UNIVERSIDAD DEL ROSARIO

UNDERGRADUATE THESIS

Enhancing Performance in Special Teams within the NFL through Reinforcement Learning: A Data-Driven Approach

---

Author:
Santiago ALVAREZ BARBOSA

Supervisor:
Alexander
CAICEDO-DORADO

A thesis submitted in fulfillment of the requirements
for the degree of Professional in Applied Mathematics and Computer Science

School of Engineering, Science and Technology
Applied Mathematics and Computer Science

August 17, 2023

Declaration of Authorship

I, Santiago ALVAREZ BARBOSA, declare that this thesis titled, "Enhancing Performance in Special Teams within the NFL through Reinforcement Learning: A Data-Driven Approach" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Analytics is all about finding ways to gain a competitive advantage, and math is the key to unlocking that advantage."

- John Urschel

UNIVERSIDAD DEL ROSARIO

Abstract

School of Engineering, Science and Technology
Applied Mathematics and Computer Science

Professional in Applied Mathematics and Computer Science

Enhancing Performance in Special Teams within the NFL through Reinforcement Learning: A Data-Driven Approach

by Santiago ALVAREZ BARBOSA

The integration of data science and artificial intelligence techniques in sports has gained significant attention in recent years. This thesis investigates the application of these techniques in the context of American Football, specifically within the special teams unit. Using the NFL Big Data Bowl 2022 dataset and OpenAI Gym API, a tailored dynamic environment was developed to facilitate the training of two distinct agents representing punt and kick-off returners, as well as pursuers.
The objective is to equip special team coaches with valuable insights into decision-making processes and effective utilization of players' skills.

The research begins by compiling and analyzing relevant data from the NFL Big Data Bowl 2022 dataset, which provides a comprehensive collection of game statistics and player tracking information. The dataset was cleaned and processed to create a custom environment suitable for agent training, and designed to simulate realistic scenarios faced by special teams in American Football. This environment enables agents to learn optimal pathfinding strategies based on reinforcement learning techniques.

In this study, via reinforcement learning, the agents learned to navigate a dynamic environment by receiving feedback in the form of rewards and penalties based on their actions. The agents' goal is to achieve their respective objectives, whether to tackle the returner (pursuer) or to maximize the field position (returners). Through the training process, agents improve their decision-making abilities and develop strategies to optimize their performance.

The effectiveness of the developed agents was evaluated by comparing their performance against predefined routes and focusing on two metrics, *ep_len_mean* and *ep_rew_mean*, which encompass the mean values of the episode length and reward. The results of this study provide insights into the behavior and performance of agents in a simulated environment. With regards to the metrics, for the Returner there was an average value of 61.846 for the mean length of episodes and a value of 7.2274 for the mean reward of episodes. However, for Pursuer, there was an average value of 60.024 for the mean length of episodes and 8.694 for the mean reward of episodes. Both agents demonstrated the ability to learn and make decisions that enabled them to achieve their goals. However, it is important to note that the results varied, with one agent exhibiting better performance than the other.

Nonetheless, the fact that both agents were able to learn and improve their decision-making capabilities was a positive outcome of this study.
Special team coaches can use these insights to make informed decisions and enhance team performance.

This study contributes to the advancement of data science and artificial intelligence in American Football. Valuable solutions are provided to the challenges faced by coaches and players. The findings of this study can improve decision-making processes, optimize player strategies, and ultimately lead to improved outcomes in American Football games. Future research could explore additional applications of Data Science and Artificial Intelligence in different aspects of sports, further enhancing their potential for performance improvement.

Acknowledgements

I would like to dedicate this space to express my deep gratitude to all the people who have been fundamental in the development of this thesis.

First, I want to thank my thesis advisor, Alexander Caicedo-Dorado for their invaluable support and guidance throughout the entire process. Their expertise and dedication have played a crucial role in shaping my academic path in the field of artificial intelligence. I am especially grateful for their willingness to listen to my ideas, constant support, and guidance.

I want to extend my heartfelt thanks to Dana. Words cannot describe how grateful I am for all the support you have always given me. Every word of encouragement, philosophical advice and gesture of trust helped, and I will always appreciate that. (P.S. I told you so! You owe me an ice cream).

Lastly, and most importantly, I would also like to acknowledge and express my deep appreciation to my entire family.
Andrés, siempre estaré agradecido por tu apoyo, ayuda y orientación en este proceso, por alentarme a cumplir mis metas. Eres la persona a quien más admiro.
Angi, my twinsy, también quiero agradecerte por todo. Sé que a veces puedo ser molesto, pero gracias por siempre saber cómo hacerme reír y por ser la mejor hermana del mundo.
Mamá y Papá, sin ustedes no sería quien soy hoy ni estaría donde estoy. Gracias por absolutamente todo, por su apoyo inquebrantable y, sobre todo, por el amor que me brindan.

A todos, gracias por confiar en mí y apoyarme en este sueño.

Contents

List of Figures

List of Tables

Dedicated to my family

Chapter 1

Introduction

1.1 NFL and NFL Big Data Bowl

The National Football League (NFL), founded in 1920 in Canton Ohio, is widely regarded as the most popular/profitable sports league in North America, and the highest level of American Football in the world. The league consists of thirty-two teams divided into two conferences: the National Football Conference (NFC) and American Football Conference (AFC). Each conference is further divided into four divisions: North, East, South and West.
During a regular season, each team plays 17 games over 18 weeks, and the first teams in each division, alongside three wildcard teams, qualify for the postseason. The postseason then culminates in the Superbowl, with the NFC champion facing off against the AFC champion.

Each American football team is comprised of a fifty-five man roster, out of which fifty-three players are listed as active and only forty-eight of those players are deemed eligible to participate in a game. Furthermore, the roster is separated into three units: offense, defense, and special teams.

1. The offensive unit is responsible for scoring points and moving the ball down the field. This unit includes different player positions, such as quarterback, running back, wide receiver, tight end, and offensive linemen.

2. The defensive unit is responsible for preventing the opposing team from advancing down the field, causing turnovers and if possible, scoring points. The players included in this unit are defensive linemen, linebackers, safeties, and cornerbacks.

3. The special teams unit is responsible for kickoffs, field goals, extra points and punts with the objective of obtaining a good field position to generate an advantage for their team. This unit consists of players, such as kickers, punters, and returners.

Recently, the NFL began to implement Machine Learning (ML) and Artificial Intelligence (AI) in order to obtain better insights on the vast amount of data generated in the following fields [1].

- Activity and game analytics: Match outcome modelling, player/ball tracking, tactical classification, sports betting, and refereeing assistance.

- Talent identification: Player recruitment based on sporting performance.

- Training and coaching: Evaluation of the effectiveness of tactical formations, player decision-making, player injury modelling, and biomechanics.

Amazon Web Services (AWS), which is the official provider of technology related to Cloud Computing and AI, allows the NFL to utilize advanced data analytic tools and aid in the development of Next Generation Statistics (NGS), a platform that grants teams a variety of general, specific, and innovative game statistics. In addition, the NFL Big Data Bowl, an annual competition focused on sports analytics organized by the league in collaboration with Sportradar, invites students and professionals to work with NFL data to develop new solutions to help coaches, players, and teams. Thus allowing for more data-driven decisions to be made within the league [2].

1.2 Artificial Intelligence and Machine Learning

Artificial Intelligence is a multidisciplinary field that encompasses computer science, mathematics, linguistics, psychology, and even philosophy. Its aim is to develop computer systems that can mimic human behavior and perform tasks that usually require human intelligence [3]. Such systems are classified into three categories: rule-based, machine-learning, and deep-learning systems. Unlike rule-based systems, which rely on a set of predefined rules to make decisions, machine learning and deep learning systems learn from data, implement models for data processing, which are able to improve their performance over time.

Machine learning is one of the main sub-fields of artificial intelligence, which focuses on the study and design of computational algorithms and the construction of statistical models based on large amounts of data. These models are designed to emulate human intelligence by learning from the surrounding environment and previous experience without being explicitly programmed [4]. There are three types of Machine Learning algorithms:

1. Supervised Learning: The algorithm generates a function that maps the inputs to the desired outputs [5]. In other words, the system is trained on labeled data, where the output is known, and the goal is to learn how to predict the correct output of unknown input data.
2. Unsupervised learning: The model is trained on an unlabeled data set with the goal being to identify patterns within the data.
3. Reinforcement Learning (RL): This algorithm learns a policy of how to act via rewards or punishments given an observation of the environment. The goal is to find the optimal policy that maximizes the expected rewards

1.3 Reinforcement Learning

As mentioned in section 1.2, Reinforcement Learning is one of the three main paradigms of learning within ML. Here an agent learns from interacting with an environment; thus, by feedback in the form of rewards or punishments, the agent is able to make appropriate decisions [6].
The Reinforcement Learning framework comprises the following components.

- Agent: Decision maker that interacts with the environment. The actions taken by the agent were derived from the state of the environment.
- Environment: Context in which the agent interacts.

- State: Representation of the environment used by the agent to make decisions.
- Action: Decision made by the agent.
- Reward: Value that represents the feedback provided by the environment given a specific action taken by the agent.

Reinforcement learning is a versatile technique that can be applied to a wide range of fields. For example, robots can be taught to perform tasks such as grasping objects and walking or navigating the environment. Using reinforcement learning, self-driving cars can be taught to make decisions in complex traffic situations. Finally, the development of trading strategies for markets, where an agent can learn to make decisions based on historical data and market trends.

Concerning sports, reinforcement learning is being increasingly used for a variety of applications, nonetheless the three main applications are the following:

1. Athlete training: Reinforcement learning can be used to help athletes or teams improve their performance in specific areas such as timing, coordination, and decision-making.
2. Strategy development: Optimal game strategies can be developed based on a team's or athlete's past performance, as well as the behavior of their opponents.
3. Game simulations: Specific scenarios can be simulated in sports, which can help coaches and players to test the effectiveness of different strategies and tactics.

Path-finding problems, which incorporate the three key applications stated above, represent an area where reinforcement learning can be applied. By utilizing RL, an agent can be trained to efficiently navigate through an environment, create optimized strategies for diverse routes, and simulate various scenarios in both static and dynamic environments.

1.4 State of the Art

The implementation of reinforcement learning in path-finding problems has been addressed previously in the literature, and significant progress has been made recently.

A study by Marco A. Wiering in *Reinforcement Learning in Dynamic Environments using Instantiated Information*, explores the challenges of using Reinforcement Learning in dynamic environments, with moving objects which make optimal planning difficult. A new approach called Instantiated Information is proposed, here additional information with respect to the moving components of the environment is provided to the agent, in order to improve decision making and re-plan model-based reinforcement learning techniques whenever information changes [7].

Concerning Neural Networks in path planning within dynamic environments, *Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning*, an article written by Xiaoyen Lei, Zhian Zhang and Peifang Dong, proposes a deep learning algorithm that integrates deep neural networks with a Q-learning

to generate optimal actions based on sensory information. The solution provided yields promising results as state-of-the-art methods are outperformed in terms of accuracy and efficiency [8]. Furthermore, neural networks can be trained to learn how to navigate through game environments based on sensors that provide information to improve performance as seen in the article *Neural Networks for Real-time Path-finding in Computer Games* by Ross Graham, Hugh McCabe and Stephen Sheridan [9].

Path finding can also be implemented in dynamic environments with more than one agent, also known as multi-agent path finding. Within the text, *Multi-Agent Path Finding using a Dynamic Distributed Deep Learning Model*, a deep learning model that can make decisions based on the current local surrounding information, is implemented. It was found that classical approaches are unfair to real-life applications because of their inability to handle uncertain and dynamic scenarios. Such model involves a combination of Convolutional Neural Network (CNNs) and Long Short-Term Memory (LSTM) networks trained by reinforcement learning. The results demonstrated the effectiveness of using these techniques in dynamic environments [10]

Another publication that deals with multi-agent path finding is *Path-finding in Multi-Agent, unexplored And Dynamic Military Environment Using Genetic Algorithm* written by Saeed Saeedvand, Seyed Nasar Razavi and Fahimeh Ansaroudi. As outlined in the manuscript, challenges that arise when finding an optimal path in these environments such as obstacles, coordination between agents, and changing conditions, can be addressed using a Genetic Algorithm. A Genetic Algorithm is a type of heuristic optimization algorithm that involves evolving a population of solutions over generations by applying selection, crossover, and mutation to find the fittest solutions in the population. By implementing this algorithm in path finding problems, multiple challenges presented can be solved with relative ease [11].

An alternative context in which reinforcement learning can be implemented, which will be explored in this thesis, is the use of this approach in sports, where environments are highly dynamic. As mentioned in Section 1.1, American Football is one of the leading sports industries in terms of machine learning and artificial intelligence, thus making it a perfect case study for the implementation of these techniques. Previous studies have been conducted with respect to the incorporation of these concepts within sports, such as in the paper *A Decision-making and Actions Framework for Ball Carriers in American Football* by Danny Jugan and Dewan T. Ahmed. This paper illustrates how existing models of ball-carrier decision-making are overly simplistic and fail to take into account various factors that influence a player's decision. Therefore, they incorporated multiple decision-making layers with situational awareness and tactical analysis for ball carriers in avoidance and hole selection when running. Their approach provided meaningful insights into the understanding of ball-carrier decisions that can lead to improved performance [12].

In terms of supervised learning and multi-agent frameworks, the articles *Supervised Learning in Football Game environments using Artificial Neural Networks* and *Multi-agent off-screen behavior prediction in football present useful information*. The first article mentions the use of artificial networks in football games in large data-sets of game footage, where the model can provide more data-driven and objective feedback to players in order to improve performance [13]. The second article mentioned

a novel approach to predict the future behavior of football players using deep reinforcement learning based on previous actions taken. Furthermore, the authors take into account other game influences like ball position and time, which allow them to improve effectiveness in their approach [14].

1.5 Problem Statement

The primary aim of this thesis is to investigate the integration of data science and artificial intelligence techniques in American Football, specifically within the context of special teams. By leveraging the NFL Big Data Bowl 2022 data-set and the OpenAI Gym API, a tailored dynamic environment will be developed to facilitate the training of two distinct agent sets. Each set will represent a unique position and function within the special teams unit and will be capable of learning optimal path-finding strategies to achieve their respective objectives.

The overall goal of this project is to equip special team coaches with valuable insights into decision-making processes, and how they can effectively utilize the skills of punt and kick-off returners and pursuers. Specifically, this involves analyzing the behavior of these agents within a simulated dynamic environment, and identifying optimal paths for achieving their respective goals - be it maximizing field position (returners), or reaching the returner (pursuer).Through a combination of Data Science, Artificial Intelligence and Reinforcement Learning methodologies, this study aims to provide valid approaches to enhance performance in special teams, ultimately leading to improved outcomes in American Football games.

