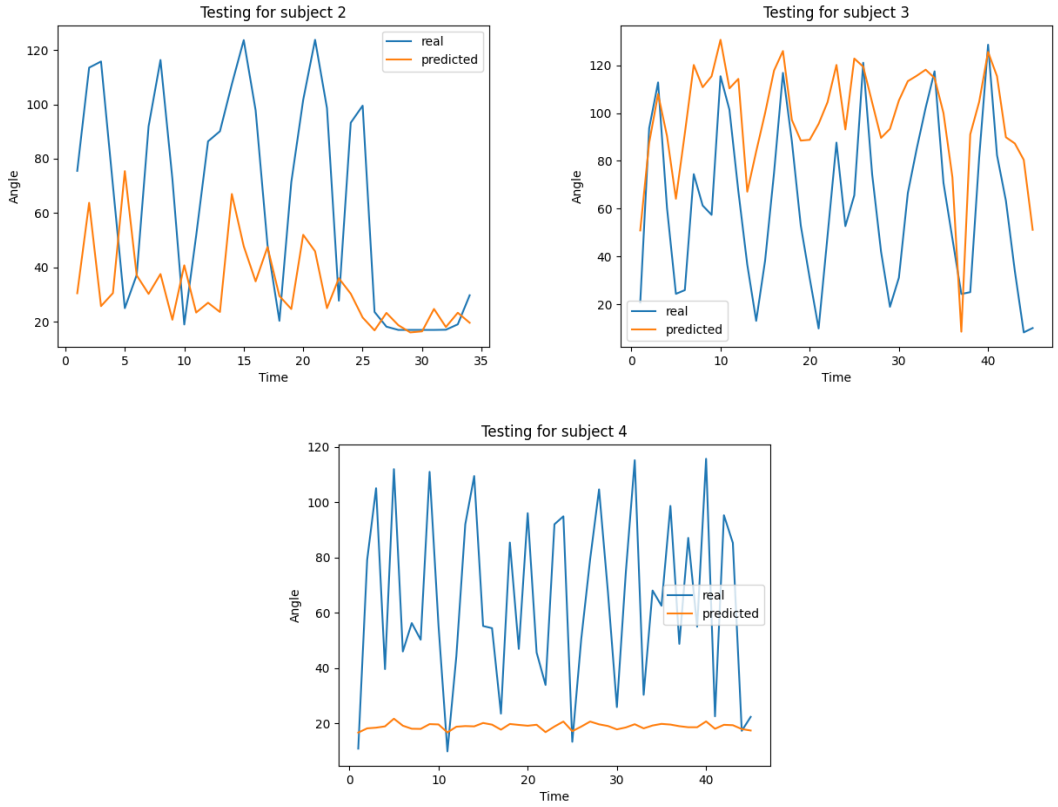

Figure 4.8: Plot of the beginning of the training, the end of the training and testing for optimised DDPG, respectively.

In Figure 4.8 it is possible to examine the generalisation performance of the model. As established, the model was first trained with the data from Subject 1. Afterwards, the EMG signals from 2 other subjects were used in the learning process, and the remaining subject was used for testing.

Starting with Subject 2, it is hard to find a correlation between the two lines in the graph. Although sometimes the lines seem to increase and decrease simultaneously, the magnitude of these changes is entirely unrelated. On other occasions, the lines behave as complete opposites.

Proceeding to the generalisation of Subject 3: the model's predictions appear to follow the movement regarding direction; however, the prediction line is almost always above the actual line, and the difference between the two can be very severely - reaching up to 80 degrees.

Finally, the prediction of S4 was very unsatisfying. The agent was not able to extrapolate any useful information from the input signal, resulting in a lousy prediction.

4.3.2 Discussion

After examining the generalisation performances, it is possible to state that the most favourable was the prediction of Subject 3. However, there is no relation between the three predictions, as the agent behaviour changed completely for each trial.

The two worse subject estimations (subject 2 and subject 4) both trained with subject 1 and subject 3. While subject 1 has already proven itself useful in previous experiments (supervised and Reinforcement Learning), this was the first use of subject 3. This could provide a possible explanation for such poor predictions.

