Playing American Football By Combining Reinforcement and Imitation Learning

Brett Keir Jones

Submitted in partial satisfaction of the requirements for the degree of
Master of Science in
Intelligent Systems and Robotics
Faculty of Computing, Engineering and Media
De Montfort University
September 2022

Abstract

The success of deep reinforcement learning methods in single-agent settings has fostered a boom in research interest which as technology progresses has now expanded into multi-agent settings. Modern methods are now able to play a number of games at a super-human level. Sports games have long been used as testbeds for AI research, a distinguishing characteristic of sports games from other genres is that dominance of the basic game mechanics is not enough to be considered successful; they are also expected to exhibit realistic human-like behaviours and be fun to play.

This dissertation project investigates the combination of reinforcement and imitation learning methods applied to a novel American football themed environment. The experimental results obtained demonstrate that there is some evidence of emergent behaviours and the combination of reinforcement and imitation learning leads to faster training and stronger game playing agents than reinforcement learning alone. The results also suggest that end-to-end reinforcement learning on its own is not enough to produce the type of explicitly coordinated actions as would be required in a commercial game.

Acknowledgements

I would like to thank Dr Jethro Shell for his invaluable advice and support during this project process. I would also like to thank Sandie Reid for her guidance and finally I would also like to thank my wife Kirsti for her continued love and patience over the last year.

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
AF	American football
AWS	Amazon Web Services
BC	Behavioural Cloning
COMA	Counterfactual Multi-Agent Policy Gradients
CSP	Competitive Self-Play
CTDE	Centralised Training Decentralised Execution
Dec-POMDP	Decentralised Partially Observable Markov Decision Process
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
IL	Imitation Learning
IRL	Inverse Reinforcement Learning
FPS	First Person Shooter
FPS	For The Win
GAI	General Artificial Intelligence
GAIL	Generative Adversarial Imitation Learning
GAN	Generative Adversarial Network
GRF	Google Research Football
HFO	Half-Field Offense
MARL	Multi-Agent Reinforcement Learning
MADRL	Multi-Agent Deep Reinforcement Learning

MA-POCA	Multi-Agent POsthumous Credit Assignment
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multi-Layer Perceptron
MOBA	Multiplayer Online Battle Arena
MVP	Minimum Viable Product
NCAA	National Collegiate Athletic Association
NFL	National Football League
PAMDP	Parameterised Action Markov Decision Process
POMDP	Partially Observable Markov Decision Process
POSG	Partially Observable Stochastic Game
PPO	Proximal Policy Optimisation
RL	Reinforcement Learning
RTS	Real Time Strategy
SAFE	Simple American Football Environment
WAR	Wins Above Replacement

Table of Contents

Chapter 1

Introduction

Deep reinforcement learning (DRL) methods have enabled the investigation of problems that were previously thought to be intractable. Over the last few years DRL methods have seen success in a number of different domains such as Autonomous Vehicles (Kiran *et al*., 2020), Robot Swarms (Hüttenrauch, Sosic and Neumann, 2018), Internet Advertising (Jin *et al*., 2018), Smart Energy Grids (Xi *et al*., 2018) and Traffic Light Control (Tan *et al*., 2020) to name but a few.

It has been claimed (Silver at al., 2021) that DRL methods could be a viable route for achieving the goal of human-like Artificial General Intelligence (AGI) (Pennachin and Goertzel, 2007) and influenced by the success of Deep Q-Networks (Mnih *et al*., 2013; 2015) in the Atari Learning Environment (Bellemare *et al*., 2012), many researchers have continued to focus on improving and applying deep reinforcement learning methods to various different computer and video games as they can provide fast, safe and suitably complex environments for experimentation. To this end there have been a number of video game based reinforcement learning competitions such as ViZDoom (Kempka *et al*., 2016; Wydmuch, Kempka and Jaskowski, 2018), the OpenAI Retro Contest (Nichol *et al*., 2018), the Obstacle Tower Challenge (Juliani *et al*., 2019) and the NeurIPS MineRL competitions (Guss *et al*., 2019;2021; Kanervisto *et al*., 2022).

The survey papers by Shao *et al.* (2019) and Justesen *et al.* (2019) provide more detail about the general application of reinforcement learning methods to games but some of the most notable examples are the *Go* playing family; evolving from methods where the rules are provided in AlphaGo (Silver *et al*., 2016) which was able to beat the worlds top professional *Go* players, and its refinement, AlphaZero (Silver *et al*., 2018) which also learned to play *chess* and *shogi* at a super-human level from scratch. The

latest iteration, MuZero (Schrittwieser *et al*., 2019), first introduced in 2019, is also able to play the suite of Atari games as is able to automatically learn the underlying rules of each game, allowing it to master unseen environments with no model provided (Schrittwieser *et al*., 2019). Indeed, 2019 can be seen as a particularly significant year for deep reinforcement learning research as it saw the emergence of a number of other advances such as the OpenAI Five agents for *Dota 2* (OpenAI *et al*., 2019), the AlphaStar agent for *Starcraft II* (Vinyals *et al*., 2019), the FTW agent for *Quake III: Capture The Flag* (Jaderberg *et al*., 2019) and the agents that learned to play hide and seek (Baker *et al*., 2019).

These examples represent important breakthroughs for the field as OpenAI Five was the first AI to beat the reigning world champions in an eSports game while AlphaStar was rated at a 'Grandmaster' level and ranked above 99.8% of the world's best human players. The FTW agents are able to reach human-level performance in an complex, team based FPS game using raw pixel input only while the hide and seek agents used an implicit autocurricula (Leibo *et al*., 2019) during training that enabled them to learn increasingly complex strategies including making use of various tools in unexpected ways.

One area that appears to have had less attention is the application of reinforcement learning methods to sports games in general (Mozgovoy, Preuss and Bidarra, 2021; Li, Fussell and Komura, 2021). For example, the sport of American football (AF) features two teams of cooperative agents competing against each other in a complex zero-sum game. As far as could be established from a review of the literature, prior to this project there were no American football environments currently freely available for conducting research on coordinated team behaviours. To address this gap, this dissertation project therefore introduces a novel sport-related reinforcement learning environment suitable for the creation and study of multi-agent AI behaviours. The motivation for selecting American football as the domain is detailed in section 1.1 below.

1.1 Motivation

Sport simulations are among the very earliest established genres of video games; *Tennis For Two* (Higinbotham, 1958) is generally thought of as one of the first video games ever created. Over the years, there has been a general trend towards the convergence between real world sports and the simulated versions (Sicart, 2013). As

video games have evolved, the fidelity of the graphical, audio and physical simulation engines have steadily continued to improve with each new generation of technology. Modern AAA sports games like Electronic Art's Madden, FIFA and NBA franchises now have advanced game engines that feature near photo-realistic graphics, life-like physically-driven animation systems and extremely polished TV-style presentation.

It has been stated (Mozgovoy, Preuss and Bidarra, 2021) that good AI systems are one of the cornerstones of a high-quality team sport game. Unlike in most other genres, because sports games represent a real world activity, they are judged on how close they are to the real thing. Therefore any AI-controlled teammates or opponents have to be believable and act in a human-like manner; if they often exhibit strange or unrealistic AI behaviours then it breaks the sense of immersion for the user (Coleridge, 1817, quoted in Roberts, 2022). For example, in his comparison between the game *FIFA 12* and real life football, Sicart (2013) noted that although the aesthetics and general simulation were satisfactory, he believed that the AI system fell short in its interpretation and understanding of the sport.

American football is a strategically complex team sport that has often been turned into video games. First released in 1988, the popular *Madden NFL* series of games are some of the most commercially successful, having sold upwards of 130 million copies as of 2018 ('Madden NFL', 2022). Marketed as a simulation, a high level of realism is expected by its players in all aspects of the game, including the AI systems. A cursory search of the internet shows that there are still a great many users who have been dissatisfied with how the AI behaves over the past few incarnations (e.g. Sin, 2013; Dcfan1992, 2021; Canes21, 2020). In November 2020 there was even a lawsuit brought against Electronic Arts due to how the AI behaves during online multiplayer matches (Sharbaugh, 2021). Examples given online as anecdotal evidence of AI problems include descriptions of situations where in real word if its 3rd down and long it would be expected to be a passing play for the offence and thus the defence would hang back to defend the line-to-gain, but because in the Madden series the AI 'cheats'; the defensive play selection is purely reactive, it waits until the user has already selected an offensive play then picks the correct defensive play to counter it. In this example because the user instead opted to run the ball (to try and pick up a few extra yards before punting the ball away) the defence opted for a run-stop play which the user flagged as unrealistic behaviour. Other issues commonly raised online are problems with the inability of the offensive line to block correctly (e.g. Baby-boy, 2021) and that sometimes the AI players either stand around doing nothing or run

around aimlessly (e.g. Canes21, 2020).

Defining the correct actions for the AI agents to in every possible situation in a video game using traditional handcrafted game AI solutions such as Behaviours Trees (Colledanchise and Ögren, 2018) or Finite State Machines (Yannakakis and Togelius, 2018) is a time consuming, complex and error prone process and often leads to either very predictable or brittle behaviours (Yannakakis and Togelius, 2018). Additionally as game worlds become ever more content rich with potentially large numbers of agents interacting with each other and the environment, it is becoming increasingly harder to scale up these traditional methods which calls for the development of alternative approaches for producing human-like and believable AI behaviours.

The issues highlighted above demonstrate the need for finding new approaches to creating the AI behaviours in games. In recent years, one approach to creating AI agent behaviours that has seen a lot of research attention is by training agents via deep reinforcement learning. DRL approaches have already been proven fruitful in a number of genres such as card games, board games, FPS, MOBA and RTS games (Silver *et al*., 2016; 2018; Jaderberg *et al*., 2019; OpenAI *et al*., 2019; Vinyals *et al*., 2019).

According to Mozgovoy, Preuss and Bidarra (2021) there are two primary reasons why (team) sport games are an important testbed for AI benchmarking; First, they argue there are several specific challenges for AI systems that are particular to only to sports games and second, the goal of the AI agents is not just to win but also contribute to player enjoyment by appearing human-like and believable. Sports-inspired environments, such as RoboCup (Kitano *et al*., 1995), have been used in general AI research and benchmarking for many years, however aside from a few higher profile projects such as the *Fever Basketball* game (Jia *et al*., 2020) and a football simulation environment released by Google (Kurach *et al*., 2020) the application of DRL methods to sports games has had relatively little consideration thus far.

As existing DRL challenges are solved, a number of authors have expressed the need for new and ever more complex learning environments in order to continue to advance the field (e.g. Juliani *et al*., 2019; Kurach *et al*., 2020). As such there has been a trend moving away from purely single and towards multi-agent environments which helps to foster research around the additional challenges encountered when multiple collaborative or competing agents are present. In terms of developing team strategies; Mozgovoy, Preuss and Bidarra (2021) declared that they believe sport games to be at least as challenging as MOBA or RTS games. Additionally, in their study investigating the application of Neuro-Dynamic programming to modelling a American football

drive, Patek and Bertsekas (1998) concluded that American football poses a challenging set of problems which can act as a "effective testbed" (Patek and Bertsekas, 1998, para. 10) for further AI research.

1.2 Dissertation Outline

The structure of the rest of this dissertation is as follows, beginning with an in-depth literature review which is divided into two chapters; the key theoretical background information that underpins both reinforcement learning and imitation learning is covered in chapter 2 while chapter 3 provides an overview of the research literature related to this project. Chapter 4 gives details of the methodology used to create the learning environment, the algorithm used for training multiple agents as well as how verification and validation was performed. Chapter 5 provides information on the experimental setup, a discussion on the results obtained. It then provides a critical review of the project outcomes as well as detailing some avenues for potential future investigation. Finally in chapter 6 a summary of the project and its findings is provided.

Chapter 2

Background

As the aim of this project is to investigate the use of reinforcement learning applied to a sports game setting, this chapter will first provide a brief overview of the core theory for single agent reinforcement learning beginning at section 2.1 below. Next the background for mulri-agent reinforcement learning is covered in section 2.4. this chapter then covers some of the key issues and challenges that need to be addressed when moving to a multi-agent setting in section 2.8. Finally this chapter also gives background detail for imitation learning and for the specific methods that are the foundation for the methodology presented in chapter 4.

2.1 Reinforcement Learning

Reinforcement learning is a goal-oriented machine learning approach in which a decision making computer program (referred to as an 'agent') automatically learns a sequence of 'actions' to take using just feedback ('rewards') received from the (potentially unknown) environment when performing a series of trial and error interactions (Sutton and Barto, 2018). The aim of the agent is to maximise its numerical reward received by selecting the optimal sequence of actions that achieve its goal (Sutton and Barto, 2018). Time is a key component of Reinforcement Learning and an important characteristic is that the reward signal for an action may be delayed until a later time step. Actions may not only affect the immediate reward received but also the next state and thus potentially all subsequent rewards (Sutton and Barto, 2018).

Reinforcement learning falls somewhere in between the supervised and unsupervised machine learning paradigms Sutton and Barto, 2018). It is not strictly supervised as there is no 'correct' set of labelled training data provided as feedback on a task, in-

stead the agent must discover for itself which actions correspond to the highest rewards through exploration. In unsupervised learning the goal is to find similarities and differences between data points while in reinforcement learning the aim is to find a decision making model for maximising the agent's total cumulative reward (Sutton and Barto, 2018).

More formally, a reinforcement learning problem is often defined as in figure 2.1, here the agent and environment are shown at a time step t.

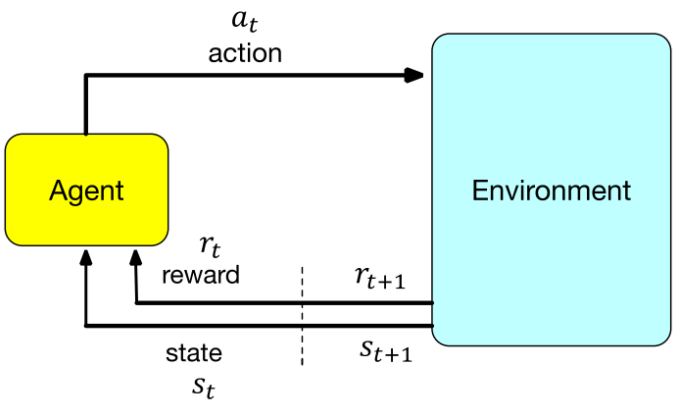

Figure 2.1: Reinforcement Learning feedback loop (Du and Ding, 2021).

At each time step the agent receives an observation of the environment's state s_t where $s_t \in S$ and uses that information to select an action a_t where $a_t \in A(s)$. In the next time step the agent then receives its numerical reward r_{t+1} (where $r_{t+1} \in R \subset \mathbb{R}$) for the previous action and the observation of the new current state s_{t+1}. This feedback loop is repeated until the agent's goal has been achieved or some other terminal state reached (Sutton and Barto, 2018).