1.6 Objectives

1.6.1 General objective

Maximize the field position of the returner and pursue the returner while avoiding maximizing field position for the pursuer.

1.6.2 Specific objectives

In order to achieve the general objective stated in Section 1.5, the following specific objectives were identified:

1. Compile and analyze relevant data from the NFL Big Data Bowl 2022 data-set to facilitate the creation and development of customized environments and agents.
2. Develop a dynamic custom environment to simulate realistic scenarios faced by special teams in American Football to allow for comprehensive agent training.
3. Implement reinforcement learning techniques to enable the agents to learn optimal path finding strategies that suit their objective.
4. Evaluate the effectiveness of developed agents in achieving their objectives, as well as measure the performance of models implemented against relative metrics.

5. Provide insights based on the results of the study to support informed decision-making and improve special team performance.

By achieving these objectives, significant contributions can be made to the advancement of data science and artificial intelligence within American Football and valuable solutions can be provided to the challenges faced by special teams.

1.7 Outline

The outline of this thesis is as follows.

- Chapter 2 provides an explanation of the theoretical framework. Definitions of concepts are given. Furthermore, the data is analyzed and thoroughly cleaned to prepare it for the development of the agents and dynamic environment.
- Chapter 3 delves into the implementation of the environment and agents. This chapter explains the concepts given in the previous sections within the context of this thesis.
- Chapter 4 explains the reinforcement learning model implemented with OpenAI Gym. This chapter covers the key features and functionalities of the model
- Chapter 5 presents the simulations and results of the study, including training, evaluation, and predictions with respect to the data.
- Chapter 6 is a discussion of the results. A comprehensive discussion of the key findings and their implications is presented.
- Chapter 7 presents the conclusions of this study, including possible improvements and future work that can be built on this research.

Chapter 2

Theoretical Framework

2.1 Concepts

To gain a comprehensive understanding of the roles played by artificial intelligence and reinforcement learning within the framework of this thesis, it is crucial to bear in mind specific concepts related to these fields. These concepts serve as a foundation for linking the different components of the framework; therefore, it is important to thoroughly understand these concepts and the connection between them.

2.1.1 Neural Networks

Artificial Neural Networks (ANNs) are computing systems inspired by early models of sensory processing in the brain. As shown in figure 2.2, they are composed of a large number of nodes that simulate a network of neurons to perform complex computations [15] [16].

The neurons can receive either input from an external source or from a number of other units. Subsequently, it processes the information by weighing each input, adding them, and comparing the total input with a certain threshold set by an activation function to output a signal that is passed to other neurons. In addition, neural networks are trained via gradient descent, and use back-propagation to compute the derivatives at different points in the network. The process begins with the weights being set to small random numbers. For each input, the network provides its respective output, and the squared difference between the model's output and the desired output is measured. This error is then used to adjust the weights of the connections between layers, thus improving the network's performance and obtaining a neural network that best suits the problem at hand. [15].

There exist different architectures of neural networks, but here the three most used ones will be explained, each with its own structure and characteristics. A feed-forward neural network, shown in figure 2.3, is often used in classification and regression tasks and is the simplest type of neural network, as the signals only flow in one direction.

Another common type is the Convolutional Neural Networks (CNN), as seen in figure 2.4, are mainly used for processing data with a known grid-like topology [18]; therefore, they are used for image processing tasks. This type of network implements a convolutional layer that applies filters to detect edges, corners, and textures within the image.

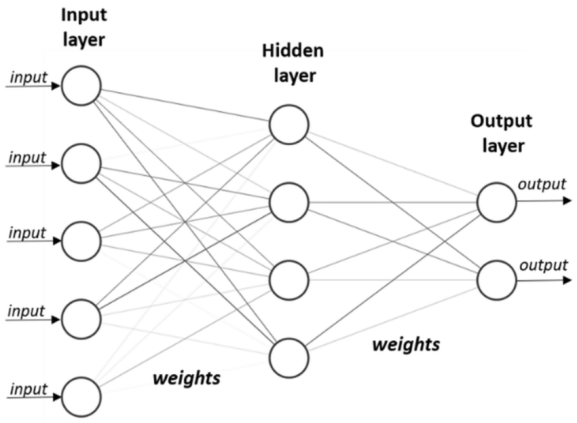

FIGURE 2.1: Back propagation process [17].

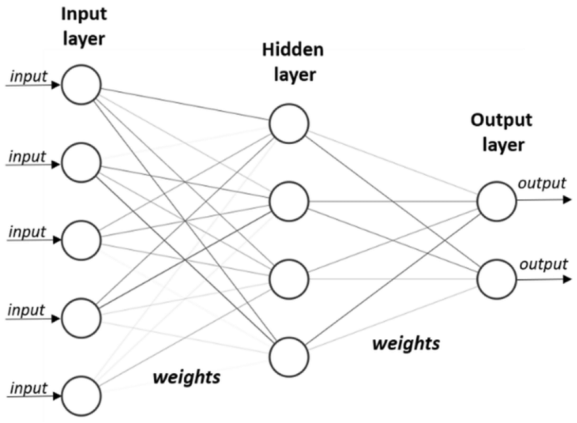

FIGURE 2.2: Neural Network [17].

Finally, Recurrent Neural Networks (RNN) are used for sequential information, where there is a sequential relationship between the input and the output [6]. This type of network has feedback connections that allow for an internal state, meaning that they have a memory that can keep track of information from previous inputs. This can be seen in figure 2.5.

2.1.2 Deep Learning

Deep learning attempts to model high-level abstractions in data by using deep networks, trained by means of supervised and/or unsupervised learning algorithms [6]. These types of models have multiple layers of representation; however, the learning process is the same as that explained in the previous section 2.1.1. Furthermore, learning complex tasks is possible because of the hierarchical representation of data through the use of ANNs with multiple layers. By doing so, deep learning models can discover intricate structures in large data-sets [22].

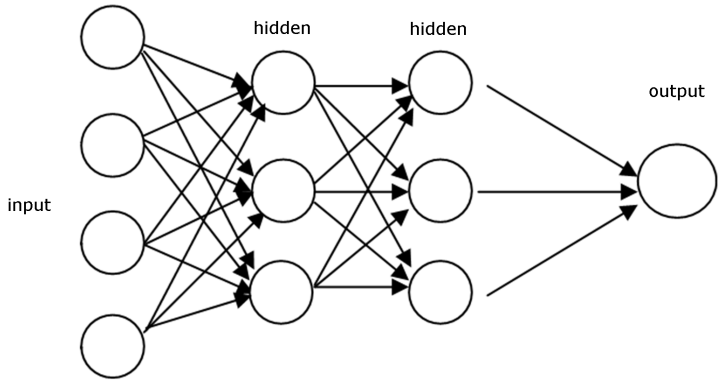

FIGURE 2.3: Feed Forward Neural Network [19].

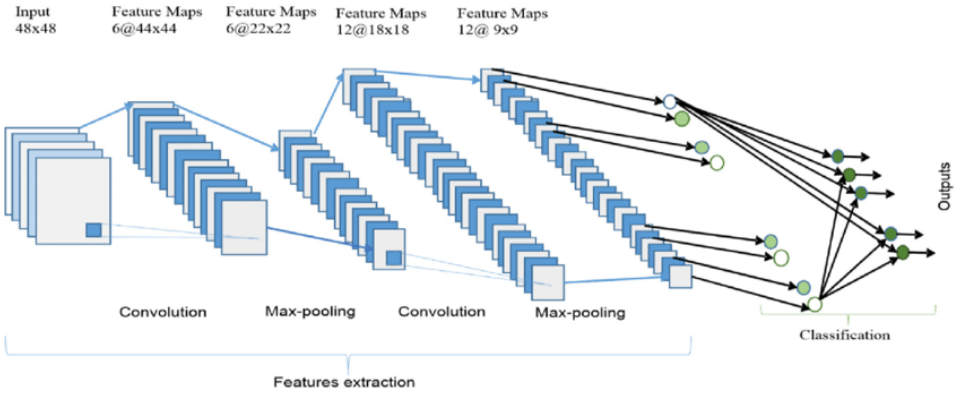

FIGURE 2.4: Convolutional Neural Network [20].

2.1.3 Markov Decision Process

The Markov property is a fundamental concept in probability theory, in which the probability distribution of future states depends only on the current state and action but not on the past. When a Reinforcement Learning problem satisfies this property and has a finite number of states and actions, the problem can be formulated as a Finite Markov Decision Process (FMDP) defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ [23].

- $\mathcal{S}$: Finite set of possible states.
- $\mathcal{A}$: Finite set of legal actions.
- $\mathcal{P}$: Transition probability function, which defines the probability of moving from one state to another, given a specific action. In mathematical notation, $\mathcal{P}(s, a, s')$ denotes the probability of transitioning from state s to s' when action a is taken
- $\mathcal{R}$: Reward function that maps the state/action pair to a scalar value. $\mathcal{R}(s, a, s')$ denotes the reward obtained when moving from state s to s' after taking action a.
- γ: Discount factor between 0 and 1 which balances the immediate and future rewards.

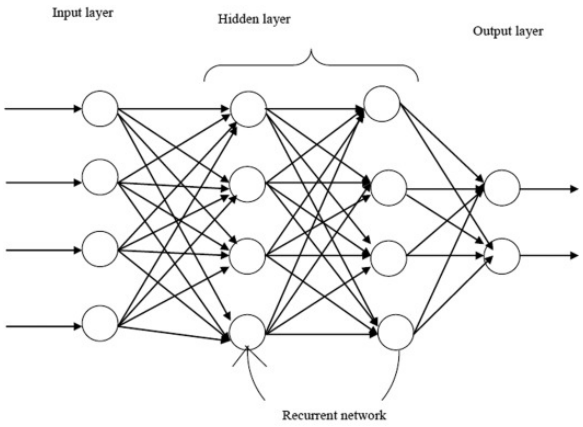

FIGURE 2.5: Recurrent Neural Network [21].

In an MDP, the agent acts as follows: at each time step $t = 0, 1, 2, \ldots$ the agent receives an environment state $s_t \in \mathcal{S}$. Based on this state, the agent performs action $a_t \in \mathcal{A}$ and forms the state action pair (s_t, a_t). Time then increases to the next time step $t + 1$, and thus the environment transitions to state $s_{t+1} \in \mathcal{S}$. As a result of action a_t taken from state s_t, the agent receives a reward $r_{t+1} \in \mathcal{R}$. For better understanding, see figure 2.6

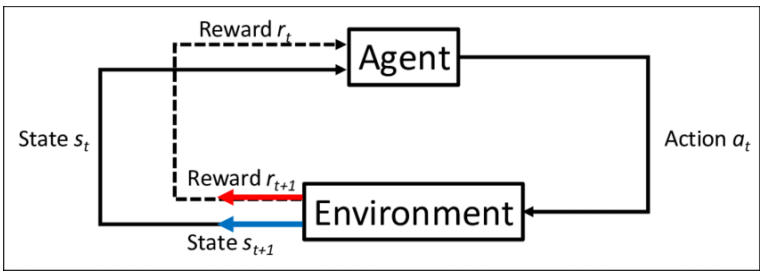

FIGURE 2.6: Agent-environment interaction [24].

Because r_t and s_t have well-defined probability distributions, all possible values that can be assigned to either variable have associated probabilities. Given an arbitrary state $s' \in \mathcal{S}$ and an arbitrary reward $r \in \mathcal{R}$, there is a probability that $s_t = s'$ and $r_t = r$ determined by the preceding state and action. Therefore, the probability of each possible next state s_{t+1}, with reward r_{t+1}, from taking action a_t in the state s_t comes from the transition distribution $\mathcal{P}(s_{t+1}|s_t, a_t)$ [7].

2.1.4 Expected Discounted Return

The concept of the expected return of rewards is introduced to formalize the aim of maximizing the cumulative reward. Let us denote the return of rewards as $\mathcal{G}_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots$. Because it is key to balance the importance of immediate rewards in comparison to future rewards, a discount factor $\gamma \in [0, 1]$ is introduced, which determines the weight given to rewards obtained in the future with respect to rewards obtained immediately. Thus, the discounted return of rewards is given as $\mathcal{G} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$ and the expected discounted return R at time t is defined as:

$$R_t = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right] \tag{2.1}$$

2.1.5 Policies and Value Function

In general, policies are considered rules for choosing actions given the values of states to which those actions will immediately lead [25], thus answering the question of the probability that an agent selects a specific action from a specific state. A policy is denoted by π and it is a function that maps a given state to the probabilities of selecting each possible action from that state. If an agent follows policy π at time t, then $\pi(a|s)$ is the probability that $a_t = a$ and $s_t = s$, i.e. at time t, under policy π, the probability of taking action a in state s is $\pi(a|s)$.

A value function is a function of states or state-action pairs that estimates how good it is for an agent to be in a given state, where the notion of goodness is given in terms of expected discounted return [25]. There are two types of value functions, each of which are defined with respect to policies:

1. State value function: The value of a state under π is denoted by v_π. Formally, the value of s under π is the expected discounted return starting at s in time t, following policy π

$$v_\pi(s) = \mathbb{E}[G_t|s_t = s] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t = s\right] \tag{2.2}$$

2. Action-value function: Denoted q_π, this function gives the value of an action under π. The value of a in state s under π is the expected discounted return from starting at state s at time t, taking action a and following policy π thereafter.

$$q_\pi(s,a) = \mathbb{E}[G_t|s_t = s, a_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t = s, a_t = a\right] \tag{2.3}$$

An RL agent wants to find a policy that achieves the greatest future reward during its execution [6]. A policy π is considered better than or the same as policy π' if the expected discounted return of π is greater than or equal to the expected discounted return of π' for all states. In mathematical notation, this states that $\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s) \forall s \in \mathcal{S}$. This policy is known as the optimal policy and is denoted as π^*.

In turn, the optimal policy has an associated optimal state-value function and an optimal action-value function. The optimal state value function is defined as $v_*(s) = \max_\pi v_\pi(s) \forall s \in \mathcal{S}$, which yields the largest expected return achievable by any policy π for each state.
With regard to the optimal action value function, $q_*(s,a) = \max_\pi q_\pi(s,a) \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$, this function provides the largest expected return achievable by any policy for each possible state-action pair. Additionally, a fundamental property of q_* is that it must satisfy the *Bellman optimality equation*:

$$q_*(s,a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} q_*(s',a')] \tag{2.4}$$

Which states that for any state-action pair (s, a) at time t, the expected return from starting in state s, setting action a and following the optimal policy after, is going to be the expected reward obtained from taking action a in state s plus the maximum expected discounted return that can be achieved from any possible next state-action pair (a', s') [25].