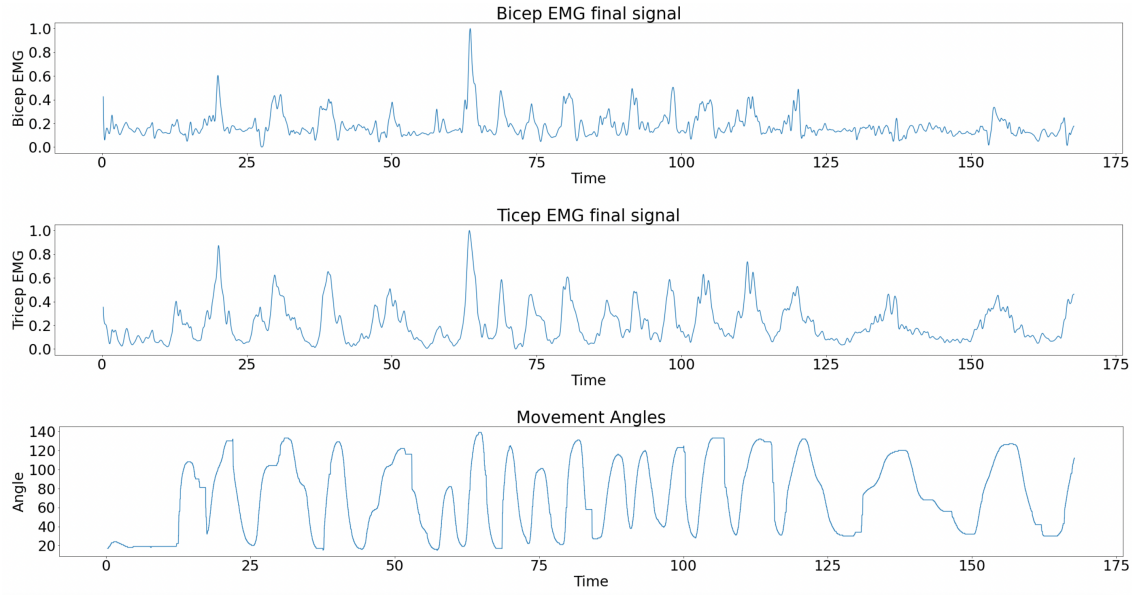

Figure 4.9: Plot of the EMG signals of the bicep, tricep and the arm angle for subject 1.

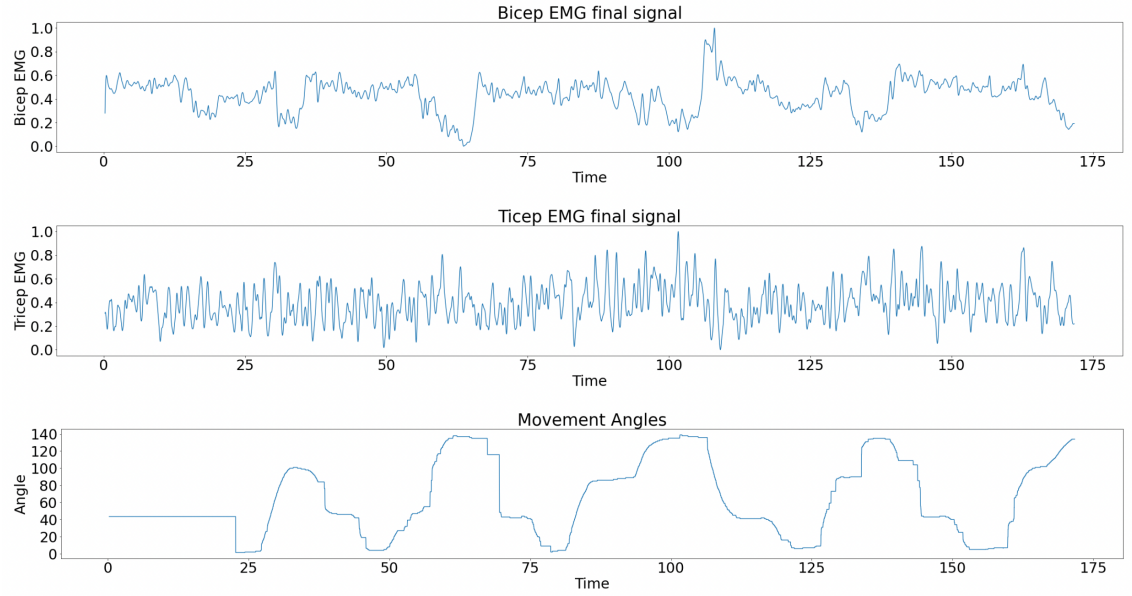

Figure 4.10: Plot of the EMG signals of the bicep, tricep and the arm angle for subject 3.

As can be seen in Figures 4.9 and 4.10, while the EMGs collected from subject 1 have a clear correlation to the angle, the readings from subject 3 appear a lot more inconsistent. There could be a lot of reasons for this discrepancy between the EMGs and the angle. For instance, the subject could have an amputated upper limb and, therefore, an EMG with less prominent oscillations. However, these results could simply be caused by the natural variations in EMGs from person to person. The different processed EMGs of each subject can be found in the appendix A.1.

4.4 Final Discussion

The supervised learning approach was very successful, not only it produced a working model with reasonably good results, but also it provided useful information that influence the rest of the experiments. Besides showing that the dataset and its preprocessing were usable to train a predictive model, it also allowed the study of the best window size and the preferable network layout.

With respect to the Reinforcement Learning methods, the DDPG agent was the most successful, but even more so when provided with the HER optimisation technique.

Finally, although it seemed promising, the application of this algorithm in the generalisation scenarios yielded subpar results. This inadequacy may be due to the model's inability to find resemblances between such distinct EMG signals. A way to tackle this could be to train with a large number of subjects. This way, even though there would be a lot a variability in the inputs, the agent might be able to pick-up on some of the similarities.

Chapter 5

Conclusion

The development of well-functioning powered prosthetics can cause a tremendous positive impact on the well-being, autonomy and overall quality of life of their users. A revision of the state-of-the-art efforts in this direction reveals that different techniques can be employed to approach the problem. This study focused on bringing the advantages of Deep Reinforcement Learning to this context. Starting with an analysis of the data used as input, the methodology relied on multiple strategies to preprocess the EMG signals. This allowed the transformation of raw data into useful, less noisy information that could be used to extrapolate valuable knowledge. A supervised learning approach followed this successful stage.

So as to improve the experimental setup, a regression model was developed to study how the network architecture and data format would benefit the results. The model showed its best performance when using a wider network and a window size of 50 (coefficient of determination of almost 80%). This valuable information was considered when developing the DRL methods.

