FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FCPortugal Multi-Robot Action Learning

Ventura de Sousa Pereira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Paulo Reis

December 22, 2020

FCPortugal Multi-Robot Action Learning

Ventura de Sousa Pereira

Mestrado Integrado em Engenharia Informática e Computação

December 22, 2020

Abstract

Simulated robotic soccer is a multi agent system environment, a field where there are still questions to answer.

Within the referred environment there are certain game situations, the set plays, that could be better explored in order to find ways to score goals. In fact, these game situations happen rather frequently and it is a waste to just concede possession or goals from them. So, facing a competitive multi-agent environment such as the previously mentioned, we intend to design, implement and test a learning engine that will allow a group of humanoid robots to perform a set of individual coordinated actions in order to execute a global plan. However we can't put aside the fact that the environment is dynamic and that we need to minimize the hypothesis of being exploited by adversaries. We can, thus, formally say that the problem is finding an efficient mechanism, in the context of a MAS environment, to make the humanoid robots learn coordinated behaviors to gain advantage in the game. In order to do that we will be using the already developed low level skills, as well as the FCPGym toolkit and a deep learning framework, Stable Baselines.

Even though some work was already developed regarding the creation of set plays and an interface for the end user was created, there are still some limitations and questions to answer. How can we achieve the desired outcome of a set play on a non stationary environment? How can we improve the learning mechanism of each agent?

Nowadays, there has been some research and advances regarding the multi-agents topics, where it is said that the complexity of the action-space is exponential to the amount of agents. Most of the work done here is applied by adapting the individual agent context to the MAS. This is done by centralizing the learning mechanism or by individualizing each agent. Furthermore, there was relative success with Deep Learning Neural Networks, using the Deep Q-Networks algorithm to the Pursuit Game and Foraging Task, regarding the multi-agent paradigm. So far there has been some interesting work developed within the mentioned fields, applied to the humanoid robot football context. Regarding the set plays, there was developed a strategy planner that allows the creation of a set play. Furthermore, the usage of reinforcement learning within the development of set plays has also been tested by the FCPortugal team, where we manage to increase performance by experience. Lastly, deep learning, in the last couple of years, within the RoboCup, has been tested, especially, for ball localization.

In this work we made use of the TD3, a deep reinforcement learning algorithm, to optimize the developed set plays, which consisted of two variations of a corner kick. We were able to make use of deep learning, which has been growing, and the idea of using past experience, to synchronize the humanoid robots into a chained sequence of actions leading to a goal. The obtained trained models show that the neural network found optimal outputs for the defined action spaces, resulting on goal scoring set plays.

Keywords: Multi-Agent System, Deep Neural Networks, Simulated Robotic Soccer, Deep-Q Networks

Resumo

Futebol robótico simulado é um ambiente de sistema de multi agentes, uma área onde há ainda muitas questões por responder.

No contexto do ambiente referido, há certas situações de jogo, as jogadas de bola parada, que poderiam ser melhor exploradas de forma a encontrar formas de marcar golos. Efetivamente, este tipo de situações de jogo acontecem com relativa frequência e é um desperdício perder a posse de bola ou sofrer golos a partir das mesmas. Desta forma, enfrentando um ambiente competitivo de multi-agentes como o anteriormente referido, pretendemos desenhar, implementar e testar um mecanismo de aprendizagem que permitirá a um grupo de robots humanóides executar uma série de acções individuais coordenadas, de forma a executar um plano global. Contudo, não podemos pôr de lado o facto de que o ambiente é dinâmico e que necessitamos de minimizar as hipóteses de ser explorados pelos adversários. Podemos, então, dizer formalmente que o problema é encontrar um mecanismo eficiente, no contexto de um ambiente MAS, para fazer com que os robots humanóides aprendam comportamentos coordenados que lhes permita ganhar vantagem no jogo. Para atingir o nosso objetivo iremos usar as habilidades de baixo nível já desenvolvidas, assim como a ferramenta FCPGym e uma framework de deep learning, Stable Baselines.

Apesar de algum trabalho já ter sido desenvolvido no que toca à criação de jogadas colectivas e uma interface para o utilizador ter sido desenvolvida, ainda há algumas limitações e questões por responder. Como podemos atingir o desfecho de uma jogada colectiva quando enfrentando um ambiente não estacionário? Como podemos melhorar os mecanismos de aprendizagem de cada agente?

Hoje em dia já há algumas pesquisas e avanços relativamente ao tópico de multi agentes, onde é dito que a complexidade do espaço-acção é exponencial para a quantidade de agentes. A maior parte do trabalho é aplicado através da adaptação do contexto de agente individual ao MAS. Isto é feito centralizando o mecanismo de aprendizagem ou individualizando cada agente. Para além disso, houve relativo sucesso com redes neuronais de aprendizagem profundas, usando o algoritmo Redes Profundas-Q no jogo Pursuit e Foraging Task, no que toca ao paradigma de multi agentes. Houve, também, algum trabalho interessante desenvolvido nas áreas mencionadas, aplicadas ao contexto de futebol robótico. No que toca a set plays, foi desenvolvido um strategy planner onde uma interface fornecia a possibilidade de criar uma jogada estudada. Para além disso, o uso de reinforcement learning, no desenvolvimento de set plays, foi já testado pela equipa FCPortugal, onde o desempenho foi melhorado através da experiência. Por último, deep learning, nos anos mais recentes, dentro da RoboCup, foi testado, especialmente, para localização da bola.

Neste trabalho usamos o TD3, um algoritmo de deep reinforcement learning, para optimizar as set plays desenvolvidas, que consistiram em duas variações de um canto. Fomos capazes de usar deep learning, que tem crescido, e a ideia de usar as experiências passadas, para sincronizar os robots humanóides numa sequência de ações que originam golo. Os modelos treinados obtidos mostram que a rede neuronal encontrou valores óptimos dentro dos espaços de ação definidos, resultando em jogadas estudadas que levam a golo.

Keywords: Sistema de multi-agentes, Redes neuronais profundas, Futebol Robótico Simulado, Redes profundas-Q

Acknowledgments

Time really does fly. It is with that in mind that I recall all the years that have gone by and I find myself thankful for all the experiences and support I have had.

Firstly, I would like to thank my parents, Daniela and Agostinho, for always having believed in me, providing me with all the conditions to succeed and have the best possible results. For all their sacrifices, patience and comprehension. For lifting me up when things got hard and sharing my happiness when it went great. To my brother André and my family, thank you. I could not have made it without you all.

Secondly, I am grateful that I found an amazing group of friends who were responsible for the time passing so rapidly, as they made it worthwhile. More than sharing the same struggles and fears, you all got me going and lifted my spirit up when I needed, you created unforgettable moments and got me missing you all on this tough period of our lives. A special thank you to Luís, Sofia, Miguel, Seixas and Julieta for putting up with the rants and helping me get here.

Lastly, to my supervising professor, Luís Paulo Reis, I want to say thank you and appreciate the availability, the patience and the guidance from the beginning, be it in this document or the developed work. I also want to thank Tiago Silva, a member of the FCPortugal3D team, for his time helping me with the many questions and doubts throughout this dissertation. Furthermore, to the whole FCPortugal3D team, who were always available to give their feedback and helping hand.

Unfortunately, I can not mention every single person that was very important for me, but you all know who you are and a word of appreciation is due.

Thank you all for everything.

Ventura Pereira

"To dare is to do."

Spurs

Contents

List of Figures

List of Tables

Abbreviations

RoboCup	Robot Soccer World Cup
ML	Machine Learning
DL	Deep Learning
DDPG	Deep Deterministic Policy Gradient
PPO	Porximal Policy Opimization
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
A3C	Asynchronous Advantage Actor Critic
A2C	Advantage Actor Critic
TD3	Twin Delayed DDPG
SAC	Soft Actor Critic
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
CREPS	Contextual Entropy Search Policy
DQN	Deep Q-Netowrks
SARSA	State Action Reward State Action

Chapter 1

Introduction

Context

The multi-agent system, often called MAS, is a very interesting field to be explored. In recent years, its paradigm and the desire to, within a competitive or cooperative environment, achieve increases in performance or imitate human-like behaviours have lead to researches and usage of machine learning algorithms that are, as well, growing recently. The RoboCup, Robot World-Cup Soccer being its full name, is an international competition that promotes AI research, as well as Robototic's. Using a 3D Soccer Simulation and NAO robots as its agent unit, it creates a system composed by multiple agents, setting up to be the ideal for the research of cooperation, coordination and learning of both individual or collective skills. This dissertation plans on working on an open problem: the learning of collective skills,set-plays, by a group of humanoid robots, making usage of developed low level skills already implemented by the FCPortugal3D's team, as well as new ones to be developed. It is a goal, furthermore, to optimize such behaviors with deep learning.

1.1 Motivation

The RoboCup promotes an ideal environment for the research of cooperation, coordination and learning of both individual or collective skills. It is this dissertation's goal to increase the performance of the FCPortugal3D's team, by optimizing the humanoids robot's behaviours using the growing subset area of machine learning, deep learning. This problem constitutes a challenge because there is a mix of necessities where there is both a cooperative and competitive environment. Firstly, it has the need for cooperation, because each robot has its own understanding and perception of the environment and function in the field, like a real football match. Therefore, there is a need for communication in order to contribute to the team to achieve its goal - score a goal know when on'es contribution is maximized. However, it also focuses on the possible penalization - suffering a goal. This leads to an exciting work that has two variants, plus the need for coordination. Furthermore, it is expected to offer ease of creation and creativity when creating a possible

sequence of actions, the low level skills. In order to do so, one of the objectives is to integrate the work to be implemented, a multi-robot learning engine, in an existing framework, offering a graphical display for the definition of the expected end result - creation and definition of a set play.

1.2 Objectives

This dissertation aims, mainly, to the ability of successfully creating a set play. It is our goal to make usage of the previously developed work by FCPortugal3D's team, such as the already existing low level skills, and to implement new behaviours. These shall be used to elaborate the sequence of actions, by an end user, forming a set play. Moreover, following the implementation, it is important to integrate engine within a framework, concerning the handing out of a graphical view, allowing the user to find ease and speed on the creation of the desired joint skill. Lastly, we hope to accomplish greater performance on the behaviours we are set to create, by optimizing them with deep learning, making full usage of our robots capacities and potential, integrating this work on the team set to compete.

1.3 Document Structure

This document follows a standard structure, with the chapters being:

- **Introduction** This chapter will introduce the given theme, offering contextualisation for the problem. It will also point out the motivations and goals for the proposed work.
- **State of the work** Throughout this section the related work regarding both the multi-agent system environment and deep learning methods will be explored. The related work regarding this two fields and work developed for RoboCup will be reviwed with some detail.
- **Problem and Proposed Solution** In this chapter the problem will be described with more depth, regarding, also, the approach it is intended to be executed in the future. A plan in regard to the scheduling of tasks composing the to be implemented solution will be outlined.
- **Conclusions** Finally, here the research and appreciations of the work done will be reflected upon. Expected results for the future work will be mentioned.

Chapter 2

State of the Art

2.1 RoboCup

Ever since 1997, there's an annual competition that sets up several countries to test their humanoid robotic teams in a 3D simulation environment, through the course of a football match. Such environment represents an ideal domain for the research of cooperation, coordination and learning engines in a multi-agent system. All the teams are set to develop their robots, with the motivation to achieve better and better performance. The possible performance increase derivates from the optimization of - or implementation of new ones - single actions, called low level skills, such as locomotion, kicking, treatment of collisions, dribbling, among others. Furthermore, over the last few years deep learning is a machine learning's subfield that is being used to achieve success in many areas. The utilisation of its algorithms, that uses multiple hidden layers of artificial neural networks to infer knowledge, is proving to be very effective and replaces long hours of coding for autonomous learning engines for complex behavirous. In the next section we will review some of the related work to these two fields, MAS and Deep learning.

2.1.1 The evolution of RoboCup

The first mention of robots playing football was in a paper called "On Seeing Robots" - [30], by the Professor Alan Mackworth, of the University Of British Columbia, Canada, which was presented in 1992 and published later on in 1993. Following this idea an independent workshop about artificial intelligence was done by a group of Japanese researchers, which led to a serious debate about using football as a way of promoting technology and science. Subsequent to this discussion a series of studies were made apace with an early draft of rules, robots and simulator systems. In 1993, some investigators, such as Yasuo Kuniyoshi, Hiroaki Kitano, Minoru Asada, launched a robot competition - Robot J-League. Due to the overwhelming international feedback the initiative was extended to an international joint project, being renamed to the Robot World

Cup Initiative or "RoboCup", for which it is known today. Parallel to this initiative some multi-agent related researches were already using soccer as a domain for the problem. Furthermore, a simulator started being developed dedicated to football matches which ended up to be the official football server of RoboCup. The robots, developed by the professor Manuela Veloso, the student Peter Stone and the Professor Minoru Asada's Lab at Carnegie Mellon University and Osaka university - explained in [8], respectively, were of major importance for this initiative. In the meantime, the first public show off of the simulator was made at IJCAI-95 - [18]. Eventually, in 1995, throughout an Internation Joint Conference on AI, at Montreal, Canada, the first ever Robot World Cup Soccer Games and conferences were announced with IJCAI-97, [14]. The call to organize the Pre-RoboCup-96 was made to identify possible problems with organizing RoboCup on a large scale. Finally the year 1997 marked a turning point regarding artificial intelligence and robotics. Alongside with the IBM Deep Blue, a computer system who beat the world champion, and with NASA's MARS Pathfinder mission, RoboCup initiated the robotic soccer project with the goal of creating a team that could beat a human World Cup champion team, with the first official games being held, with 40 teams participating and over 5000 spectators attending.

Figure 2.1: RoboCup logo.