2.1.6 Exploration vs. Exploitation

Exploration and exploitation are two fundamental concepts within the field of RL where there is often a trade-off between them. Exploration refers to the process of an agent trying different actions to gain more information about its surrounding environment, and thus determine which actions lead to the highest rewards. On the other hand, exploitation refers to the process of an agent choosing actions it already knows to yield positive results based on knowledge gained via the environment.

An agent can choose and obtain a balance between the two concepts by implementing an *epsilon-greedy strategy*. This strategy defines an exploration rate $\epsilon = 1$, which represents the probability of whether the agent will explore rather than exploit. As the agent begins to learn, ϵ decays, ensuing the agent to explore the environment as it collects more information, but with time making the agent greedy and more likely to exploit. To determine whether the agent explores or exploits, a random number, $r \in [0, 1]$, is generated. If $r > \epsilon$, it will exploit, meaning it will choose the action with the highest q-value. Otherwise, it explores and randomly chooses an action. By doing so, the agent is able to gather information, improve performance via exploration, and maximize immediate rewards via exploitation.

2.1.7 Key Takeaways

The Reinforcement Learning agent interacts with the environment over time. At each time step t, the agent receives a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$, following a policy $\pi(a_t|s_t)$, which is the agent's behavior (i.e., a mapping from state s_t to actions a_t), receives a scalar reward r_t, and transitions to the next state s_{t+1}, according to the environment dynamics, or model, for reward function $\mathcal{R}(s, a)$ and state transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$, respectively. In an episodic problem, this process continues until the agent reaches the terminal state and then restarts. The return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the discounted accumulated reward with a discount factor $\gamma \in [0, 1]$. The agent aims to maximize the expectation of long-term returns from each state [23].

Therefore, this framework allows the agent to learn from trial and error by receiving rewards with the sole purpose of selecting actions that maximize the expected cumulative reward over time [25].

2.2 NFL Definitions

In addition to the aforementioned concepts, it is essential to consider context-specific definitions of the National Football League that pertain to this thesis.

1. Special Teams: Unit responsible for kicking, punting and returning aspects of the game.

2. Kicker: Player who specializes in kicking field goals, extra points and kickoffs.
3. Punter: Player who specializes in punting the ball.
4. Returner: Player who specializes in returning kickoffs and punts for the receiving team.
5. Coverage player or pursuer: Player in charge of preventing the returner from gaining significant yardage or scoring a touchdown by tackling them as quickly as possible.
6. Touchdown: A scoring play worth six points that occurs when a team manages to advance the ball into their opponents end zone either by carrying the ball across the goal line, catching a pass in the end zone or recovering a loose ball in the end zone.
7. End zone: Rectangular area 10 yards deep at each end of the field bounded by the goal line, side lines, and end line.
8. Field goal: Scoring play worth three points in which a kicker attempts to kick the ball through the opponents' goal posts.
9. Extra point: Scoring play worth one point in which a kicker attempts to kick the ball through the opponents goal posts after a touchdown is scored.
10. Onside Kick: Kick that is intentionally made to travel a short distance in order to give the kicking team a change to recover the ball
11. Kickoff: Play that starts the game at the beginning of each half and occurs after a scoring play. The kicker kicks the ball downfield to the receiving team, and the receiving team returns the ball until they are tackled, go out of bounds or score a touchdown.
12. Punt: A team kicks the ball to the opposing team, usually on the fourth down, in an attempt to gain field position. The opposing team can either fair catch the ball, return it or let it bounce in the hope of resulting in a touchback.
13. Fair catch: Signal made by the returner to catch a punt without being tackled
14. Touchback: Ruling that occurs when the ball bounces or is caught in the end zone. This results in the ball being placed at the 25-yard line
15. Blocked punt: Play in which a player from the opposing team successfully blocks a punt
16. Fake punt: Play where the kicking team appears to attempt a punt but instead run or throw the ball.
17. Fumble: When a player who has possession of the ball loses control of it before the play is over.

2.3 Data

NFL coaches often state that special teams make up a third of the game, as they have a significant contribution with respect to the improvement of a team's field position. However, this unit remains understudied, thus creating an opportunity to better understand its impact on the game.

To shed light on the performance of special teams, the 2022 NFL Big Data Bowl offers five distinct data-sets. These data-sets encompass various types of data, including next-generation statistics (NGS) tracking, play information, game data, player data, and scouting data from the Pro Football Focus (PFF). However, for the purpose of this thesis, only two of these data-sets were deemed relevant and will be explored in more significant depth.

NGS tracking data are one of the core data-sets used in this study. It provides detailed information about the movement and positioning of players during special team plays from 2018 to 2020. These data enables the ability to analyze player trajectories, speeds, accelerations, and other important metrics, offering valuable insights into the spatial and temporal aspects of special team plays.
The second dataset of significance was play data. This dataset contains comprehensive information about each individual special team play, including the teams involved, the type of play (such as kickoff, punt, or field goal), the yardage gained or lost, and other play-specific details. By examining this dataset, a deeper understanding of the strategic choices made by teams, outcomes of different types of plays, and overall effectiveness of special team units can be gained

By focusing on these two data-sets, the aim is to uncover the patterns, trends, and key factors that contribute to the success or failure of special team plays. The combination of NGS tracking data and play information offers a comprehensive perspective on the performance of special teams, allowing the identification of factors that influence field position, scoring opportunities, and game outcomes.

The availability of the 2022 NFL Big Data Bowl data-sets presents an exciting opportunity to delve into the realm of special team analyses and can teach important things about the effectiveness and influence of special teams by concentrating on NGS tracking data and play information. This will help understand the game better and, as stated in this thesis, will give coaches useful information to employ.

2.3.1 Play Data

This data-set contains detailed play-level information. The data-set comprises 19,979 observations and 25 columns, where the key variables are gameId and playId. The following is a brief description of all the variables within the data-set [26]:

- gameId: Game identifier, unique (numeric)
- playId: Play identifier, not unique across games (numeric)
- playDescription: Description of play (text)
- quarter: Game quarter (numeric)
- down: Down (numeric)

- yardsToGo: Distance needed for a first down (numeric)
- possessionTeam: Team punting, placekicking or kicking off the ball (text)
- specialTeamsPlayType: Formation of play: Extra Point, Field Goal, Kickoff or Punt (text)
- specialTeamsResult: Special Teams outcome of play dependent on play type: Blocked Kick Attempt, Blocked Punt, Downed, Fair Catch, Kick Attempt Good, Kick Attempt No Good, Kickoff Team Recovery, Muffed, Non-Special Teams Result, Out of Bounds, Return or Touchback (text)
- kickerId: nflId of placekicker, punter or kickoff specialist on play (numeric)
- returnerId: nflId(s) of returner(s) on play if there was a special teams return. Multiple returners on a play are separated by a ; (text)
- kickBlockerId: nflId of blocker of kick on play if there was a blocked field goal or blocked punt (numeric)
- yardlineSide: 3-letter team code corresponding to line-of-scrimmage (text)
- yardlineNumber: Yard line at line-of-scrimmage (numeric)
- gameClock: Time on clock of play (MM:SS)
- penaltyCodes: NFL categorization of the penalties that occurred on the play. A standard penalty code followed by a d means the penalty was on the defense. Multiple penalties on a play are separated by a ; (text)
- penaltyJerseyNumber: Jersey number and team code of the player committing each penalty. Multiple penalties on a play are separated by a ; (text)
- penaltyYards: yards gained by possessionTeam by penalty (numeric)
- preSnapHomeScore: Home score prior to the play (numeric)
- preSnapVisitorScore: Visiting team score prior to the play (numeric)
- passResult: Scrimmage outcome of the play if specialTeamsPlayResult is "Non-Special Teams Result" (C: Complete pass, I: Incomplete pass, S: Quarterback sack, IN: Intercepted pass, R: Scramble, ' ': Designed Rush, text)
- kickLength: Kick length in air of kickoff, field goal or punt (numeric)
- kickReturnYardage: Yards gained by return team if there was a return on a kickoff or punt (numeric)
- playResult: Net yards gained by the kicking team, including penalty yardage (numeric)
- absoluteYardlineNumber: Location of ball downfield in tracking data coordinates (numeric)

2.3.2 Tracking Data

Within this dataset is player tracking data from each season between 2018 and 2020. Each season's set of data includes 18 variables with the key variable being nflId. However, the number of observations varies between seasons. The variables are as follows [26]:

- Time: Time stamp of play (time, yyyy-mm-dd, hh:mm:ss)
- x: Player position along the long axis of the field, 0 - 120 yards. (numeric)
- y: Player position along the short axis of the field, 0 - 53.3 yards. (numeric)
- s: Speed in $\frac{yards}{second}$ (numeric)
- a: Acceleration in $\frac{yards}{second^2}$ (numeric)
- dis: Distance traveled from prior time point, in yards (numeric)
- o: Player orientation (deg), 0 - 360 degrees (numeric)
- dir: Angle of player motion (deg), 0 - 360 degrees (numeric)
- event: Tagged play details, including moment of ball snap, pass release, pass catch, tackle, etc (text)
- nflId: Player identification number, unique across players (numeric)
- displayName: Player name (text)
- jerseyNumber: Jersey number of player (numeric)
- position: Player position group (text)
- team: Team (away or home) of corresponding player (text)
- frameId: Frame identifier for each play, starting at 1 (numeric)
- gameId: Game identifier, unique (numeric)
- playId: Play identifier, not unique across games (numeric)
- playDirection: Direction that the offense is moving (left or right)

2.4 Data Cleaning

Data Cleansing is a crucial step in ensuring that reliable and accurate data is provided to a machine-learning model. This activity is the process of detecting and removing inaccurate, incomplete, or irrelevant data, as well as correcting errors and inconsistencies. These errors and inconsistencies can be formatting issues, missing values, duplicates, and outliers, all of which can affect the accuracy and quality of the data and lead to less reliable decision-making [27].
To detect the data to be removed, a thorough and detailed data analysis is required, where manual inspection of the data is used to gain better insight into the properties of the data [28]. In this thesis, only the play data were analyzed and cleaned as the tracking data depended on this previous dataset. This resulted in a reduction in dimensionality and a dataset with 4361 observations and 13 variables instead

of 19,979 and 25 observations and columns, respectively. The process was as follows.

As mentioned in section 1.1 and 2.2, the special teams unit is responsible for kick-offs, field, goals, extra points, and punts. However, the purpose of this thesis is to find an optimal path for a punt or kick-off returner to maximize their teams' field position, this for the Returner. For the Pursuer, to try to avid the returner to geain a significant amount of field advantage.

Therefore, the first process focused only on the data centered around returned punts and kick-offs, thus omitting plays that have different outcomes such as fumbles, kicks out of bounds, touchbacks, and fair catches. In addition, apart from general variables such as game/play identifiers and descriptions, only a certain number of variables are relevant for the study as they are considered to influence decision making. Only the following variables are taken into account: textcolortealquarter, down, possessionTeam, specialTeamsPlayType, gameClock, penaltyCodes, kickLength, kickReturnYardage, playResult, and absoluteYardlineNumber.

As can be seen, one of the variables kept is penaltyCodes. A penalty in the NFL is defined as a rule violation committed by either a player or a member of the coaching staff before, during, or after a play that can result in the loss of either yardage or a down. Having said this, once a penalty is given, the play in question is nullified; therefore, these plays are filtered from the dataset to ensure that the data represent the true outcome of the play. Another important aspect to keep in mind is onside kicks since, although there are occasional instances where these kicks are returned, the vast majority are fair caught, meaning they are of no use for the research and were promptly removed.

Visually, after filtering the dataset to include only punts and kicks, the resulting bar graph in figures 2.7 and 2.8 provide a clear representation of the change in the quantity of observations. The bars depict the frequency of each event, with distinct colors indicating the different seasons. This visualization allows for an easy comparison of event frequencies across the specified seasons.

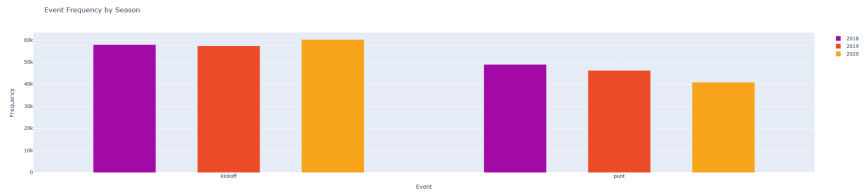

FIGURE 2.7: Quantity of Punts and Kickoffs before cleaning

By performing this process, the dataset can provide insightful information to a model that can have greater efficiency and generalization.

2.5 Visualization

Once the data was processed and prepared for analysis, the next crucial step in the research was to simulate different plays for each agent. This involved several key components, including the creation of a virtual field, implementation of specialized

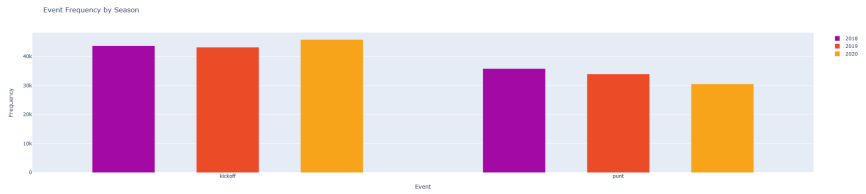

FIGURE 2.8: Quantity of Punts and Kickoffs after cleaning

functions to extract pertinent information from the dataset, and ability to animate player movement over the field.

Creating a virtual field served as the foundation for the play simulations. The field needs to accurately represent the dimensions and layout of a football field, complete with end zones, yard markers, and other essential markings.
Specific functions were implemented to effectively analyze the data and extract relevant information for the simulations. These functions were designed to access the processed dataset and retrieve the necessary data points for each play. This information typically includes the initial positions of the players, the direction of play and the length of the play. All being information that influences player movement and decision-making.
Once the relevant data was extracted, the next step was to animate the player movements over the virtual field. This involved utilizing the collected data points to accurately represent the trajectory, speed, and acceleration of each player during play.

By combining all these elements, a clear visual representation of the data can be utilized for each agent. These simulations allows for a detailed examination of agents' behaviors, decision-making, and performance in various game scenarios.

2.5.1 Field and Play Animations

A regulation NFL field has a length of 120 yards and a width of 53.3 yards. A field consists of two 10 yard end zones, each at opposite sides, hash marks that are used to determine the ball placement during plays and yard lines that are marked every 5 yards and numbered every 10 yards. Using this information, a scaled replica of the field was created, as shown in figure 2.9.

To simulate play from the tracking data, the following functions were created:

- total_frames: Obtains the total number of frames of a specific play from the moment a kick is fielded to the moment the play ends.
- simulation_returner: Plots players and football positions per frame on the field and primarily focuses on the returner, meaning it identifies him as well as traces his movement throughout the duration of the simulation
- simulation_pursuer: Plots players positions per frame on the field and primarily focuses on a specific pursuer by identifying him and tracing his movement.