For the DRL problem formalisation, two different algorithms were used: Deep Q-Network and Deep Deterministic Policy Gradient. Using these two methods allowed the comparison of a value-based and a policy-based approach that relied on similar practices and architectures. The DDPG model showed the best results. The solution also explored state-of-the-art sample efficiency techniques to improve learning under sparse reward settings. With this in mind, the Hindsight Experience Replay method was adopted in the training process. Finally, there was also a generalisation attempt. In this stage, the data collected from three subjects was used to predict the movements of another unseen individual. Multiple training and testing combinations were analysed.

The proposed solution succeeded in facing the problem of prosthetic movement as a continuous control problem. Moreover, the models exhibited good performance, especially when relying on the sample efficiency optimisation. The agent clearly learned the motion it was supposed to execute with little delay to changes in direction. With respect to the generalisation process, the performance did not meet the expectations though. Therefore, further research should be done by relying on more accurate data collected from a larger pool of subjects.

5.1 Main Contributions

The work described aimed to contribute not only to the field of powered prosthesis, from the application perspective, but also to the Deep Reinforcement Learning domain. As such, the contributions include:

- Implementing an EMG data preprocessing pipeline that sought to produce less noisy and more valuable signals. This part of the work could be helpful in other fields besides ML.
- Exploration of the effect of different network architectures and window sizes on the model's performance.
- The adaptation and use of the same dataset in both RL and supervised learning experiments.
- Exploratory work on the adoption of DRL procedures in a continuous control problem.
- Extending traditional methodologies in the literature by developing value-based and policy-based algorithms for comparison.
- Implementation and result discussion of optimisation techniques such as HER in a practical problem.
- Practical generalisation experiment with Reinforcement Learning.
- Experimental work with empiric evaluations.

5.2 Future Work

Even though this project can be considered successful within the boundaries of the proposed research methodology, there are always things that can be done to improve or even magnify its significance. For starters, capturing the EMGs could significantly improve the results obtained. The input used to test the models was an open-source dataset found online. This means there was little information about how they were acquired and the subjects associated with the generated data. On top of that, the signals described in the dataset were noisy. In an attempt to retrieve the best quality EMGs possible, the first stages of this work consisted of performing a substantial amount of preprocessing. When a signal is filtered to eliminate the noise, some important EMG information may be lost. The acquisition of the signals in a controlled environment could be a way to minimise the amount of noise introduced. This improvement would not only enhance the initial quality of the input as well as reduce the necessary computation power.

The next step for this work to be helpful in practical scenarios would be to adapt it to real-time constraints. More specifically, one could provide a stream of real-time EMG readings. The signal would be acquired while a subject moves their arm, making it possible to assess if the virtual prosthetic mimics the gesture and its delay and precision. Finally, by taking the work even further, one could recreate these tests with an authentic robotic arm.

To have a successful real-time prosthesis, it should have little to no delay. One option for this to be possible would be to include the preprocessing in the Reinforcement Learning pipeline. Furthermore, resorting to a Convolutional Neural Network architecture would allow the model to extract features directly from the signals without needing hand-engineered techniques like the current solution. This approach would create an end-to-end solution that would acquire the data, process it and then run it through the algorithm to get an appropriate reaction.

Although it is a compelling experience, further improvements are necessary to accomplish generalisation. As the EMGs used appeared overly dissimilar, this could be one of the reasons for the subpar results. This heterogeneity in data once again stresses the importance of acquiring the data under a controlled set of conditions.

The final and most crucial stage would be testing the model with an upper limb amputee or someone with a non-functional upper limb. It is common for people who lost a body member to show different EMGs from the others since, after some time, the body loses the ability to send meaningful signals to the muscles. Consequently, in an amputee's EMG, there is not such an obvious connection between the arm angle and the EMG values. Evidently, this depends on the person and how much time has passed from losing the body part mobility.

References

[1] S. Akwei-Sekyere. Powerline noise elimination in biomedical signals via blind source separation and wavelet analysis. *PeerJ*, 3:1086, July 2015.

[2] M. R. Al-Mulla, F. Sepulveda, and M. Colley. A review of non-invasive techniques to detect and predict localised muscle fatigue. *Sensors*, 11(4):3545–3594, March 2011.

[3] O. Alagoz, H. Hsu, A. J. Schaefer, and M. S. Roberts. Markov decision processes: A tool for sequential decision making under uncertainty. *Medical Decision Making*, 30(4):474–483, December 2009.

[4] N. Amrutha and V. H. Arul. A review on noises in emg signal and its removal. *International Journal of Scientific and Research Publications*, 7(1):27, May 2017.

[5] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. 30:11, July 2017.

[6] M. Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, May 2016.

[7] T. Chai and R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)? – arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250, June 2014.

[8] A. Chalard, M. Belle, E. Montané, P. Marque, D. Amarantini, and D. Gasq. Impact of the EMG normalization method on muscle activation and the antagonist-agonist co-contraction index during active elbow extension: Practical implications for post-stroke subjects. *Journal of Electromyography and Kinesiology*, 51:102403, April 2020.

[9] L. Chen, T. Guo, Y. Liu, and J. Yang. Survey of multi-agent strategy based on reinforcement learning. In *2020 Chinese Control And Decision Conference (CCDC)*. IEEE, August 2020.

[10] B. Dellon and Y. Matsuoka. Prosthetics, exoskeletons, and rehabilitation [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):30–34, March 2007.

[11] eMathZone. Positive and negative correlation. https://www.emathzone.com/tutorials/basic-statistics/positive-and-negative-correlation.html. Accessed: 2021-06-02.

[12] S. Feike. Multiple time series classification by using continuous wavelet transformation. https://towardsdatascience.com/multiple-time-series-classification-by-using-continuous-wavelet-transformation-d29df97c0442, January 2020. Accessed: 2021-06-01.