2.1.2 The 3D league

Within the RoboCup 2.1 initiative and the context of this dissertation the 3D league presents itself as sub-league, alongside the 2D simulation. Both are included in the SSIm - RoboCup Soccer Simulation League. This simulation raises the level of realism in the simulation environment used in other leagues by adding an extra dimension and increasing the complexity of the physics involved, whilst creating an excellent opportunity within the context of experimenting learning and optimization techniques as well as applying them on a multi-agent system environment, with the coordination and cooperation methodologies. When it first started it used a spherical agent, up until 2006. However, in 2007 the first humanoid model was used in this context - the Fujitsu HOAP-2, as the competition has its feeback describe on [9]. In 2008, however, the NAO robot model was introduced to the simulation. This model, from Aldebaran robotics, has been used as

the official robot for the Standard Platform League since 2008. Using this model in the simulation environment is a great opportunity for testing before applying them in the real world. At this point, the football match reached the 11vs11 of a real game with variations within the NAO robots - heterogenity- and has shown a growing variety of technical challenges like running , passing, shooting, among others. With the usage of an humanoid model, the aim of this competition changed its course from designing strategic behaviors to the control of low level skills like walking, kicking, turning up, standing and others, by the humanoid robots. In this dissertation we expect to incorporate this aim and the work developed in the scope of this aim to within the strategic behaviors in a football match.

Figure 2.2: Representation of the 3D league environment during a game

2.1.2.1 Competition Process

With the next competition being from June 23 to June 29, 2020, in France, another opportunity to experiment with humanoid robots, represented during a game in 2.2, approaches. This year, regarding the qualification process, the evaluation will be done based on the team's current performance, previous achievements - where it is interesting to mention that the FCPortugal team achieved very good results, throughout the editions being the Champions in 2006 and getting four 3rd places in 2013, 2015, 2016 and 2018 -, scientific contributions in relevant areas and development of the simulator. The competition has 24 teams participating and the top 3 teams of 2019 will be automatically classified. Within the simulation there is a set of rules, that can be fully seen on [19], where we can name some:

- **Self-collision** Due to unnatural moves being performed when parts of the same robot collided a fould is given to a robot that violates the normal physical rules by using this collision mechanism.

- **Set-up** After a round starts it is no longer possible to make changes to the agents until the next round.

- **Team format** The teams must have 11 players.
- **Faulty agents** When a match has already started and the agent shows abnormal behavior or gets disconnected the problem can:
  - Cause the restart of the match if it happens within the first 30 seconds, up until 2 times.
  - Force a timeout if the problem persists, up until 2 minutes to be fixed.
  - Cause the loss of 1-0 to the team if the team ends up with less players than the minimum or cause the team to play with less elements on the field if above.
- **Kickoff** Goals directly scored from this are not accepted and the agent that performs this can't touch the ball untill another has touched it.
- **Obstructing the Ball** Using the player's body, arms pr legs to prevent the progress of the game, by covering the ball, carrying it or holding it between the arms or legs, or lying in the front of the goal in an attacking situation will be penalized if intentional.
- **Illegal defense** At any time during the match, only 3 players, at maximum, of a team, may be inside their own penalty area. If A 4th player enters the penalty aream it will be placed outside of the field.
- **Penalty shoot-outs** During a penalty shoot-out, each team only has a single player in the field: the goalkeeper, for the defense and the striker. The attacking team has 40 seconds to try and score and the defending team myst have its goalkeepr inthe penalty area.

In this dissertation, as the goal is to develop set plays some of these rules, among others that can be found within the RoboCup's rules document, must be taken into consideration.

2.1.3 Humanoid Robot Skills

Much like in a real world football match, there many game level components that comes into account. Firstly, it is of vital importance to implement ways of simulating a real life player's behavior. This refers to certain skills that will allow the robot to move through the field, perform actions with the ball and interact with the team or the opposing robots. All of this refers to **low level skills** - the unity habilities that allow a robot to walk, run, shoot or pass, amongst others. The development of such capabilities was taken on by the FCPortugal team, which implemented them with success. Secondly, there are moments of the game where, due to the set of rules mentioned in 2.1.2.1, set pieces - or set plays - will be awarded. On those situations - the **high level skills** - come in place, as it is of the team's desire to implement set plays, where the robots cooperate between them to score a goal following the set piece awarded. These kinds of plays require the usage of the previous skills mentioned, the low level skills.

2.1.4 Low Level Skills Package

The main actions that have been developed and continue to be optimized are walking, running, kicking - with several variations - and getting up after falling - had to suffer alterations due to updated self collision rules.

- **Walking** This skill has a various range of options to be used:
  - **Omnidirection walk** This uses a 3D-LIPM based model to perform. It is subdivided into a trajectory generator module which is used to plan the route of the movements that the robot will do, a Foot Planner to decide where to set the foot and an Active Balance module in order to try and attempt to achieve balance so the robot doesn't fall. The implementation of this skill can be found on the "Development of an Omni-directional Walk Engine for Soccer Humanoid Robots" paper - [37].
  - **Fourier series** This type of walking uses Fourier series formulations, using, as well, a genetic algorithm to optimize it. With this method the step distance, the frequency of them and the walking pattern is required. Such work was developed in [17]
- **Running** Making usage of the omnidirectional walking skill, it can be extended into a running skill, if we are not to consider the mass center as a constant. This skill as been highly improved and is going to be of good use to the team's high level skills.
- **Kicking**
  - **Front Kick** It is based on a simple key frame and goes up to a maximum of 6 meters, when wanting full range. If power is prefered it goes down to 5 meters. Its setup is somewhat long, so that the robot finds himself in the correct position to perform it and it is very limited, regarding the distance range.
  - **Omnidirectional Kick** This skill excludes the phase of the setup, making calculations on how to kick it in order to achieve the desired location. Therefore, it is executed faster and in any wanted direction. It makes use of an Inverse Kinematics module - determines the value for each joint -, a Path Planning module -takes care of the trajectory -, and a Stability module - to stabilizing the robot. It has a total of 8 parameters which can be manually tuned. The work developed can be found in [16].
  - **Controlled Kick** The main trait of this variation is its flexibility, which allows the ball to be kicked in a much more variety of distances, in different conditions regarding, especially the robot's angle to the ball and position. It uses contextual policy search and, even though it is slower to the omnidirectional kick - as it requires setup - it is far more accurate. To find the optimization parameters it uses a Policy function and a Kick Controller, which is responsible for the robot joints.
  - **Long distance kick** AS the name suggests, this kick is used for long distance purposes, such as scoring a goal from a distance or kicking the ball into the opponent's half. It isn't very accurate and takes some time to setup.

- **Getting up** This skill has two slight variations, depending on whether the robot falls backwards or forwards. Based on a simple key frame model, it achieved good results. It is currently under development.

2.1.5 FCPortugal's Team

2.1.5.1 A brief history

The beginning of this project began early into the year 2000, by February. This project unites both the University of Porto with regards to LIACC - Luís Paulo Reis - and the University of Aveiro laboratories - Nuno Lau. There were also more members that were involved at the beginning of the project and some others that joined later on. Throughout the years the team achieved great results, namely:

- **Winning the Euro RoboCup (Amsterdam)** The results were impressive as the team managed to score 86 goals without conceding.
- **Winning the RoboCup 2000 (Melbourne)** Improving from the latter achievement, the team got 84 goals without suffering a goal.
- **Winning the German Open 2001**
- **3rd in RoboCup world Championship 2001**
- **Winning the RoboCup 2002 coach competition**
- **5th in RoboCup World Championship 2002**
- **Winning the RoboCup Dutch Open 3D 2006**
- **Winning the RoboCup World 3D 2006**
- **4th in RoboCup wolrd Championships 2014**

2.1.5.2 The team's work

In order to achieve the mentioned results, the team developed and did a lot of research in order to improve and implement innovative solutions.

Studying a series of papers it is possible to understand some of the approaches that were taken on by the team. There are some papers extensively cited through this document, but some others are also important such as [1, 38, 25, 26, 6].

2.1.5.3 Walking

One of the main low level skills is **walking**. This skill allows the robot to move throughout the pitch. One of the crucial aspects to implement this is to find a way where the robot maintains balance. This skill presents itself as vital as the robots, in order to accomplish their goals, in

a dynamic environment, must possess the ability to present a dynamic locomotion. In order to achieve so, research and work was done, as it can be seen on the 2013 paper [36]. So that the robot finds stability, regarding balance and avoiding falling down, the ZMP - Zero Moment Point - is used, hence providing a way to measure dynamic stability in biped locomotion. Furthermore, alongside the ZMP, a foot planner, a CoM - Center of Mass - generator and an Active balance loop are added, creating an omnidirectional walk engine. The presented mechanism begins with the Foot planner realizing the wished walk speed vector, generating as an output the future steps to be taken. These will act as input to the ZMP, which uses a cart-table model and a preview controller, in order to preview the CoM reference trajectory. Lastly, with the active balance loop will help the robot not to fall, when facing disturbances. Nonetheless, this skill, as well as any other skill, is in constant development to find possible ways of improving. Thus, in a continuous search for better results, more ways of tackling this problem were found, such as the analytical approach explained in [35]. In this paper, to find the CoM trajectory values, an analytical approach is presented. Like so, using a Fourier approximation of the ZMP, it is proposed to use a new time segmentation method to parametrize the double support phase. This approach proved to be, in a RoboCup 3D simulation environment, favourable, requiring less complexity and better accuracy.

2.1.5.4 Kicking

Another important skill is the action of kicking. In fact, when talking about humanoid robotic football, kicking, alongside walking, are the most important actions a robot could perform, as this allows to achieve the main objective of the team: to score a goal. When it comes to kicking, there are several variations, as there are many factors to take into account, such as the desired distance, the desired accuracy, position, orientation, the time it takes to perform it. As mentioned before, in section 2.1.4, there are numerous variations, for different use case scenarios. One of the most recent variations was worked on this year, 2020. In order to turn the games into more dynamic scenarios and surpass the opponents with faster kicks, the paper [12] refers to a new important variation - **kick in motion**. s In fact, the usual kick requires some time of preparation where the robot will be static, preparing the kick. A kick in motion presents itself as a big and important improvement, as it shortens the time required to kick, coming from a non static behavior, such as walking or running. Accounting for different possibilities of usage, this work made use of reinforcement learning, by utilizing the PPO - Proximal Policy Optimization - algorithm to create reliable skills. Another important aspect of this implementation is the hypertunning of the neural network that is important and will be talked about afterwards.

Another variation of the kick is the **controlled kick**. As stated before, one of the desirable scenarios would be to achieve the desired accuracy when kicking the ball. In the paper [3] the work to, regarding multiple variables, such as the robot's position related to the ball, implement a flexible humanoid robot kick controller is explained. By making use of one of the machine learning techniques, contextual policy search, a parametric function was found. This function, given a desired distance output the, with the highest possible accuracy, optimal controller parameters.

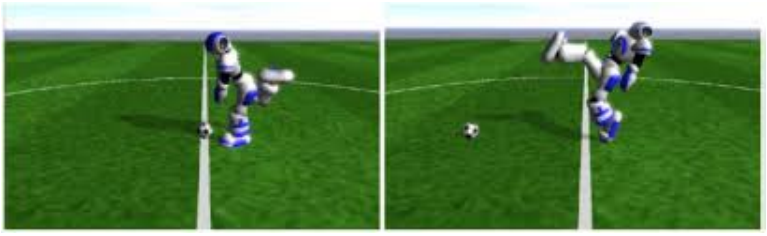

Figure 2.3: An humanoid robot kicking a ball in a simulation.

2.1.5.5 FCP Gym

One of the most important developed tools, which is rather recent, is the **FCPGym** tool.

A framework for implementing Reinforcement Learning algorithms already existed, called **OpenAI Gym**. This toolkit is used to test such algorithms, with the advantage of doing no assumptions when it comes to the environments used. It is important to notice that it offers excellent scalability, as it allows multiple episodes to be run at the same time.

However, despite being a very good tool, there was a huge margin of improvement that could be achieved. The main issue was due to the fact that the communication between the FCP agent code, which is done in C++, and the Python code - required for the optimization process - was done through TCP sockets. Making use of ad hoc files, the cost of this communication was big, even for small changes. With this in mind, Tiago Silva, a member of the FCPortugal3D team, developed a new toolkit - the **FCPGym**. This new tool retains all the prime aspects of the OpenAI Gym, but abstracts the communications between the two layers, using a third-party library. This, ultimately, led to the possibility of migrating many aspects of the reinforcement learning process to the c++ code, such as the definition of rewards, action and observation spaces, amongst others. Furthermore it added the functionality of fetching more data points, regarding the positions of object positions, joint angles and other values that are then sent back in a vector to the optimizer.

2.2 Machine Learning

Machine learning, often seen as a subset of artificial intelligence, is the study of computer algorithms that provides the ability to learn and improve from experience without being explicitly programmed. It is something that is already very present in our society, as you can see in the examples [10, 42, 33]. In order to do so, the process starts with observations, direct experience or instructions with the objective of detecting patterns and make better decisions in the future.

In fact, regarding the process of learning, one can mention three different manners:

- **Supervised machine learning** Within this method, the program is given labeled input data alongside the desired output results. The model is, thus, trained to detect patterns and possible relations with the goal with presenting good results with new input. It is a process of learning that uses guidelines.

- **Unsupervised machine learning** With this approach, it is intended for the algorithm to find patterns which one can't easily indentify. Unlike supervised learning, there is no labeled data, thus being unsupervised. One of its main usages is for cluster analysis.
- **Semi-supervised learning** This model is a combination of the previous two that were mentioned, as it makes use of labeled data - a small portion - and a large amount of unlabeled data.
- **Reinforcement learning** This approach is going to be the most relevant for this dissertation, thus being further explored in section 2.2.2. Having a dynamic environment, the program will interact with it, having a defined goal. Throughout the training interactions, some feedback will be given - regarding the process of rewarding - which will be looked to be maximized.

During the last few years a lot of work and research was done combining robotics and machine learning. In this section we will briefly review the most common algorithms and give some examples, in regards to the RoboCup context, where they were used.