A basic taxonomy of reinforcement learning algorithms is presented in figure 2.2.

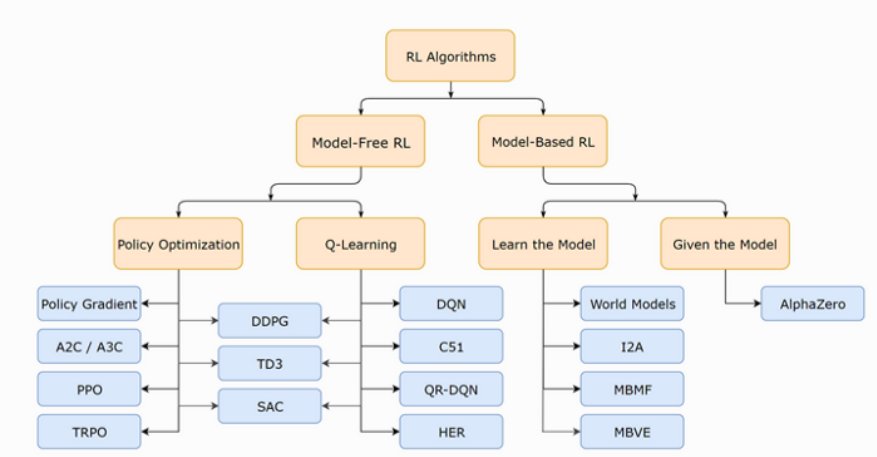

Figure 2.2: A taxonomy of reinforcement learning methods (OpenAI, 2018).

2.1.1 Model-Based vs Model-Free

In reinforcement learning, algorithms are categorised as being either 'model-based' or 'model-free'. In model-based methods the agent either has access to or learns a model of the environment's underlying dynamics (Sutton and Barto, 2018). If the model is provided the reward function and the transition process can be accessed directly by the agent (Sutton and Barto, 2018) which enables it to be able to use forward planning techniques when selecting actions. Notable examples of methods that use a given model are the AlphaGo family of algorithms (Silver *et al.*, 2016; 2018). In methods that learn the model from experience, agents first interact with the environment and then iteratively use the feedback received to build and refine a model which in turn is used to improve the policy (Sutton and Barto, 2018). Examples of methods that learn a model are the Imagination-Augmented agents (Racanière *et al.*, 2017) and World Models (Ha and Schmidhuber, 2018) algorithms.

A key advantage of model-based methods is that they can leverage planning methods to look ahead to potential future states and rewards; which helps agent to make better decisions (Sutton and Barto, 2018). An issue with model based methods is that if there are any inaccuracies or bias in the model then it will lead to learning behaviours that appear to be satisfactory in the simulated environment but turn out to be inadequate in the real environment (Sutton and Barto, 2018). In practice an accurate model of the environment is usually not available and the system dynamics are too complex or opaque for an agent to learn (Sutton and Barto, 2018).

Model-free algorithms do not try to construct a model of the environment, instead

agents interact with the environment directly on a trial-and-error basis in order to improve their policies (Sutton and Barto, 2018). Compared to model-based methods, model-free methods are generally more straightforward to implement (as there is no model to maintain) but are not as efficient and may require a large number of environment samples to learn an adequate policy (Sutton and Barto, 2018). Q-learning (Watkins, 1989; Watkins and Dayan, 1992) is a classical model-free method.

2.1.2 On vs Off Policy

Reinforcement Learning algorithms are also classified as being either 'on-policy' or 'off-policy'. On-policy algorithms attempt to improve a single policy which used to both make decisions and explore behaviours by interacting with the environment (Sutton and Barto, 2018). Off-policy algorithms can evaluate and improve a decision making policy which is different from the one used to generate the sample data. An advantage of on-policy methods is that they can perform direct optimisation and convergence is generally faster and more stable (Sutton and Barto, 2018). A potential issue is that they can select sub-optimal actions during exploration that are then used to update the policy even if the resulting behaviour is substandard (Sutton and Barto, 2018). SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 2018) is a notable on-policy algorithm which uses the current policy to select an action, executes the selected action and then uses the resulting feedback gained to update the policy.

Off-policy methods can make use of prior experience or the experience of other agents in order to improve its policy which can make exploration more efficient (Sutton and Barto, 2018). Off-policy algorithms do not have to use the resulting data generated when the exploration selects an inferior action, instead it can use the best previously seen action and avoid tainting the policy to keep it optimal (Sutton and Barto, 2018). Off-policy learning can be unstable in practice, to mitigate this most approaches randomly sample a replay buffer of stored experience data; the Q-learning family of algorithms (Watkins, 1989; Watkins and Dayan, 1992; Mnih *et al.*, 2015) are typical of this type of approach.

2.1.3 Value-Based vs Policy-Based

Reinforcement Learning can further be grouped into 'value-based' or 'policy-based' optimisation methods. Value-based methods aim to implicitly construct an optimal policy by learning to maximise the state-action function whereas Policy-based methods

attempt to directly optimise the policy via gradient ascent without requiring additional information from the underlying Markov Process (Sutton and Barto, 2018). Value-based algorithms can have issues when the state/action spaces are high dimensional and often suffer from oscillation issues during conversion (Sutton and Barto, 2018). As they only make small incremental changes, policy-based algorithms tend to be more stable but less efficient (Sutton and Barto, 2018). A benefit of policy-based algorithms is that they can learn both stochastic and deterministic policies, whereas value-based algorithms can only handle deterministic policies (Sutton and Barto, 2018).

Again Q-learning (Watkins, 1989; Watkins and Dayan, 1992) and its derivatives are a classic example of a value-based approach to reinforcement learning, while REINFORCE (Sutton and Barto, 2018) is a classical example of a policy-based approach.

2.1.4 Actor-Critic Frameworks

A hybrid approach is the actor-critic framework (Konda and Tsitsiklis 2000; Sutton and Barto, 2018) that simultaneously learns a policy function and a value function and seeks to combine the advantages of both value and policy based methods and is suitable for either a discrete or continuous action space. The actor-critic framework uses its policy-based methods to learn the policy function which proposes a set of possible actions (the actor) and its value-based methods to evaluate actions taken by the actor based on the given policy (the critic) (Konda and Tsitsiklis 2000; Sutton and Barto, 2018). Actor-critic algorithms generally involve a trade-off between the strengths and weaknesses of either side of the approach, sample efficiency can be improved but there can be issues with insufficient exploration and over optimistic estimation (Konda and Tsitsiklis 2000; Sutton and Barto, 2018).

2.2 Deep Reinforcement Learning

A serious limitation with classical reinforcement learning as described above is that for anything other than simple environments it becomes computationally intractable to calculate, store and update all possible states and values in a memory based lookup table. To overcome this limitation, a combination of deep learning and reinforcement learning methods can be used; referred to as deep reinforcement learning (DRL) (Li, 2018; Lapan, 2018; Zai and Brown, 2020; Morales, 2021; Plaat, 2021). In DRL, instead of using a lookup table to store values as in the classic tabular approach, multi-layer non-

linear neural networks can be used to represent the state (or state observation) itself as well as the policy or value functions. For example with the value or policy functions, the neural networks take a representation of the environment state (such as custom vectors or raw pixel information) as the input and then output an approximation of the value and policy function results. Doing this enables DRL based systems to perform a wide range of complex decision-making tasks that were previously impossible due to having very large state or action spaces. (Li, 2018; Lapan, 2018; Zai and Brown, 2020; Morales, 2021; Plaat, 2021).

2.3 Markov Decision Processes

If the next state in a reinforcement learning problem depends only on the current state of the environment and the actions currently available then the problem is said to satisfy the Markov property (Oliehoek and Amato, 2016) as no prior history or additional outside information is required, and can thus be represented as a Markov Decision Process (MDP) (Oliehoek and Amato, 2016). MDPs are the theoretical foundation that underpins reinforcement learning and can be defined as a five-tuple (S, A, P, R, γ) where:

- S is the set of all possible states the environment can be in; every timestep the agent observes the current state of the environment where $s \in S$.
- A is the set all of all possible actions available to the agent; every timestep the agent selects an action a where $a \in A$.
- P is the probability that action a in state s at time t leads (transitions) to state s' at time t_{+1}.
$$P_a(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$$
- R is the reward received after an action a transitions state s to state s'. $R_a(s, s')$
- γ where $\gamma \in [0, 1]$ is a discount factor representing the difference between present and future rewards.

In an MDP, P and R together define the world model and represent the environment's dynamics and the long-term reward for each strategy. The strategic decision making process for selecting an action based on the current observation and reward received is referred to as the agent's policy function π which maps states to actions. The

policy function can either be deterministic $\pi : s \mapsto a$ which always returns one action for each state, or stochastic $\pi(a_t|s_t)$ which gives the probability of taking action a in state s.

The goal in reinforcement learning is to find the optimal policy function π^* that gives the best action to take for each state that leads to the greatest overall total reward. In order to find the optimal policy, there needs to be some method of determining how 'good' a particular state is so the agent can determine whether or not it is desirable to reach that state. In reinforcement learning there are two standards methods, via the state-value function $V^\pi(s)$ or via the state-action-value function $Q^\pi(s,a)$, both of which are defined by the Bellman equations (e.g. Sutton and Barto, 2018).

The Bellman equations provide a recursive decomposition of the problem by breaking the functions into two parts; the value of the current timestep and the predicted value of all the remaining steps:

The state-value function $V^\pi(s)$ returns the predicted total discounted future reward for a particular starting state s if policy π is followed.

$$V^*(s) = max_a \Sigma_{s',R} P(s',R|s,a)[R + \gamma V^*(s')]$$

The state-action-value function $Q^\pi(s,a)$ returns the predicted total discounted future reward for a particular starting state s if action a is taken first and then policy π is followed thereafter. This is the basis of the Q-learning (Watkins, 1989; Watkins and Dayan, 1992) family of algorithms.

$$Q^*(s,a) = \Sigma_{s',R} P(s',R|s,a)[R + \gamma max_{a'} Q^*(s',a')]$$

Both the state-value and state-action-value functions have respective optimal forms (Sutton and Barto, 2018) that will maximise the cumulative rewards as much as possible:

The optimal state-value function is the one with the highest value across all policies.

$$V^*(s) = argmax_\pi V^\pi(s)$$

The optimal state-action-value function is also the one that returns the highest values.

$$Q^*(s,a) = argmax_\pi Q^\pi(s,a)$$

As stated above the goal is to find an optimal policy π^*. An optimal policy is one in which the expected cumulative discounted reward for every state is greater than or

equal to those returned by any other policy (Sutton and Barto, 2018).

2.3.1 Partially Observable Markov Decision Processes

It is important to note that the MDP formalisation above assumes full observability of the environment state. This assumption may be true for board games such as chess but is often not the case in many situations such as in in poker or bridge, where the player does not have access to all the card information and especially in robotics where sensor readings may be interrupted or noisy. Imperfect information problems where some information may be hidden from some agents, some of the time, can be modelled via an extension to standard MDPs known as Partially Observable Markov Decision Processes (POMDP) (Åström, 1965; Oliehoek and Amato, 2016). POMDPs violate the Markov Property (Oliehoek and Amato, 2016) by incorporating the history of observations made by an agent and their probability of occurrence within each possible environment state. The agent no longer knows the exact state of the environment but instead maintains a Bayesian belief distribution over the possible states that is used for decision making (Kaelbling, Littman and Cassandra, 1998).

2.4 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is the combination of reinforcement learning and multi-agent systems. MARL draws from a number of different fields such as game theory, machine learning, stochastic control theory, psychology, statistics and optimisation (Neto, 2005; Busoniu, Babuska and De Schutter, 2008; 2010). Historically most reinforcement learning research has focused on either single or two agent problems; when extending DRL from the single to multi agent paradigm there are several additional challenges that need to be addressed (Neto, 2005; Busoniu, Babuska and De Schutter, 2008; 2010). In a multi-agent setting agents may need to collaborate or compete with one another, the environment is no longer deterministic but stochastic as agents may move simultaneously and it is difficult to ascertain which action(s) by which agent(s) were the ones that lead to the greatest reward (Neto, 2005; Busoniu, Babuska and De Schutter, 2008; 2010).

In MARL there are two or more autonomous agents attempting to simultaneously learn within a shared interactive environment as shown in figure 2.3. As with the single agent case, each agent is trying to solve a sequential decision-making problem via a

trial and error approach with the goal of finding an optimal policy that maximises its cumulative discounted reward received from the environment (Neto, 2005; Busoniu, Babuska and De Schutter, 2008; 2010).

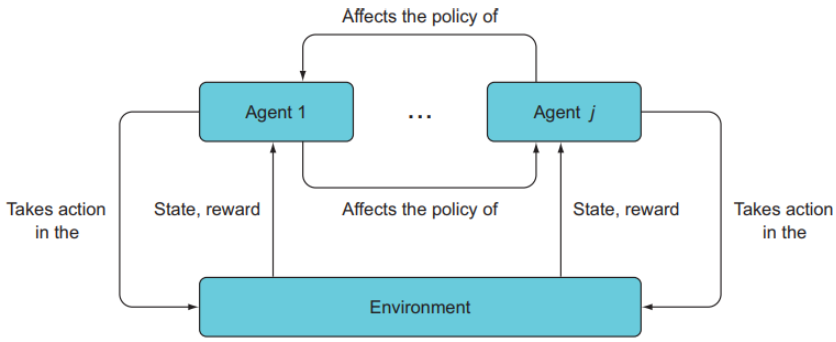

Figure 2.3: Multi-agent reinforcement learning (Zai and Brown, 2020, p. 244).

However, agents now may potentially interact repeatedly with each other as well as the environment, which makes successful reinforcement learning in a multi-agent domain fundamentally more complex than learning with only a single agent (Neto, 2005; Busoniu, Babuska and De Schutter, 2008; 2010). An overview of some of the most common challenges and issues that need to be addressed in multi-agent reinforcement learning is provided later in this chapter in section 2.8.

2.4.1 Multi-Agent Deep Reinforcement Learning

Since the advent of the deep learning paradigm, as with the single agent case, research is now in the so-called “Second Era” (Lanctot, 2019) of MARL in which deep neural networks are commonly used as powerful non-linear function approximators (Li, 2018; Plaat 2022). This new sub field of ‘Multi-Agent Deep Reinforcement Learning’ (MADRL) has enabled research into more complex problems and can handle environments with large action or state spaces that were previously unfeasible to address using traditional tabular methods (Plaat, 2022).