[13] F. Fischer, M. Bachinski, M. Klar, A. Fleig, and J. Müller. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports*, 11(1):14445, July 2021.

[14] P. F. P. Fonseca. Validation of two types of textile electrodes for electrocardiography and electromyography measurement applications. Master's thesis, FEUP, Porto, Portugal, 2012.

[15] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[16] Z. Gao, R. Tang, L. Chen, Q. Huang, and J. He. Continuous shared control in prosthetic hand grasp tasks by deep deterministic policy gradient with hindsight experience replay. *International Journal of Advanced Robotic Systems*, 17(4):172988142093685, July 2020.

[17] C. Gorino. A short history of prosthetics. https://synergypo.com/blog/a-short-history-of-prosthetics/, September 2020. Accessed: 2022-02-05.

[18] E. Granados. Granados et al., estimation of elbow flexo-extension angle using semg. data: Biceps/triceps semg recording with angle measurement and estimated angle., 2017. Accessed: 2022-02-04.

[19] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. November 2016.

[20] B. Jang, M. Kim, G. Harerimana, and J. W. Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7:133653–133667, 2019.

[21] M. Khezri and M.Jahed. Real-time intelligent pattern recognition algorithm for surface EMG signals. *BioMedical Engineering OnLine*, 6(1):45, 2007.

[22] Z. Khodzhaev. Spectrogram - Practical Guide. Technical report, EU-CELTIC - Project FU5ION, February 2020. Page 2.

[23] L. Kidziński, S. P. Mohanty, C. Ong, J. Hicks, S. Francis, S. Levine, M. Salathé, and S. Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In Sergio Escalera and Markus Weimer, editors, *NIPS 2017 Competition Book*. Springer, Springer, 2018.

[24] P. Konrad. The abc of emg. *A practical introduction to kinesiological electromyography*, 1, January 2005.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. September 2015.

[26] C. J. Luca, L. D. Gilmore, M. Kuznetsov, and S. H. Roy. Filtering the surface EMG signal: Movement artifact and baseline noise contamination. *Journal of Biomechanics*, 43(8):1573–1579, May 2010.

[27] J. MacDonald. A brief history of prosthetic limbs. https://daily.jstor.org/a-brief-history-of-prosthetic-limbs/, July 2017. Accessed: 2022-02-15.

[28] E. Massa, M. Jonker, K. Roes, and A. Coolen. Correction of overfitting bias in regression models. April 2022.

[29] D. Maulud and A. M. Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1(4):140–147, December 2020.

[30] L. Mesin. Crosstalk in surface electromyogram: literature review and some insights. *Physical and Engineering Sciences in Medicine*, 43(2):481–492, May 2020.

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *ICML 2016*, 48:1928–1937, February 2016.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. December 2013.

[33] D. H. Nam. Electromyography. https://encyclopedia.pub/entry/7298, 2021. Accessed: 2022-04-01.

[34] M.A. Oskoei and H. Hu. Support vector machine-based classification scheme for myoelectric control applied to upper limb. *IEEE Transactions on Biomedical Engineering*, 55(8):1956–1965, August 2008.

[35] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. page 10, May 2017.

[36] LeTourneau Prosthetics. Prosthetics – an education of artificial limbs and limb replacement for amputees. https://www.llop.com/prosthetics/. Accessed: 2022-02-06.

[37] S. Rand and V. Vanodia. Upper limb amputations. https://now.aapmr.org/upper-limb-amputations/, January 2013. Accessed: 2021-11-30.

[38] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin. Techniques of EMG signal analysis: detection, processing, classification and applications. *Biological Procedures Online*, 8(1):11–35, December 2006.

[39] E. Rocha, D. Cantergi, A. Bonezi, D. Soares, C. Candotti, and J. Loss. Smoothing emg signals: Implications on delay calculation. *Revista Portuguesa de Ciências do Desporto*, 12:60, January 2012.

[40] F. A. B. Sampaio. Low-cost simultaneous and proportional myoelectric control of powered upper limb exoskeletons. Master's thesis, FEUP, Porto, Portugal, March 2021.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. page 12, July 2017.

[42] S. Shen and C. Mei. Estimation of the variance function in heteroscedastic linear regression models. *Communications in Statistics - Theory and Methods*, 38(7):1098–1112, March 2009.

[43] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. December 2017.

[44] Y. Smirnov, D. Smirnov, A. Popov, and S. Yakovenko. Solving musculoskeletal biomechanics with machine learning. *PeerJ Computer Science*, 7:e663, August 2021.

[45] M. Srivastava. Improving signal resolution and reducing experiment time in electron spin resonance spectroscopy via data processing methods. Master's thesis, Cornell University, Ithaca, NY, August 2018.

[46] T. Triwiyanto, I. Pawana, B. Irianto, T. Indrato, and I. D. H. Wisana. Embedded system for upper-limb exoskeleton based on electromyography control. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(6):2992, December 2019.

[47] N. N. Unanyan and A. A. Belov. Design of upper limb prosthesis using real-time motion detection method based on EMG signal processing. *Biomedical Signal Processing and Control*, 70:103062, September 2021.

[48] M. Veskovic, A. Vulovic, D. Vujicic, B. Popovic, and M. Milutinov. Precision full-wave rectifier - practical realization in discrete technology. In *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–5. IEEE, March 2020.

[49] A. Vieira. Clinical reabilitation of upper limb in chronic stroke in portugal-a cross sectional survey. *International Journal of Physiotherapy*, 3(1):124–131, February 2016.