2.2.1 Deep Learning

Deep learning can be seen as a sub-field, 2.4, of Machine Learning [21], where the learning method can be either supervised or unsupervised. The former receives training data which is labeled whereas the latter is given unlabeled data. Using artificial neural networks it is divided into many layers in order to process non-linear information for classification and pattern analysis classification. [13] It is a field which is growing as of recently, as it is being further explored and increasing in usage for current problems. Some usages can refer to medical image analysis [29], sensor-based activity [43], data mining [15].

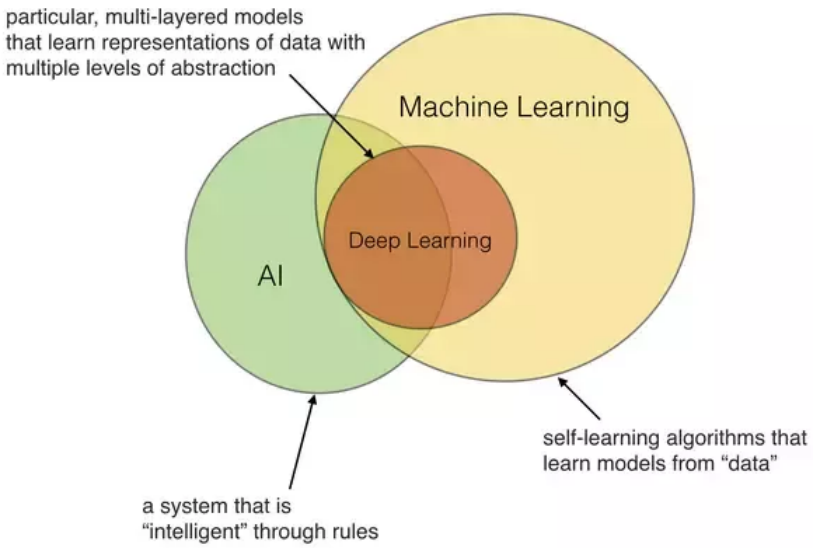

Figure 2.4: A sub-field of Machine Learning

2.2.1.1 Artificial Neural Networks

Neural networks are commonly used, nowadays, to solve computational problems. These type of systems were developed in order to create what can be called as "intelligent programs", a way of imitating the way the human brain works. One of its characteristics is the ability to work in parallel, communicating through multiple interconnections. To better understand this, some terms should be highlighted:

- **Neurons** An ANN is composed of artificial neurons. Each and every one of them has certain inputs and produce a single output. The latter can be sent to many other neurons. Throughout the layers, some of these outputs will just be used as inputs for other neurons, however, the ones that come from the final layer of neurons are the ones that proved the result of the task. The process of finding the output of the neuron consists of: getting the weighted sum of all the inputs, adding a bias term to it, passing it through an activation function.

- **Connections** As mentioned, the ANN is based on neurons and their connections. The connections are responsible for providing the output of a neuron as the input onto the next. Each connection has a **weight**.

- **Propagation function** This function computes the value of the input, making use of the inputs and their connections, regarding the weighted sum.

- **Hyperparameter** These type of parameter refers to values which can be tuned before the training process. These can refer to the **learning rate** - size of corrective steps that the model takes to adjust for errors in each observation -, the number of hidden layers and batch size.

The ANNS are generally organized with multiple layers. The two main layers are the input layer and output layer and, in between them, there are a certain number of hidden layers - zero or more. One way we can look at them is as a directed graph in which its edges have wights and the nodes are representative of artificial neurons. The connections would represent an output becoming the input of the next node. [23]

There are two possible architectures in which we can sub-divide ANNs:

- **feed-fordward** where loops do not exist.

- **recurrent** Also known as RNN where the feedback of the connections cause the existence of loops.

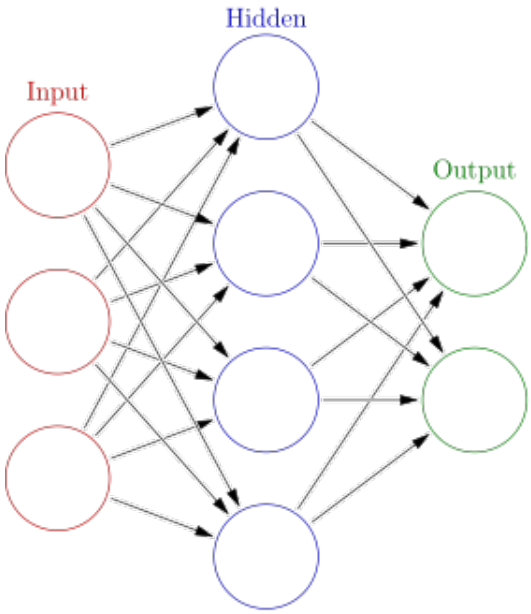

Figure 2.5: An example of an artificial neural network.

2.2.1.2 Learning in ANNs

One of the objectives of an ANN, and this dissertation, is the ability to learn. When using artificial neural networks, learning can be viewed as the "how to update the network architecture and connection weights so that a network can efficiently perform a specific task", In order to do this, it should, foremost, learn the weights of the connection from the available training patterns. This will lead to an improvement as time goes alongside iterations.[23]

In the next image 2.6 one can see the broadness of the terminology within machine learning, with special focus to Deep Learning.

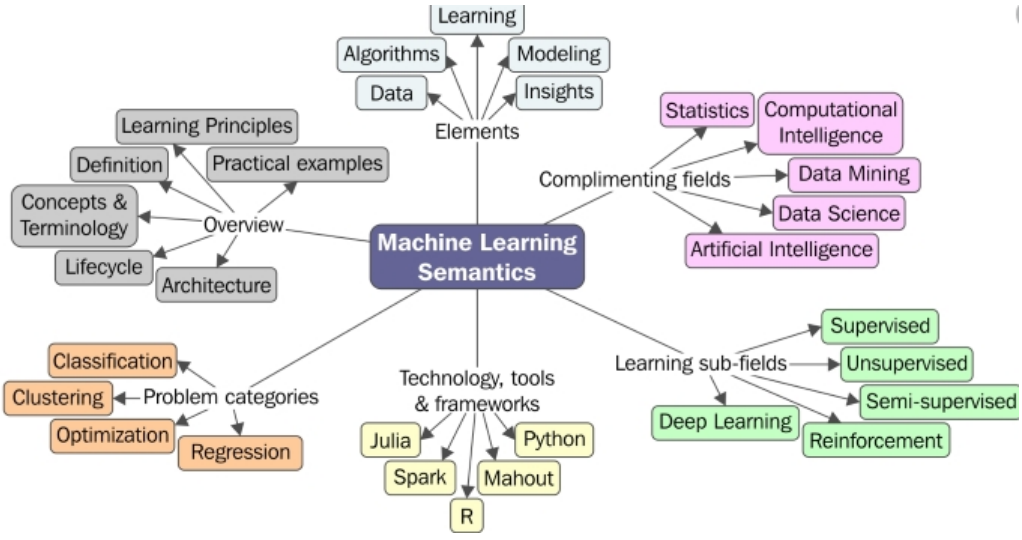

Figure 2.6: Some machine learning terms.

2.2.2 Reinforcement Learning

While we talk about Machine Learning it is often common to talk about unsupervised, supervised or even semi-supervised learning. However Reinforcement Learning presents itself as a different Machine Learning sub-field. It uses behaviours as a mean to achieve an end, looking to maximize or minimize, depending on the context of the problem, a so called reward, numerical, usually. It does not find or aims at fitting a model, but to set behaviors. Reinforcement learning, fairly well explained in the work if Sutton - [41] -, constantly tries to expand its reach to improve or replace outdated solutions. It is an area in which sizeable state and measures spaces, such as the multi-agent system can be, are finding particular interest do to its ability to overcome the possible problems. Either cooperative or competitive environments can adopt this field and RoboCup is no different. A reinforcement learning approach can be seen as a MDP - Markov Decision Process - where the Agent interacts with the environment. This structure can be defined by:

- **Agent** The agent is the one who takes actions.
- **Action A** is the set of all possible moves an agent would choose from. Normally, this set is a dist of discrete actions such as turning left or right, crouching or jumping.
- **Discount factor** This factor is calculated my multiplicating it by the possible future rewards as they come to be known by the agent so as to lessen or raise the influence of these rewards on the agent's course of decisions. it is often represented as the letter γ.
- **Environment** It represents the world in which the moves of the agent will be made. It receives the current state of the agent as input, as well as the a action, and returns, as output, the corresponding reward and its next state.
- **State S** S is a concrete and in the moment event in which the agent is. It involves the relation of the agent with other significant instances of the environment.
- **Reward R** The reward can be seen as the feedback by which it is possible to measure the success or failure of an agent's actions in a given state. At any state an agent can send output with his actions within the environment and, when receiving the new state it will also receibe the rewards, if there are any. They can be immediate or delayed.

There is also some more important concepts such as:

- **Policy(Π)** This can be seen as a strategy that the agent utilises to determine the next step, the action, based on its current state. It is used as a map that connects states to actions that are supposed to give the highest reward.
- **Value V** Opposed to R, which was a short term reward, V represents the expected long-term return with discount. VΠ(s) is the anticipated long-term return of the at the moment state under the policy Π.

With the need of improving its team's performance year after year, by implementing or optimizing new skills, low or high level, individual or joint, this field is becoming more and more relevant. One of the possible application examples is making a NAO robot run faster. Regarding this approach there are several algorithms available to explore within the multi-agent context.

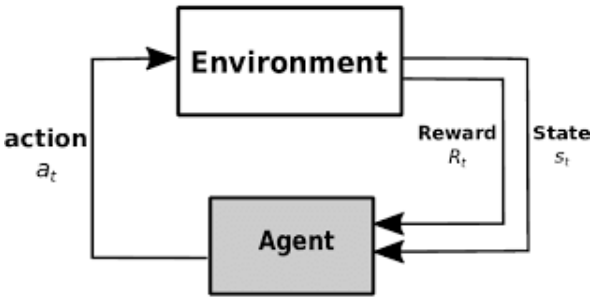

Figure 2.7: The procedure within reinforcement learning.

2.2.2.1 Types of models

Within the environment of a reinforcement learning approach, the model can be seen as the existing dynamic transitions. However, there are two approaches in wich the process can be divided into:

- A model-free approach.
- A model-based approach.

When talking about **model-free** approaches, this means that the agent will try to maximize the expected reward only from real experience, having no prior experience or prior model. It can't, hence, predict the next state when choosing an action and will only care about the associated reward. These type of methods are usually less computational-heavy, as they are only trying to find the optimal policy and will update theur values, even if the reward is not positive.

Regarding **model-based methods**, the agent will firstly learn the model by taking actions and observing the next state and immediate reward. Using a reduced number of interactions with the environment during the learning phase, aiming to construct a model used to simulate the next episodes - not in the real environment but by applying the actions in the constructed model. The advantages of this type of approach is that the learning is faster, as one does not need to wait for the environment to respond or reset. However, if the model is inaccurate the results of the training can be completely different from the desired results. ANther interesting aspect to this is that model-based methods use model-free algorithms in order to construct the model.

The following image 2.8 shows the difference in the process and further analysis was researched in: [24, 40, 27, 45, 44].

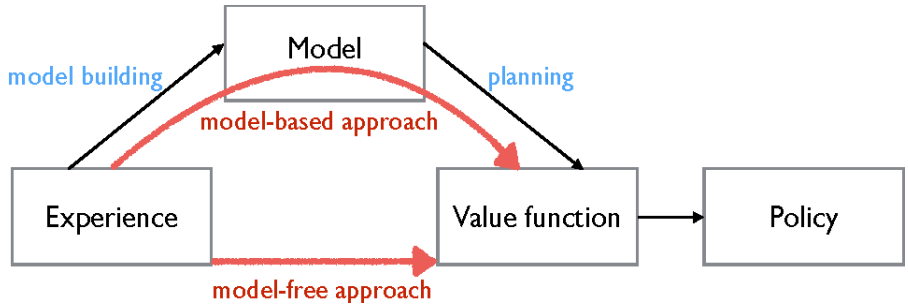

Figure 2.8: The relations of model-free and model-based approaches.

2.2.2.2 On and Off Policy

When it comes to Reinforcement Learning there are also dissimilarities when talking about the policy of algorithms, being **off-policy** or **on-policy**.

The difference between them is that **on-policy** methods will evaluate the policy that is being used to make decisions, while the **off-policy** will evaluate a different policy from the one that is being used to generate the data.

2.2.2.3 Policy-based vs Value-based

With the explained policy methods explained above in 2.2.2.2, one can also refer the possibility of having a value-based approach.

In fact on **policy-based** methods there is an explicit representation of a policy, that maps states to actions and saves it into memory during learning.

However, in **value-based** methods, the only thing that it is stored is a value function.

2.2.3 Multi Agent Learning

One of the aspects that this disseration will be focusing on is the learning on a MAS environment. This aspect mainly uses machine learning while increasing the number of agents, which results on the discussed problem of exponentially increasing the complexity of the problem. Furthermore, when wanting to implement high level interactions, that naturally happen with the sequence of the low level actions made by each agent, the how to question stands with some possible answers. The main traits of these answers are Reinforcement Learning and stochastic search. One of the approaches within this scenario uses a centralized manner, as discussed previously. Being called "team learning", it uses one agent only to generalize the whole team, learning for it. However, this tends not to hold up in high complex environments with a high level of agents. Another possible way of approaching such issue is concurrent learning. This methodology uses all agents - they all learn at the same time. The main barrier to this method is the fact that the environment is dynamic, which requires adaptation, especially to the fact that other agents can also be learning and changing their policies. Within the RoboCup context, it is also important to talk about the way the agents are organized. Thus, we can refer to teams that can be either homogenous or

heterogeneous, as well as hybrid. Such categories depend to the encoded behaviours within each agent - single behaviour or with agent specialization,, such as distinguishing a goalkeeper from an attacker. Another problem that needs to be attended and represents an issue is to understand how far an agent contribution should be extended to and how to assign the reward. This problem is due to the fact that if we reward every single agent the same way, dividing the reward among all the agents, there's a possibility we might be rewarding an agent that did a non-optimal action and, thus, rewarding and amplifying bad behaviours. Lastly, continuing on the high level behaviours, it becomes of great importance the prediction of teammates behaviours. For this to be effective there must be an use of time constraints.