Once these functions are defined using Python's *FuncAnimation*, it is possible to visualize the play. As previously stated, there are two agents, meaning that there are

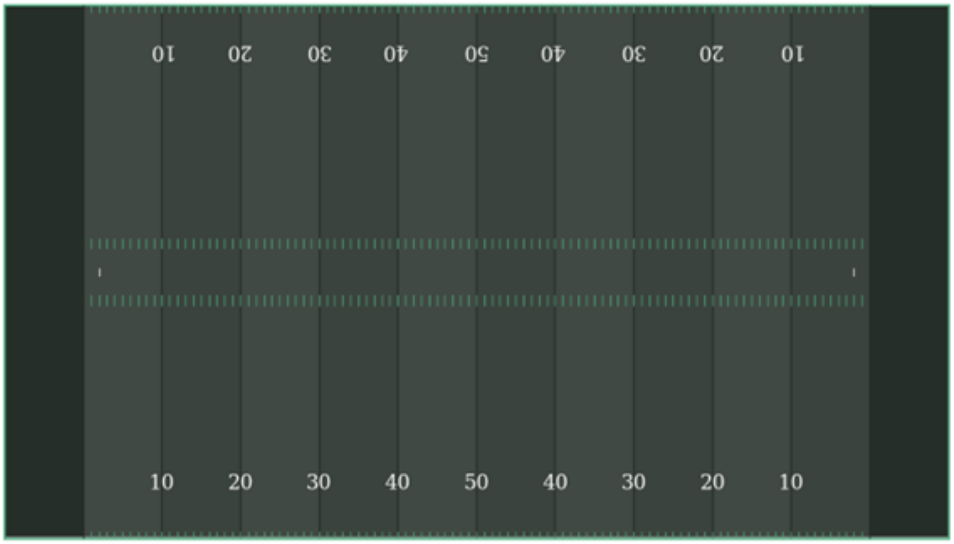

FIGURE 2.9: Scaled NFL field

two separate simulations. For each simulation, the teams are identified with the colors orange and red, the football with the color yellow, while the agent is identified with the color cyan, and a trail of the same color can be seen, depicting the trajectory of the agent within the environment.

The primary focus of this thesis will be on five different environments chosen arbitrarily, each of which depicts a different play scenario. The upcoming information illustrates the information of each of these environments separately.

Figures 2.10, 2.11 display the initial frame corresponding to the following data:

- gameId: 2018111104
- playId: 1873
- season: 2018

Figures 2.12, 2.13 display the initial frame corresponding to the following data:

- gameId: 2020120200
- playId: 2011
- season: 2020

Figures 2.14, 2.15 display the initial frame corresponding to the following data:

- gameId: 2019122910
- playId: 3160
- season: 2019

Figures 2.16, 2.17 display the initial frame corresponding to the following data:

- gameId: 2018093004

- playId: 2415
- season: 2018

Figures 2.18, 2.19 display the initial frame corresponding to the following data:

- gameId: 2018092300
- playId: 11435
- season: 2018

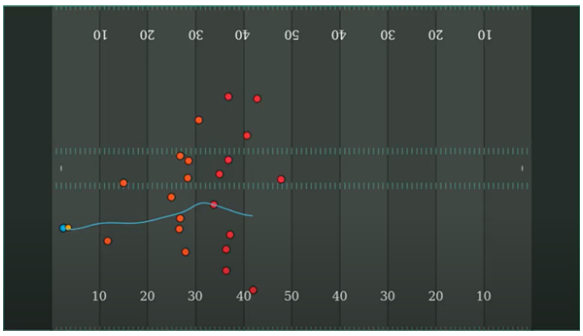

FIGURE 2.10: Environment 1: Returner agent simulation

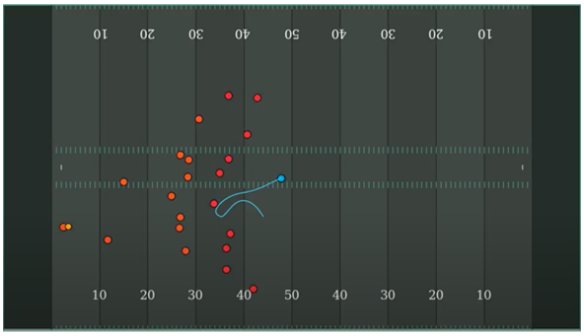

FIGURE 2.11: Environment 1: Pursuer agent simulation

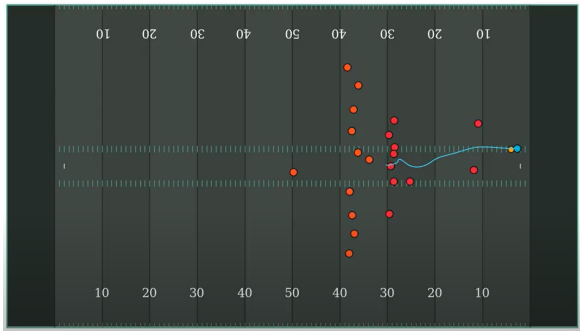

FIGURE 2.12: Environment 2: Returner agent simulation

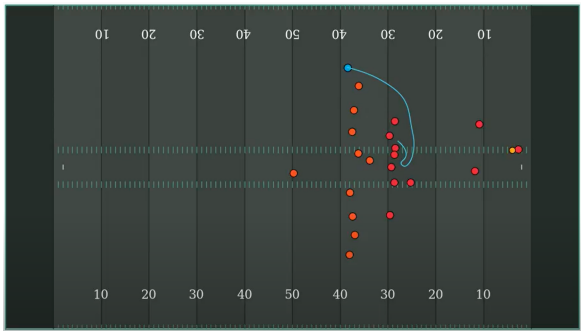

FIGURE 2.13: Environment 2: Pursuer agent simulation

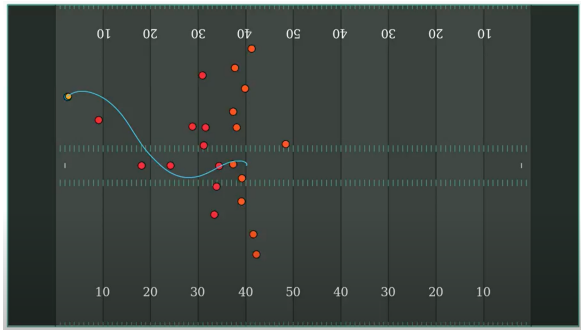

FIGURE 2.14: Environment 3: Returner agent simulation

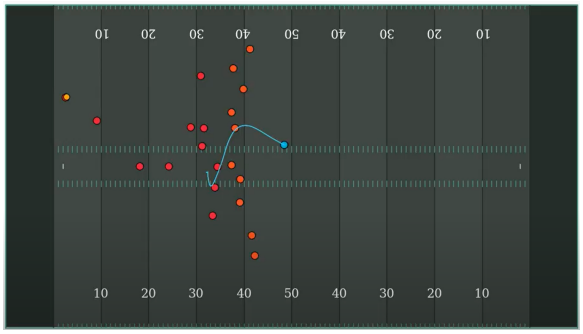

FIGURE 2.15: Environment 3: Pursuer agent simulation

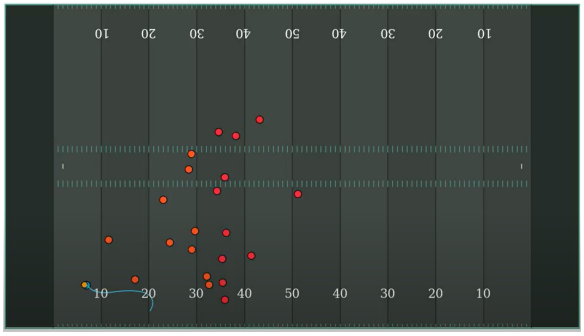

FIGURE 2.16: Environment 4: Returner agent simulation

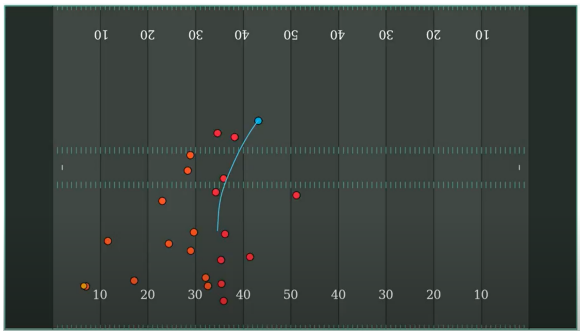

FIGURE 2.17: Environment 4: Pursuer agent simulation

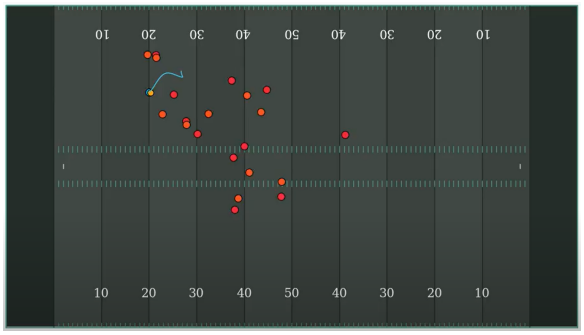

FIGURE 2.18: Environment 5: Returner agent simulation

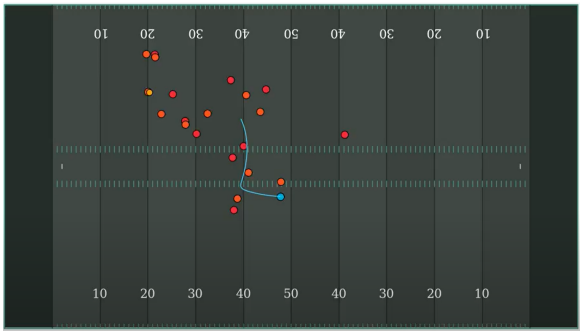

FIGURE 2.19: Environment 5: Pursuer agent simulation

As can be seen, each image presents the different agents and their corresponding movements. In later sections, these agents are discussed in greater detail.

The first three pairs of figures show a typical punt return in which the agent is able to gain a significant number of yards and improve their team's field position. By contrast, the last two figures show plays in which the opposition has the upper hand and is able to tackle the returner or force them off the field.

Chapter 3

Dynamic Learning Environment

In this chapter, the focus is on the concept of a dynamic learning environment and its significance in the realm of sports and reinforcement learning. The integration of technology and reinforcement learning in sports training opens up new possibilities for personalized and adaptive coaching. Athletes can engage in interactive simulations that mimic game situations, allowing them to develop decision-making skills, strategic thinking, and situational awareness.

3.1 Dynamic Environments

In recent years, there has been a growing interest in addressing optimal path-finding problems in dynamic environments. Dynamic environments are characterized by the fact that the conditions and components of the environment can change over time [29]. Unlike static environments, which remain constant throughout a task, dynamic environments introduce additional challenges as obstacles and goals can appear or move unpredictably. Traditional path-finding algorithms are limited in their ability to adapt to such changes, making them less effective in these scenarios. Therefore, a good approach to this problem is to apply reinforcement learning, as it offers a promising solution to path-finding in dynamic environments and it enables agents to learn from experience and adapt in real-time to changing conditions over time.

In this thesis, the use of RL for path-finding in the dynamic environment is investigated. The specific environment considered is an American football field, where players from both teams serve as moving obstacles and there are two agents. Returners are responsible for catching the ball and running it back for a touchdown while pursuers are responsible for tackling the returner and stopping them from scoring. Its worth mentioning that one of these agents focus on reaching a dynamic goal while the other is concerned with reaching a static goal in the environment.

3.2 Custom Reinforcement Learning Framework

Given the context presented in this thesis, an MDP is an appropriate framework for describing the reinforcement learning problem because it satisfies the five key characteristics presented in this section 2.1.3.

First, the states $\mathcal{S}$ involved in the problem have a finite set of possibilities, as they are defined by the number of frames in the simulation. Although the set of possible states may vary depending on the length of the play, each is composed

of the positions of all 22 players in the field. In addition, the agent is restricted to choosing from a limited set of eight actions $\mathcal{A}$ in a given state that allows him to move one yard in a specific direction. The eight actions are as follows:

1. Move Forward
2. Move Backwards
3. Move Left
4. Move Right
5. Move Forward Right
6. Move Forward Left
7. Move Backwards Right
8. Move Backwards Left

The transition probability $\mathcal{P}$, which specifies the likelihood of moving from one state to another given action, is influenced by the movement of opposing players. Furthermore, a well-defined reward function $\mathcal{R}$ assigns a numerical value to state-transition pairs. As each agent has a different goal, their respective reward functions differ and are explained in the following section 3.4.

Finally, the discount factor γ is implemented, and the Markov property holds because the future state and reward depend solely on the current state and action [30].

3.3 Custom Environment Class

Two distinct customized custom environments for each agent were developed using Python: CustomEnvironment_Returner and CustomEnvironment_Pursuer. Both inherit information from the gym.env class provided by the OpenAI Gym library. This class serves as a foundation for the creation of environments for reinforcement learning as it offers a standardized interface, basic structure, and functionality for simulating game environments.

The custom environments in this thesis consist of four key aspects that form their core functionality.

- The *__init__* method initializes the environment by setting up the initial positions of the agent, teammates, opposition players, and variables, such as play length, total reward, frame, yard steps, and nearest players.
- The *step* method considers an action as input and updates the environment states based on the action taken by the agent. Furthermore, it also calculates the reward and checks if the episode has reached a stopping criteria, such as time expiration, agent going out of bounds or the goal is met
- The *add_position* method appends the current position of the agent and players to their respective lists to collect the positions throughout the episode, which is used by the animation method to generate a visual representation of the game play.

- The *animation* method can use the positions recorded during an episode to create a visual representation of the game play.

A crucial aspect worth highlighting is the step method, which plays a pivotal role within the custom environment because since each class is agent-specific, this method determines the reward function and state transitions of the environment based on actions.

3.4 Agents

This thesis focuses on the accomplishment of goals by two key *agents*: **the Returner** and **the Pursuer** in the dynamic environment of an NFL pitch. The Returner's objective is to navigate the field, evade opponents, and advance toward the end zone to maximize field position. Conversely, the Pursuer aims to track down and prevent the Returner from reaching their goal. Techniques from reinforcement learning are employed to train the agents in a dynamic learning environment, enabling them to learn from experiences and adapt their strategies.

3.4.1 Returner

Recall that in football, a returner specializes in returning kicks with the primary objective of maximizing field position and, if possible, scoring a touchdown. While rare, there are certain plays in which any player from the receiving team can become the returner; therefore, in the context of this study, the returner is defined as the player who makes initial contact with the ball.