2.5 Agent Relationships

Depending on the scenario and reward structures involved, agents may need to cooperate or compete with each other or they may even require a mixture of competitive and cooperative behaviours to succeed. In general, MARL algorithms can therefore be

categorised into three corresponding groups; fully cooperative, fully competitive, or a mixture of both (Busoniu, Babuska and De Schutter; 2008; 2010; Plaat, 2022).

2.5.1 Fully Cooperative

In a fully cooperative setting, all agents collaborate to achieve a shared goal; success for one agent is success for all agents. In this setting it is common for all the agents to receive the same reward for state transitions, making it a team joint-return (Busoniu, Babuska and De Schutter, 2008; 2010). As all agents have the same reward function, the overall learning objective is to maximise the common long-term discounted cumulative reward received (Plaat, 2022). Thus agents are motivated to work together and try to avoid individual failures in order to optimise the overall performance of the team (Plaat, 2022). A more general case also exists where agents may have different reward functions (Plaat, 2022) and the overall system reward function is a vector of all the individual agent rewards (Plaat, 2022). In this heterogeneous context it is common for the average reward of the team to be used as the optimisation target (Plaat, 2022).

2.5.2 Fully Competitive

In a fully competitive setting the reward of all the agents sums up to zero; one agent's gain is a loss for the other agents in the environment. Agents are encouraged to act in a purely selfish manner and will aim to maximise their individual rewards while minimising the reward received by other agents (Busoniu, Babuska and De Schutter, 2008; 2010). Historically much of the research in a competitive setting has been focused on two-player games as the computational complexity increases significantly with each new agent present in the environment (Plaat, 2022).

2.5.3 Mixed Cooperative-Competitive

Also known as general sum games (Busoniu, Babuska and De Schutter, 2008; 2010), in a mixed cooperative-competitive setting, no restrictions are imposed on the inter-agent relationships (Plaat, 2022). Agents may or may not choose to collaborate with others depending on what strategy appears to yield the greatest reward (Plaat, 2022). This setting includes team-based games where to be successful agents on the same team must work together while also acting in an adversarial manner to agents on the opposing team (Plaat, 2022). At the team-to-team level it is therefore a zero-sum

competitive game (Plaat, 2022).

2.6 Stochastic Games

Stochastic Games, (Shapley, 1953) also often referred to as Markov Games, (Littman, 1994) generalise the Markov Decision Process formalism from the single to the multi-agent case, and like MDPs, assume the environment to be fully observable. A Stochastic game is a collection of simultaneous action normal-form games (Shapley, 1953) that are repeated multiple times for the agents present; the game state can be represented by a payoff matrix which is used to determine the joint action to be selected. Formally, a Stochastic Game is defined by the tuple (I, S, A, R, T, γ) where:

- I is the set of N agents.
- S is the set of all possible states for the environment.
- A is the joint action space of the N agents. $(A = A_1 \times A_2 \times \cdots \times A_N)$
- R is the joint reward for all the agents. $R = (r_1, r_2, \ldots r_N)$ where $R_i : S \times A \to \mathbb{R}$ is each agent's individual reward function.
- T is the state transition function. $T : S \times A \times S \to [0, 1]$
- γ is the discount factor where $\gamma \in [0, 1]$.

Stochastic Games can be cooperative, competitive or mixed; in cooperative games all the agents share the same reward $r1 = r2 = \cdots = RN$ while a competitive or mixed team game is zero-sum.

2.6.1 Partially Observable Stochastic Games

As with single agent MDPs, in practice the environment is often not fully observable and can therefore be defined by Partially Observable Stochastic Games (POSGs) (Oliehoek and Amato, 2016). Also whether agents cooperate or not is dependent on the agent's individual reward functions. The case in which all agents share the same reward function $R_1 = R_2 = \cdots = R_N$; for example, one side in a team sports game where all agents on team share the same goal, is known as a 'Decentralised Partially Observable Markov Decision Process' (Dec-POMDP) (Bernstein *et al.* 2000; 2002; Oliehoek and Amato, 2016).

2.6.2 Decentralised Partially Observable Markov Decision Processes

The Dec-POMDP framework (Bernstein *et al*. 2000; 2002; Oliehoek and Amato, 2016) can be used to model stochastic cooperative team games that have partial observability, while they may still have different individual goals, instead of agent's optimising their individual reward functions, in a Dec-POMDP all the agents are attempting to maximise the joint reward received by the team (Bernstein, Zilberstein and Immerman, 2013; Oliehoek and Amato, 2016).

A Dec-POMDP can be formally described (Oliehoek and Amato, 2016) by the tuple $(I, S, A, P, \Omega, O, R, \gamma)$ where:

- I is the set of N agents.
- S is the set of all possible states for the environment.
- A is the joint action set of all the N agents, where A_i denotes the action space for an individual agent i.
- P is the set of state transition probabilities; $P(s'|s,a)$. This is the probability of transitioning to state s' while in state s and taking action a.
- Ω is the set of joint observations; Ω_i denotes a set of observations for a single agent i. $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_N$ and $o_i \in \Omega_i$
- O is the set of observation probabilities; $O(o|s,a)$, the probability of agent's having observations o, while in state s and taking action a.
- R is the expected immediate reward function; $R(s, a_1, \ldots, a_N)$.
- γ is the discount factor where $\gamma \in [0, 1]$.

At every time step, each agent takes an action and receives both an immediate joint reward and a local observation. Each agent has its own local policy which is a mapping from its individual observations to actions. Under partial observability an agent's observation history may be used to help determine the current state which again violates the Markov property (Oliehoek and Amato, 2016). A joint policy is a collection containing each individual agent's local policy; decisions for each agent are made solely on local information, but these decisions can affect the overall global reward as well as each agent's transitions and observations (Oliehoek and Amato, 2016). As in the

Stochastic Games framework, the objective is to maximise the expected return by selecting an optimal joint policy (Oliehoek and Amato, 2016). However, in this case, the policy performs a mapping from local observations to actions; thus, the local policy of agent i can be written as $\pi^i : O^i_t \rightarrow A^i$. The Dec-POMDP model is very general and has been applied to a wide range of applications such as solving riddles (Foerster *et al.*, 2016), real-time strategy games such as *Starcraft II* (Vinyals *et al.*, 2019) or *Dota 2* (OpenAI *et al.*, 2019) as well as first person action games such as *Quake III: Capture the Flag* (Jaderberg *et al.*, 2019). Solving Dec-POMDPs is a challenging problem that has been shown to NEXP-complete in the worst case (Bernstein, Zilberstein and Immerman, 2013). Many recent solution algorithms for Dec-POMDPs take advantage of a key assumption; that training can be centralised as long as execution is decentralised. Using a centralised controller that can access the observations from all agents therefore converts the problem into a POMDP which is much easier to solve (Oliehoek and Amato, 2016).

2.7 Multi-Agent Training Architectures

A major challenge in MADRL is to find an agent training scheme that is both efficient and can deal with the non-stationarity and partial observability problems (Wong *et al.*, 2021). As per figure 2.4 below, the most common approaches can generally be divided into fully centralised, fully decentralised or hybrid architectures (Wong *et al.*, 2021).

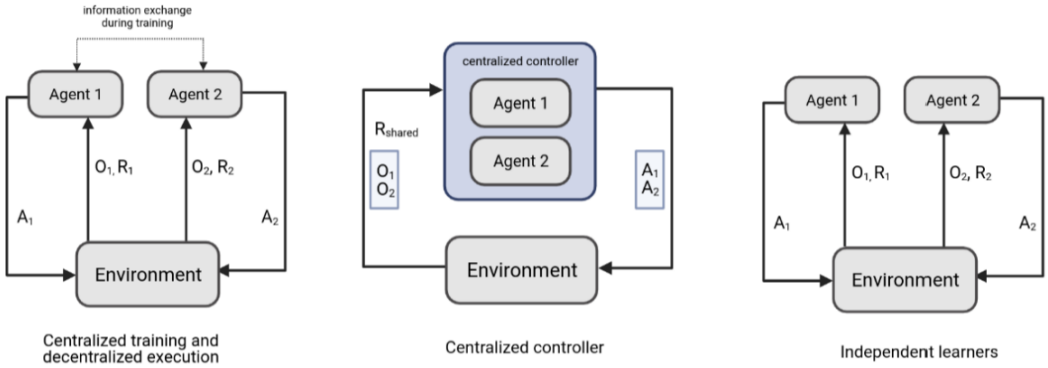

Figure 2.4: Multi-agent training schemes (Wong *et al.*, 2021).

2.7.1 Fully Centralised

In a fully centralised approach, all the agents are treated as a single joint entity (Bowling and Veloso, 2002) which effectively reduces the multi-agent scenario to a single-

agent problem where standard reinforcement learning algorithms can be applied. In this approach all agents send their observations and policies to a central controller, which then decides what action to take for each agent (Wong *et al*., 2021). The input is the joint state which is the set of individual observations from all agents, and the output is a joint action which consists of the combination of the actions to be taken (Wong *et al*., 2021).

Using a centralised controller is one of the simplest multi-agent training schemes to implement (Foerster, 2018) and can help to mitigate issues with partial observations in scenarios where agents receive incomplete state information (Foerster, 2018). While this approach can work well when there are only few agents present it is computationally expensive and does not scale as the joint action space increases exponentially with the number of agents. Additionally in many real world settings, such as with autonomous vehicles, agents have to make decisions based solely on their local observations making a centralised control scheme impossible (Foerster, 2018).

2.7.2 Fully Decentralised

In a fully decentralised approach each agent is trained independently and considers other agents as part of the environment (Tan, 1993). A direct extension of single-agent learning to a multi-agent environment, in this architecture each agent learns an individual policy based solely on its local observations. Instead of having to contemplate the joint-action space of all agents present, each agent only needs to consider its own action space, which means the problem can conveniently be modelled as a Decentralised Partially Observable Markov Decision Process (Oliehoek and Amato, 2016) as detailed above in section 2.6.2. Although a decentralised architecture does not suffer from scalability issues caused by the growth of the number of agents, they effectively discard any explicit multi-agent coordination and instead relies on suitable problem solving behaviours emerging by chance. This approach is therefore sensitive to issues arising from partial observability, credit assignment and environment non-stationarity due to multiple agents switching policy at the same time (Hernandez-Leal et al., 2017; 2019; Du and Ding, 2021). While fully decentralised approaches can be successful in some scenarios, in general, they often fail to learn complex behaviour between agents, such as communication and cooperation, that are required to achieve goals in more difficult environments (Foerster, 2018).

2.7.3 Centralised Training and Decentralised Execution

First introduced by Kraemer and Banerjee (2016), a popular hybrid training architecture is known as 'Centralised Training and Decentralised Execution' (CTDE). In this approach, during training agents can access global information that is normally unavailable such as all the other agent's observations, rewards, (policy) gradients and parameters. During inference, the extra information is removed and agents execute their learned policies in a decentralised manner based on local observations only. A CTDE approach helps to mitigate problems with non-stationarity and partial observability and stabilises the agent learning process (Kraemer and Banerjee, 2016), while avoiding the exponential explosion in action and state spaces normally associated with a centralised approach (Kraemer and Banerjee, 2016). CTDE methods can be divided into value-based methods and policy-based methods (Wong *et al*., 2021). Value-based methods typically focus on decoupling centrally learned value functions and how to apply them with decentralised execution (Wong *et al*., 2021). Most policy-based methods have an actor-critic architecture where a centralised critic is to train decentralised actors (Lyu *et al*., 2021). While CTDE has proven to be an effective solution in situations such as in games or robotics where a simulator is usually available (Foerster, 2018), it is not a blanket solution as it is not applicable to many real-world problems that cannot be accurately simulated.

2.8 Common Challenges and Issues in MADRL

Multi-agent reinforcement learning is known to be challenging both technically and conceptually, and demands a clear understanding of both the overall objective and how the system will be measured (Shoham, Powers and Grenager, 2007). Alongside traditional single-agent concerns, such as the Exploration-Exploitation trade-off, Sparse rewards and Sample Efficiency (Sutton and Barto, 2018), the transition to a multi-agent setting introduces a new set of challenges and issues that must also be addressed (Hernandez-Leal *et al*., 2017; 2019; Nguyen, Nguyen and Nahavandi, 2020; Du and Ding, 2021; Canese *et al*., 2021; Wong *et al*., 2021; Plaat 2022).

Some challenges such as environment partial observability, non-stationarity and system scalability are universal in the MADRL domain (Nguyen, Nguyen and Nahavandi, 2020; Du and Ding, 2021; Canese *et al*., 2021; Wong *et al*., 2021) while others such as the learning of explicit coordination and communication or the (tempo-

ral) credit assignment problem are more applicable to cooperative settings (Gupta *et al.*, 2017; Oroojlooy and Hajinezhad, 2019).

2.8.1 Partially Observable Environments

In a partially observable multi-agent setting agents do not receive the full global information about the state of the environment on each transition (Zhang, Yang and Başar, 2019; Canese *et al.*, 2021; Wong *et al.*, 2021). Instead agents are forced to make the 'best' decisions they can based on their local observations only (Zhang, Yang and Başar, 2019; Canese *et al.*, 2021; Wong *et al.*, 2021). Partial observability can arise in number of ways; especially in competitive or simultaneous move games. Here, it is common for some information to be kept private to each agent (Wong *et al.*, 2021). For example, in poker the cards held will be private to each agent and in real time action games all the agents will be determining their next move concurrently so they need to make a decision without information about what actions other agents are taking. Additionally, depending on the setting and individual agent sensor configurations involved; cooperative agents may receive different fragments of the information available for a particular state and then need to use some form of communication mechanism (Foerster *et al.*, 2016) to share their knowledge and obtain the full picture. Partial observability also exacerbates the credit assignment problem as it is more difficult to relate a change in the environment to an agent's own action when other agents' rewards and actions are not accessible. Commonly used formal models for handling partial observability are POMDPs for general settings and Dec-POMDPs for cooperative settings, including team games, that have a shared reward function (Oliehoek and Amato, 2016).

One way the partial observability problem can be tackled is by adopting a CTDE structure as demonstrated by Lowe *et al.* (2017) with MADDPG and Foerster *et al.* (2017a) with COMA. Li *et al.* (2019) introduced a minimax extension to the MADDPG algorithm while Foerster *et al.* (2016) also proposed two other potential methods; Reinforced Inter-Agent Learning (RIAL) which is based on Deep Recurrent Q-Learning (Hausknecht and Stone, 2015) and Differentiable Inter-Agent Learning (DIAL) which relies on messaging and gradient sharing between agents. By using some form of communication to exchange information about the environment; agents can compensate for their limited individual knowledge (Foerster *et al.*, 2016). Peng *et al.* (2017) introduced the bidirectionally-coordinated network which uses a vectorised actor-critic framework to facilitate inter-agent communication. Jiang and Lu

(2018) used an approach incorporating an attention mechanism (Vaswani *et al.*, 2017) to enable on-demand and targeted multi-agent communication respectively.