2.2.4 Deep Reinformecent Learning

"Neural networks are function approximators". This comes in hand in reinforcement learning when the action or state space is far too big to be fully known. It can be used to approximate in a value or policy function. Instead of using lookup tables or index states, which isn't highly scalable, we could train a neural network with the samples of the state to learn how to predict the value of our target in reinforcement learning. One good example is the usage of a deep neural network as an approximator in the Q algorithm. This would use each action's reward as an output and the agent's discernments as inputs. Furthermore, due to the many layers divisions, this method is very useful in complex environments because it can separate lower level traits in lower level layers and higher level traits in higher layers - many levels of abstraction. Applied to this problem, given, for example, an image that designates a state a network could rank the actions that are possible to execute in it.

2.2.5 Q learning

In order to find good approaches in a cooperative multi-agent environment, Q-learning is in a good place regarding its acceptability by the scientific community as many approaches are derivated from it. Known as a Value-based Method, this algorithm trains to estimate the Q-Value of each state-action pair, making the policy to be greedy in order to maximize or minimise a certain value at every step. This type of algorithm allows the reusage of data, turning it into one of the most sample efficient methods, compared to the alternatives that do not allow the reusage of data. [11] In a single-agent reinforcement learning environment, these algorithms will be based by a Markov Decision Process - MDP. It is also model free because it starts learning without prior knowledge. [11] Through an action-value function, the Q-function is the expected return of policy based on the actual state and performed action. For each action there's a discount factor, from 0 to 1, that gives the importance of each action to the future reward. With a greedy policy, the agent will then choose its actions wanting to increase the return from the policy function. [11] One of the examples of these algorithms is DQN - Deep Q-Networks - which was already used and experimented with a single agent that was able to learn how to play Atari games, without prior

knowledge, and surpass human performance. [39] This method can be derivated to the multi-agent paradigm following a Decentralized Partially-Observable Markov Decision Process (Dec-POMDP). Such process intends to specify the number of agents, the set of possible exponential dimensional states of the environment, possible agent actions, joint actions and the transiction probability amongst them. It also marks observations from each agent so that it associated the action to a reward function. Another example is the DDQN, Double Deep Q-Networks. This algorithm was presented by Google Deepmind, who published an article where the flaws of the original DQN were shown and presented DDQN as an improved version of it. [39] This approach failed to show itself practical in high dimensional tasks, due to its lack of generalization for the estimation of the action-value function (calculated separately for each state). Following all these problems one interesting approach (multi agent double deep q networks) was applying the DDQN to the multi-agent paradigm. With the Joint-Action Learners (JAL- learner's Q-values are based on join-actions of all the agents) and Independent Learners(IL - each learner's Q-values are based solely on its own actions)- description of multi-agent Q-Learning by Claus and Boutilier, there was relative success applying it to cooperative scenarios like the pursuit game. With the following formula 2.9 it is possible to update the values, using the weighted average of the old value and the new information.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Big)}^{\text{temporal difference}}$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Figure 2.9: Q learning method to update its value.

2.2.6 Proximal Policy Optimization

The search for optimization is continuous and, therefore, further ways of achieving such goal are tested.

": Proximal Policy Optimization (PPO), which perform comparably or better than state-of-the-art approaches while being much simpler to implement and tune."

OpenAi

The Proximal Policy Optimization - PPO - is a reinforcement learning algorithm developed by OpenAi which tries to achieve a mid-term between performance and simplicity. It uses a method based on trust regions to ensure that in the non-stop search of improvement it keeps in track with the previous policy, so it doesn't lose performance. Policy Gradient methods deal with convergence problems by the natural policy gradient. Due to it involving a second-order derivative matrix, thus limiting its scalability into large scale problems, PPO was found to be better as it uses

a different variation of adressing the problem. Opposed to imposing a hard constraint, it defines it as a penalty in an objective function. This way we can use a first-order optimizer - Gradient Descent method - To optimize.[7] With this in mind we can talk about two variants of PPO:

- **PPO-Clip** This variant doesn't have a KL-divergence term in the objective and no constraint. It relies in clipping its objective function to abolish motivations for the novice policy to distance itself from the old policy.

- **PPO-Penalty** This solves a KL-constrained problem, but penalizes its divergence in the objective function instead of making it a hard constraint. It tunes the penalty coefficient as the train progresses so that it scales smoothly.

Regarding the area in which this dissertation will be focusing, this algorithm was used recently to try develop a "kick in Motion Ability" and to increase the speed at which the NAO robot could run.

2.2.7 Actor-Critic Methods

The Actor-Critic Methods are temporal-difference procedures, 2.10, that use distinct memory structures to represent the policy and the value function. Thus, the policy will be known as the actor, due to the fact that it is the one that selects actions, while the estimated value function is known as the critic, as it evaluates, by criticizing the actions made by the actor. This method can be either on-policy or off-policy, depending on the implementation of the TD error.

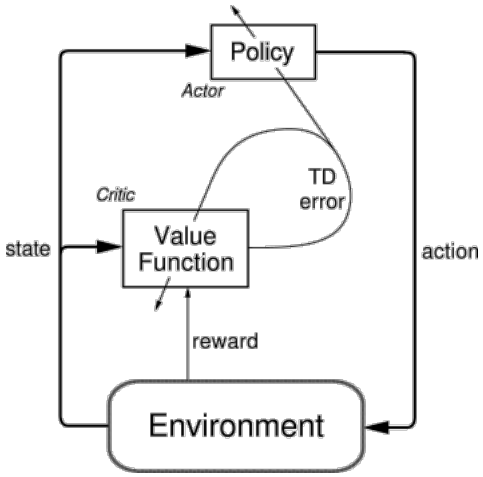

Figure 2.10: The actor critic process.

Usually, in this approach, the critic is a state-value function in which, after each action selection, it evaluates the new state to understand if things went better or worse than predicted.

2.2.8 CREPS

CREPS is an acronym for the Contextual Relative Entropy Policy Search. This algorithm seeks for the best parameters of a policy in one process x optimization [4]. Therefore, optimizing and generalizing the parameters of a controller happen in parallel.

2.2.9 CMA-ES

The Covariance Matrix Adaptation Adaptation Evolution Strategy is an algorithm which is considered to be the state of the art in stochastic optimization. Its main goal is to find the minimum of a function, which makes it an useful algorithm when it comes to learning problems as it can be used to find the optimal values in a fitness function, thus allowing us to find the best parameters that best result for those. It can be applied to unconstrained or bounded constraint problems in the continuous domain. The CMA-Es uses a second order approach to define a positive definite matrix. It is often extended to contextual problems due to its complementarity with algorithms such as CREPS. Such is due to CMA-ES benefits, like the stability, avoidance of premature convergence, step size control and low parameter tuning. FCPortugal3D's team already made use of this algorithm, in the low level skill walking [2].

2.2.10 CREPS-CMA

Following the two previous algorithms, in order to surpass the drawback of CREPS's search distribution - premature convergence due to the collapse of the distribution in a point estimate, a team, in 2016, combined the benefits of CREPS with the update rules of CMA-ES resulting to the contextual relative entropy policy search with covariance matrix adaptation. This approach was then used to optimize, in the context of humanoid robot soccer, the kick controller, so that the robot could kick the ball with a continuous scope of wanted kick distances.[5]

2.2.11 SARSA

This algorithm acronym stands for "state-action-reward-state-action". It is an algorithm that works with a model-free on-policy approach, with discrete action and state spaces.

2.2.12 DQN

The Deep Q Network is an algorithm that is also model-free, off-policy and works with a discrete action space but can operate with a continuous state space. It makes use of deep neural networks [28, 32, 31].

2.2.13 A3C and A2C

The Asynchronous Advantage Actor-Critic algorithm and the Advantage Actor-Critic are actor critic methods - they use both aspects of value-based (The Critic) and policy-based (The Actor).

The main difference between A2c and A3c is that in A3C there are several independent agents with different copies of the model. However in A2C the work is done in a synchronous manner, with the model sending data after each step [34].

2.2.14 DDPG

The Deep Deterministic Policy Gradient is an algorithm, which origin can be traced back to the work of Jonathan Hunt, [28], in which, jointly, both a Q-function and a policy are learned. While doing so it uses off-policy data and the Bellman equation to learn the Q-function and the Q-function to learn the policy. This process can be seen as linked to Q-learning. It is, in fact, driven by the same principle - if one is to know the best action-value function $Q*(s,a)$, then, for any given state, the best possible action $a*(s)$ is discovered by solving:

$a^(s) = \arg\max_a Q^(s,a)$

DDPG interleaves learning an approximator to $Q*(s,a)$ with learning an approximator to $a*(s)$. This method is done in a fashion that is extremely suitable for continuous action spaces. However, the way the computation of the max over actions in the equation defines whether this algorithm is adapted specifically for continuous action spaces environments. In fact, if an user is working within a finite number of discrete actions, there is no issue when using the max. This is because the Q-values, for each action, can be computed separately and be compared among themselves. However, when working in a continuous environment context, it would be feasible to exhaustively evaluate the space. Not only it would be a very expensive subroutine, in computation terminology, it would not be tolerable to run this procedure every time the agent wants to take an action. With this in mind, the function $Q*(s,a)$ is presumed to be differentiable with respect to the action argument. This sets up an efficient, gradient-based learning rule for a policy $\mu(s)$ which exploits that fact. This way, instead of running the previous optimization every time it is required to compute the $\max_a Q(s,a)$ one can approximate it with $\max_a Q(s,a) \approx Q(s,\mu(s))$.

2.2.14.1 Q-learning DDPG

Firstly, it is important to review the Bellman equation where the optimal action-value function, $Q*(s,a)$, is described. This equation is as it follows:

$Q^(s,a) = \underset{s'\sim P}{\mathrm{E}}\left[r(s,a) + \gamma \max_{a'} Q^(s',a')\right]$

Here, the $s' \sim P$ means that th next state, s', is sampled by the environment from a distribution $P(\cdot|s,a)$ The Bellman equation above is the foundation for learning an approximator to $Q*(s,a)$. As one is talking about deep reinforcement learning, assume the approximator is a neural network $Q\phi(s,a)$ with parameters ϕ and we have a set $\mathscr{D}$ of transitions (s,a,r,s',d). With this it is possible to set up a mean-squared Bellman error - **MSBE** - function, which will allow us to evaluate the proximity in which $Q\phi$ satisfies the Bellman equation:

$$L(\phi,\mathscr{D}) = \underset{(s,a,r,s',d)\sim\mathscr{D}}{\mathrm{E}}\left[\left(Q\phi(s,a) - \left(r+\gamma(1-d)\max a' Q_\phi(s',a')\right)\right)^2\right]$$

The approximator is heavily based on minimizing the MSBE loss function. In order to do this there are some tricks used:

- **Replay Buffers** This trick is used by all standard algorithms that train a deep neural network to approximate $Q*(s,a)$. It makes use of the off-policy approach DDPG uses, by saving old experiences, regardless of the followed policy. This is desired because the Q-function should satisfy the Bellman equation for all possible transitions.

- **Calculating the Max over Actions in the Target** As previously mentioned, the computation of the maximum over actions in the target is a challenge for continuous spaces based environments. In order to deal with this, DDPG used a target policy network. Much like the target Q-function, an average of the policy parameters, over the course of training, is done.

2.2.14.2 Policy Learning DDPG

The goal of this side is to learn a deterministic policy $\mu_\theta(s)$ which provides the action that maximizes $Q_\phi(s,a)$. In order to perform gradient ascent - regarding policy parameters - we assume that the Q-function is differentiable with respect to action, as the action space is continuous. Thus, and making attention to the fact that the Q-function parameters are treated as constants, the equation to solve is:

$$\max \theta \underset{s\sim\mathscr{D}}{\mathrm{E}} [Q\phi(s,\mu_\theta(s))]$$

2.2.15 TD3

This algorithm is one of the most used as it allows one to train environments in which the observation space is either discrete or continuous and the action space is continuous. Its origin can be traced to the work of Scott Fujimoto in [20]. It presents itself as a model-free and off-policy deep reinforcement learning method. The former refers to the approach used by the algorithm. In this case, a model-free can be seen as a "trial and error" procedure, as it does not use the transition probability distributon associated with the Markov decision process. The latter means that, in order to evaluate the action to take, it uses a buffer of data that contains information about other policies, independently of the current one. Thus instead of being stuck on the current "best" policy, which can lead to a local optimum, the agent has more freedom to explore. Nonetheless this is an extension of the DDPG algorithm. Even though DDPG can grant high levels of performance in numerous occasions, it has shown itself rather inflexible when it comes to tuning hyperparameters - such as the size of the Neural Network. Another common issue is that the learned Q-function starts to abruptly overestimate Q-values. This will eventually converge to a policy breaking situation, as it explores the errors in the said Q-function. The Twin Delayed DDPG - TD3 - implements three techniques which are critical to address this issue. The three techniques are:

- **Clipped Double-Q Learning** Instead of using only one Q-function, it makes use of two. It also uses the smaller value of them to form the targets in the Bellman error loss functions.

- **"Delayed" Policy Updates** While the Q-functions update rather frequently, the TD updates the policy in a delayed manner, in a one policy update per two Q-function updates.
- **Target Policy Smoothing** In order to make it harder for the policy to exploit the errors on the Q-function it adds noise to the target action, so that it smoothes Q along changes in action.