[50] Y. Wang, K. S. Won, D. Hsu, and W. S. Lee. Monte carlo bayesian reinforcement learning. 2:8, June 2012.

[51] Wikipedia. Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network. Accessed: 2021-06-01.

[52] P. Wojtczak, T. G. Amaral, O. P. Dias, A. Wolczowski, and M. Kurzynski. Hand movement recognition based on biosignal analysis. *Engineering Applications of Artificial Intelligence*, 22(4-5):608–615, June 2009.

[53] D. Zhang. A coefficient of determination for generalized linear models. *The American Statistician*, 71(4):310–316, October 2017.

[54] Q. Zhang, R. Hosoda, and G. Venture. Human joint motion estimation for electromyography (emg)-based dynamic motion control. *IEEE Engineering in Medicine and Biology Society. Conference*, 2013:21–24, July 2013.

Appendix A

Further Results

This chapter aims at showing further results obtained with the preprocessing techniques and reinforcement learning algorithms.

A.1 EMG Preprocessing

In this section, we can find the EMG signals plotted after the preprocessing phase for each agent. There will also be a plot with the arm angle, so it is easier to correlate to the EMG changes.

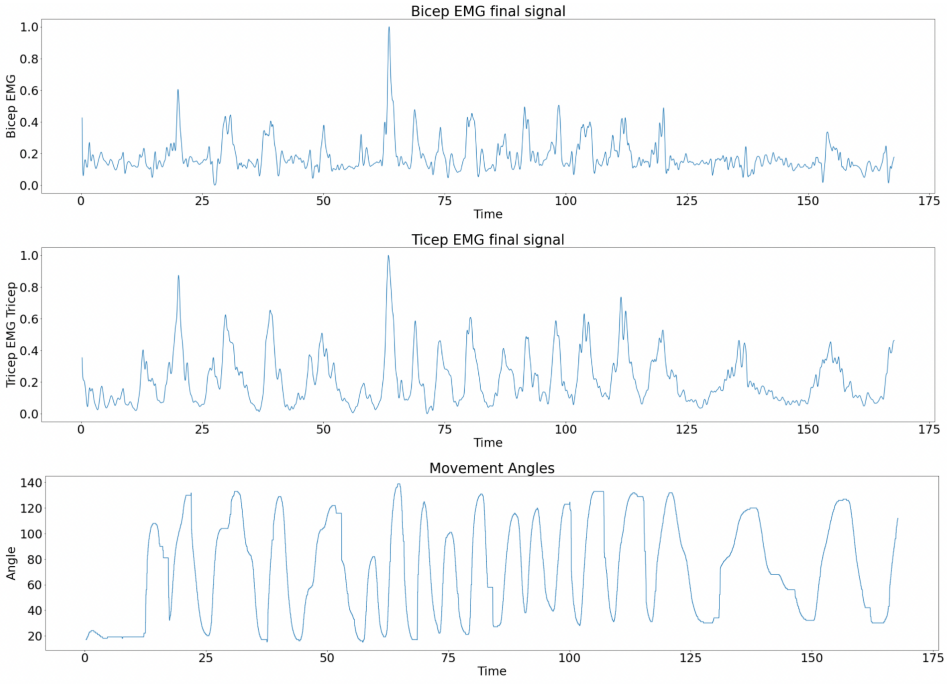

Figure A.1: lot of the EMG signals of the bicep, tricep and the arm angle for subject 1.

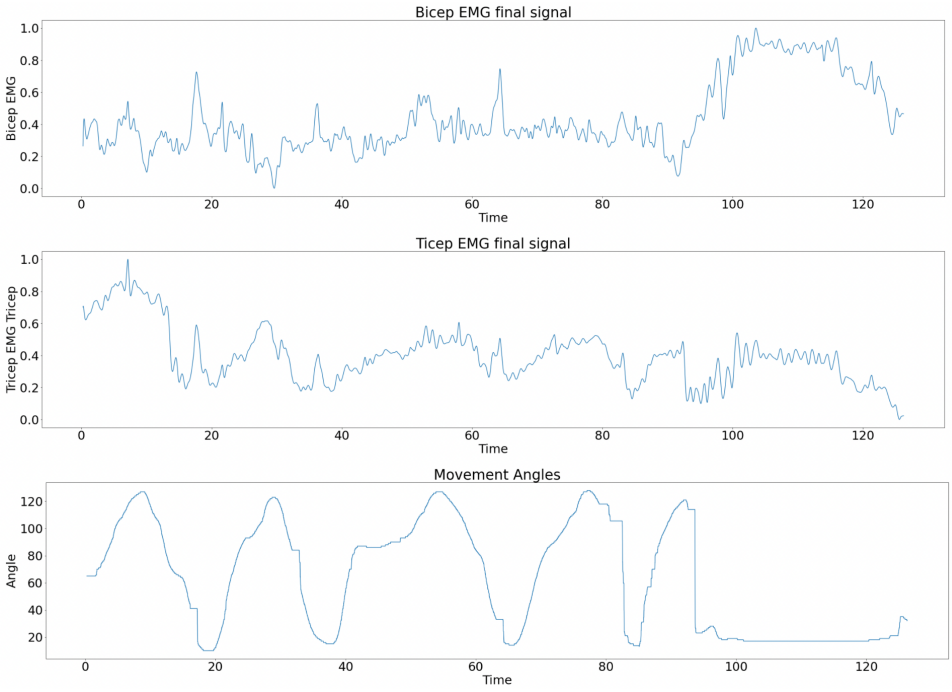

Figure A.2: lot of the EMG signals of the bicep, tricep and the arm angle for subject 2.