The reward function to guide the behavior of the returner agent is defined as follows:

$$r_t = \begin{cases} 1, & \text{if the agent moves closer to the goal} \\ -0.5, & \text{if agent is too close to opponents} \\ -1, & \text{if agent moves away from the goal} \\ -3, & \text{if agent went out of bounds} \\ -2, & \text{if agent was tackled} \\ 4, & \text{if agent touchdown is scored} \end{cases} \tag{3.1}$$

This function encourages the agent to make decisions that maximize the field position while avoiding penalization by performing unfavorable actions. The agent receives positive rewards for advancing in the direction that optimizes the field position but incurs penalties if the agent gets too close for opposing players, is tackled by opponents, or steps out of bounds. In the case where the agent is able to score a touchdown, the maximum reward is given.

3.4.2 Pursuer

In contrast to the returner, a pursuer, also known as a coverage player, is tasked with preventing the returner from gaining significant yardage or scoring a touchdown. Thus, in this study, any player from the kicking team is considered a pursuer.

To guide the behavior of the pursuer agent, the reward function considers the distance between the agent and the target before and after the agent's movement. If the agent moves closer, meaning that the distance decreases, then it receives a reward. On the other hand, the agent is penalized if the distance remains the same, increases, or moves out of bounds. In addition, if the agent successfully tackles the returner by being inside a catch radius that represents the distance needed to perform a tackle, the agent receives a substantial reward.

To understand this better, the reward function is explicitly stated as follows:

$$r_t = \begin{cases} 5, & \text{if new distance is smaller than old distance} \\ -2, & \text{if new distance is equal to old distance} \\ 20, & \text{if target is within catch radius} \\ -3, & \text{if agent went out of bounds} \\ -10, & \text{otherwise} \end{cases} \tag{3.2}$$

3.5 Agent-Environment Interaction: Agent oriented

For each agent, a visual representation of the agent-environment interaction process was created in order to provide a concise summary of the previously discussed key elements as seen in figures 3.1 and figure 3.2. The actions, rewards, and states that are a part of the interaction are all displayed in this thorough visualization [31].

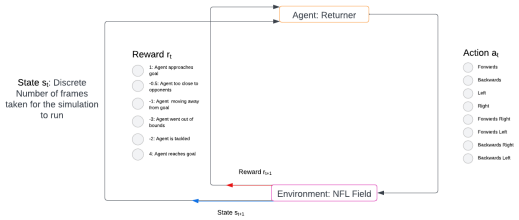

FIGURE 3.1: RL agent-environment interaction: Returner

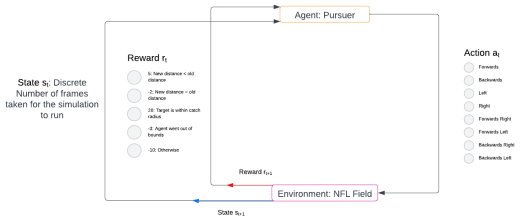

FIGURE 3.2: RL agent-environment interaction: Pursuer

Chapter 4

Reinforcement Learning Model

In this chapter, the reinforcement learning model employed in this study to address the challenges of the dynamic environment in football is presented.

The reinforcement learning model utilized in this study was based on the Proximal Policy Optimization (PPO) algorithm. The PPO combines the strengths of the advantage actor–critic (A2C) and Trust Region Policy Optimization (TRPO), providing stability and performance in training agents to navigate dynamic environments. This model architecture includes a multilayer perception (MLP) policy and value functions to guide the agent's decision-making process.

Throughout this chapter, topics such as the network architecture, hyperparameter configurations, and rationale behind their selection are presented. Also the key characteristics of the PPO algorithm are discussed, such as its ability to maintain policy stability while updating to better adapt to changing conditions.
By implementing this reinforcement learning model, the is aim to equip the agents, with the necessary tools to successfully achieve their goals in a dynamic football environment. Therefore, this chapter serves as a crucial foundation for understanding the mechanisms behind the intelligent decision-making capabilities of agents and their ability to adapt to evolving situations in the field.

4.1 Model

As mentioned above, the simulations in this study employed the Proximal Policy Optimization reinforcement learning algorithm that combines the advantages of the Advantage Actor–Critic and Trust Region Policy Optimization.

The Advantage Actor Critic method is a synchronous, model-free algorithm that combines elements of both policy based and value based methods, therefore, it can be regarded as a framework of algorithms with parameterized *Actor* and *Critic* components. The *Actor* component represents the policy function $\pi(a|s)$ which acts as the basis of the agents action. On the other hand, the *Critic* component refers to the value function that estimates the action-value function to guarantee training $q_\pi(s, a)$ [32]. Currently, the actor–critic method is usually viewed as a family of related techniques; however, A2C incorporates the term *Advantage* as an improvement to the algorithm. The Advantage Function $A^\pi(s, a)$, which is the difference between the estimated value of the state-action pair and the expected value, is proposed to upgrade the gradient in order to significantly reduce the variability of the results.

Trust Region Policy Optimization is designed to optimize the policy in a stable and efficient manner, thus allowing more precise control of the expected policy improvement through the introduction of surrogate loss [33]. The main goal of this algorithm is to find a policy that maximizes the expected cumulative reward (notation) by iteratively updating the policy that guarantees a significant improvement.

The procedure that TRPO uses is as follows: it first computes the advantages, same concept as in the previous algorithm, of different actions based on a given policy by estimating the empirical return minus the baseline. Then, a constraint is placed on the maximum change in the policy distribution, typically using a measure called the *Kullback-Leibler* (KL) divergence, which is a step size parameter that controls how much the policy is allowed to change per iteration [33].
Determining the policy update in TRPO that maximizes the expected reward while adhering to the defined KL divergence constraint is known as the optimization step. This is accomplished by utilizing numerical optimization methods such as conjugate gradient descent. To maintain stability throughout learning, TRPO enforces a constraint to ensure that the policy update does not stray too far from the current policy distribution.

Therefore, to summarize, A2C is an algorithm that simultaneously trains the policy (the agent's decision-making process) and value function (an estimate of the expected future rewards) [34]. On the other hand, TRPO aims to find an optimal policy by iteratively improving it while ensuring performance [31].

The PPO is particularly suitable for this task because it ensures that the updated policy remains close to the previous one and, by doing so, prevents destabilization. This is achieved by performing multiple policy updates while carefully constraining the extent of the update.
As this model implements policy and value functions, the network architecture consists of three hidden layers of 50, 25, and 10 nodes, respectively, see figure 4.1 but note that the model shown has been simplified in order to provide a more comprehensive overview of its structure. It is also important to note that the policy used is the **MlpPolicy**, a multi-layer preceptron.

Several hyperparameters were utilized in the model, each serving a specific purpose, Among the numerous optimization algorithms available, one particular method stands out as being highly significant and deserving of further attention. This is the Adam Optimizer, a stochastic gradient descent method based on the adaptive estimation of first- and second-order moments commonly used in deep learning [35]

The name stands for Adaptive Moment Estimation and combines the advantages of AdaGrad and RMSProp, two other optimization methods. A neural network model's parameters can be effectively updated using the Adam optimizer based on the gradients calculated during the back propagation procedure.

The fundamental principle of the Adam optimizer is to adaptively modify the learning rate for each parameter in the model while taking into account the momentum and historical gradients. For each parameter, it keeps a set of adaptive learning rates that are updated based on the magnitudes of the gradients. This enables the optimizer to modify the learning rates for various parameters automatically based on their individual traits, resulting in quicker convergence and better optimization

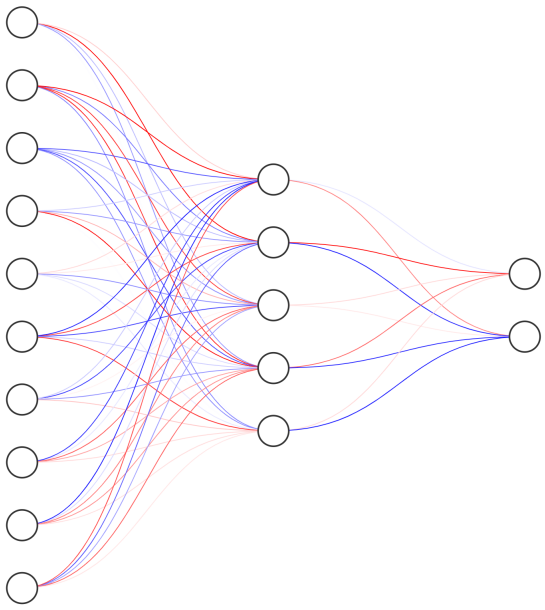

FIGURE 4.1: Scaled Model Architecture

results.
Momentum and adaptive learning rates are the two main concepts used by the Adam optimizer. By building up a running average of previous gradients, the momentum term speeds up the optimization process by smoothing out variations and assisting the optimizer in navigating through challenging and distracting optimization environments. The estimated first and second moments of the gradients, which reveal the magnitude and direction of the gradient changes, are used to calculate the adaptive learning rates. As a result, depending on the gradient conditions at any given time, the optimizer is able to dynamically modify the learning rates for each parameter.

Next is a list of the hyper parameters presented in a concise matter:

- **Adam optimizer**: This optimizer was used as it converges faster and adapts to the learning rate of each parameter
- **Rectified Linear Unit**: A computationally effective and simple non-linear activation function used in multi-layer neural networks [36], that allows learning of complex relationships within the data. See figure 4.2
- **Learning rate**: Determine the step size. A value of 0.0003 was chosen as its common value for deep learning tasks.
- **Number of steps**: 1024 steps were taken to collect experience before performing an update. Because this value is relatively large, it allows the model to gather more information before updating it.
- **Batch size**: Number of experiences used in each training update. 256 was chosen because it provided a good balance between efficiency and accuracy.

By utilizing these hyperparameters and the PPO algorithm, the model can learn and improve its decision-making capabilities over time, ultimately enabling returner and pursuer agents to achieve their respective goals in the dynamic football field.

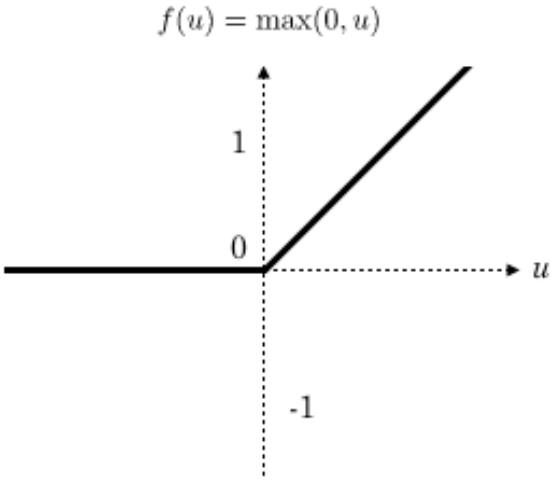

FIGURE 4.2: Rectified Linear Unit Graph [37]

Chapter 5

Simulations and Results

To provide a comprehensive understanding of the data-cleaning process and the subsequent steps of model creation, training, prediction, and evaluation, two Jupyter notebooks were developed. These notebook can be accessed through the following link: https://github.com/SantiagoAlvarezb/Thesis.git

The purpose of this section is to focus on the steps taken after data cleaning to obtain a better overview of the entire process and highlight the significance of each step in achieving the results.

5.1 Procedure

After completing the data-cleaning process and conducting the initial simulations of the environment without any modifications, the next step involves the implementation of two distinct sets of functions. The incorporation of this specific set of functions enhanced the efficiency of the model but also significantly reduced the computational time required for running the simulations. This was done by extracting useful information beforehand, making it more manageable for the computer to optimize resource usage.

initial_positions_returner and initial_positions_pursuer are the first set of functions. Given a specific season, game, and play, can retrieve and process tracking data, specifically the initial positions of the returner, home team, and away team, along with other information, such as time and direction. Based on the input argument, function first determines the season. Subsequently, by looking for specific occurrences in the tracking data, the first frame can be found, and to ensure that each frame contains the right number of entries for the 10 players in the field, the function uses data-manipulation techniques. Next, the function identifies the agent as either the player that is near the football's coordinates, if the agent is a returner, or any other player on the kicking team, as they are the pursuers. Later, the duration of the play is calculated by finding the difference between the maximum and the first frame as well as obtaining the direction of play, as this determines which direction the agent must take to reach their goal. Finally, pertinent tracking information for the home team, away team, and specific play is returned.

The second set of functions are data_frames_returner and data_frames_pursuer .They are responsible for retrieving and processing tracking data given a specific frame. These functions have the same base functionality as the previous set, because they first check the season and then filter the clean tracking data for each player. However, because this function is called upon within each step that the agent takes, the tracking data returned for all players is based on the given frame.

It is also worth mentioning that, for both sets, error handling is included in cases where the information given is not found or when the agent's position cannot be found.

To create the custom environment and ultimately train the model, a simulation was run where the agent had random movement and the other players continued their usual motion. The agent exhibited random motion, as no information about the environment was given; therefore, he had no perception of a reward or penalty. Figures 5.1, 5.3, 5.5, 5.7 and 5.9 for the returner and figures 5.2, 5.4, 5.6, 5.8 and 5.10 for the pursuer illustrate this phenomenon. Compared to earlier photos, it is clear that the agent follows a random trajectory, as indicated by the blue line.