2.8.2 Non-Stationary Environments

In the single-agent domain, it is only the agent's own decisions that change the state of the environment: the agent is always observing the effect of its own actions only. Here, all state transitions can be clearly accredited to the agent (Hernandez-Leal *et al.*, 2017), and everything outside the agent's direct influence is simply regarded as part of the underlying system processes (Hernandez-Leal *et al.*, 2017). Even though the environment itself may be dynamic, the learning problem remains stationary as the state transition and reward functions do not change over time (Hernandez-Leal *et al.*, 2017).

In the multi-agent domain all agents are learning, interacting and changing the environment concurrently. The agents are all continually trying to improve their own policies according to their self-interest (Canese *et al.*, 2021; Wong *et al.*, 2021; Gronauer and Diepold, 2021). It is possible for an agent to receive a different state or observation each time it takes the same action in a particular state depending on the actions of the other agents at that time. The agent becomes unable to tell whether the state transition and associated reward was the result of its own actions or if it was influenced by another agent (Canese *et al.*, 2021; Wong *et al.*, 2021; Gronauer and Diepold, 2021). Consequently previously learned associations between action and rewards may become obsolete and require updating. Therefore the environment becomes non-stationarity from the viewpoint of each agent as they are faced with a moving-target problem; the best policy changes as the other agents' policies change over time (Canese *et al.*, 2021; Wong *et al.*, 2021; Gronauer and Diepold, 2021).

A non-stationary environment invalidates the Markov assumption (that the individual reward and current state depend only on the previous state and action taken) required for most single-agent algorithms to guarantee convergence. Unhandled non-stationarity generally leads to a poorly learned policy for any individual agent (Canese *et al.*, 2021; Wong *et al.*, 2021; Gronauer and Diepold, 2021); additionally in the worst case, it is also possible for the learning process to get stuck in a loop and never properly converge (Canese *et al.*, 2021; Wong *et al.*, 2021; Gronauer and Diepold, 2021). As well as addressing partial observability; communication has also been used to handle non-stationarity. Foerster *et al.* (2016) proposed Deep Distributed Recur-

rent Q-Networks, while Sukhbaatar *et al*. (2016) proposed the CommNet architecture for agent communication, this was later extended to a competitive setting by Singh *et al*. (2019) who proposed the Individualised Controlled Continuous Communication Model. Handling non-stationarity is a large and widely researched topic; further information can be found in the surveys by Papoudakis *et al*. (2019) and Hernandez-Leal *et al*. (2017) which focus solely on examining potential solutions.

2.8.3 Credit Assignment

In fully or partially cooperative multi-agent settings, the (temporal) credit assignment problem (Weiss, 1995; Wolpert and Tumer, 1999) arises because even with a fully observable state space it is not always possible for individual agents to determine if their actions had an overall positive or negative effect on the global reward signal (Minsky, 1961). Additionally, agents may have to handle time delays before a reward signal is received; many subsequent state transitions may have occurred before the consequences of a particular action are revealed (Weiss, 1995; Wolpert and Tumer, 1999). Not being able to identify which agents and actions contributed to the reward outcome makes learning a good policy more difficult and can often lead to a related issue; the lazy agent problem (Sunehag *et al*., 2017). In this issue if one agent has learned a particularly strong policy other agents can be discouraged from further exploration so as to not interfere with its performance (Sunehag *et al*., 2017). The difficulty of the credit assignment problem has been shown to be connected to the frequency that feedback is provided to the agents (Hernandez-Leal *et al*. 2019; Wong *et al*., 2021). If agents have to wait a long time for feedback and many actions/transitions have taken place then this is known as receiving sparse rewards (Hernandez-Leal *et al*. 2019; Wong *et al*., 2021). Conversely if agents receive meaningful feedback on a regular basis (dense rewards), it is easier for them to associate actions to state transitions (Hernandez-Leal *et al*. 2019; Wong *et al*., 2021). A number of approaches exist to address the multi-agent credit assignment problem. For fully centralised, cooperative settings with discrete action spaces, Foerster *et al*. (2017a) proposed 'Counterfactual Multi-Agent Policy Gradients' (COMA). Inspired by the concept of difference rewards (Wolpert and Tumer, 2001) COMA calculates the advantage of a single agents action by comparing the estimated reward for a joint-action to a computed counterfactual baseline. The baseline is used to estimate of what would have happened on average had the agent selected a different action, which allows the system to determine the impact of

a single action (Foerster *et al*., 2017a). More information on COMA is provided towards the end of this chapter in section 2.11. Cohen *et al*. (2021) present a novel architecture based on COMA which uses self-attention (Vaswani *et al*., 2017) instead of a fully connected layer. Referred to as Multi-Agent POsthumous Credit Assignment (MA-POCA); the architecture can efficiently compute counterfactual baselines for both homogeneous and heterogeneous agents. Additionally MA-POCA is able to handle an arbitrary number of agents that share a reward function and are created or destroyed within an episode (Cohen *et al*., 2021). The MA-POCA training algorithm is explored further in chapter 4.

2.8.4 Scalability

In modern reinforcement learning, 'scalability' mainly refers to the extension from single-agent environments to multi-agent environments (Du and Ding, 2020; Canese *et al*., 2021; Zhang, Yang and Basar, 2021; Gronauer and Diepold, 2022) which is an essential factor when trying to solve many real-world problems as these can often involve large numbers of agents (Du and Ding, 2020; Canese *et al*., 2021; Zhang, Yang and Basar, 2021; Gronauer and Diepold, 2022). An issue that affects reinforcement learning in general is the so called 'curse of dimensionality' (Bellman, 1957) which is when a problem has very large state or action spaces (Sutton and Barto, 2018) making it computationally challenging. The problem is exacerbated as further agents are introduced into a scenario, the joint-action space grows, potentially exponentially (Sutton and Barto, 2018). The adoption of DNNs as function approximators can help mitigate the computational complexity but it is still challenging to scale a system to include multiple learning agents; especially when they are heterogeneous (Du and Ding, 2020; Canese *et al*., 2021; Zhang, Yang and Basar, 2021; Gronauer and Diepold, 2022).

Centralised approaches to training often require large amounts of computational resources and memory to be able to work with more than a handful of agents which is not always feasible; even when using a DNN based approach. Researchers have therefore proposed using various different decentralised training schemes such as using independent learners (Tan, 1993) however this approach is known to be weak in dealing with the non-stationarity environment problem (Claus and Boutilier, 1998). A popular approach is to try and reduce the computation complexity via knowledge reuse which can be applied by parameter sharing (Foerster *et al*. 2016) or transfer learning (Taylor and Stone, 2009). If the learning agents are homogeneous then they can make use

of parameter sharing; that is they can share the same weights across multiple neural networks and thus reduce the overall number of inputs that and variables to be trained (Da Silva and Costa, 2019). This boosts the learning process (Gupta *et al.* 2017) and enables more efficient scaling up to an arbitrary number of agents. Parameter sharing has been shown to be effective when multiple homogeneous agents are learning to communicate with one another (Foerster *et al.* 2016; Jiang and Lu 2018; Peng *et al.* 2017; Sukhbaatar *et al.*, 2016).

Transfer learning is where experience already acquired from learning one task may then be applied to a different (but still related) task (Da Silva and Costa, 2019) thus reducing the overall training time required (Omidshafei *et al.* 2019). In a similar vein, learning from demonstrations has also been shown to be an effective way to speed up the learning process (Da Silva *et al.*, 2018). Other proposed solutions are to use curriculum learning (Bengio *et al.*, 2009) and divide the learning process into progressively more complex stages by slowing increasing the number of agents present (Gupta *et al.* 2017; Long *et al.* 2020; Narvekar *et al.* 2020). One other notable approach is by using a mean-field method (Yang *et al.*, 2020) to reduce the overall system complexity by approximating the interactions between a single agent and comparing it with the average impact of its neighbouring agents.

2.9 Imitation Learning

In order to learn the optimal policy, reinforcement learning techniques often requires a large number of environmental interactions (Sutton and Barto, 2018). One way to speed up training is by leveraging existing prior knowledge; it is often more intuitive for a human expert to transfer their knowledge by providing demonstrations of the desired behaviours they wish the learning agents to emulate (Hester *et al.*, 2017; Hussain *et al.*, 2017). The combination of reinforcement learning and imitation learning has been noted as a promising direction for more efficient learning and faster training (Ding, 2021).

Related to reinforcement learning, imitation learning (IL) is not a new concept, one of the earliest examples dates back to 1989 when it was used to train ALVINN; one of the worlds first autonomous vehicles (Pomerleau, 1989). IL is a type of supervised machine learning in which the goal is for the agent(s) to try and learn an optimal policy by reproducing the actions taken in a set of demonstrations, encoded as a series of state-action pairs, provided by an expert teacher (Hussain *et al.*, 2017). In practice,

just learning a direct mapping between states and actions on its own is often not enough to achieve the desired behaviours (Hester *et al*., 2017; Ding, 2021). This may be down to a number of reasons such as poor quality or insufficient demonstrations, as well as minor differences in the environment. Another common issue in IL is the risk of overfitting which can occur if the learning agent only tries to copy the demonstration exactly, instead of learning to generalise to handle unseen situations. Overfitting can however be mitigated by the use of a large set of demonstrations (Hester *et al*., 2017; Ding, 2021), preferably recorded by multiple experts that covers the full scope of the problem domain (Hussain *et al*., 2017; Arulkumaran and Lillrank, 2021).

There are two main types of IL algorithms; Behavioural Cloning and Inverse Reinforcement Learning which are explored in subsequent sections.

2.9.1 Behavioural Cloning

Behavioural Cloning (BC) is the simplest form of imitation learning (Osa *et al*., 2018; Ding, 2021) which reduces the problem to a supervised learning task. BC does not attempt to reason about any effects of taking particular actions (Pomerleau, 1991); it only tries to derive a policy from the provided demonstrations. If the sequence of given sample state-action pairs are sub optimal then the resultant learned policy will also be sub optimal. Any actions recorded that do not related to achieving the goal will still be reproduced (Ding, 2021). Behavioural Cloning is known to fail in cases where there is not enough expert data available (Ross and Bagnell, 2010); however in simpler applications BC on its own can work quite well and is known to be efficient (Ding, 2021); BC is not robust to minor errors or variations encountered during training so in most cases its likely to encounter issues, especially when the training data is not independent and identically distributed (i.i.d) (Ross and Bagnell, 2010). Another issue with BC is that it is unable to generalise to similar tasks (Ding , 2021); if an agent trained with BC makes a mistake and then starts to drift away from the demonstrated states due to an accumulation of errors then it is unable to get back on track as it does not have any defined behaviours for the previously unseen states (Ross, Gordon and Bagnell, 2011; Ding 2021) thus leading to a catastrophic failure of the system (Arulkumaran and Lillrank, 2021). In order to be successful, pure BC approaches require a large amount on high quality demonstration data, covering every possible state (Arulkumaran and Lillrank, 2021).

2.9.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a variant of imitation learning that first attempts to identify and learn the underlying environment reward function from the expert demonstrations (Osa *et al*., 2018; Arora and Doshi, 2018; Ding, 2021). It then uses reinforcement learning methods to maximise the reward function and find the optimal policy. Unlike with behavioural cloning, in IRL compound errors are not an issue because it is able to consider and prioritise or discard entire trajectory rollouts, not just single-time steps (Osa *et al*., 2018; Arora and Doshi, 2018; Ding, 2021). A potential drawback of IRL is that training can take a very long time if the reward function is sparse (Osa *et al*., 2018; Arora and Doshi, 2018; Ding, 2021). In practice there are often more than one reward function that can lead to the same optimal policy, in order to find a unique solution, it was proposed by Ziebart *et al*. (2008) to choose the solution with the largest entropy which has shown improve the overall robustness of the system (Eysenbach and Levine, 2022).

2.9.3 Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) is an occupancy matching algorithm that extends the concept of Generative Adversarial Networks (GANs) (Goodfellow *et al*., 2014) and applies it to imitation learning. GAIL obtains an policy by attempting to minimise the difference between the data provided by the expert and the data generated by the learning agent. As with GANs, the approach uses two neural networks, a generator and a discriminator. The generator network adds new sample data of an agents behaviour into the distribution and the discriminator network then serves as a reward function by determining whether or not the observations it receives are similar to those from the expert data set (Goodfellow *et al*., 2014). This means that learning agents are rewarded for acting in a manner similar to the provided demonstrations (Ho and Ermon, 2016).

More formally, to find a saddle point π, D for the expression:

$$\mathbb{E}_{\pi}\big[log(D(s,a))\big] + \mathbb{E}_{\pi E}\big[log(1-D(s,a))\big] - \lambda H(\pi)$$

GAIL uses DNN function approximation for π and D, fitting a parameterised policy π_θ with weights θ and has a discriminator network: $D_w : S \times A \rightarrow (0,1)$ with weights w. During training GAIL alternates between an Adam optimiser step (Kingma and Ba, 2014) on w and a TRPO (Schulman *et al*., 2015) step on θ.

Algorithm 1 Generative Adversarial Imitation Learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy & discriminator params θ_0, ω_0
2: **for** $i = 0, 1, 2, ...$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from ω_i to ω_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}\Big[\nabla_\omega log(D_\omega(s,a))\Big] + \hat{\mathbb{E}}_{\tau_E}\Big[\nabla_\omega log(1 - D_\omega(s,a))\Big]$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO (Schulman *et al.*, 2015) rule with cost function $logD_{\omega_{i+1}}(s,a)$
6: Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{tau_i}\Big[\nabla_\theta log\pi\theta(a|s)Q(s,a))\Big] - \lambda\nabla_\theta H\pi_\theta$$

7: where $Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{tau_i}\Big[log(D_{\omega_i+1}(s,a)|s_0 = \bar{s}, a_0 = \bar{a}\Big]$
8: **end for**

Ho and Ermon (2016) note that while GAIL is generally sample efficient in terms of the amount of expert demonstrations required, as a it is a model-free method it is not particularly sample efficient in terms of the amount of environment interactions required during training. One potential drawback to using GAIL is that in practice, due to instabilities inherent in adversarial approaches, the training process can be quite complex and brittle (Kurach *et al.*, 2018).

GAIL is able outperform methods such as behavioural cloning by training the reinforcement learning agents to match the provided demonstrations over a longer time horizon and by incentivising agents to select actions which will return them to known states when they encounter new, unseen states (Ho and Ermon, 2016) which helps to avoid catastrophic failure as seen in BC. However to improve training performance Ho and Ermon (2016) recommend the use of BC in combination with GAIL to initialise the policy parameters (Ho and Ermon, 2016).