2.2.15.1 TD3 Key Equations

As for the **Target Policy Smoothing** technique, mentioned before, the problem it addresses is the lack of regularization of the DDPG algorithm. This is if the Q-function approximator is induced into developing an incorrect sharp peak for some actions, the policy will explore that peak, leading to a wrong approach. In order to avoid such problem, the TD3 procedure uses Q-function smoothing, which is done by adding clipped noise on each dimension of the action. After doing so - adding clipped noise - the target actions is also clipped to lie in a certain approved range. The equation for the arget actions is then:

$a'(s') = \text{clip}\left(\mu\theta\text{targ}(s') + \text{clip}(\varepsilon, -c, c), aLow, aHigh\right), \varepsilon \sim \mathcal{N}(0, \sigma)$

This equation shows us that the action a' for the state s' will be the result of clipping the target policy $\mu\theta$targ, using a range of valid actions a.

When it comes to the **Clipped double-Q learning** the end result is very similar to the DDPG one. However, due to the fact that the policy is less updated than the Q-functions, the volatility that is normally generated in DDPG - as the change of a policy changes the target - is damped. Regarding this issue, both Q-functions will utilize a target, making use of the smaller value provided by the "twin" Q-functions. The target equation is:

$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q\phi_{i,\text{targ}}(s', a'(s'))$

Afterwards, both are learned by regressing to the previous target:

$$L(\phi 1, \mathscr{D}) = \underset{(s,a,r,s',d)\sim\mathscr{D}}{E}\left(Q\phi_1(s,a) - y(r,s',d)\right)^2$$

$$L(\phi 2, \mathscr{D}) = \underset{(s,a,r,s',d)\sim\mathscr{D}}{E}\left(Q\phi_2(s,a) - y(r,s',d)\right)^2$$

With this approach, the overestimation in the Q-function will be less problematic than what it is in DDPG. To finish, the policy is learned by maximizing one of the Q-functions:

$\max\theta \underset{s\sim\mathscr{D}}{\text{E}}\left[Q\phi 1(s, \mu\theta(s))\right]$

2.2.16 SAC

Being released around the same time as the TD3 algorithm was, by Tuomas Harnooja and some more members in [22], the Soft Actor Critic optimizes a stochastic policy in an off-policy way. This constitutes a bridge inbetween the DDPG's approach and the stochastic policy optimization. Just like the TD3 it makes use of one of its techniques - the **Clipped Double Q-learning**. Combining this usage alongside the stochasticity of the policy in SAC it grants an approximate target

policy smoothing aswell. One of the key features of this method is **entropy regularization**. In order to understand this one needs to know the definition of:

- **Expected return** Usually represents the cumulative rewards, it is what drives the decisions it makes.
- **Entropy** It is a measure of randomness in the information being processed. The harder the entropy, the harder it is to draw any assertive conclusions from the given information.

With this in mind, in this algorithm, the policy is trained to maximize a trade-off between both of the above mentioned concepts. This is rather useful regarding the avoidance of converging into a bad local optimum, as the increasing of entropy can result in more exploration. The constraints of this model is that it is bounded to continuous action spaces.

2.2.16.1 SAC key Equations

In this algorithm the equations vary due to the fact that a new setting must be introduced. The **Entropy-Regularized Reinforcement learning** As stated before, entropy determines the randomness of a variable. Considering x to be a random variable with a density function P, the entropy H of x is computed as:

$H(P) = \underset{x\sim P}{E} -\log P(x)$

Furthermore, the agent will be awarded with a bonus reward at each time step, correlated to the entropy of the policy in that given timestep. Thus, the reinforcement learning problem is now given by:

$\pi^* = \arg\max \pi \underset{\tau\sim\pi}{E} \sum t = 0^{\infty} \gamma^t \left(R(s_t, at, st+1) + \alpha H\left(\pi(\cdot|s_t)\right) \right)$

where $\alpha > 0$ represents the trade-off coefficient. From now on, the equations change in order to include the entropy bonuses from each timestep. Hence, the Value function is given by:

$V^{\pi}(s) = \underset{\tau\sim\pi}{E} \sum_{t=0}^{\infty} \gamma^t \left(R(s_t, at, st+1) + \alpha H\left(\pi(\cdot|s_t)\right) \right) \Big| s_0 = s$

And the policy only excludes the first timestep:

$Q^{\pi}(s,a) = \underset{\tau\sim\pi}{E} \sum_{t=0}^{\infty} \gamma^t R(s_t, at, st+1) + \alpha \sum_{t=1}^{\infty} \gamma^t H\left(\pi(\cdot|s_t)\right) | s_0 = s, a_0 = a$

Lastly, the Bellman equation for the policy is:

$Q^{\pi}(s,a) = \underset{s'\sim P}{E} \underset{a'\sim\pi}{E} R(s,a,s') + \gamma(Q^{\pi}(s',a') + \alpha H\left(\pi(\cdot|s')\right)) = \underset{s'\sim P}{E} R(s,a,s') + \gamma V^{\pi}(s')$

2.3 Tools

In our problem we focus on a certain environment, making use of the tools that exist and were developed, alongside with already defined skills. Being the aim to develop a set-play, a sequence of actions - low level skills - it is mandatory that we make use of the existing skills, as a set play must use them. Further developments regarding the actions might be taken. As such, the set-plays to be developed will mainly focus on the offensive aspect of a football match. In this part we will talk about the simulation environment, its unit of testing and further tools that will be useful when using machine learning algorithms.

2.3.1 SimSpark

Regarding the multi-agent system environment, SimSpark presents itself as a physical multi-agent simulator for 3D environments. This was developed as part of the RoboCup initiative, by Marco Kögler and Oliver Obst, being the RoboCup's official 3D simulator, and it has been used as a flexible platform. Regarding the communication within it, UDP or TCP can be both used and, therefore, any language with the support for such sockets can be used to implement it. It allows multiple agents to run in one simulation, thus being the simulator used. Regarding the football matches, the simulation is defined as:

- 30 by 20 meters pitch
- A goal 0.8 meters tall, 2.1 by 0.6 meters - it was recently increased.
- A penalty area 6 by 1.8 meters.
- The midfield center circle with a diameter of 4 meters.
- A ball with 0.08 diameters and 26 grams of mass
- Markers in each corner and in each goal post.

Using this simulator to test our work will allow us to understand if the optimizations will run seamlessly in official matches. It is relevant to say that SimSpark allows the creation of a custom scenario using its scene description language.

2.3.2 NAO Robot

The NAO robot, 2.11 is a humanoid robot, developed by SoftBank Robotics and its the unit of the simulated scenarios, regarding the RoboCup initiative. Each robot as certain characteristics that vary from team to team, like the maximum running speed. However the model of the robot has common assets to all. Naming a few, we can denote:

- A height of 57 centimetres.
- A weight of 4.5 Kg.
- 22 degrees of freedom.

Furthermore it is equipped with some tools, 2.12, that allow him to be the ideal unit for this simulation environment. It has one gyroscope in the torso, one accelerometer also in the torso, force resistance preceptors, one in each foot, one vision preceptor in the center of the head, one say effector and one hear preceptor for communication purposes, one games tate preceptor that is used to denote the state of the game in which it currently is.

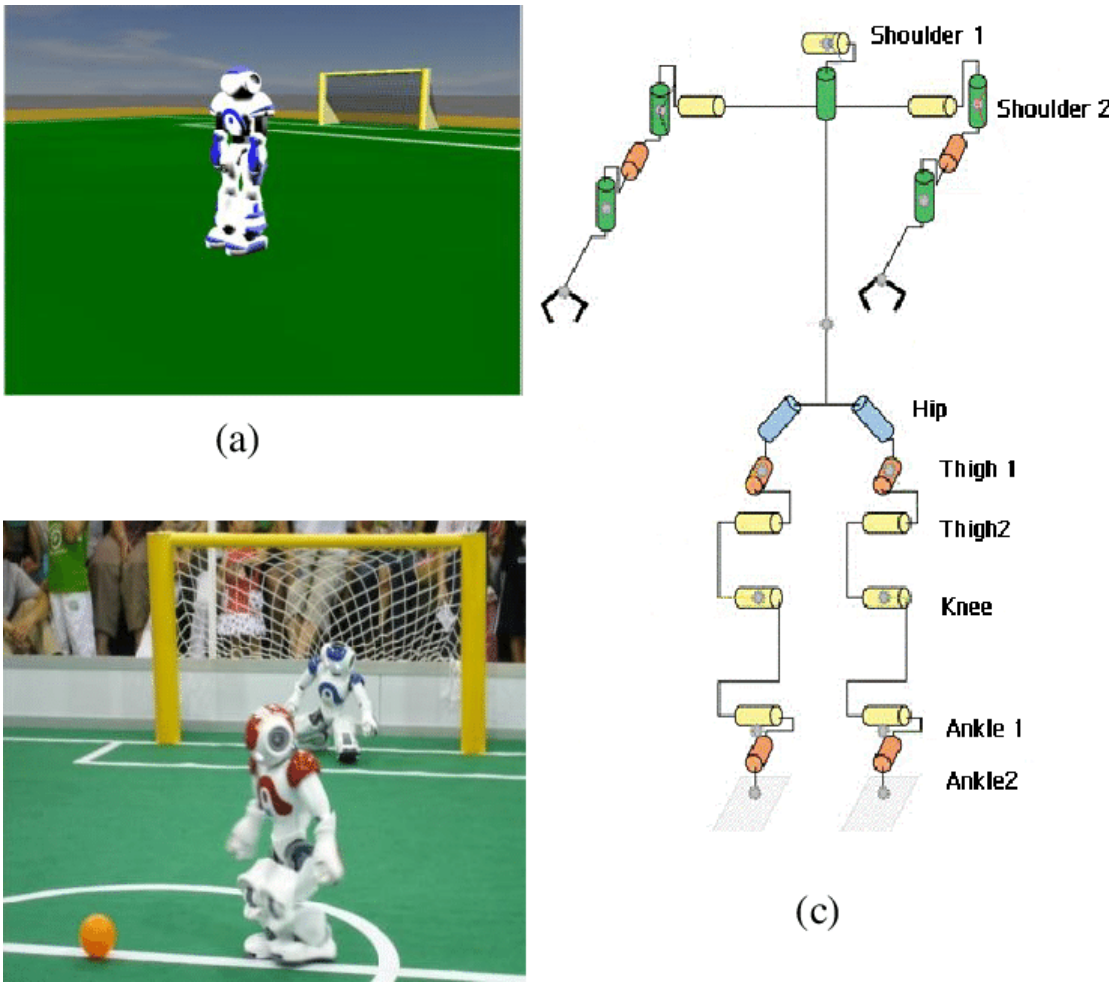

Figure 2.11: An humanoid robot.

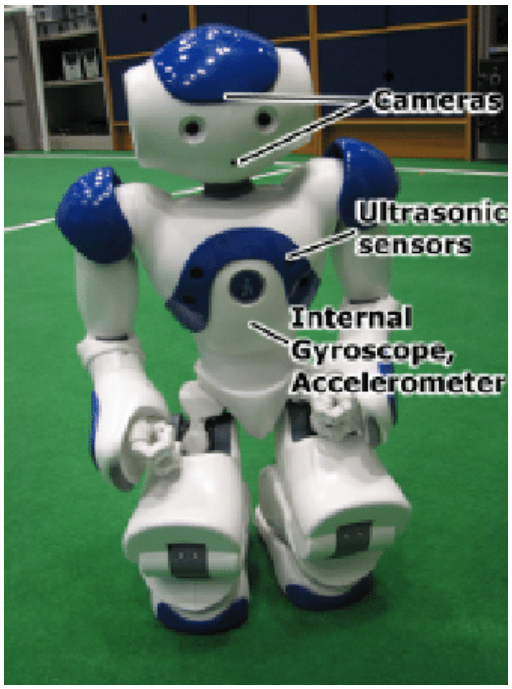

Figure 2.12: Some of the tools available.

2.3.3 TensorFlow

TensorFlow is a machine learning framework, open-source and python-oriented, used to design, build and train deep learning models, developed by Google. The name itself is derived from the operations which neural networks perform on multidimensional data arrays or tensors. It uses dataflow graphs in order to represent the mathematical operations and the communicated data amongst edges. It also revealed to be a very scalable and flexible tool, making it ideal for academic purposes. Furthermore, it provides with graphical aid, supplying log data and graphs of the inner structure, which allows one to analyse what it is displayed.

2.3.4 Stable Baselines

Following the implementation of OpenAi Baselines, there was a need of improving certain aspects. This need, that came from the desire of tweaking certain aspects of the usages of it, led to its creation. With this tool it is far easier to make use of the different algorithms, due to the common methods like train, save, load. Furthermore it implements full integration with tensorboard, which allows the user to visualize numerous graphics, in order to track the evolution of the reward throughout training, learning curves, coefficient values, loss values. It also includes a very important feature - tweaking hyperparameters, such as the the policy, allowing the user to define a new network structure, by changing the number of layers and nodes. In conclusion, this tool is an improved implementation of OpenAi Baselines, offering more documentation, a bigger use-case usage and making it easier and faster to approach reinforcement learning and deep reinforcement learning problems.

2.3.5 FCP Gym

": Gym is a toolkit for developing and comparing RL algorithms. It makes no assumptions about the structure of your agent, and is compatible with any numerical computation library, such as TensorFlow or Theano."

http://gym.openai.com/docs/

It is of this dissertation interest to, as the FCPortugal team as been doing, seek for improvements and to optimize the skills and the performance of the team. In order to do so it is our plan to use, on top of the framework that allows us to create the set plays, FCP Gym as a tool that will allows us to train our deep reinforcement learning algorithms. This tool was developed by Tiago Silva, a member of The FCPortugal team. This tool allows one to fetch from the FCportugal code a series of data points and send them to the optimizer, which stimulates the creation of new learning scenarios, thus allowing us to greatly increase the usage of our algorithms.

2.4 Conclusions

By the end of this chapter, a wide variety of possible approaches, in regards of Machine Learning, were reviewed in order to tackle the problem that concerns this dissertation - creating a set play, which is, in its essence, a machine learning problem, facing the learning of many agents simultaneously, with a continuous action space.

Thus, concerning all the concepts and algorithms that were mentioned and explored, the procedures that were of higher relevance were the Twin Delayed DDPG and the Soft Actor Critic algorithms.