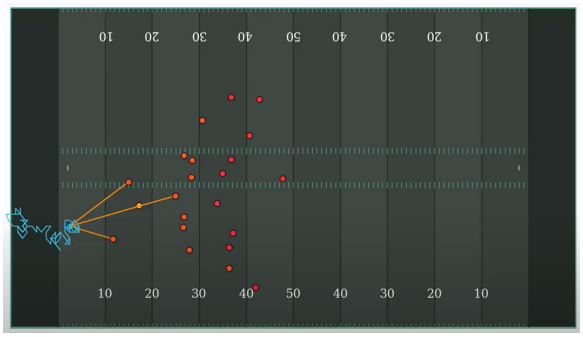

FIGURE 5.1: Environment 1: Returner random simulation

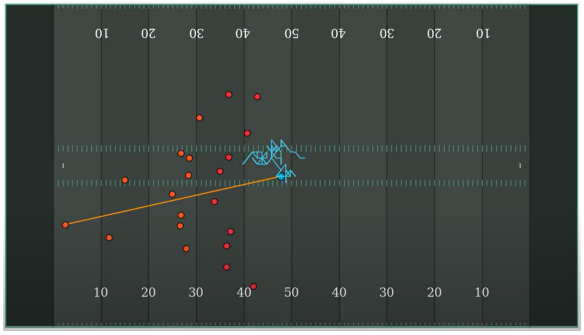

FIGURE 5.2: Environment 1: Pursuer random simulation

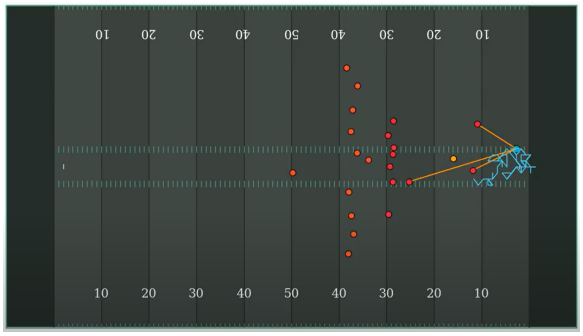

FIGURE 5.3: Environment 2: Returner random simulation

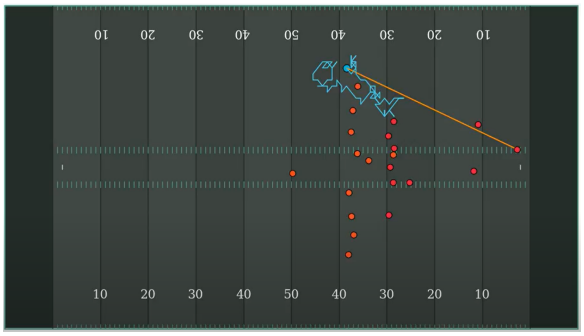

FIGURE 5.4: Environment 2: Pursuer random simulation

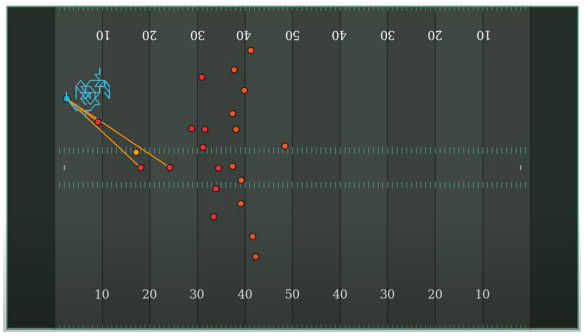

FIGURE 5.5: Environment 3: Returner random simulation

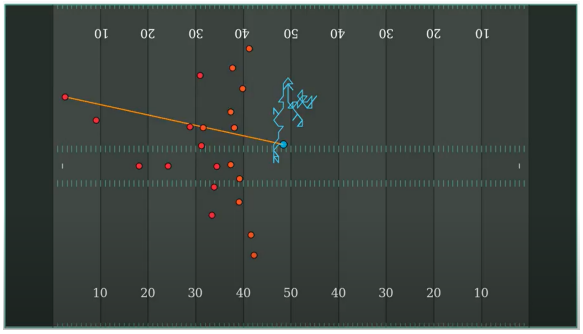

FIGURE 5.6: Environment 3: Pursuer random simulation

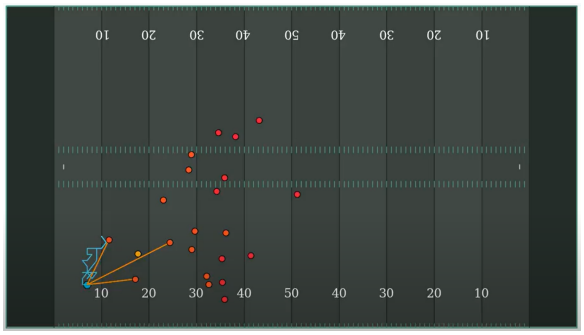

FIGURE 5.7: Environment 4: Returner random simulation

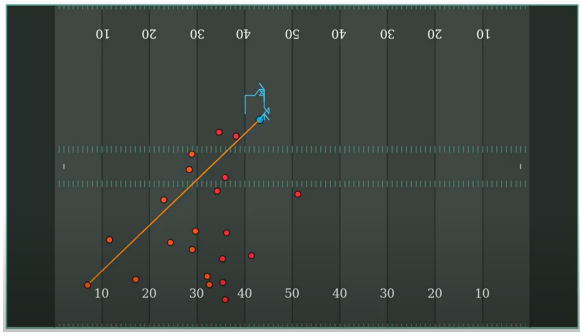

FIGURE 5.8: Environment 4: Pursuer random simulation

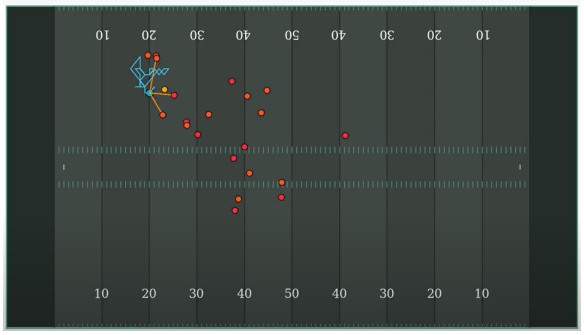

FIGURE 5.9: Environment 5: Returner random simulation

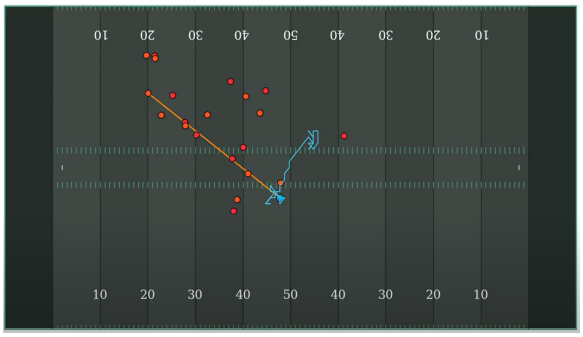

FIGURE 5.10: Environment 5: Pursuer random simulation

5.1.1 Training

The model underwent an extensive training process consisting of 5000 epochs to enhance its prediction skills and improve its ability to anticipate outcomes. Through this prolonged training, the model learned from patterns and makes more accurate predictions. Each epoch involved multiple iterations of updating the model's parameters based on the observed rewards and optimizing its performance.
Upon completing the training process, the trained model was saved to preserve its learned knowledge and avoid the need for retraining in subsequent uses. This saved model served as a valuable asset, encapsulating the learned insights and predictive abilities that the model had developed through the training phase. Next, the trained model was applied to the same environment. By incorporating the model's predictions into the agent's decision-making process, it was anticipated that the agent's performance within the environment would significantly improve. The model's insights and predictions acted as an additional resource for the agent, augmenting its decision-making capabilities and enabling it to make more informed choices.

By repeatedly using the trained model in the dynamic environment, the aim was to continually enhance the agent's performance over time. This iterative process ensured that the agent's behavior was continuously refined and improved, leveraging the knowledge and insights gained from the trained model. As a result, the agent could adapt and make better decisions based on the model's predictions, ultimately leading to more successful outcomes in the dynamic football environment.

5.1.2 Flowchart y Pseudocode

Figure 5.11 is a flowchart was made to depict the key steps in the procedure so that it could be understood more clearly. The order of the actions and choices made during the process are graphically represented.

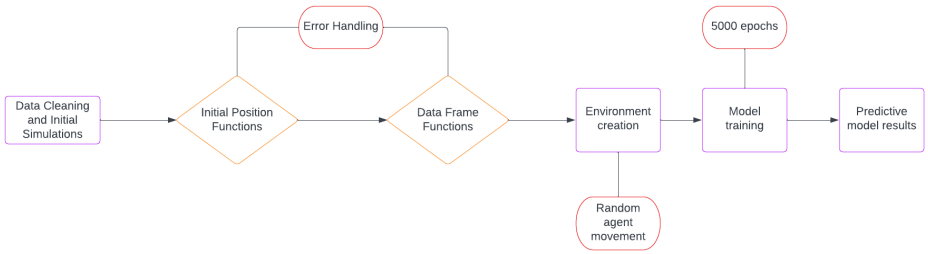

FIGURE 5.11: Flowchart

The following pseudocodes were also created to help with training and visualization process comprehension. The algorithms and steps used to train the reinforcement learning models and visualize their performance are outlined in these pseudocodes step-by-step.

Algorithm 1 Reinforcement Learning Training

```
procedure TRAINING PSEUDOCODE(inputs)
    Initialize model parameters
    for epoch ← 1 to 5000 do
        Sample a batch of experiences
        Update model using the sampled batch
        Store the updated model parameters
    end for
    Return result
end procedure
```

Algorithm 2 Reinforcement Learning Visualization

```
procedure VISUALIZEMODEL(model, environment)
    Initialize model parameters
    for episode ← 1 to N do
        Reset the environment
        Observe the initial state
        while not terminal state do
            Select an action using the model's insights
            Execute the action in the environment
            Observe the next state and reward
            Update the agent's knowledge
        end while
    end for
end procedure
```

5.2 Results

The predictions yielded satisfactory results, showing notable improvements both visually and in terms of the score achieved in each simulation. The incorporation of predictions into the agent's decision making capabilities proved to enhance its ability to learn and make optimal decisions in order to reach the goal.

The following images are displayed to help gain a better understanding of these findings. They show the initial trajectory and the trajectory of the model predicted for each agent for the five environments that were explained previously in section 2.5.1 in order to compare.