2.10 Competitive Self-Play

A key element for successfully training adversarial agents in a competitive team-based environment is the use of Competitive Self-Play (CSP). Competitive Self-Play encourages the agents to develop complex behaviours (Bansal *et al.*, 2017; Silver *et al.*, 2018;

Bai and Jin, 2020; Zhong, Zhou and Peng, 2020) by fixing the network of one of the teams for a period while the other team continues to improve then vice-versa which makes the learning process more stable by making the environment stationary (Bansal *et al*., 2017; Silver *et al*., 2018; Bai and Jin, 2020; Zhong, Zhou and Peng, 2020). As training progresses and each team continues to improve it breaks the learning process into a number of distinct phases referred to as an Autocurriculum (Leibo *et al*., 2019; Baker *et al*., 2019). A self-play approach provides the agents with an adaptive sequence of challenges (Leibo *et al*., 2019; Baker *et al*., 2019) because no matter the skill level of a particular agent, the opposing agents in the environment will be of a comparable strength which facilitates more efficient learning (Leibo *et al*., 2019; Baker *et al*., 2019).

2.11 Counter Factual Multi-Agent Policy Gradients

Foerster *et al*. (2017a) presented 'Counterfactual Multi-Agent Policy Gradients' as a potential solution for solving the multi-agent credit assignment problem for settings where all agents are cooperative and share the same goal. COMA is a model-free, on-policy, fully centralised actor-critic method designed to overcome difficulties with coordination that Foerster *et al*. (2017a) encountered previously when trying an approach using independent learners (Tan, 1993). For efficiency, in COMA parameters are shared among agents; there is only one actor and only one critic, which are used by all the agents present. As COMA is centralised, only the actor is needed for execution and only the critic is required during the learning process. The critic can access either the full global state s or the joint action-observation histories τ for partially observable environments. During learning, each actor only uses its own local action-observation history τ^a so they will still be able to manifest different behaviours. Instead of each actor following an estimated error gradient from a critic where the Temporal Difference (Sutton and Barto, 2018) error only considers the overall global rewards and does not consider how any one particular agents' actions contribute to that reward like so:

$$g = \nabla_{\theta\pi} log\pi(u|\tau_t^a)(r + \gamma V(s_{t+1}) - V(s_t))$$

COMA is influenced by the concept of difference rewards (Wolpert and Tumer, 2001) and calculates a separate counterfactual baseline for each agent $A^a(s, u^a)$ that allows it to reason about what would happen if a different action had been selected and therefore

workout an agents individual contribution to the global reward. This approach does not require knowledge of the reward function or any form of default action (Foerster *et al.*, 2017a). COMA estimates Q-values for the joint action $\mathbf{u}$ influencing the central state s to learn a centralised critic $Q(s,\mathbf{u})$. An advantage function is computed for each agent a which compares the Q-value for the current action u^a to a counterfactual baseline that discounts u^a, while fixing other agents' actions $\mathbf{u}^{-a}$ in place (Foerster *et al.*, 2017a):

$$A^a(s,\mathbf{u}) = Q(s,\mathbf{u}) - \sum_{u'^a} \pi^a(u'^a|\tau^a) Q(s,(\mathbf{u}^{-a},u'^a))$$

Algorithm 2 Counterfactual Multi-Agent (COMA) Policy Gradients

Initialise θ_1^c , $\hat{\theta}_1^c$, θ^π
for each training episode e **do**
  Empty buffer
  for $e_c = 1$ **to** $\frac{BatchSize}{n}$ **do**
    $s_1 =$ initial state, $t = 0$, $h_0^a = 0$ for each agent a
    while $s_t \neq$ terminal **and** $t < T$ **do**
      $t = t + 1$
      for each agent a **do**
        $h_t^a = Actor(o_t^a, h_{t-1}^a, u_{t-1}^a, a, u; \theta_i)$
        Sample u_t^a from $\pi(h_t^a, \varepsilon(e))$
      end for
      Get reward r_t and next state s_{t+1}
    end while
    Add episode to buffer
  end for
  Collate episodes in buffer into a single batch
  {processing all agents in parallel via a single batch}
  for $t = 1$ **to** T **do**
    Batch unroll RNN using states, actions and rewards
    Calculate TD(λ) targets h_t^a using $\hat{\theta}_i^c$
  end for
  for $t = T$ **down to** 1 **do**
    $\Delta Q_t^a = y_t^a = Q(s_t^a, \mathbf{u})$
    $\Delta\theta^c = \nabla_{\theta^c}(\Delta Q_t^a)^2$ {calculate critic gradient}
    $\theta_{i+1}^c = \theta_i^c - \alpha\Delta\theta^c$ {update critic weights}
    Every C steps reset $\hat{\theta}_i^c = \theta_i^c$
  end for
  for $t = T$ **down to** 1 **do**
    $A^a(s_t^a, \mathbf{u}) = Q(s_t^a, \mathbf{u}) - \sum_u Q(s_t^a, u, \mathbf{u}^{-a})\pi(u|h_t^a)$ {calculate COMA}
    $\Delta\theta^\pi = \Delta\theta^\pi + \nabla_{\theta^\pi} log\pi(u|h_t^a)A^a(s_t^a, \mathbf{u})$ {accumulate actor gradients}
  end for
  $\theta_{i+1}^\pi = \theta_i^\pi + \alpha\Delta\theta^\pi$ {update actor weights}
end for

Chapter 3

Related Work

In sports, much historical research focus has been on the recognition and classification of observed team behaviours. This includes sports such as American football (Intille and Bobick, 1999), baseball (Tango, Lichtman and Dolphin, 2007), basketball (Bhandari *et al*., 1997; Jug *et al*., 2003) and Robocup football simulations (Riley and Veloso, 2002; Kuhlmann *et al*., 2006) as well as the real thing (Tuyls *et al*., 2020). There is a long and rich history of academic interest in American football, from its beginnings in works such as *Operations Research on Football* (Carter and Machol, 1971) and *The Hidden Game of Football* (Carroll, Palmer and Thorn, 1988) to more recent activity such as the joint NFL and AWS 'Digital Athlete' injury prediction and detection programme (NFL, 2022). Over the years a number of researchers have investigated the application of various machine learning methods to the domain of American football; such as automatically detecting plays and player actions from raw video sources (Hess, Fern, and Mortensen, 2007; Hess and Fern, 2009; Stracuzzi *et al*., 2011), learning strategies by observation (Li *et al*., 2009) as well as opponent modelling (Laviers *et al*., 2009).

After a brief discussion of an American football learning environment in section 3.1 This chapter then presents a rundown of prior research on American football from a reinforcement learning perspective in section 3.2. Section 3.3 contains an overview of related work in imitation learning. Finally, section 3.4 provides a review of literature detailing the application of reinforcement learning methods to sports video games.

3.1 American Football Simulation Environments

There is a lack of freely available simulators suitable for empirically researching algorithmic performance in this domain. The only example uncovered during this dissertation was *Rush 2008* (Molineaux, Aha and Sukthankar, 2009), which is a heavily modified version of a previously released open source hobbyist game. Molineaux, Aha and Sukthankar (2009) created the *Rush 2008* simulator to investigate the effectiveness of using plan recognition techniques to identify the defensive team's strategy in an American football scenario and to improve the results of their case-based Q-learning algorithm. The environment was a stochastic, partially observable, simplification of an American football game using only eight players for each team. Only the quarterback agent was controlled by the learning algorithm, the other agents on offense and all defensive agents instead used a semi-random predefined simple heuristic for control. The problem investigated was learning how best to control the behaviour of a quarterback agent in a pass play situation, where the goal is to maximise the overall yardage gained on a repeated single play. Every play, the defense randomly selected one of five given strategies that began from the same in initial formation (Molineaux, Aha and Sukthankar, 2009).

The adversarial action selection task considered in this paper had a large state space (4.3^{109} states), which again was intractable for a standard Q-learning algorithm on the hardware available at the time. The authors thus sought methods to reduce the state dimensionality under consideration and also used an approximation of the Q-value. The paper presents the 'Case-Based Q-Lambda with Plan Recognition' (CBQL-PR) algorithm which was based on the $Q(\lambda)$ algorithm (Sutton and Barto, 2018); it used a set of case bases to approximate the Q-function and then an Expectation-Maximisation clustering algorithm to add opponent plan information to the state. CBQL-PR periodically selects an action that either maximises the expected return (exploiting learned knowledge), or improves its knowledge of the value space (exploring the environment) so as to maximise the long-term reward. Unfortunately the *Rush 2008* simulator is no longer available which is one of the motivating factors for undertaking this dissertation project.

3.2 Research on American Football

Historically, much of the research on American football focuses on predicting the outcome of a match. For example Min *et al*. (2008) used Bayesian inference coupled with rule-based reasoning to make predictions, while Purucker (1996) and Kahn (2003) both applied artificial neural networks to make NFL game predictions; Although an issue is that both authors only trained on a total of two weeks of games which was not enough data to be truly accurate. The finer-grained problem of situational play calling has been also been studied by a number of researchers. For example, Patek and Bertsekas (1998) formulated the task of play selection in American football as a shortest path MDP optimisation problem; more specifically, the authors focused on the decision making problems faced by a quarterback when attempting to maximise the score gained on an abstracted version of an infinitely long offensive drive (ignoring any absorbing/termination states). The authors considered a simplified version of American football which used a discretisation of the playing field by dividing the field into a number of squares, giving approximately $15,000$ position states. At each state, the quarterback must pick one of four play options: run, pass, punt, or kick a field goal. As they only used a fairly small number of states the authors were able to compute an optimal solution numerically as a baseline for comparison against various methods of approximation including the use of a multi-layer perceptron (MLP) trained by back-propagation which they noted as being the most effective; demonstrating that a neural network based approach for approximating the solution to MDPs showed promise even back then.

Goldner (2012) also created a similar Markov model of American Football which was used to simulate offensive drives with historical statistical information as the input. In this paper a football drive is coarsely discretised to give 349 states for which the calculated absorption probabilities lead into an expected points model, which was then used to measure the efficiency of individual plays. Jordan, Melouk and Perry (2009) approached play calling from a game theoretical perspective. They specifically treated the sport as a two-player zero-sum game where one players loss is the others gain and the players take actions according to the rewards they expect to gain from their behaviour. In 2010, Laviers and Sukthankar (Laviers and Sukthankar, 2010) used the *Rush 2008* football simulator to demonstrate a Monte Carlo Upper Confidence Tree method for generating new plays. In general, the analysis of fourth down attempts and related play-calling has proven to be a popular topic of investigation for academic

research (Romer, 2006; Alamar, 2010; Yam and Lopez, 2019).

A more recent example of using a reinforcement learning approach to play calling is provided by Biro and Walker (2021). The authors argue that there is now enough play-by-play analytical data available such that each individual drive can be treated as a self-contained sub unit within the larger game (Biro and Walker, 2021). In this work a drive is viewed as a sequence of states, where a state can be characterised by the current down, current distance needed for a first down,and the current line of scrimmage. The authors posit that the knowing the rewards associated with reaching a terminal state as well as having the various transition probabilities are enough to select optimal actions (Biro and Walker, 2021). The authors also discuss modifications to the analysis that would lead to specific desired outcomes such as scoring a set number of points in a particular time frame or running out the game clock. As opposed to a simulation based approach as used in this dissertation project, the paper instead details data driven and machine learning techniques for optimal prediction and action selection (Biro and Walker, 2021). The authors then analyse the performance of teams as a percentage of their actions as they relate to the optimal choices.

It should be noted that, aside from American football, various reinforcement learning techniques have long been applied to the study and analysis of a number of sports, both real and simulated. This includes basketball (Cervone *et al*., 2016; Sandholtz and Bornn, 2020), ice hockey (Liu and Schulte, 2018; Luo *et al*., 2021; Routley and Schulte, 2015) and association football (Fernandez, Bornn and Cervone, 2021; Hirotsu and Wright, 2002; Liu *et al*., 2020).

3.2.1 Valuing Individual Player Actions

There have been a number of studies into the evaluation of individual player actions and how they contribute to a sports teams overall performance, which is closely related to the credit assignment problem in MADRL. For example, Routley and Schulte (2015) applied the Stochastic Game (Shapley, 1953) formalism to develop a context-aware approach to valuing player actions, positioning, and overall team performance in ice hockey. The applied model had over $10,000$ states and 148 potential actions. Dynamic Programming (Sutton and Barto, 2018) is then used to learn value functions that quantify the impact of actions on goal scoring. In 2019, Burke proposed DeepQB (Burke, 2019), a deep learning approach to the evaluation of quarterback decision-making and overall performance. DeepQB makes use of two seasons worth of tracking data gained

from American Football played at the professional level. The system uses a deep neural network trained on this data for predicting which receiver the quarterback should be targeting to maximise the chance of completion. Quarterback passing skills are then assessed in isolation from the offensive players by comparing metrics of actual play success to the metrics of success as predicted by the model. Burke reports that when passes targeted the model's predicted receiver, passes are completed 74% of the time, compared to 55% when the quarterback targeted any other receiver (Burke, 2019).

In American football, measuring the performance of so-called 'non-skilled' players such as offensive lineman is an oft underexplored area. To address this, Alamar and Weinstein-Gould (2008) proposed an original model for exploring the relationship relationship between the abilities of an individual offensive lineman and the effectiveness of the passing game. The performance of each lineman was recorded on every pass play from the first three games of the 2007 NFL season for seven teams, as well as the amount of time the quarterback was given to make a throw (Alamar and Weinstein-Gould, 2008). The model uses a series of regressions to determine the likelihood of a specified offensive lineman successfully blocking his opponent, in relation to the time it took for the quarterback to throw the ball. The results of these regressions were then used to simulate the impact of different linemen on the passing game and the proposed model is able to estimate the correlation between successful blocking and pass completion rate (Alamar and Weinstein-Gould, 2008).

More recently and in a similar vein, Yurko, Ventura and Horowitz (2018) presented a 'Wins Above Replacement' (WAR) framework for player evaluation; wherein an individual player's statistics are compared to a fictional baseline 'replacement' player which represents the league average at a particular position (Yurko, Ventura and Horowit, 2018). The paper also considers the division of credit on a per play basis; allocating a portion of a play's value to each player on the field (Yurko, Ventura and Horowit, 2018).