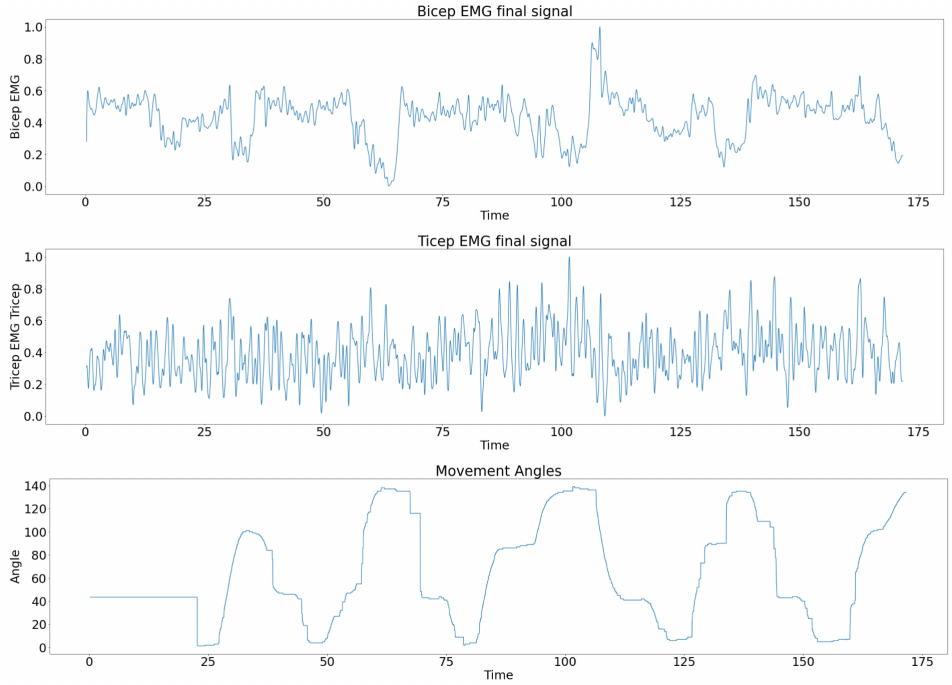

Figure A.3: lot of the EMG signals of the bicep, tricep and the arm angle for subject 3.

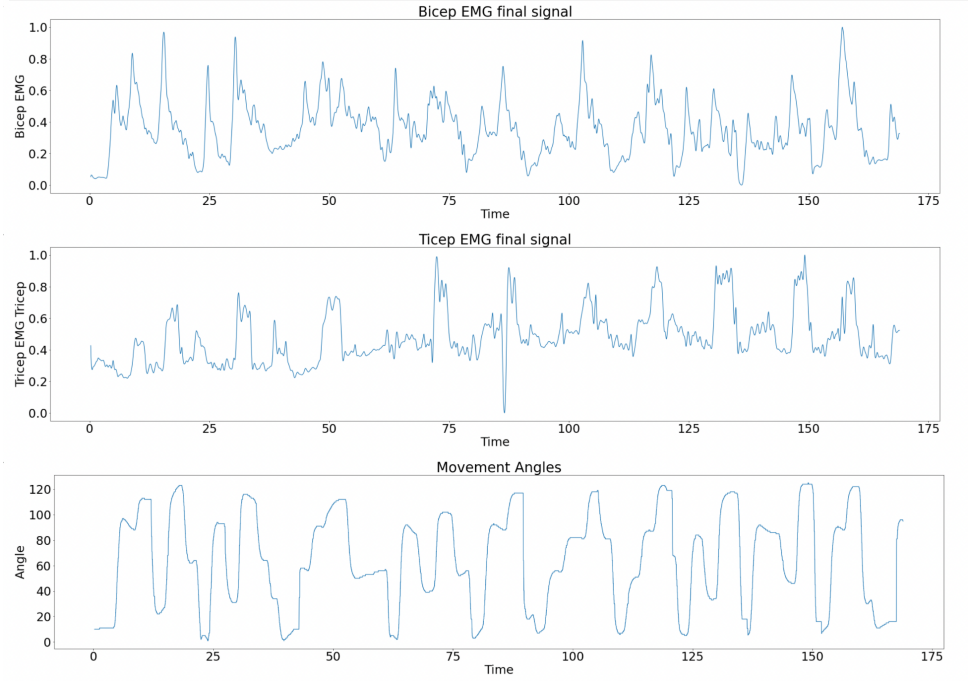

Figure A.4: lot of the EMG signals of the bicep, tricep and the arm angle for subject 4.

A.2 RL Rewards

This section shows the plot of the received reward in each time step of the Reinforcement Learning algorithms.

A.3 Generalization Rewards

This section shows the reward plotted for every generalization scenario.