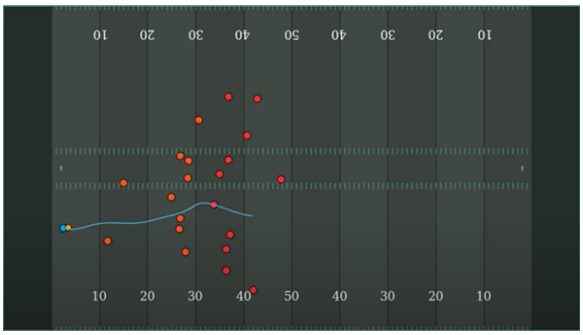

FIGURE 5.12: Environment 1: Returner simulation

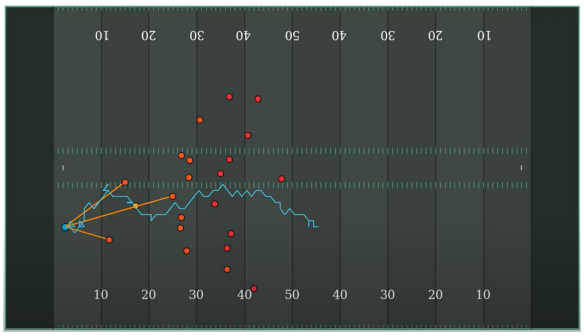

FIGURE 5.13: Environment 1: Simulation prediction

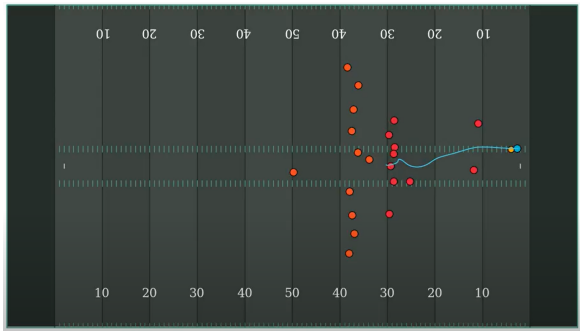

FIGURE 5.14: Environment 2: Returner simulation

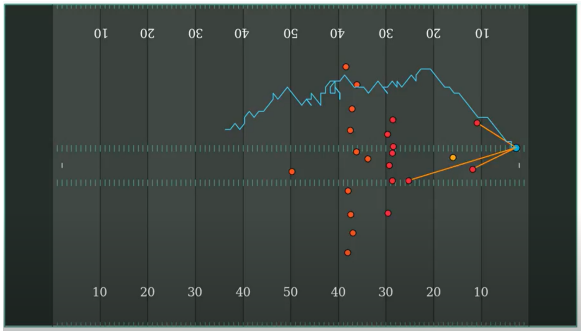

FIGURE 5.15: Environment 2: Simulation prediction

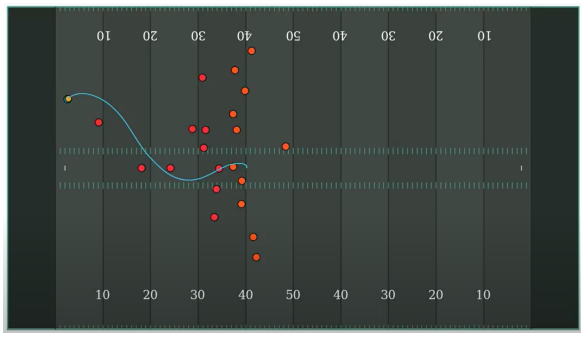

FIGURE 5.16: Environment 3: Returner simulation

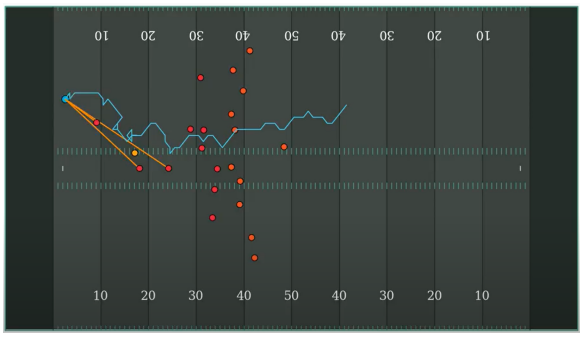

FIGURE 5.17: Environment 3: Simulation prediction

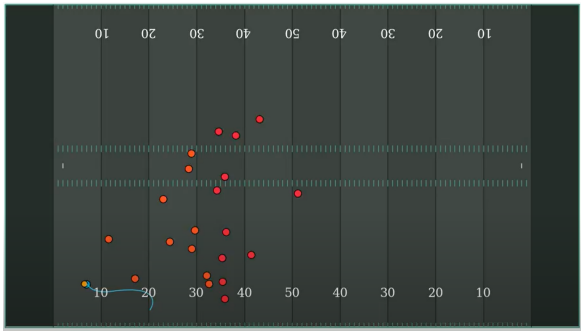

FIGURE 5.18: Environment 4: Returner simulation

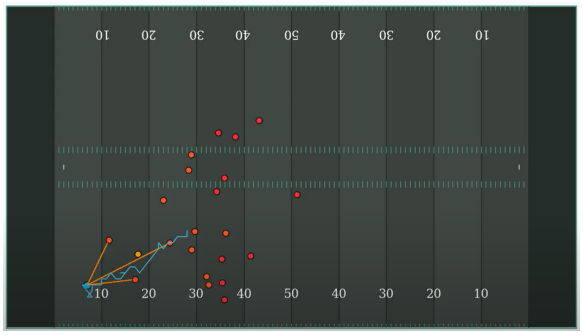

FIGURE 5.19: Environment 4: Simulation prediction

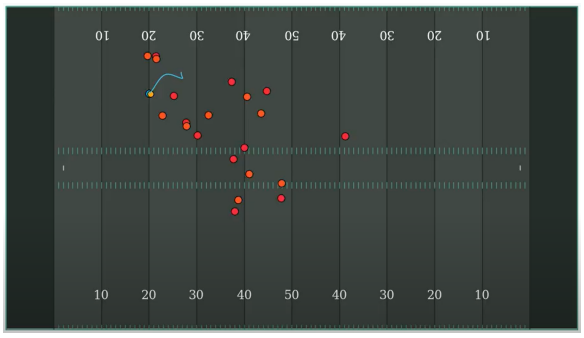

FIGURE 5.20: Environment 5: Returner simulation

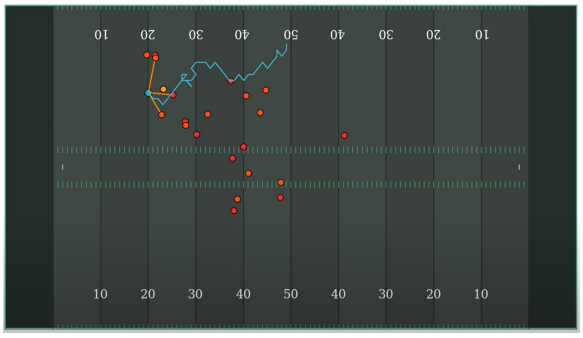

FIGURE 5.21: Environment 5: Simulation prediction

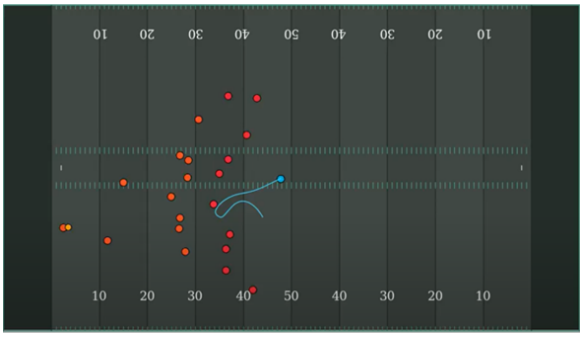

FIGURE 5.22: Environment 1: Pursuer simulation

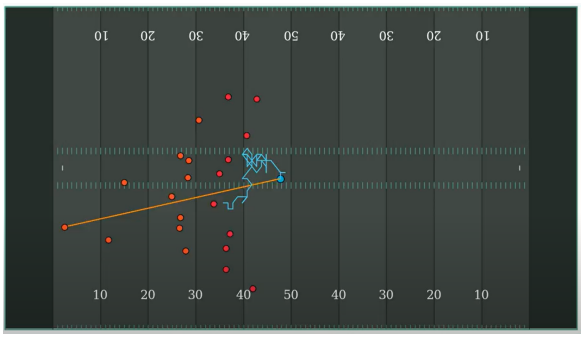

FIGURE 5.23: Environment 1: simulation prediction

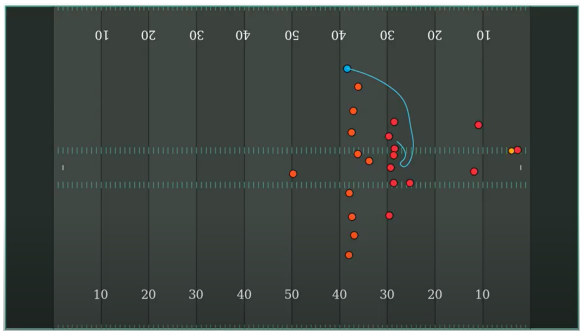

FIGURE 5.24: Environment 2: Returner simulation

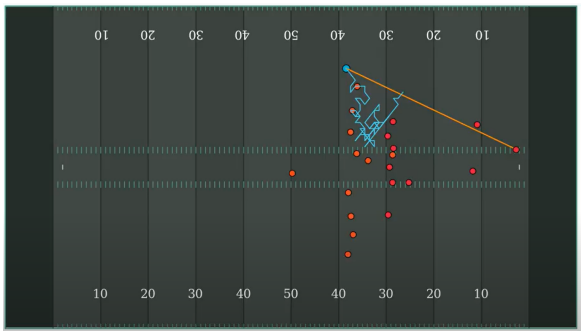

FIGURE 5.25: Environment 2: simulation prediction

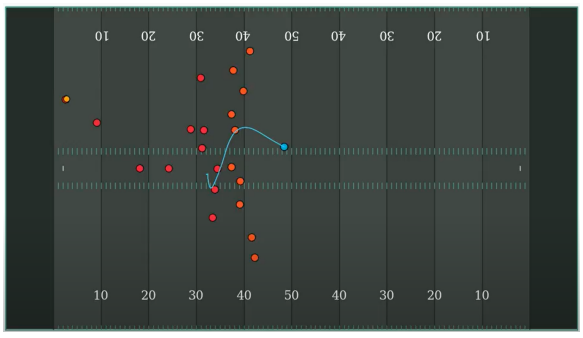

FIGURE 5.26: Environment 3: Pursuer simulation

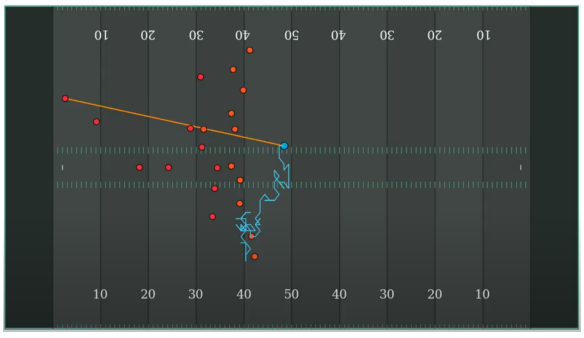

FIGURE 5.27: Environment 3: simulation prediction

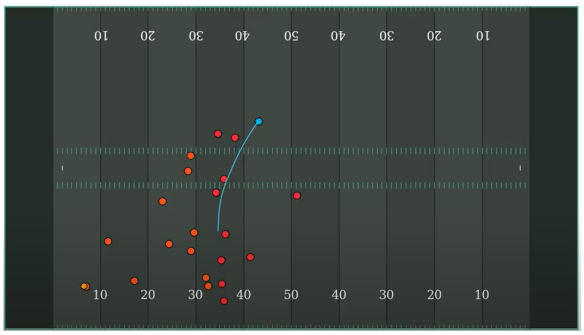

FIGURE 5.28: Environment 4: Pursuer simulation

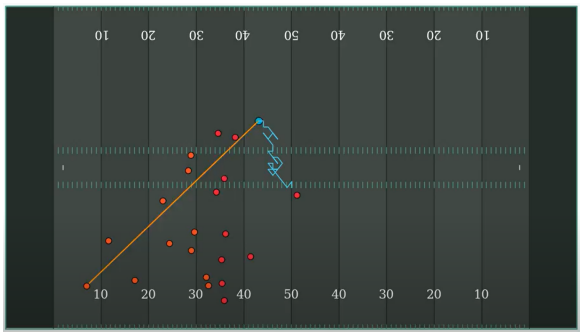

FIGURE 5.29: Environment 4: simulation prediction

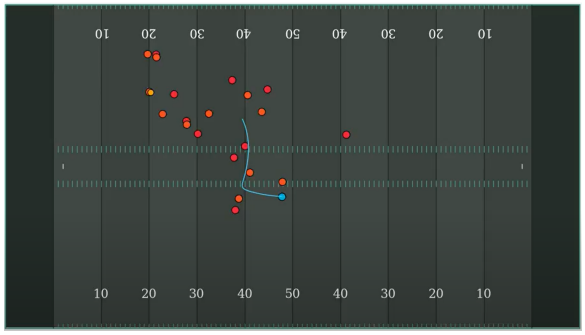

FIGURE 5.30: Environment 5: Pursuer simulation

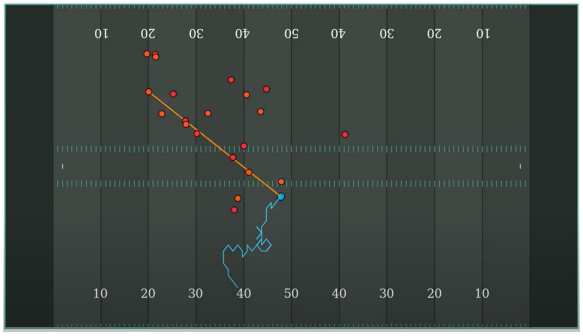

FIGURE 5.31: Environment 5: simulation prediction

5.2.1 Returner

The results revealed several interesting patterns and trends in the performance of this agent. Owing to the reward functions provided, the agent was able to demonstrate the ability to maximize the field positions of its team. Furthermore, the agent showed a pattern of successful maneuvering towards open spaces and avoiding defensive obstacles. Most importantly, the agent exhibits the ability to adapt its trajectory based on the dynamics of the game.

A comparative and visual analysis was also conducted to evaluate the agent's performance in maximizing field position, which involved comparing the agent's trajectory given by the model with the actual trajectory. Even though the agent's movement can be considered somewhat erratic, the visual representation of the movement shows patterns such as agile cuts and strategic running to fulfill the goal.

Across all five trials, the agent demonstrated the ability to perceive the opponent's movements and adjust accordingly to avoid tackling. Notably, in figure 5.19, the agent strategically chose to move in the opposite direction of the opponents; even though the other players follow a predetermined route, the decision reflects the agents' awareness of the need to avoid the opposing players. Interestingly, in

other instances, as can be seen in the figures 5.13, 5.15, 5.17 and 5.21, the agent followed a trajectory similar to that of the original path with a minimal amount of deviation. Thus, highlighting the adaptive behavior of the agent in different situations.

In addition, the results for this agent reveal instances where the agent's decision making is optimal in comparison to the original agent's decision making. Nonetheless, in other cases, it provides valuable insights that can be addressed regarding an agent's movement with factors such as direction. a coach can address the return movement. The adaptability of the agent and potential areas for further development are highlighted by the specific movement changes shown in the following figures.
Therefore, by securing a better field position, the returner agent consistently outperformed the baseline.

5.2.2 Pursuer

By contrast, this agent exhibited varying levels of success in different scenarios. While there were cases in which the agent performed well, in most instances, it fell short of achieving the desired outcome. Upon conducting a comparative and visual analysis of the agent's movement, it becomes evident that the agent possesses the ability to recognize the target's movement and make adjustments to the position itself in the general vicinity where it predicts the target will be at. This observation is supported by the analysis of the figures 5.23 and 5.25, where the agent opted to move in proximity to the inferred target trajectory, anticipating its position. However, in most cases, the agent's overall pursuit of the target proved to be challenging, as presented in figures (state them), as there was an erratic movement lacking a clear sense of direction towards achieving the goal.

In the worst cases, depicted in figures 5.27, 5.29 and 5.31 the agent exhibited movements that deviated from the intended goal. These movements can be attributed to several factors, including time constraints imposed by the simulation itself. Given the limited duration of the simulation, the agent may not have had enough time to learn effective strategies to reach the objective. Nevertheless, the primary factor influencing the agents' sub-optimal performance is most likely the design of the reward function 3.2 because the reward function only considers the distance before and after the agent's movement with the expectation that the agent would implicitly learn to reduce the distance to the target by pursuing it. However, as evidenced by the results, it may have been more advantageous to the study if an explicit reward was given so that the agent directly moved towards the target.

5.3 Model Evaluation

After examining the results, it is crucial to evaluate the models in order to extract valuable insights from them. This evaluation process allows us to identify areas for improvement and gain a deeper understanding of the models' performance. Importannt to keep in mind the following information to understand the graphs:

- **PPO_1**: Represents environment 1 and is identified by the color grey.
- **PPO_2**: Represents environment 2 and is identified by the color blue.
- **PPO_3**: Represents environment 3 and is identified by the color pink.

- **PPO_4**: Represents environment 4 and is identified by the color yellow.
- **PPO_5**: Represents environment 5 and is identified by the color purple.

For both figure 5.32, which corresponds to the returner agent, and figure 5.34 the image illustrates the performance in relation to the average length of the episode. For figures 5.33 and 5.35, these correspond to the mean of rewards earned in each episode by the agent.
In general, an increase in average rewards is observed, indicating positive progress in the agent's learning. However, there may be other relevant aspects that can be analyzed in relation to these graphs.

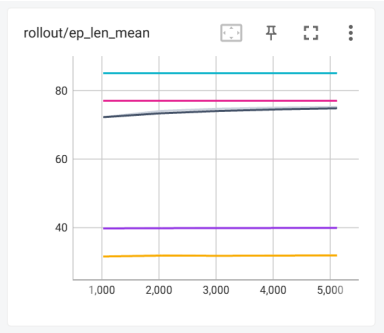

FIGURE 5.32: Returner episode length mean

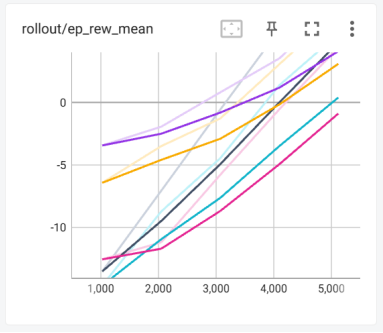

FIGURE 5.33: Returner episode reward mean

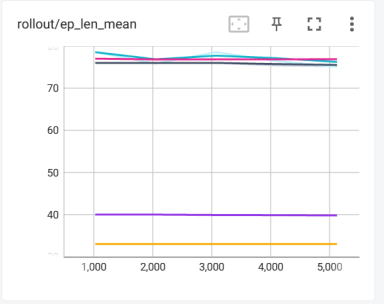

FIGURE 5.34: Pursuer episode length mean

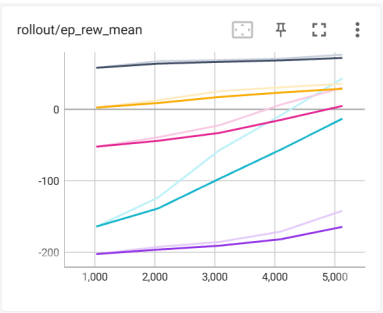

FIGURE 5.35: Pursuer episode reward mean

With regards the the Returner's models, in the *ep_len_mean* metric, a constant trend is observed in the different PPO models. Since environments 4 and 5 were the ones with least duration, the values of mean length of training of their respective models, **PPO_4** and **PPO_5** are lower, hovering between values 20 and 40. Meanwhile, the values for the other models are between 70 and 90. Furthermore, it stands out that **PPO_1** shows a steeper rise compared to the others, and the PPO with the maximum value is **PPO_2**.