3.3 Imitation Learning Approaches

Researchers have also leveraged various imitation learning (IL) techniques to simulate human behaviour as mimicking the past movements of humans has been shown to provide more natural looking behaviours compared to a rule-based programming approach (Hussein *et al.*, 2017). For example, Seidl *et al.* (2017) proposed a sketching tool for basketball play-by-play analysis, where they also use imitation learning

to synthesise NBA defenses. Takayanagi *et al*. (2022) propose a stochastic inverse reinforcement learning (IRL) algorithm for tackling sports games such as American football where the true reward functions are often hard to define. In their paper, the simulation results from an maximum entropy version of IRL are then compared to those from a mathematical two-stage stochastic optimisation process. The authors propose a Markov model of football where agents learn a reward function from sampling an expert demonstration dataset constructed from play-by-play information gathered during the 2017 NFL season. A Mixed Density Network (Bishop, 1994) is then used for estimating the next state from a probability distribution when given an input state and action. The results presented show that the agents were indeed able to learn the reward function; evidenced by similar trajectory mappings between the expert demonstration and the learning agent (Takayanagi *et al*., 2022).

Liu and Schulte (2021) combine inverse reinforcement learning and Q-learning to create a novel RL-based player ranking method that treats a proprietary play-by-play dataset of professional play as expert demonstrations when are then used for learning an underlying reward function. Their paper models ice hockey as a semi-episodic task (Sutton and Barto, 1998) and addresses the sparse rewards issue in reinforcement learning by applying IRL with domain specific knowledge to recover reward from the underlying game dynamics (Liu and Schulte, 2021). Based on the recovered reward function and subsequent Q-value calculations, the authors created a context-aware performance metric that takes all player actions into account. The authors note that while this approach is especially effective in low-scoring games such as football or ice hockey; it can be applied to any sport that can be formalised via a Markov model (Liu and Schulte, 2021).

Motivated by the original 'ghosting' work reported by Lowe (2013) about the use of analytics within the Toronto Raptors basketball team, Le, Yue and Carr (2017) presented an automatic data-driven approach applied to tracking data captured over a full season from a professional football league. This allows their system to be able to estimate how different teams might approach various defensive situations, where players may occupy a number of different roles, without the need to manually annotate a huge volume data as in the original work (Lowe, 2013); indeed the authors report that their system can be trained from scratch in just a few hours (Le, Yue and Carr 2017). To address the problem of learning from demonstrations for a group of coordinating agents, the paper proposes a joint approach that simultaneously learns a latent coordination model along with the individual policies for each agent. The proposed training method

uses a semi-supervised framework that integrates unsupervised latent, structure learning with conventional imitation learning. To make learning tractable and efficient, the paper uses an alternating optimisation method that switches between the latent structure model and individual policies (Le, Yue and Carr, 2017).

3.4 Reinforcement Learning in Team Video Games

Aside from working with the real world equivalents, reinforcement learning methods have also been applied to a number of sports video games and simulations. Zhao *et al.* (2019) describe a work-in-progress hierarchical AI solution for an ice hockey video game. The stated goal of the paper is to train agents that play like humans do; both in terms of specific tactical play and more general strategies. The authors specifically consider the problem of making the agents fun and engaging to play against; not just superhuman and unbeatable (Zhao *et al.*, 2019). At the lower level, agents need to take actions that are human-like and believable; while at a higher level the agents should be seen to be following some form of plan. The papers proposed solution uses a combination of imitation learning and reinforcement learning across multiple levels of abstraction and takes a curriculum learning approach (Bengio *et al.*, 2009) during training to achieve faster convergence. The paper reports on experimenting with multiple training algorithms including PPO (Schulman *et al.*, 2017), Deep Q-Networks (DQN) (Mnih *et al.*, 2013; 2015) and RAINBOW (Hessel *et al.*, 2017). The authors report that the RAINBOW algorithm actually failed to train agents in this environment; and that they found PPO (Schulman *et al.*, 2017) converged considerably faster than DQN (Mnih *et al.*, 2013; 2015). The authors also report that sparse rewards alone did not provide a strong enough learning signal; so they had to apply more and more refined reward shaping in order to get the agents to exhibit the desired offensive and defensive behaviours. The authors believe that end-to-end model-free RL is unlikely to provide human-like and believable agent behaviour (Zhao *et al.*, 2019) and so the proposed solution therefore resorted to using a hierarchical approach coupled with expert demonstrations.

Jia *et al.* (2020) introduced *Fever Basketball*, a novel asynchronous open-source reinforcement learning environment where agents are trained to play the game of basketball. *Fever Basketball* is a complex and challenging environment that supports multiple characters and positions; and also features both single-agent and multi-agent modes. An important difference between *Fever Basketball* and most other reinforce-

ment learning environments is that the execution time of actions is not fixed and varies between the different player characters available (Jia *et al*., 2020). The paper evaluates several commonly used multi-agent algorithms, featuring both independent and joint-action learners, in game scenarios of varying difficulty. The authors also propose two heuristic methods and an integrated curricula training framework to better handle the extra non-stationarity arising from the asynchronous actions in this class of problem. As the asynchronised actions in *Fever Basketball* are still primitive actions and the decision-making still focuses on a low level of granularity, it presents a difficult learning problem (Jia *et al*., 2020). The results presented in the paper show that the game is indeed challenging for MADRL methods, especially for methods that use a centralised training approach (e.g. Sunehag *et al*. 2018; Rashid *et al*. 2018). and that existing MADRL algorithms failed to entirely solve the environments asynchronicity. The authors note however that the *Fever Basketball* agents were still found to reach a win-rate of approximately 70% during a 300 day online evaluation period against human players.

A concrete example of RL applied to a AAA commercial title is Ubisoft's *Roller Champions* (Iskander *et al*., 2020). Built in *Unity*, *Roller Champions* is a 3 on 3 competitive sports game played in an oval-shaped arena. Iskander *et al*. (2020) demonstrated that RL can be a powerful and practical AI tool for creating modern, non-trivial video games. Implemented via the *Unity ML-Agents Toolkit* (Juliani *et al*., 2018), the author's state that their goal was to develop an AI that is not just able to win, but that would also be "fun to play with and against" (Iskander *et al*., 2020, para. 10), i.e. the win-rate is secondary to the overall player experience (Iskander *et al*., 2020). The policy network used was quite small, containing only 3 hidden layers of 512 neurons each which helps make the system highly efficient in terms of memory and CPU usage. The agents make simple observations derived directly from the game state; indeed the whole system was designed with agile, fast-paced development in mind, taking between 1–4 days to train a new model using PPO (Schulman *et al*., 2017) and Competitive Self-Play (Bansal *et al*., 2017) after any changes are made. The paper states that by keeping the rewards simple, the authors found that the agents were able to more readily adapt to changes in gameplay mechanics (Iskander *et al*., 2020); the paper notes this also aided game balance by identifying any over or under-powered mechanics. The authors observe that despite the simplicity of both the policy network and reward function, the agents were able to develop sophisticated and coordinated behaviours suitable for a fast-paced online multi-player game (Iskander *et al*. 2020).

Won, Gopinath and Hodgins (2021) developed a learning framework that generates low-level control policies for physically simulated athletes who have many degrees-of-freedom and are actuated by joint torques. The authors developed a policy model which is based on an encoder-decoder structure that uses an auto-regressive latent variable, and a mixture-of-experts decoder (Jacobs *et al*., 1991). Won, Gopinath and Hodgins (2021) proposed a two-step approach that first learns an imitation policy and then transfers it into policies suitable for two-player competitive sports. The proposed approach uses a prerecorded demonstration dataset that contains the basic skills of a symmetrical sport, such as boxing or fencing, performed by a single expert without an opponent present. An imitation policy is then learned by using single-agent deep reinforcement learning. This policy is then used as the basis for both agent's competitive policies via transfer learning and then further trained using MADRL methods. To learn the imitation and competitive policies the proposed approach uses a decentralised and distributed variant of PPO (Schulman *et al*., 2017), called DD-PPO (Wijmans *et al*., 2019). In DD-PPO, to reduce communication overhead each machine in the cluster first computes the gradients which are then shared over the network to compute the average. In their experiments, Won, Gopinath and Hodgins (2021) used twelve machines for training, and each machine had two CPUs. The paper reports that it took around a day each to perform the imitation learning phase for both sports, and then at least three days of RL training on top of that before the agents could learn both tactical and natural-looking behaviours. Finally the paper notes that, due to the computational complexity and the vast number of samples required during training the proposed system currently struggles to scale up to handle sports that involve multiple players (Won, Gopinath and Hodgins, 2021).

Robotic Soccer is a classic testbed environment for artificial intelligence agent research and reinforcement learning is no exception. Kalyanakrishnan, Liu and Stone (2007) presented the *Half-Field Offense* (HFO) subtask of RoboCup simulated soccer (Kitano et al., 1995) formulated as a reinforcement learning problem. The HFO environment poses a challenging multi-agent reinforcement learning problem as it features sparse rewards, a continuous state space and multiple agents with potentially noisy actions. The papers proposed solution is to use inter-agent communication for facilitating information sharing among the robot agents, which enables more frequent and reliable updates. This work predates the deep learning paradigm and so each agent uses particular type of a neural network called a Cerebellar Model Arithmetic Computer (CMAC) (Albus, 1975) for approximating the Q-function and then uses SARSA (Rummery and

Niranjan, 1994; Sutton and Barto, 2018) as its training method. Unlike most neural networks, CMACs cannot represent arbitrary non-linear functions and so are only able to use predefined high-level actions (Albus, 1975).

Hausknecht *et al*. (2016) revisited the *Half-Field Offense* domain and released it as an open source platform for experimentation that supports both single and multi-agent reinforcement learning algorithms. Formally represented by Parameterised Action Markov Decision Processes (PAMDP) (Masson and Konidaris, 2015) and in contrast to the previous work (Kalyanakrishnan, Liu and Stone, 2007), this updated version of HFO instead uses a parameterised low-level action space where agents first need to decide which (discrete) action to select, and then consider exactly what (continuous) parameters should be used. Additionally agents can choose whether to receive a low or high level representation of the environment state which affects the difficulty of the task. The paper's proposed baseline solution makes use of Tile Coding (Sutton and Barto, 2018) in order to discretise the state space and again uses SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 2018) as the training algorithm.

A recent example of another robot football simulation environment is provided by Hong *et al*. (2021) who presented the AI World Cup; a set of AI competitions based around various football mechanics. Built on top of the Webots Robot Simulator (Michel, 2004), the AI World Cup is a platform for simulating robotic soccer. Components include AI Soccer which is intended as a more beginner friendly environment for conducting multi-agent research. The authors state that the primary goal of developing the AI Soccer challenge was to simplify the game so that participants could focus on the core challenge of creating complex multi-agent cooperative/competitive behaviours among two teams of five two-wheeled robotic agents instead of having to worry about the various low-level actuator adjustments required to provide physical locomotion (Hong *et al*., 2021). The simulation environment supports both a coordinate-based representation of game state as well as a raw visual image representation. The resulting simulated soccer game is still a very challenging environment for AI algorithms as it is highly dynamic and also simultaneously both cooperative and competitive. The AI World cup has now been running for a number of years; in 2018 the winning team used a DRL based approach which was trained via PPO (Schulman *et al*., 2017) and used multibatch experience replay (Han and Sung, 2017) to choose between sixteen discrete actions. Each robot attempts to maximise their individual rewards based on the distance to their team mates, the ball position and the overall game score. This information is then stacked across four consecutive frames to use as the input state for

learning. In 2019, there were another two competitions held and the winners of both used a deep reinforcement learning approach. The first winning team combined Deep Q-Learning (Mnih *et al.*, 2013; 2015) with a set of knowledge-based rules to decide the best set of parameters to use; while in the second competition the winning team used the multi-agent deep deterministic policy gradient (MADDPG) algorithm (Lowe *et al.*, 2017).

Liu *et al.* (2022) developed a method for training teams of physically simulated humanoid agents to play a simplified 2 on 2 football game in a realistic virtual environment. The described approach combines Imitation Learning, single- and multi-agent reinforcement learning as well as population-based training, it also uses of transferable representations of behaviour for enabling decision making at different levels of abstraction (Liu *et al.*, 2022). As opposed to the high or mid-level action abstractions used by the bulk of prior research presented for the team sports domain; in this work the agents have a fully articulated, physically simulated humanoid body requiring the agents to first learn low-level fine motor-control to produce human-like movements such as walking, running or turning (Liu *et al.*, 2022). The agents then need to acquire mid-level domain specific skills such as dribbling and shooting and finally, they must develop an awareness of the other agents present and learn to play as a coherent team by factoring in different levels of spatial and temporal abstraction when coordinating behaviours (Liu *et al.*, 2022). The provided simulated environment thus provides a number of very challenging problems that need to be addressed, including behaviour specification, credit-assignment and the ever present exploitation-exploration dilemma (Liu *et al.*, 2022). The proposed solution uses a combination of imitation learning and CSP Autocurricula (Bansal *et al.*, 2017; Baker *et al.*, 2019) in order to provide prior knowledge where possible and also enable the discovery of complex emergent solutions that would otherwise be difficult to specify through reward shaping alone. Similar to the approach used by Jaderberg *et al.* (2019); the solution presented also performs outer-loop optimisation with Population-Based Training (Jaderberg *et al.*, 2017) using Nash Averaging (Balduzzi *et al.*, 2018) as the selected fitness measure. The inner loop then uses a DRL-based optimiser. The training approach described in the paper has significant computational demands; learning is performed on a central 16-core TPU-v2 machine where one core is dedicated per agent in the population. Policy interactions are spread across large pool of 4096 CPU workers while actual model inference when playing a match is performed via requests to a cluster of 128 dedicated servers. The chosen architecture also supports inference batching and per-

episode re-sampling which the authors claim offers greatly improved efficiency over previous systems (Liu *et al*., 2022). Despite the powerful compute hardware available, it is notable that an independent, rather than joint-action, approach to training was still used as well as that the system was restricted to two agents per team when playing a match. This suggests that the computational requirements for a full 11 on 11 game using joint-action learning derived coordinated set pieces are still too high for the current generation of machines. Finally authors also note that although they focused on a football domain, the underlying principles presented in their paper are general (Liu *et al*., 2022) and should be applicable in other domains such as other team sports or perhaps collaborative work scenarios.