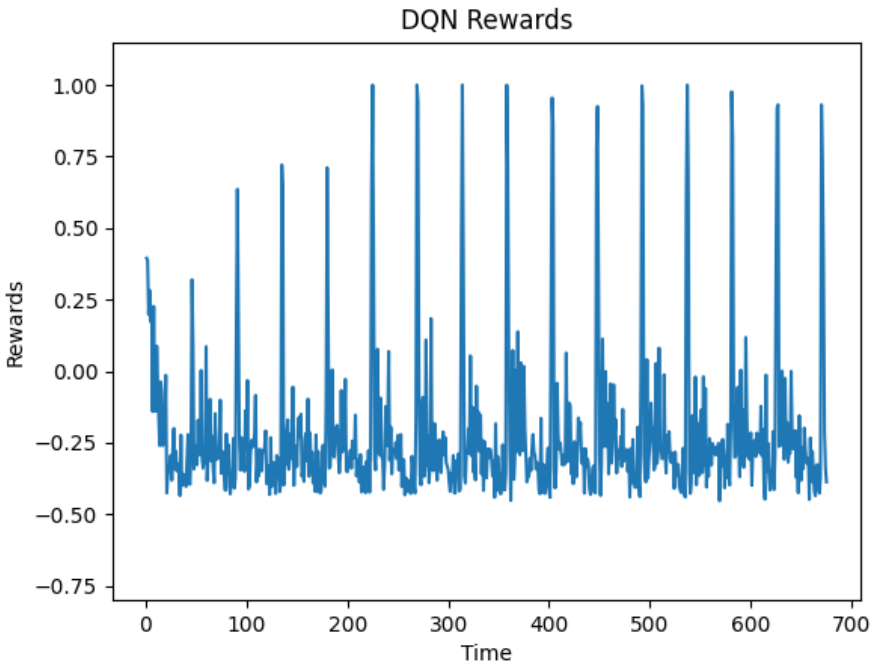

Figure A.5: Rewards received with DQN algorithm.

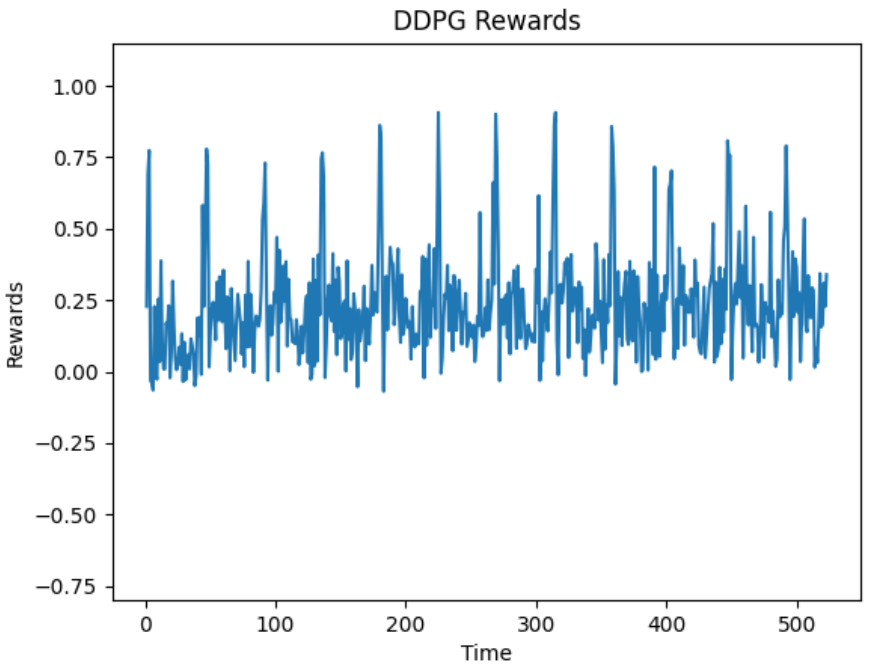

Figure A.6: Rewards received with DDPG algorithm.

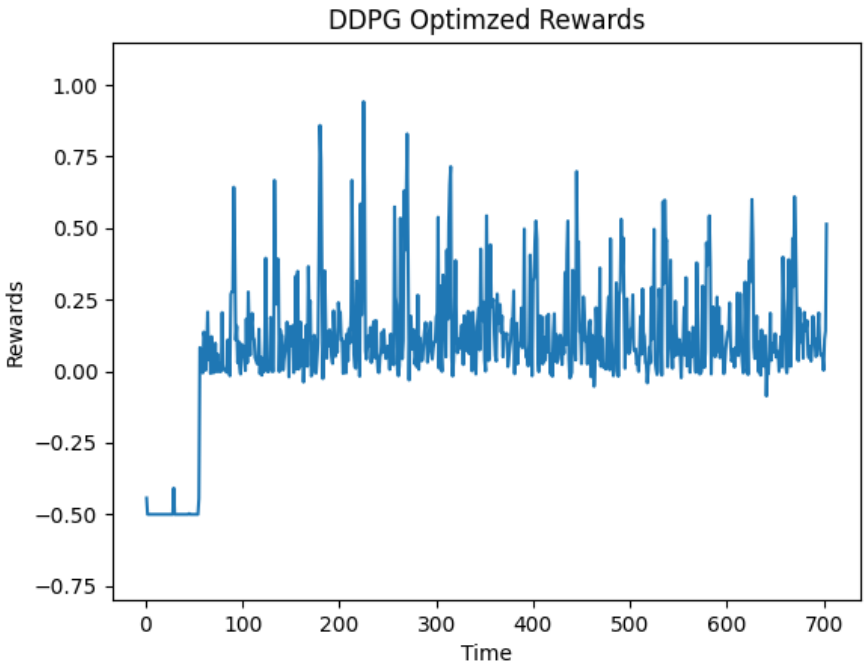

Figure A.7: Rewards received with an optimized DDPG algorithm.