On the other hand, in the *ep_raw_mean* metric, the values are not so consistent among the different PPO models. Here, the values vary from -15. Both **PPO_1** and

PPO_2 show linear increases, while the others show variations. Nevertheless, this is an encouraging sign as it demonstrates that the agent was able to learn in all of the models. This is evident from the consistent increase in the mean reward value over epochs, which aligns with the displayed results.

With the Pursuers' models, In relation to the *ep_len_mean* metric, similar behaviors to the other graphs can be observed. However, the difference lies in **PPO_1**, **PPO_2** and **PPO_3**, where their values are closer to each other compared to the other graphs. In addition, it should be noted that **PPO_2** shows variability and a slightly downward trend, while the other models remain constant. This downward trend is quite interesting as it can be inferred that the agent was reaching a stopping criteria quickly and did note solely depend on the duration of the simulation itself.

On the other hand, in the *ep_rew_mean* metric, interesting aspects are observed. The range of values now starts at -200, and no models with linear growth are found. In addition, none of the models present a noticeable steep change, which means that even though the agent was learning, it wasn't learning sufficient information to accomplish its goal. **PPO_2** shows the highest variability compared to the other models while there are models with positive values, specifically **PPO_1**, **PPO_3** and **PPO_4**, yet those are the ones that show sub-optimal results. Once again, the observations seen here with the model evaluations are consistent with those found in the simulations.

The following table 5.1 summarizes the findings. In which the first and final mean episode length and rewards for each model are stated to show the improvement overtime of each model during training.

Mean Length and Rewards per Model				
Model and Agent	First *ep_len_mean*	First *ep_rew_mean*	Final *ep_len_mean*	Final *ep_rew_mean*
PPO_1: Returner	7.21	-13.5	7.21	11.07
PPO_2: Returner	85	-14.64	85	5.458
PPO_3: Returner	77	-12.54	77	4.394
PPO_4: Returner	31.63	-6.422	32.02	7.32
PPO_5: Returner	39.8	-3.44	40	7.895
PPO_1: Pusuer	76	58.08	75.24	76.4
PPO_2: Pusuer	78.54	-164.02	75.07	43.22
PPO_3: Pusuer	66	-52.31	76.39	30.15
PPO_4: Pusuer	33	2.419	33	35.7
PPO_5: Pusuer	40	-202.6	39.82	-142

TABLE 5.1: Results table for final mean length and reward

5.4 Considerations and Challenges

Several significant challenges were encountered during the implementation of the model. One challenge is that the model is computationally expensive and requires an extensive amount of time to execute. The average run time was 3–4 h per run. This prolonged run time poses limitations that further hinder the efficiency of the

model.

In addition, the most noteworthy limitation arises from the restrictions imposed by the API used in the model, most specifically, the inability to train the model with various observations [38]. One possible reason why multiple environments were not run simultaneously is related to the computational resources and parallelization capabilities of the software. The ability to run in multiple environments depends on the availability of multiple cores (processing units) that can handle parallel computation; however, the machines used throughout the duration of the thesis simulations were only equipped with one core. Thus, the model could not be trained with multiple observations, meaning that the model's ability to learn was hindered as it only took into account one simulation. The model was able to learn relatively with the given simulation, while the model could still be employed for other similar simulations, and lacked the advantage of generalizing across various plays in order to predict movement to agents in different scenarios.

The discrete rather than continuous character of an agent's movement presents another noticeable problem and is a crucial factor to consider. It is essential to keep in mind that the output of the model is used to provide coaches with guidance. A smooth trajectory might not be visible because of the unpredictable behavior of the agent throughout the movement. Coaches' ability to extract important and insightful information from the model may be hampered by the agent's lack of coherence in their movement.

The need to utilize computational resources and overcome API limitations to increase the model's efficiency and enable more comprehensive learning is highlighted by these considerations and challenges.

Chapter 6

Discussion

In this section, a detailed analysis of the data-cleaning process and the subsequent steps involved is delved into. The objective is to provide a comprehensive understanding of the entire process and to emphasize the significance of each step in achieving the desired outcomes.

To facilitate comprehension of the process, two Jupyter notebooks were developed and these serve as valuable resources for anyone interested in gaining insights into methodology and technical implementation.
The procedure section outlines the specific steps taken after completing the data cleaning process and conducting the initial simulations without any modifications. The subsequent step involves the implementation of two distinct sets of functions that play a pivotal role in enhancing the efficiency of the model and reducing the computational time. By extracting relevant information beforehand, these functions optimize resource usage, making it more manageable for the computer to process data effectively.

Due to the creation of a custom environment and models, a simulation is run in which the agent exhibits random movement, while the other players continue their usual motion. As the agent has no prior information about the environment, it follows a random trajectory devoid of any perception of rewards or penalties.
Once the training phase begins, the model undergoes 5000 epochs to enhance its prediction skills. Extensive training empowers the model to analyze the data thoroughly and effectively anticipate outcomes.

Once the model was trained, it is then applied to the same environment where it generates movement predictions for the agent. **However, its very important to emphasise that the train model can be applied to any other environment, yet since it was only trained with a specific play, it may no yield the best results**. By incorporating these predictions into the decision-making process, an agent's performance in the environment is expected to improve significantly.

With this, the results obtained from the simulations can be considered mixed since for one of the agents achieved the expected outcomes, while the performance of the other agent highlighted significant room for improvement. Thus the incorporation of predictions into the agent's decision-making process proved to be instrumental in enhancing its learning ability and enabling optimal decision-making to reach the goal. Furthermore, the importance of defining the reward function must be emphasized because it is a key factor in determining an agent's capacity for learning. The reward function's structure and formulation directly affect the agent's capacity

for learning and decision-making.

In summary, the simulations and results show the effectiveness of the implemented model in improving agent performance, and thus can be considered as a potential tool that can be used by coaches to make themselves better decisions and improve their teams special teams unit. By integrating predictions into the decision-making process, both agents exhibited an improved capacity to learn and make optimal decisions, albeit with varying degrees of success. One agent demonstrated a superior performance in leveraging predictions to enhance its decision-making abilities compared to the other agent.

However, further details and metrics are necessary to provide a more comprehensive evaluation of the performance of the model in achieving the desired outcomes due to the limitations found throughout the development of the study.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The findings of this thesis demonstrate the potential of using Reinforcement Learning techniques for special team evaluations in the National Football League. The results revealed that, of the agents, the returner had successfully learned and provided valuable insights into the decision-making of players, while the other agent, the pursuer, showed ample room for improvement, specifically in the definition of a new reward function that is able to better teach the agent to accomplish their goal. By achieving the goals of this study, it can be established that reinforcement learning can effectively contribute to the decision-making process in special teams.

The obtained results indicate that with further refinement and optimization, the developed models have the potential to be valuable tools for coaches in the NFL. These models can provide meaningful insights and aid in making informed decisions to improve the performance of team units. The success of one agent in learning serves as compelling evidence of the potential benefits of tools, such as this in the league.

In conclusion, this thesis shows that applying reinforcement learning to special teams evaluations is both feasible and beneficial. Although one agent showed promising results, the discovery of flaws emphasizes the importance of further study and development in this field. The results of this study serve as a springboard for additional research and practical applications of reinforcement learning strategies in NFL, ultimately enhancing coaches' capacity for judgment and raising the effectiveness of special team units.

7.2 Future Work

This thesis' section on future work seeks to pinpoint potential areas for additional study and advancement in this topic. Given that the study has addressed a number of objectives and offered insightful information, it is critical to be aware of future opportunities to build on the findings and advance knowledge in general. This section will highlight important directions for future work while outlining specific areas where additional research, improvement, and innovation can improve the comprehension and applications of reinforcement learning in sports, specifically the NFL.

- **Refining Reward Functions**: Further refinement of the reward functions for both the returner and pursuer agents could lead to better performance. Explicit rewards that directly incentivize desired behaviors, such as moving towards

the target for the pursuer agent, could improve pursuit strategies. Experimenting with different reward structures and considering additional factors, such as player velocities or proximity to opponents, can be explored.

- **Optimizing Computational Efficiency**: The computational cost and execution time of the model were identified as significant challenges. Future work could focus on optimizing the computational efficiency of the model to reduce the time required for the simulations. This could involve exploring parallelization techniques, utilizing distributed computing, implementing different neural networks, such as a recurrent neural network, or optimizing the code implementation to make it more efficient.

- **Enhancing Model Generalization**: Addressing the limitations of training the model with only one simulation and exploring ways to generalize the model's learning across different plays and scenarios is essential. Future work could involve training the model with multiple simulations, incorporating diverse play scenarios, and exploring transfer-learning techniques to enhance the model's ability to generalize its predictions. A solution worth exploring for this is the use of Amazon Web Services, one of which is the Amazon Elastic Compute Cloud (EC2), which can provide a scalable virtual server in the cloud. EC2 instances can be launched with high-performance CPUs or GPUs, and they allow the user to configure the desired number of cores, memory, and storage capacity, enabling better performance of the model.

- **Improving Agent's Coherence**: The discrete nature of the agent's movement poses challenges in extracting meaningful insights from the model. Future work could focus on generating smoother trajectories for the agent's movement or providing additional visualizations or guidance for coaches to interpret the agent's behavior more effectively.

- **Real-World Application**: Translating the findings and insights from the simulations into real-world application is an important direction for future work. Collaborating with football teams or coaches to validate the effectiveness of the model and implementing it in live game scenarios could provide valuable feedback and further refinement.

- **Incorporating Player Interaction**: Currently, the model focuses on the pursuit of a single target. Future work could explore scenarios in which multiple pursuer agents interact with each other and coordinate their movements to achieve a collective objective, that is, a multi-agent environment. This could involve developing cooperative strategies, communication mechanisms, or even adversarial scenarios where pursuers compete against each other.

By addressing these areas of future work, the performance and applicability of the model can be further enhanced, leading to more effective decision making and strategic planning in football scenarios.

Bibliography

[1] N. Chmait and H. Westerbeek, "Artificial intelligence and machine learning in sport research: An introduction for non-data scientists," *Frontiers in Sports and Active Living*, p. 363, 2021.

[2] U. Brefeld, J. Davis, M. Lames, and J. J. Little, "Machine learning in sports (dagstuhl seminar 21411)," in *Dagstuhl Reports*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, vol. 11, 2022.

[3] J. McCarthy, "What is artificial intelligence," 2007.

[4] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[5] T. O. Ayodele, "Types of machine learning algorithms," *New advances in machine learning*, vol. 3, pp. 19–48, 2010.

[6] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: An overview," in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*, Springer, 2018, pp. 426–440.

[7] M. A. Wiering *et al.*, "Reinforcement learning in dynamic environments using instantiated information," in *Machine Learning: Proceedings of the Eighteenth International Conference (ICML2001)*, 2001, pp. 585–592.

[8] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *Journal of Robotics*, vol. 2018, 2018.

[9] R. Graham, H. McCabe, and S. Sheridan, "Neural networks for real-time pathfinding in computer games," *ITB J*, vol. 5, no. 1, p. 21, 2004.

[10] M. Sinkar, M. Izhan, S. Nimkar, and S. Kurhade, "Multi-agent path finding using dynamic distributed deep learning model," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, IEEE, 2021, pp. 1–6.

[11] S. Saeedvand, S. N. Razavi, and F. Ansaroudi, "Path-finding in multi-agent, unexplored and dynamic military environment using genetic algorithm," *Journal of World's Electrical Engineering and Technology*, vol. 2322, p. 5114, 2015.

[12] D. Jugan and D. T. Ahmed, "A decision-making and actions framework for ball carriers in american football.," in *MAICS*, 2015, pp. 109–116.

[13] O. Baykal and F. N. Alpaslan, "Supervised learning in football game environments using artificial neural networks," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, IEEE, 2018, pp. 110–115.

[14] S. Omidshafiei, D. Hennes, M. Garnelo, *et al.*, "Multiagent off-screen behavior prediction in football," *Scientific reports*, vol. 12, no. 1, p. 8638, 2022.

[15] A. Krogh, "What are artificial neural networks?" *Nature biotechnology*, vol. 26, no. 2, pp. 195–197, 2008.

[16] B. M. Wilamowski, "Neural network architectures and learning," in *IEEE International Conference on Industrial Technology, 2003*, IEEE, vol. 1, 2003, TU1–T12.

[17] A. Drewek-Ossowicka, M. Pietrołaj, and J. Rumiński, "A survey of neural networks usage for intrusion detection systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 497–514, 2021.

[18] K. G. Kim, "Book review: Deep learning," *Healthcare informatics research*, vol. 22, no. 4, pp. 351–354, 2016.

[19] M. Reza, "Galaxy morphology classification using automated machine learning," *Astronomy and Computing*, vol. 37, p. 100 492, 2021.

[20] M. Z. Alom, T. M. Taha, C. Yakopcic, *et al.*, "A state-of-the-art survey on deep learning theory and architectures," *electronics*, vol. 8, no. 3, p. 292, 2019.

[21] B. Kumaraswamy, "Neural networks for data classification," in *Artificial intelligence in data mining*, Elsevier, 2021, pp. 109–131.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[23] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[24] P. Dell'Aversana, "Reinforcement learning in optimization problems. applications to geophysical data inversion," *AIMS Geosci*, vol. 8, pp. 488–502, 2022.

[25] R. S. Sutton, A. G. Barto, *et al.*, "Reinforcement learning," *Journal of Cognitive Neuroscience*, vol. 11, no. 1, pp. 126–134, 1999.

[26] T. N. F. League. "Nfl big data bowl 2022: Help evaluate special teams performance." (2022), [Online]. Available: https://www.kaggle.com/competitions/nfl-big-data-bowl-2022/overview.

[27] S. Porwal and D. Vora, "A comparative analysis of data cleaning approaches to dirty data," *International Journal of Computer Applications*, vol. 62, no. 17, pp. 30–34, 2013.

[28] E. Rahm, H. H. Do, *et al.*, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.

[29] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 109 544–109 554, 2019.

[30] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[32] Y. Liu, D. Zhang, and H. B. Gooi, "Data-driven decision-making strategies for electricity retailers: A deep reinforcement learning approach," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 2, pp. 358–367, 2020.

[33] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*, PMLR, 2016, pp. 1329–1338.

[34] L. Espeholt, H. Soyer, R. Munos, *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International conference on machine learning*, PMLR, 2018, pp. 1407–1416.

[35] A. Tato and R. Nkambou, "Improving adam optimizer," 2018.

[36] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," 2020.

[37] L. Pauly, D Hogg, R Fuentes, and H Peel, "Deeper networks for pavement crack detection," in *Proceedings of the 34th ISARC*, IAARC, 2017, pp. 479–485.

[38] M. Abadi, P. Barham, J. Chen, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *Osdi*, Savannah, GA, USA, vol. 16, 2016, pp. 265–283.