Kurach *et al*. (2020) introduced *Google Research Football* (GRF), a novel open-source reinforcement learning environment where agents learn to play football in an advanced, physics-based 3D simulator which is modelled on popular football video games. In GRF, agents have to learn how control their teams players, how to pass between them and how to overcome the other teams' defence in order to score goals and thus win the game. Unlike the work by Liu *et al*. (2022); the GRF environment focuses on high-level actions instead of low-level motor control. GRF provides a challenging RL problem as success at football requires a balance between short-term control and high level strategy while incorporating learned concepts such as passing, dribbling and shooting the ball (Kurach *et al*., 2020). The environment also supports a curriculum learning approach by including a set of diverse reinforcement learning scenarios that get progressively harder. The paper also provides a set of comparison benchmark results on tasks of varying difficulties for several popular algorithms; PPO (Schulman *et al*., 2017), IMPALA (Espeholt *et al*., 2018) and Ape-X DQN (Horgan *et al*., 2018). The authors reported that training was able to hit 24 Million steps per day; PPO was trained using sixteen parallel worker processes on a single machine, while IMPALA and Ape-X DQN were both run on a distributed cluster with 500 and 150 actors respectively. It is important to note the approaches described in this paper were not true multi-agent reinforcement learning solutions as although there are 22 players on the pitch in GRF, only a single agent is controlled by the AI system at a time (Kurach *et al*., 2020). Making use of a modified version of GRF, Huang *et al*. (2021) presented 'TiKick', an end-to-end AI System for simultaneously controlling all 22 players on the pitch. The proposed solution uses demonstrations provided by an earlier solution, which was designed to control a single agent via DRL and makes use of imitation learning and self-play. The demonstrations are then compiled into an expert replay

dataset which is then fed into a novel a distributed learning system; implemented via a series of deep recurrent neural networks and using behaviour cloning techniques and a carefully hand-crafted algorithm it is able to build a pretrained multi-agent model from the single-agent dataset. This offline model was then used as the starting point for performing further training in a series of multi-agent scenarios in order to obtain the final model. The experimental results presented in the paper show that using transfer learning via an offline pretrained model can accelerate the reinforcement learning process (Huang *et al*., 2021).

Chapter 4

Methodology

The primary goal of this project is to investigate how reinforcement learning methods could be leveraged to train agents to play a (simplified) variant of American football and evaluate the resulting behaviours. This chapter provides information on the proposed approach below in section 4.1 and then discusses the architecture, algorithm and methods used in section 4.2. Section 4.3 details the process of creating the learning environment, the approach to validation and verification is described in section 4.5 and finally section 4.6 touches on some specific implementation details of note.

4.1 Investigation Of Potential Methods

To create game agents capable of playing a game of American football, the potential solution investigated in this dissertation is to use a combination of reinforcement and imitation learning methods; however to do this required the creation of a new learning environment as no there were no open source American football games or simulators available for conducting research with. Although it is possible to create a reinforcement learning environment from scratch, for this project it made more sense to leverage the power of existing simulation engines and reinforcement learning libraries and tools to be able to get up and running as quickly as possible. Development then proceeded using an agile iterative prototyping approach to incrementally add additional functionality.

The selected software for this project is the *Unity* game engine and the *ML-Agents Toolkit* plugin (Juliani *et al.*, 2018). *ML-Agents* provides a novel multi-agent architecture for COMA (Foerster *et al.*, 2017a), detailed further in section 2.11, which is designed to train groups of agents and which provides functionality for training coop-

erative behaviours.

4.2 Unity ML-Agents Toolkit

Unity is a state-of-the-art, cross-platform general development platform (Juliani *et al.*, 2018) with a workflow beneficial for single developers. Modern versions of *Unity* are able to support realistic real-time 2D/3D rendering and physics simulations. The underlying engine is not restricted to any particular game genre or simulation type (Juliani *et al.*, 2018), and *Unity* has seen widespread adoption in a number of different sectors such as gaming, robotics, engineering, as well as in the automotive and film industries (Juliani *et al.*, 2018). As of 2020, software built with *Unity* was running on more than 1.5 billion devices (Juliani *et al.*, 2018). *Unity* exposes a rich scripting system via the C# language. This system simplifies the creation of multi-agent environment and enables almost any imagined problem scenario to be defined and dynamically controlled. Additionally, as the rendering and physics systems are decoupled, updates can be made in an asynchronous manner which means it is possible to increase the speed of a simulation without needing to also increase the rendering frame rate. If rendered visuals are not required, it is also possible to run simulations in a headless mode (Juliani *et al.*, 2018).

To support research into improved artificial intelligence behaviours via the *Unity* platform, the developers have released the *Unity Machine Learning Agents Toolkit* (Juliani *et al.*, 2018) which is an open-source editor plug-in that facilitates the use of games and simulations as learning environments in which to train intelligent agents (Juliani *et al.*, 2018).

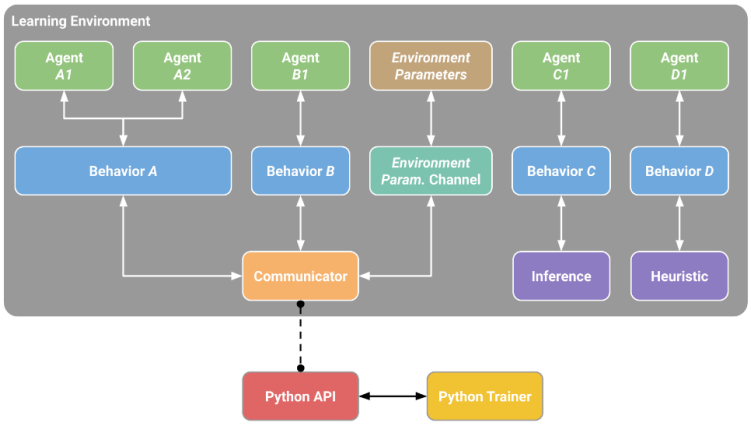

Figure 4.1: Unity ML-Agents Toolkit 2.0 system architecture (Unity Technologies, 2021)

As shown in figure 4.1 above; the *Unity ML-Agents Toolkit* consists of several key components, namely the learning environment, a low-level Python API, an external communicator as well as the various bundled algorithm implementations and a special OpenAI gym (Brockman *et al*., 2016) wrapper. The learning environment consists of the scene geometry and all the various dynamic objects and scripts contained within. The *ML-Agents Toolkit* SDK allows any scene to be converted into a learning environment where make observations, perform actions, and gain new behaviours. As of version 2.0, the *ML-Agents Toolkit* provides a number of state-of-the-art reinforcement learning algorithm implementations such as Soft Actor-Critic (Haarnoja *et al*., 2018), Proximal Policy Optimisation (Schulman *et al*., 2017) and a novel MADRL architecture called MA-POCA (Cohen *et al*., 2021).

The primary reason the *Unity Ml-Agents Toolkit* was selected over alternatives such as using the *Unreal Engine* with the *MindMaker* plugin (Krumins, 2020) is that the *ML-Agents Toolkit* has been under continual development for a number of years and is now a mature and stable software library which has been used for a number of projects such as *Arena* for *Unity* by Song *et al*. (Song *et al*., 2019), the *Obstacle Tower Challenge* (Juliani *et al*., 2019) and at least one AAA commercial game title; *Roller Champions* (Iskander *et al*., 2020).

4.2.1 MA-POCA: Multi-Agent POsthumous Credit Assignment

An extension of COMA (Foerster *et al*., 2017a), which is detailed in section 2.11, MA-POCA (Cohen *et al*., 2021) is a model-free, on-policy algorithm designed to handle the problem of credit assignment in multi-agent settings where an agent may or may not be removed from the environment before the end of the episode. Conceptually similar in a way to death-masking is used by Yu *et al*. (2021) but without the use of absorbing states, MA-POCA is able to handle varying numbers of active agents in the environment. Agents that are removed mid-episode are still able to learn how their individual actions affect the group reward; allowing agents to learn behaviours that will benefit the group as a whole, even if the outcome is negative for an individual.

Falling inside the CTDE framework, MA-POCA uses a centralised actor-critic architecture for training, it uses a value function for estimating the expected discounted return for a team of agents. To handle a changing number of agents the algorithm uses a self-attention mechanism (Vaswani *et al*., 2017) over all the agents that are still active in the critic network.

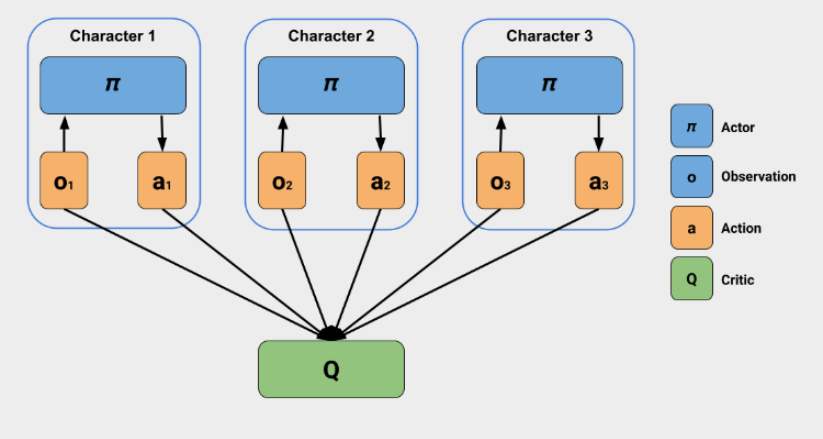

Figure 4.2: Unity ML-Agents Toolkit 2.0 system architecture (Berges *et al.*, 2021)

The self-attention mechanism (Vaswani *et al.*, 2017) enables the architecture to efficiently compute counterfactual baselines for groups of both homogeneous and heterogeneous agents. Unlike the original implementation of COMA (Foerster *et al.*, 2017a) which assumes discrete actions only, MA-POCA can handle both continuous and discrete action spaces. The self-attention mechanism (Vaswani *et al.*, 2017) used in MA-POCA considers distinct observation spaces as entities for encoding. Two agents i, j that share the same observation space $O_i = O_j$, will have their observations embedded in the same encoder, otherwise they are embedded with different encoders. Observations and observation-action pairs are considered separate entities; the observations and actions are concatenated before being embedded. More formally, $g_i : O_i \rightarrow E$ is an encoding network for observations $o_i \in O_i$ where E is the embedding space and if $O_i = O_j$, then $g_i = g_j$. Separate DNNs are used to approximate the value function and counterfactual baseline.

4.2.2 MA-POCA Value Function

Instead of a separate vector containing all the global information or the joint observation information of all agents $o_t \in O$ MA-POCA considers only the joint observations of *active* agents in the environment as its approximation of the global state. The observations of all active agents are encoded:

$$g_i(o_t^i)1 \leq i \leq k_t$$

Where k_t represents the number of active agents at time step t. The encoded observations are passed through an Recurrent Self-Attention (RSA) block. Each agent's embedded observations are normalised using Layer Normalisation (Ba, Kiros, and Hinton,

2016) and then further embedded into a query format: Query : Q, Key : K and Value: V using a fully connected network (Baker *et al*., 2019) which is then fed into the scaled dot-product multi-head attention (Vaswani *et al*. 2017). The original embedded observations are summed together with the processed ones, normalised again, then averaged together to form the final fixed size embedding. Formally, the centralised state value function (with parameters denoted by ϕ) has the form:

$$V_\phi(RSA(g_i(o_t^i)_{1\leq i\leq k_t})) \tag{4.1}$$

and is trained with $TD(\lambda)$ (Sutton and Barto, 2018):

$$J(\phi) = (V_\phi(RSA(g_i(o_t^i)_{1\leq i\leq k_t})) - y^{(\lambda)})^2 \tag{4.2}$$

where

$$y^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)} \tag{4.3}$$

and

$$G_t^{(n)} = \sum_{l=1}^{n}\gamma^{l-1}r_{t+l} + \gamma^n V_\phi(RSA(g(o_{t+n}^i)_{1\leq i\leq k_{t+n}})) \tag{4.4}$$

where k_{t+n} is the number of agents that are active at time $t+n$.

As several agents could have been spawned or terminated early at time step t; it is possible for k_{t+n} to be greater or less than k_t which is how the expected value from time $t+n$ may pass to an agent that terminated at time t.

4.2.3 MA-POCA Counterfactual Baselines

The counterfactual baselines used in MA-POCA leverage the concept of difference rewards (Wolpert and Tumer, 2002) and uses per-agent baselines so that the calculated advantage is indicative of the individuals contribution to the total group reward (Foerster *et al*., 2017a). The actions of individual agents are negated from the state-action value function in order to compute the baseline:

$$b_i(s,a) = \mathbb{E}_{a'\sim\pi_i(\cdot|o_i)}\left[Q_\pi(s,(a^{-i},a'))\right] \tag{4.5}$$

where a^{-i} is the joint action without entry i. The advantage of agent i is:

$$Adv_i = Q^\pi(s,a) - b_i(s,a) \tag{4.6}$$

and the update for agent i is:

$$\nabla_{\theta_i}J(\theta_i) = \mathbb{E}_{s\sim\rho^\pi}\left[\nabla_{\theta_i}log\pi_i(a^i|o^i)(Q^\pi(s,a) - b_i(s,a))\right] \tag{4.7}$$

where $a^i \sim \pi_i$

In the original implementation of COMA the system is limited to problems dealing with discrete actions only which must have a fixed number of agents (Foerster *et al.*, 2017a). To overcome these constraints MA-POCA proposes an alternative approach via an self-attention mechanism, which considers observations and observation-action pairs as distinct entities. Formally, let $f_i : O_i \times A_i \rightarrow E$ be an encoding network for observation-action pairs, the counterfactual baseline can be explicitly learned by estimating the expectation in equation (4.5) with Monte Carlo samples (Foerster *et al.* 2017a). For an agent j, the counterfactual baseline can be determined by learning a value function that is conditioned on the observation-action pairs of all agents i such that $1 \leq i \leq k_t$ where $i \neq j$ (but only the observation of agent j).

Using an RSA block and entity encoders for observations and observation-actions, the baseline parameterised by ψ for agent j has the form:

$$Q_\psi(RSA(g_j(o_t^j), f_i(o_i^i, a_t^i)_{\substack{1 \leq i \leq k_t \\ i \neq j}})) \tag{4.8}$$

The baseline objective is:

$$J(\psi) = (Q_\phi(RSA(g_j(o_t^j), f_i(o_i^i, a_t^i)_{\substack{1 \leq i \leq k_t \\ i \neq j}})) - y^{(\lambda)})^2 \tag{4.9}$$

which has the same target $y^{(\lambda)}$ as the value function update in equation 4.2.

A single joint observation $\mathbf{o} = (o^1, \ldots, o^N)$ generates up to N different samples on which to update equation 4.9, one for each $j, 1 \leq j \leq N$. Cohen *et al.* (2021) state this is the key reason for using separate sets of parameters for the value function and baseline.

The advantage for an agent j to be used in the counterfactual update in equation 4.7 is given by:

$$Adv_j = y^{(\lambda)} - Q_\psi(RSA(g_j(o_t^j), f_i(o_i^i, a_t^i)_{\substack{1 \leq i \leq k_t \\ i \neq j}})) \tag{4.10}$$

4.3 Creating the Learning Environment

As far as could be determined there are no freely available or open source simulators for conducting research into American Football; therefore the first step in this project was to create a new learning environment. Inspired by the *Fever Basketball* (Jia *et al.*, 2020), *Google Research Football* (Kurach *et al.*, 2020) and *Half-Field Offense* (Kalyanakrishnan, Liu and Stone, 2007) learning environments, this project introduces

SAFE: the *Simple American Football Environment* which is envisioned as eventually becoming a spiritual successor to the *Rush2008* simulator (Molineaux, Aha and Sukthankar, 2009).