Chapter 3

Multi-Robot Learning System

Set-Plays Creation

Our goal is to implement a Multi-Robot Learning System engine that will allow us to create collective skills, feasible of being learnt and optimized with machine learning algorithms. This open problem will be approached in a two phased solution.

With the above graphic, A.1 we can see an early approach of the projected work plan. Firstly there will be further investigation and research to fully understand where and how problems of similar context were approached. Not only that, but to have a better understanding of the technologies and frameworks that are to be used throughout the whole process. Secondly, it is important to integrate myself with the FCPortugal team, in order to fully understand the common goal, the dynamics and the work so far developed. As this process evolves, the practical work shall begin with the creation and definition of the set plays we wish to create, taking into consideration what is and what can be developed, regarding the mechanisms that allows us to create a sequence of interesting skills. Furthermore, it is important, not only to create and develop the plays but to optimize them with deep reinforcement learning, allowing us to, afterwards, compare the team's performance to understand if there was, in fact, improvements and if the results are what we expect under variant constrains. Lastly, further documentation regarding the developed work and research done will be made alongside the process.

3.1 Development of Set-Plays

This project is, in its core a collaborative one, as it is part of the FCPortugal team. Therefore, the first phase of our solution will consist on the creation itself of the set plays, with the usage of the already developed low level skills. Further low level skills might be added by the team focusing on that subject. After implementing the set plays making use of the FCPGym tool, the goal is to optimise the set plays, making use of Deep Reinforcement Learning algorithms, provided by the stable baselines algorithms. In order to visualize the results or the training phase Roboviz shall

and will be used, with the ability to change the velocity of which the execution occurs. This allows one to speed the process up when training a model and to see the outcome in a real time fashion.

3.1.1 Problem

Every team searches for constant improvement and ways to outplay its opponents. FCPortugal's team desires to develop set plays which will lead to goals or opportunities to score. One important and rather common set play is the corner kick. This set play consists on one agent, which can only touch the ball once, standing next to ball on a given position - extremity of the x-axis and halfway between the goal and the exit line, regarding the y-axis - with the objective of passing it to teammate. Additionally there are more agents which can receive the ball and continue the set play. The main objective is to score goal, in the fastest and most effective way possible, thus the set play can and should be performed by the interaction of two agents of the same team.

3.1.2 Observation Space

Each episode requires an observation space. This observation space is what defines the evaluation of the performed actions by the agent throughout the episode. Regarding the main goal of the context of this problem, the most important observation that can be made is the ball's position. It is so because it is what defines if the goal was accomplished or not. The ball's position is given within a continuous value, its coordinates. The important values here are the initial position in which the ball should be in order for the set play to begin, which is [15,6] or its symmetric values, and the objective position which would be the values between X>15.0 and -1.0 < Y < 1.0. The latter represents the area of the goal.

3.1.3 Episode Layout

This episode requires the coordination of two agents, thus the episode consists of two connected and coordinated episodes. Each one refers to one specific agent, as explained in 2. Being so, there are two cycles:

- The crossing agent.
- The kicking/tap-in agent.

3.1.3.1 The Crossing Agent Cycle

This agent is responsible for crossing the ball into a teammate with the purpose of providing the best opportunity to score. At the beginning of the episode the robot will spawn into a starting position, within a few meters from the ball's initial position. Once it does, if the ball is in position with its velocity equal to zero, the agent will use one of the FCPortugal's functions to cross the ball. This will initiate the passing behavior. Firstly, the agent will move towards the ball. Once it is in a close proximity he will move its joints in order to perform a kick of a rather short distance -

max of 14 meters. The coordination with the other agent is key, therefore the cycle will be relying on the ball's position and on the agent's balance. First of, as the objective is to score goal and the faster, the better, the episode will reset automatically if the agent falls before kicking the ball. Second of, after kicking the ball the agent will then wait for the other one to execute and finish its episode. Once it does the ball will reset to its starting position, triggering the reset of the episode going onto the next one. Furthermore, as stated previously, the goal is to be fast and effective. In order to do so there is another variable that requires attention - the kicking agent's height. If the agent who is supposed to kick the ball has fallen than the set play is probably ruined as the ball would be taken by an opponent while the agent gets up. With this mind, if the kicking agent falls before performing its action the episodes also reset.

3.1.3.2 The Crossing Agent Reward

As explained before there is a continuous search for efficiency and speed. In this scenario the evaluation of the episode will take into account, as its primary parameter, the ball's position. Taking this into account, the reward function defined was $-dist$, where dist refers to the distance between the agent to whom the pass was intended to and the ball. This means that the higher the distance between them, the worst the reward is going to be, while the closest it gets, the better the reward.

3.1.3.3 The Kicking Agent Cycle

This agent will have the finishing role, regarding the set play. It is his function to be the end receiver of the crossed ball and score the goal. At the beginning of its cycle, this robot will also be beamed into a starting position, close to the box. The first step of this cycle begins when the crossing agent starts its run. As this happens, the kicking agent will also initiate a run. This run will be made into a position on the box, trying to predict the pass or gain advantage by escaping the opposition. Afterwards, the agent will wait for the ball to arrive to its area of action and, once it stops, it will, if necessary, get closer to it and kick it into the net. Following the other episode's explanation, this one also requires to follow certain constrains to execute in a timely fashion, coordinating it. Thus, the episode reset function will be triggered when one of the agents fall without performing their actions. Lastly, the reset will also be triggered once the ball stops, after the kick. Additionally, a variation of this behavior was tested, as, instead of kicking the ball, the agent would simply walk towards the ball in order to tap it in into the goal. This was implemented because the kick had a big time of preparation, which led to a non usable behavior within a competitive environment, as the ball could be stolen by the opponent.

3.1.3.4 The Kicking Agent Reward

Both episodes have the same purpose, which is the end objective of the whole set play: to score. This goal will be achieved if the whole set play is smooth, regarding the contribution of all its agents. The reward for this episode will, as well, rely on the ball's position - more specifically if

whether the ball is in the goal area or not. In order to incentivise the agent to learn how to properly position the ball, the reward is $goalLine + posts$, where the goalline parameter refers to whether the ball has surpassed the goal line - if the goal line is 15.0, a goal shall be considered when the ball has a superior X-coordinate value. However, to be a goal the ball must also be in between the posts, thus a K value is added, positive when checking the condition of goal and negative when not. Thus, if the ball doesn't go into the net, a penalty is given, while if it is a goal it maximizes the reward.

3.1.4 Corner Kick Optimisation

The purpose of this thesis was not only to create new set plays but also to optimise them using deep reinforcement learning on a multiagent environment. As such the optimisation of this set play was done using algorithms that rely on deep neural networks, such as TD3 and SAC. Both of these algorithms were used, making use of the stable baselines framework and taking into consideration the fact that our action space is a continuous set of values - the coordinates of where to place the ball. In this sort of scenario, where the actions of all agents rely on existing functions - to run and to kick - the Neural Network is not on control of the entire episode. This is because the Neural Network can not output anything that will change the output of the running action. Thus, the optimisation of the set play is onde in a way here the reward is only received at the end of its episode. This is usually done once the ball stops after an action is performed. With the intention of optimising a set play each agent will have different action spaces which will be the placeholders of the Neural Network's output. Thus, this process begins with the output of values from the Neural Network which will then be used as input for the functions to be utilized by the robots. Once the episodes finish, the signal reward will be awarded and the process resets up until a predefined number of training repetitions. Furthermore, as explained before, the TD3 algorithm relies on a Mlp Policy, which uses a Neural Network of 2 layers, each one having 64 nodes. For this experiment, custom policies were created to test the speed of convergence and the improvement of the results obtained. When creating custom policies, layers were added, as well as nodes, finalizing the tests with 3 layers, two of them possessing 256 nodes and the middle layer 128.

Separating each agent the parameters - the action space - to optimise were, for the crossing agent:

- The coordinates of where the ball is going to be crossed to.

And for the kicking agent:

- The coordinates of where to make the run for.
- The orientation in which the agent should do the run for.
- The coordinates of where to kick the ball or where to walk into.

For the crossing agent the action space had the constraint of the ball having to be kicked somewhere between the kicking agent and the goal line. It also needed to be between the posts, so it would not create a much diagonal shot. Lastly, regarding the kicking agent, the constrains required the agent to make a run up until the goal line and never to get further away from the box, but into it. A specific region of the goal was also predifined.

3.1.5 Results

This section intends to show the achieved results that were obtained for the experiments regarding the Corner Kick set play, using a Neural approach, concerning the TD3 and SAC algorithms, as well as custom policies. As stated before, due to the nature of what a set play is and the way it works - the results are sparse, as the goal is to make use of low level behaviors like shooting or passing, which relies on giving rewards once the episode is finished and outputs can't change the outcome while an episode is already running -the Neural Networks will provide input for the utilized functions, evaluating them once it finishes. Once the training iterations are over, the resulting trained model will be saved with the best values obtained throughout the training phase. As such, running in a coordinated fashion, both the kicking and crossing agent, the results should show a successful set play, where the crossing agent places the ball near the kicking agent and the latter shoots into a goal.

3.1.5.1 Results Kicking

The kicking agent had the most variables to optimize. In fact, the functions of this player was to make a run into the box, face a specific orientation and, finally, kicking the ball using a controlled kick. Furthermore, the action space faced several constraints in order to avoid unwanted behaviors such as running further away from the box and failing the goal when kicking. Taking into account the fact that we are using a continuous action space there are certain range of values that could be considered optimal. The following graphic shows, with a high level of smoothness, the reward along with the training iterations. It should be noticed that due to the conditions that trigger the reset function, such as the falling of agent - regarding synchronization issues and the goal of the set play - some rewards refer to unwanted episodes, which reflects in some punctual variations.

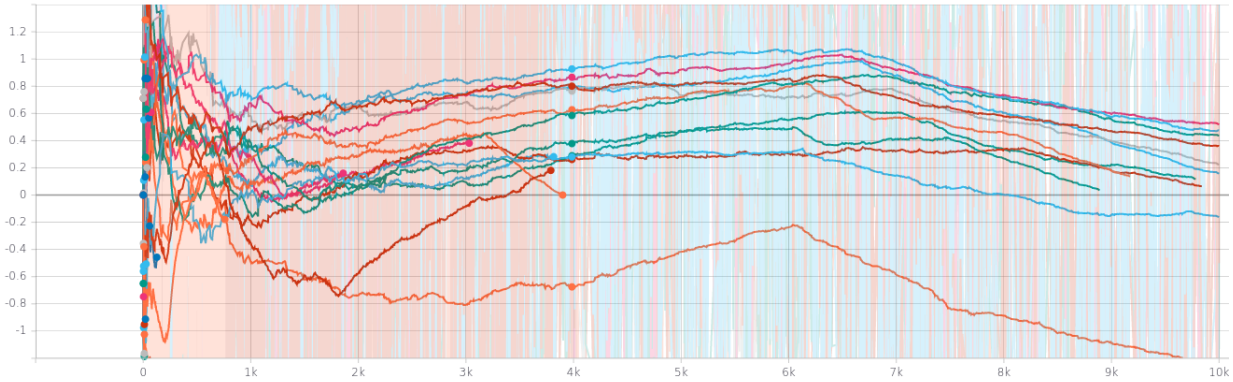

Figure 3.1: Episode reward during training.

As seen on the image 3.1, each line represents one training process with the values being very close to each other. The variations, as stated before, reflected the unwanted episodes and also the multiagent system factor. This is because when optimising multiple agents towards one common and main goal, the variations of all the action spaces may lead to various results. Nontheless, it is clear that the robot managed to achieve positive rewards. Collecting results from the trained model it can be seen that the coordinates for the kicking revolved around $X \in [15.8, 15.9]$ and $Y \in [1.8, 1.9]$ while the average reward value was 1,42. Furthermore, the values regarding the position of where to make the run to were $X \in [13.8, 14.0]$ and $Y \in [-0.92, -1.05]$. As for the walking variation, the maximum reward as 1,16, the coordinates of where to make the run for were $X \in [13.8, 14.0]$ and $Y \in [-0.8, -1.0]$ and the tap in coordinates were $X \in [15.27, 15.3]$ and $Y \in [-0.79, -0.85]$

3.1.5.2 Results Crossing

The crossing agent had the least, but not less important, variables to optimise. Having to cross the ball into the box and near the desired teammate, its action space was the coordinates of where to place the ball - serving as input for a passing function, ideally a short kick function (short refers to the meters that the ball has the ability to travel, regarding the force and position of the robot when touching the ball). When loading the trained model it can be seen that the crossing agent places the boal very near the goal line, as well as near the robot, making it easier to score the goal. Some punctual events led to the ball touching the desired agent and getting into the goal, without the need of kicking the ball.

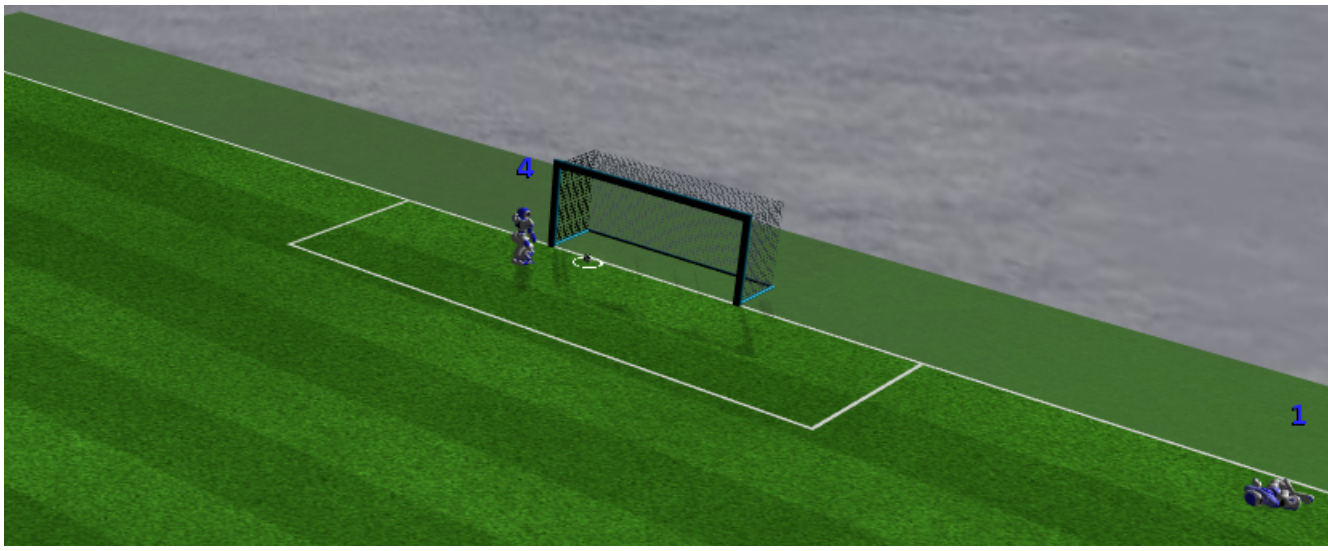

Figure 3.2: Ball crossed by agent 1.