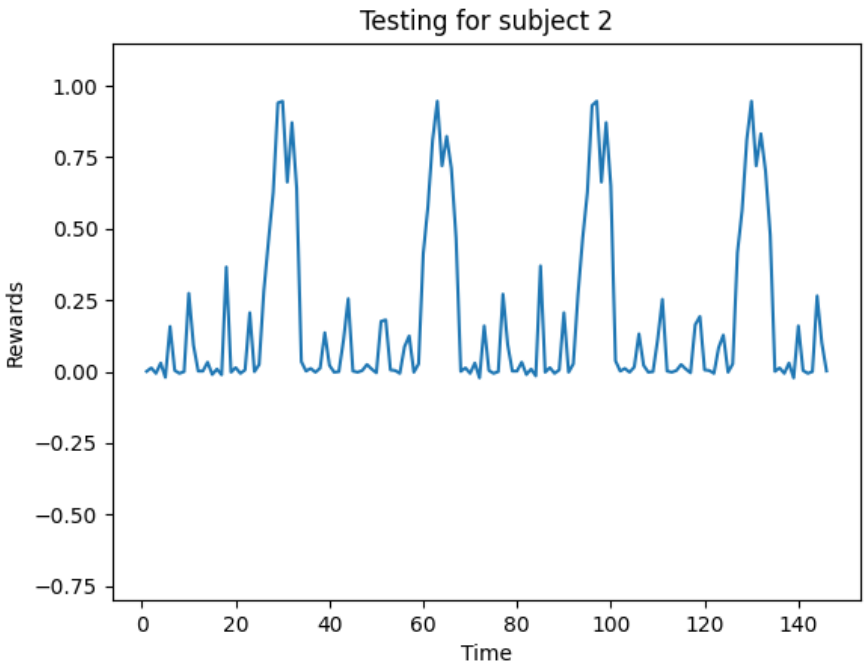

Figure A.8: Rewards received while testing for subject 2.

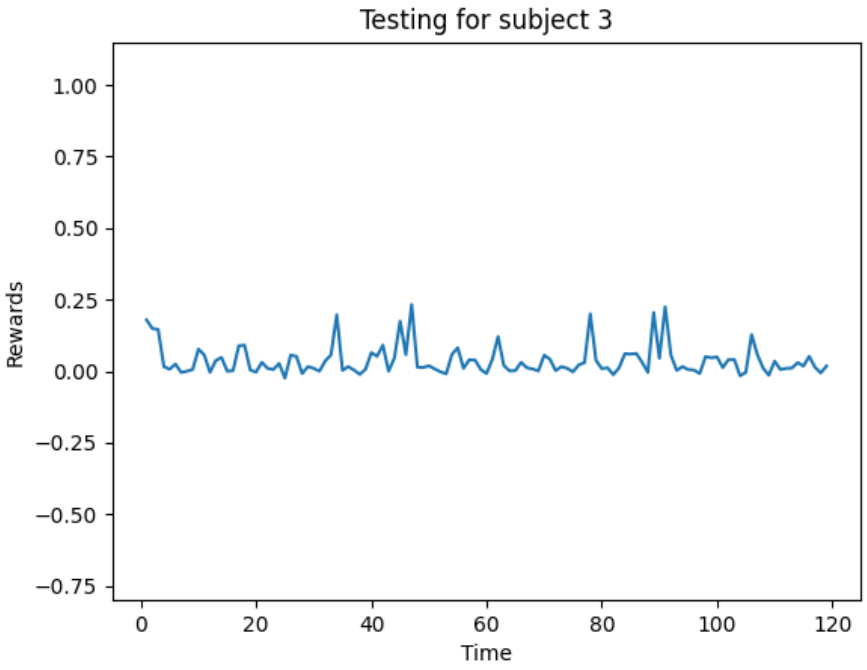

Figure A.9: Rewards received while testing for subject 3.

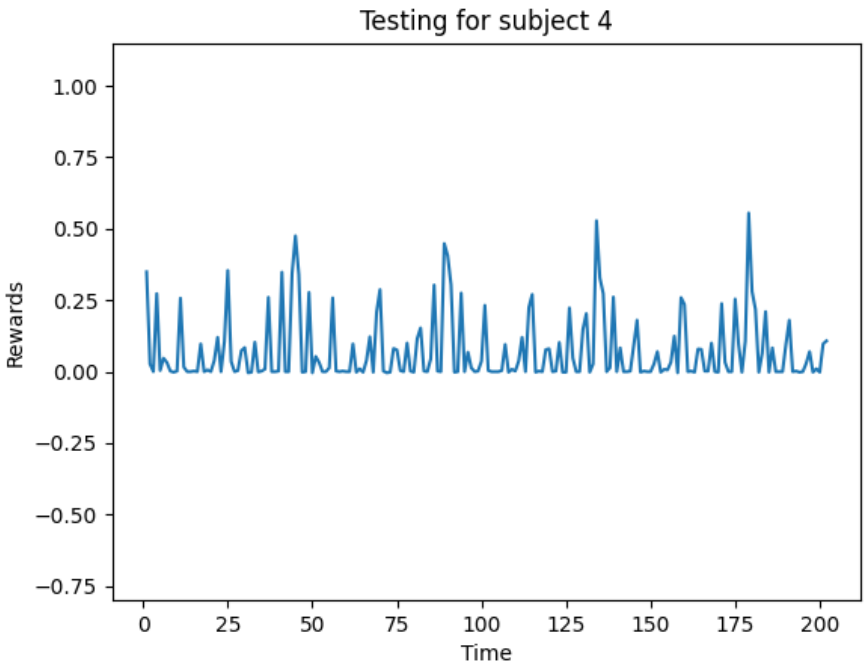

Figure A.10: Rewards received while testing for subject 4.