The figure above, 3.2, shows the position of the ball once it stops, after the crossing agent kicking the ball. It is clear that it creates a good goal occasion. Note to the fact that the crossing agent has fallen, this happens due to a lost of balance when kicking the ball. However, as its function - crossing - was already completed the episode does not need to trigger the reset function.

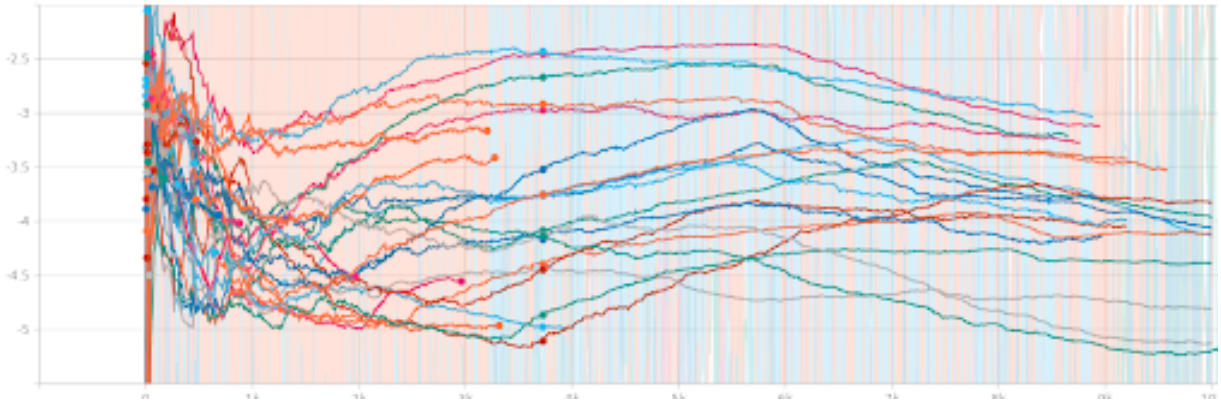

Figure 3.3: Episode reward druing training.

The graphic above, 3.3 shows that within few episodes it manages to stabalize and find an optimal range of values to which its contribution helps to score the goal. The ideal values revolved around $X \in [14.8, 14.9]$ and $Y \in [-0.8, -1.0]$ while the average reward reflects the distance to the other agent, having an average of -3.3.

3.1.5.3 Results set play

The results regarding the whole set play can be seen as a junction of the results of the optimisation of the group of all involved agents. Thus, the following images will allow to visualize the process

of a corner kick. The sequence is as shown, starting with image 3.4, followed by 3.5 and finish on 3.6.

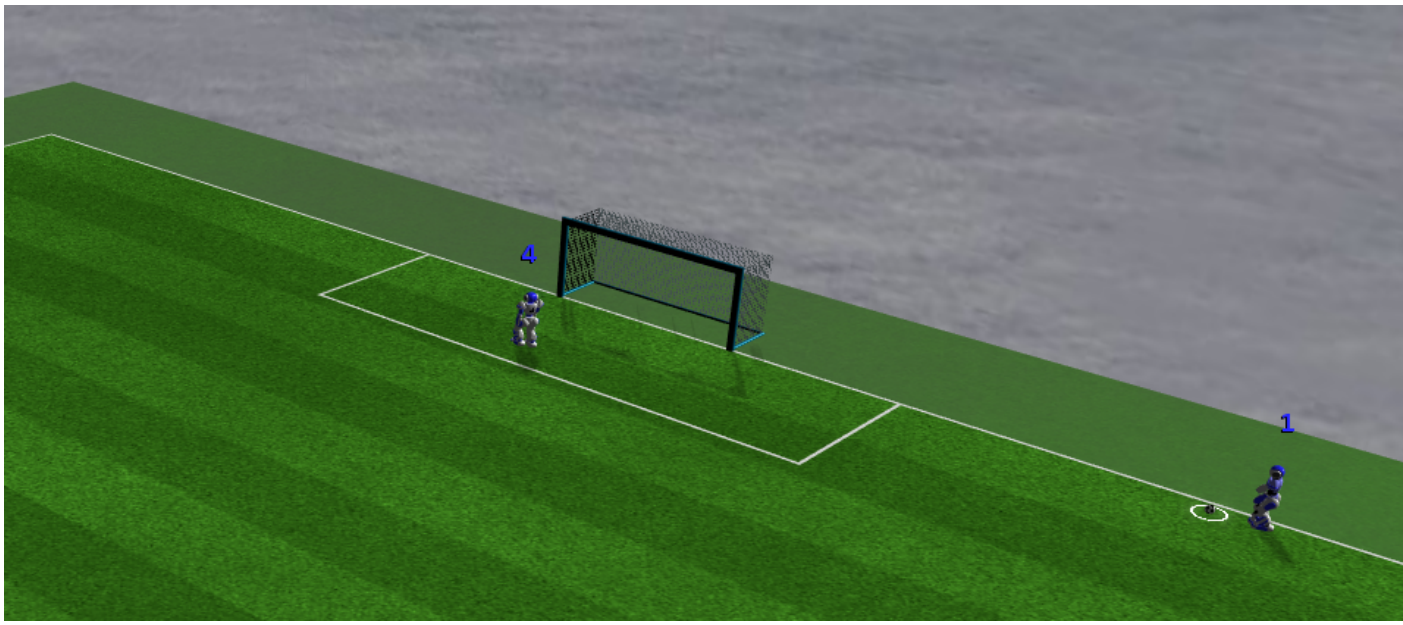

Figure 3.4: Beginning of the set play.

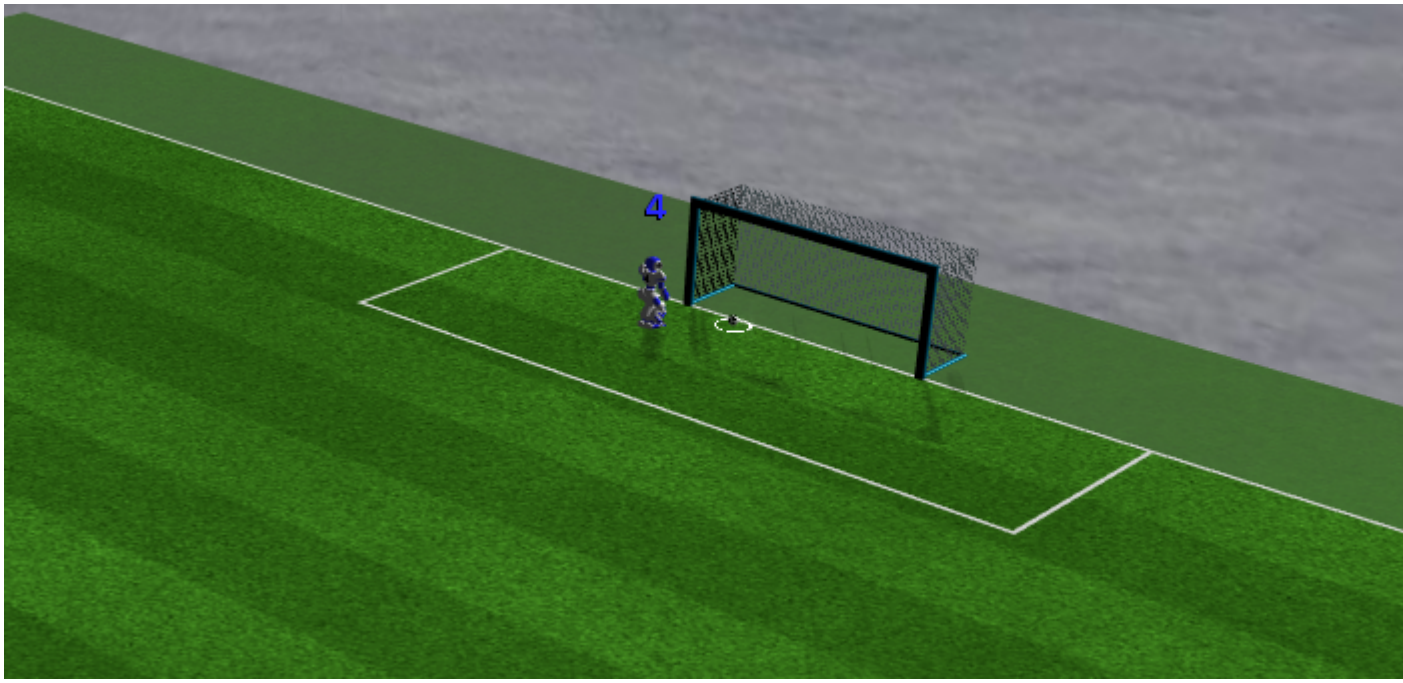

Figure 3.5: Ball crossed to kicking agent.

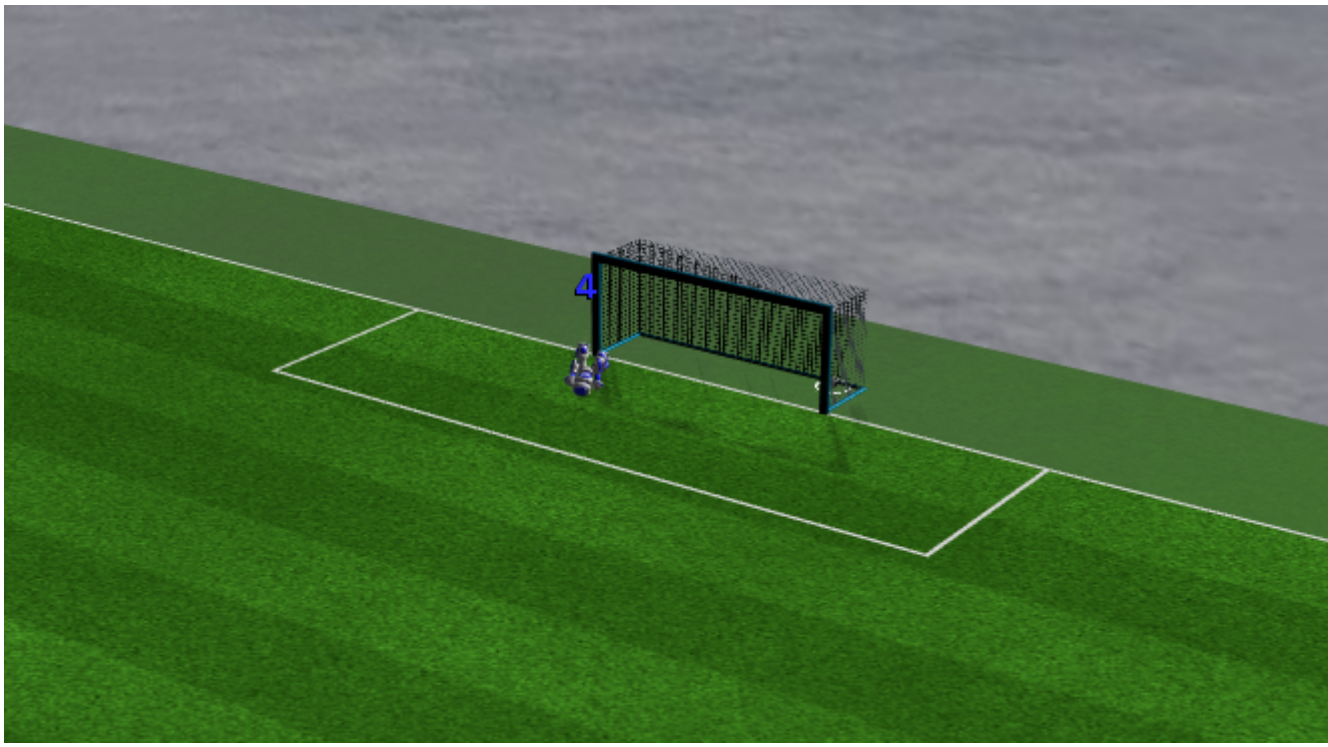

Figure 3.6: Ball kicked to goal by the kicking agent.

With the usage of the tensorboard, which stores information that it uses to generate several plots, additional parameters were allowed to be evaluated.

Firstly, on the graphic showed in 3.7, one can analyse the mean magnitude of the policy loss function. This graphic is related to how much of the process of deciding actions is changing. Ideally thisvalue should decrease during a successful training session andbe less than 0. It is a value that oscillates. Look to the following image, with regards to the specifications of our environment we can see that during the training the value decreases, representing a successful training. Furthermore, the graphic was smoothed, but it is still possible to notice the oscillating values.

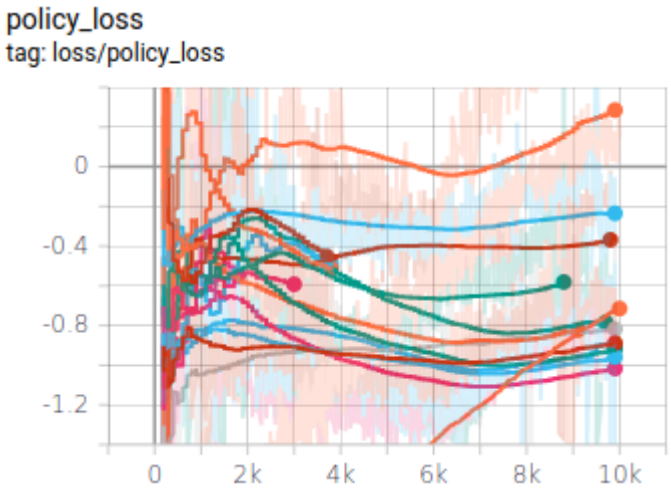

Figure 3.7: Graphic that represents, throughout the training, the policy loss.

Secondly, on the image 3.8 the learning of the robot is evaluated, as it represents the mean loss of the value function update. Thus, this graphic should increase as the robot learns and decrease when the reward stabilizes. With the learning curves in 3.3 and 3.1, we can see that the reward rapidly becomes stable. Hence a drop appears early.

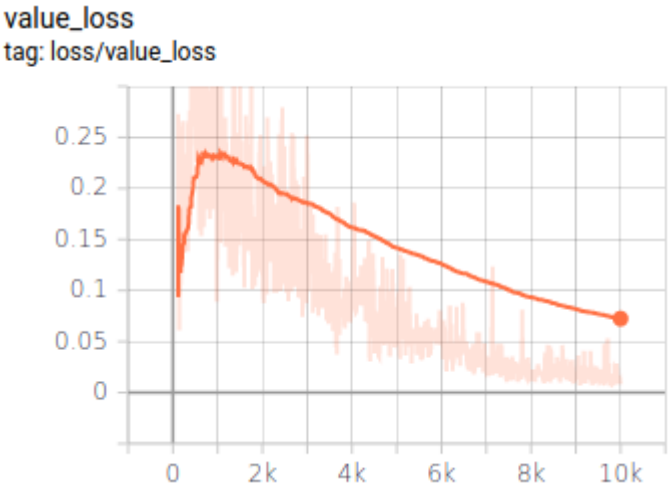

Figure 3.8: Plotof the value lossfor a training session.

Chapter 4

Conclusions and Future Work

4.1 Conclusion

At the beginning of this dissertation the team established the goals that were going to be worked upon. These included the investigation of a currently growing area - the multi agent system environment, regarding machine learning - and the RoboCup, explained in the 2.1 section. The prior required a study of the state of the art tools to face the several inherent problems, while the latter included the study of the team's code and the story behind the competition. Following this plan I was able to learn about the problems that surround the topic - the definition of rewards in a MAS environment, the coordination of the robots. Furthermore, I learned a lot about a fascinating competition, the RoboCup, and how the FCPortugal3D team works. After some study within these subjects, the goal was to find a way, where we could connect several aspects. First of, we wanted to use an innovative area, increasing in use within machine learning - Deep Learning, 2.4. The idea was to make use of deep reinforcement learning algorithms to create trained models of usable behaviors for the team. Lastly, after a very useful tool being developed, the FCPgym 2.1.5.5, it was also a goal to make use of it. Combining these tools, this dissertation ultimately aimed at creating aset play, where several robots cooperated among them - and train it with DRL algorithms. By using the code of the team I was able to create a simpleset play, such as the corner kick, which consisted of a robot crossing the ball and another one kicking it into the goal. Some variations were also tested, such as a tap-in , where, instead of crossing, the agent would just run into the ball to score a goal, and also, using the same logic, creating a throw in. With the FCPGym tool, each agent consisted of one environment. The challenge was to create reward definitions where all combined would lead to a successful set play. Furthermore, some challenges were faced when trying to synchronize the agents and the neural network outputs - these were mostly overcome by using the ball coordinates, creating action zones for the agents and using sync methods for the neural network outputs. Additionally, with the help of the team and the tool mentioned above, I was able to discover the stable baselines framework which offered a good ammount of deep reinforcement algorithms. By studying the problem and knowning I needed support for continuous action spaces, the choice landed between SAC and TD3. The main difficulties were mainly due

to the weird times we are in, as it delayed and made it hard for me to fully integrate within the source code and process. However, do to a very, always around and available team, the model was successfully trained, mostly with TD3, creating a corner set play with the goals mentioned before. After looking at the results we could see that the agent effectively learned, as the reward plots show, and with the help of the simulator we could see the trained model in action.

4.2 Future Work

With the end of this dissertation I feel like improvements were made on the subject but there is still work to be done and a lot can still be achieved.

Firstly, one goal for future work to be done would be to increase the complexity of the set plays. As we achieved in creating simple set plays, one could increase the number of robots involved in the play, or create more innovative strategies to gain advantage over the opponents. Following the rules of the competition and having the work of this disseration in mind, the structure could be used to create a more complex model such as a kick off that leads into a fast goal.

Secondly, better testing with better hardware could be done, allowing an increase on the number of episodes for training and efficiency, for better testing and training of the model.

Appendix A

Gantt planning

In this appendix, the image, which followed the proposed work in the 3 before the start of this dissertation, is presented. The following image A.1 shows the initial work plan, which consisted in a review of the state of art, the FCPortugal3D code. Afterwards the development of a model would start alongside it's optimization with depp learning.

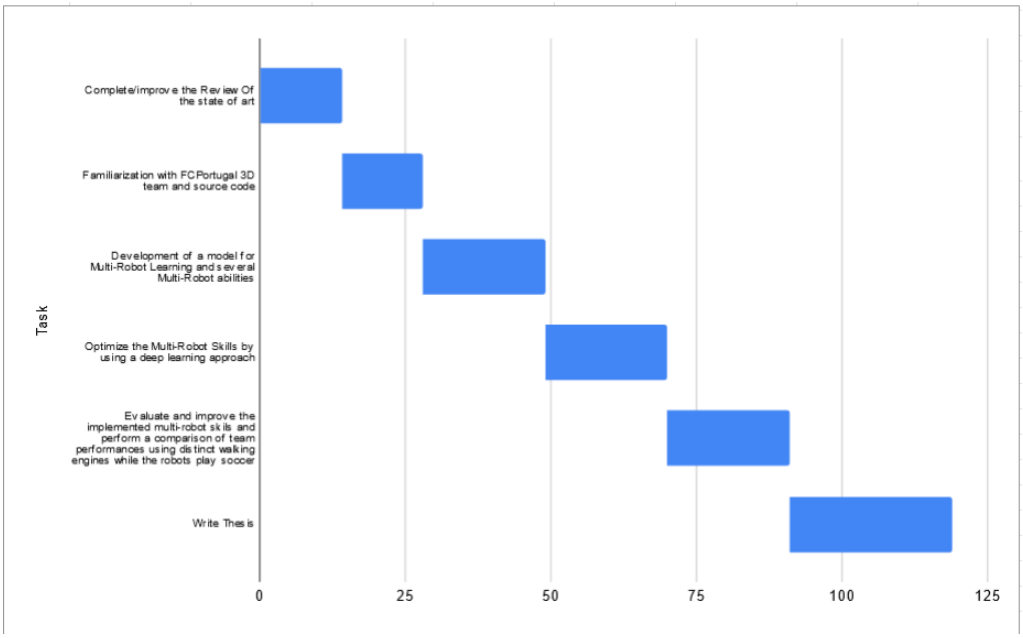

Figure A.1: Gantt plan of the work to develop

References

[1] Abbas Abdolmaleki, B. Price, N. Lau, L. Reis, and G. Neumann. Deriving and improving cma-es with information geometric trust regions. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017.

[2] Abbas Abdolmaleki, Bob Price, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Deriving and improving cma-es with information geometric trust regions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 657–664, 2017.

[3] Abbas Abdolmaleki, D. Simões, N. Lau, L. Reis, and G. Neumann. Learning a humanoid kick with controlled distance. In *RoboCup*, 2016.

[4] Abbas Abdolmaleki, D. Simões, N. Lau, L. Reis, B. Price, and G. Neumann. Stochastic search in changing situations. In *AAAI Workshops*, 2017.

[5] Abbas Abdolmaleki, David Simões, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Learning a humanoid kick with controlled distance. In Sven Behnke, Raymond Sheh, Sanem Sarıel, and Daniel D. Lee, editors, *RoboCup 2016: Robot World Cup XX*, Cham, 2017. Springer International Publishing.

[6] M. Abreu, N. Lau, Armando Sousa, and L. P. Reis. Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning. *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–8, 2019.

[7] Miguel Abreu, Nuno Lau, Armando Sousa, and Luis Paulo Reis. Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–8. IEEE, 2019.

[8] Sorin Achim, P. Stone, and M. Veloso. Building a dedicated robotic soccer. 1996.

[9] Stephen Balakirsky. Results and performance metrics from robocup rescue 2007. 2007.

[10] Justin Boyan, Dayne Freitag, and Thorsten Joachims. A machine learning architecture for optimizing web search engines. In *AAAI Workshop on Internet Based Information Systems*, pages 1–8, 1996.

[11] Tiago Rafael Ferreira da Silva. Humanoid low-level skills using machine learning for robocup. 2019.

[12] Henrique da Silva Teixeira, T. Silva, M. Abreu, and L. P. Reis. Humanoid robot kick in motion ability for playing robotic soccer. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 34–39, 2020.

[13] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[14] Mark d'Inverno, M. Luck, and M. Wooldridge. Proceedings of the fifteenth international joint conference on artificial intelligence, ijcai 97, nagoya, japan, august 23-29, 1997, 2 volumes. In *IJCAI*, 1997.

[15] Hassan Ismail Fawaz, G. Forestier, Jonathan Weber, L. Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33:917–963, 2019.

[16] Rui Ferreira, Luís Paulo Reis, António Paulo Moreira, and Nuno Lau. Development of an omnidirectional kick for a nao humanoid robot. In *IBERAMIA*, 2012.

[17] Rui Ferreira, Nima Shafii, Nuno Lau, Luís Paulo Reis, and Abbas Abdolmaleki. Diagonal walk reference generator based on fourier approximation of zmp trajectory. *2013 13th International Conference on Autonomous Robot Systems*, pages 1–6, 2013.

[18] R. Firby, Peter N. Prokopowicz, M. Swain, R. Kahn, and D. Franklin. Programming chip for the ijcai-95 robot competition. *AI Mag.*, 17:71–81, 1996.

[19] Keith D. Foote. A brief history of deep learning. https://ssim.robocup.org/3d-simulation/3d-rules/. Accessed: 2020-09-07.

[20] Scott Fujimoto, H. V. Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018.

[21] I. Goodfellow, Yoshua Bengio, and Aaron C. Courville. Deep learning. *Nature*, 521:436–444, 2015.

[22] T. Haarnoja, Aurick Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[23] A. K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, March 1996.

[24] B. Kurdi, S. Gershman, and M. Banaji. Model-free and model-based learning processes in the updating of explicit and implicit evaluations. *Proceedings of the National Academy of Sciences*, 116:6035 – 6044, 2019.

[25] N. Lau and L. Reis. Fc portugal - high-level coordination methodologies in soccer robotics. 2007.

[26] N. Lau, L. P. Reis, N. Shafii, R. Ferreira, and A. Abdolmaleki. Fc portugal 3 d simulation team : Team description paper. 2011.

[27] Chong Li. Model-free reinforcement learning. 2019.

[28] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

[29] G. Litjens, Thijs Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. V. D. Laak, B. Ginneken, and C. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[30] Alan K. Mackworth. On seeing robots. In *Computer Vision: Systems, Theory and Applications*, 1993.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, S. Petersen, C. Beattie, A. Sadik, Ioannis Antonoglou, H. King, D. Kumaran, Daan Wierstra, S. Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[33] Joni Salminen, Juan Corporan, Roope Marttila, Tommi Salenius, and Bernard J. Jansen. Using machine learning to predict ranking of webpages in the gift industry: Factors for search-engine optimization. In *icist 2019*, 2019.

[34] Mohit Sewak. Actor-critic models and the a3c. 2019.

[35] N. Shafii, Abbas Abdolmaleki, N. Lau, and L. P. Reis. International journal of advanced robotic systems development of an omnidirectional walk engine for soccer humanoid robots regular paper. 2016.

[36] Nima Shafii, Abbas Abdolmaleki, Rui Ferreira, Nuno Lau, and Luís Paulo Reis. Omnidirectional walking and active balance for soccer humanoid robot. In *EPIA*, 2013.

[37] Nima Shafii, Abbas Abdolmaleki, Nuno Lau, and Luís Paulo Reis. Development of an omnidirectional walk engine for soccer humanoid robots. *International Journal of Advanced Robotic Systems*, 12:193, 2015.

[38] Nima Shafii, Nuno Lau, and Luis Paulo Reis. Learning to walk fast: Optimized hip height movement for simulated and real humanoid robots. *Journal of Intelligent & Robotic Systems*, 80(3-4):555–571, 2015.

[39] David Simões, Nuno Lau, and Luís Paulo Reis. Multi-agent double deep q-networks. In Eugénio Oliveira, João Gama, Zita Vale, and Henrique Lopes Cardoso, editors, *Progress in Artificial Intelligence*, pages 123–134, Cham, 2017. Springer International Publishing.

[40] Z. Song and W. Sun. Efficient model-free reinforcement learning in metric spaces. *ArXiv*, abs/1905.00475, 2019.

[41] R. Sutton and A. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 2005.

[42] Subramaniyaswamy Va, Logesh Rb, Vijayakumar Vc, and Indragandhi Vd. Automated message filtering system in online social network. *Procedia Computer Science*, 50:466–475, 2015.

[43] Jindong Wang, Y. Chen, Shuji Hao, Xiaohui Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *ArXiv*, abs/1707.03502, 2019.

[44] Tingwu Wang, Xuchan Bao, I. Clavera, Jerrick Hoang, Yeming Wen, Eric D. Langlois, S. Zhang, Guodong Zhang, P. Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *ArXiv*, abs/1907.02057, 2019.

[45] Sinan Çalisir and M. K. Pehlivanoglu. Model-free reinforcement learning algorithms: A survey. *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2019.