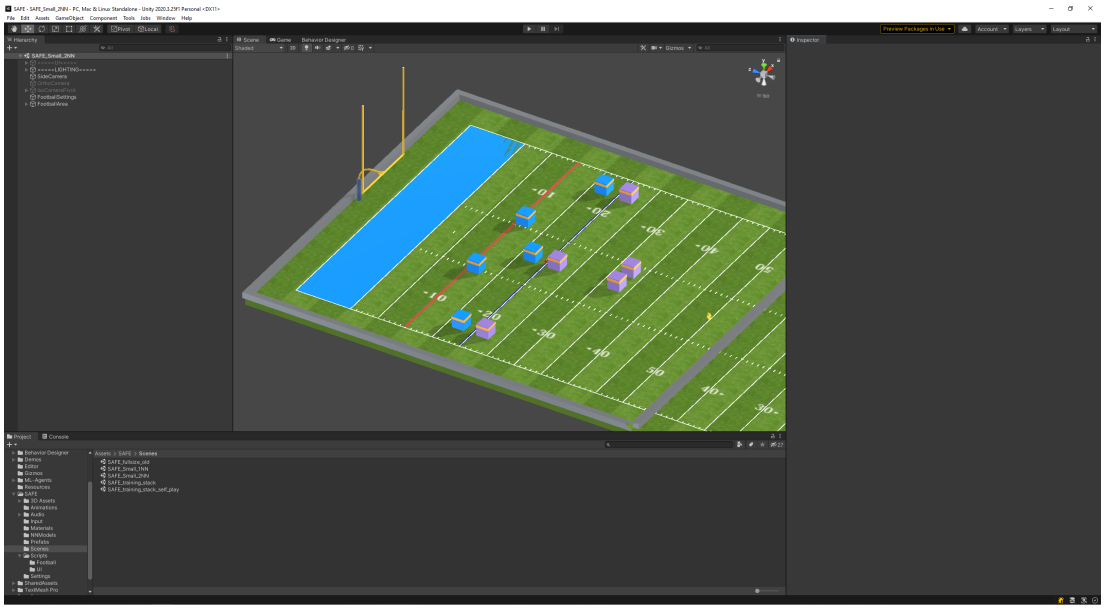

Figure 4.3: The Simple American Football Environment (SAFE).

Currently a work in progress and developed via *Unity*, SAFE implements a variant of American football which is based on the game *Backyard Football* (Humongous Entertainment, 1999) originally released for the Nintendo GameCube. *Backyard Football* is a simplified 5 on 5 version of the sport in which there are no penalties and turnovers can be disabled. A number of further simplifications are made in SAFE; the environment transitions are currently deterministic, there are no special teams or kicking game present and there is no game clock to manage. As in *Half-Field Offense*, the action takes place on half an American football field to make it easier to train the agents. As American football is a sport of role specialisation; SAFE supports both homogeneous and heterogeneous agents, an agent's type is currently defined by its individually assigned action and reward role profile.

To speed up development SAFE uses a number of visual assets taken from the *Unity ML-Agents* demonstration environments as well as some purchased from the Unity Asset Store for a small fee. Although this project leverages reinforcement learning and imitation learning, the SAFE environment itself is not limited to only one approach to creating intelligent agents; as it is created in the *Unity* editor which means it is easily extendable using custom C# scripts or downloaded asset store projects to add new

features as desired.

As this project is studying whether emergent behaviours will arise, unlike the real sport, in SAFE no proscribed play selection or movement plan is currently given to the agents; for this project they have to work out a coordinated sequence of actions on their own via training with rewards. There are two learning scenarios currently available; 'Goal Line Offense' and 'Red Zone Drive'. Both scenarios feature two competitive teams of five cooperative agents and is regarded as zero-sum at the team level. To make training simpler one team is the always defense and the other is always offense. Depending on the scenario, the offence's objective is to either score a touchdown in a single play, or to advance the ball down the field then score. The defence is always trying to stop the Offense by tackling the ball carrier for as small a gain as possible.

A training episode ends and the environment resets when a number of conditions are met:

- A touchdown is scored by the offense.
- The ball is carried or thrown out of bounds (Goal Line Offense).
- The defense makes a tackle short of the goal line (Goal Line Offense).
- The offense fails to gain 10 yards in four attempts (Red Zone Drive).
- The maximum number of environment steps (default: 1200) is reached.

4.3.1 Scenarios

- **Goal Line Offense:** The ball is placed on the defense's 10 yard line. The Offense has 1 attempt to score.
- **Red Zone Drive:** The ball is placed on the defense's 30 yard line. The Offense has 4 attempts to gain 10 yards or score, if they do so then they get another 4 attempts to gain 10 yards or score and so on.

4.3.2 Rewards

The *ML-Agents* documentation recommends using as sparse a reward profile as possible (Unity Technologies, 2021), as such SAFE only supports two different reward profiles, sparse and shaped.

- **Sparse:** The winning team receives a reward of $+1$ while the loser gets -1.

- **Shaped:** As above but with an additional positive/negative reward equal to the amount of yards gained or lost.

4.3.3 Actions

American football differs from many other domains as the action space is actually quite small. There are only five discrete branched actions corresponding to forward, backward and side-to-side movement, as well as rotation, throwing the ball and a set of context sensitive actions such as snapping the ball, tackling and catching.

4.3.4 Observations

SAFE currently does not use any visual observations but rather uses a feature vector. For positioning itself and detecting agents and other objects in the environment each agent has 20 forward facing ray-casts distributed over a 100 degree arc and 3 backward ray-casts distributed over 45 degrees. Each ray can detect several different object types, along with the detected object's distance from the originating agent. Agents have an additional unstacked state vector with 33 items that holds miscellaneous information about the game state, the ball and team mates which corresponds to 249 total observations per agent.

4.4 Training Agents To Play American Football

As end-to-end reinforcement learning is not very sample efficient (Sutton and Barto, 2018), in order to reduce training times a scene with multiple copies of the game area is used as recommended in the *Unity ML-Agents* documentation (Unity Technologies, 2021). Additionally *Unity ML-Agents* supports creating standalone builds that allow multiple instances to be run in parallel.

All training was performed on a single machine with a Intel i7-10700 CPU, 16GB RAM and a Nvidia GeForce RTX 2070 Super and after some experimentation it was found that by using twenty copies of the game area running in five concurrent instances, for a total of one hundred sets of multi-agent experience being generated simultaneously, the system was able to train 10000 steps roughly every thirty seconds achieve around 24 million steps each per day. This configuration resulted in satisfactory training times and also tracks with the training times reported in the Google Research Football paper (Kurach *et al.*, 2020).

As the environment was further developed and tested, to keep the results consistent the same network configuration and hyperparameters was used for every training run. All agents were trained via self-play and extrinsic rewards; the full set of values used can be found in Appendix B and are the defaults recommended by the Unity Technologies team and have been used to successfully train two teams of agents to play a first person dodgeball game in a complex multi-agent environment (Berges, 2021).

4.4.1 Agent Training Discussion

The initial training runs uncovered a few issues such as a bug with the code for rotation as an axis was not locked correctly meaning the Quarterback (QB) agent would give the ball to the Running Back (RB) agent but then attempted to rotate on the vertical towards the ball which caused it to fall over and get stuck. Another notable bug was that agents would keep 'stealing' the ball by bumping into the ball carrier due to a flaw in how the collisions were handled. To begin with the reward profile was quite dense, with additional individual rewards for various actions such holding the ball, throwing and catching or tackling. This only appeared to have an measurable effect with the Center agent, who was rewarded for staying engaged face-to-face with a defensive agent.

The first few runs also included an existential penalty to encourage the agents to perform actions faster, but their was no noticeable difference in the number of training steps required to converge. A hypothesis for this is that because the episode is actually quite short in our implementation, being only between $70-120$ steps on average, there is not enough time for it to be effective.

After some additional reward shaping experimentation and after 1 million training steps, run number nine was the first to display recognisable, intentional behaviour wherein the Center would snap the ball to the QB which then attempted to take the ball into the end zone while actively evading the defensive agents, who would roughly arrange themselves along the goal line to try and stop it. After observation is appeared that the QB never tried to give the ball to the other agents on offense. It would appear that the QB agent was acting greedily because the scoring agent was also receiving a $+1$ reward. To make the learning signal stronger, encourage passing and make debugging simpler, the system was changed to only use sparse rewards for win/loss, as recommended in the *Unity Ml-Agents* documentation (Unity Technologies, 2021). Finally it is of note that run 15 shows the Elo score (Elo, 1978) steadily going up as the

number of steps increase, but then after 50 million or so steps it crashes right back down, this looks like it could be an example of catastrophic forgetting (McCloskey, 1989).

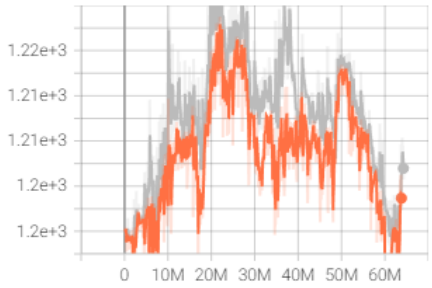

Figure 4.4: An example of catastrophic forgetting.

4.4.2 Improving Agent Training via Imitation Learning

Even after training for a large number of steps (150 million plus) the QB agent still did not attempt to pass on a regular basis, instead preferring to keep the ball and try to score on its own. Passing the ball successfully requires the correct temporal coordination of positioning and individual action selection between two agents. It is suspected that the system has learned that, as can be true in the real world, passing is a risky option because the vast majority of attempts will have ended in failure.

Unity ML-Agents supports two types of single agent imitation learning, BC and GAIL. Although *Unity ML-Agents* does not strictly support the combination of MA-POCA with these methods on a multi-agent basis, as a workaround this project explores the use of per-agent demonstrations to see whether imitation learning could still be used speed up training, improve Elo scores (Elo, 1978) and facilitate the passing game. A series of expert demonstrations were therefore recorded by a human taking control of each agent in turn and playing through a number of episodes in the unity editor to show the agent how its role should be played. These demonstrations were then used by both the BC and GAIL algorithms during training (it is recommended to use both together (Berges *et al.*, 2021) and each team did show a significant improvement in the resulting Elo scores (Elo, 1978) as can be seen in figures 4.5 and 4.6.

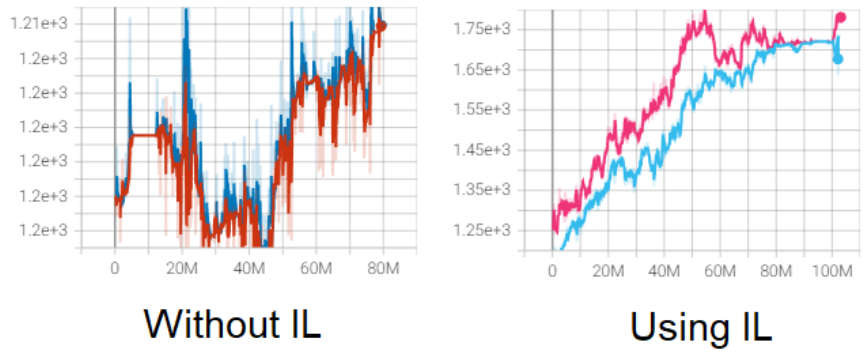

Figure 4.5: Scenario 1 Elo scores.

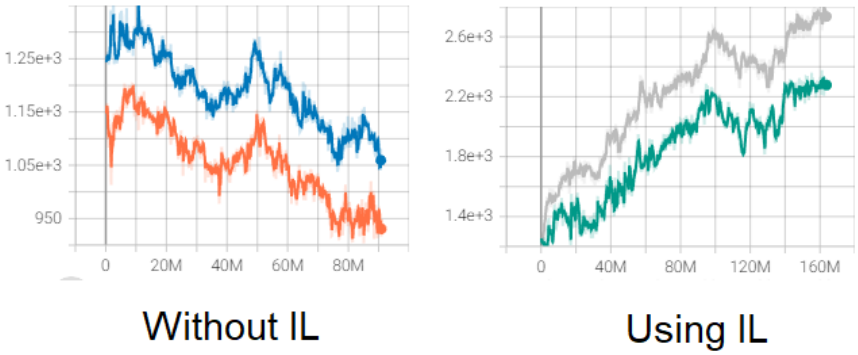

Figure 4.6: Scenario 2 Elo scores.

4.5 Validation and Verification

Performing validation and verification for machine learning is generally known to be difficult (Pullum at el., 2018) or even impossible at times (van Wesel and Goodloe, 2017) and this is especially true for DNN-based systems such as model-free deep reinforcement learning (Kazak *et al.*, 2019). As noted by van Wesel and Goodloe (2017) there is little point in applying traditional static verification methods to an online learning system as the system is continually updating thus invalidating any previous validation performed (van Wesel and Goodloe, 2017).

As the action selection mechanism is deterministic, runtime verification (Bartocci and Falcone, 2018; Wesel and Goodloe, 2017) methods can be used to check for violations instead. By attaching a debugger and using log messages manual verification was performed to ensure that agents will not select illegal actions for their current state. For example, only the agent with the Center role is allowed to use the 'Snap' action and no passing actions are allowed once the ball crosses the line of scrimmage.

There is no formal algorithmic proof of convergence available for MA-POCA so for this work an empirical simulation approach was used to verify and validate the system. As this is a simulation for a sports game there are no real-world safety concerns to consider, the main requirement for the agents in SAFE is that they act as realistically as possible and do not exhibit overly indecisive or erratic behaviours. Alongside manually observing the resulting behaviours demonstrated by agents, the progress of several key variables was tracked to validate the training process. Algorithmic training was assumed to have completed when the Elo score (Elo, 1978) had plateaued and no longer continued to improve.

The version of the system that was trained via MA-POCA only, without any IL methods, could be viewed as a form of ablation study (Meyes *et al.*, 2019). It should also be noted that as it is common for most machine learning algorithms to assume the data samples they learn from are independent and identically distributed (i.i.d), however in RL and IL this assumption is commonly violated to some degree, especially if the training environment differs from the production one in some way (van Wesel and Goodloe, 2017) and if the algorithm does not use some form of experience replay buffer (Foerster *et al.*, 2017b) when training.

4.6 Implementation Notes

The agents only use extrinsic rewards for curiosity. In an attempt to boost the exploration abilities initially the *Ml-Agents* Intrinsic Motivation module was enabled, which is noted as being experimental, in conjunction with the MA-POCA algorithm; however in practice this proved to be too unstable and repeatedly crashed out leading to a loss of training data for that run. When training, the simulation runs at $20\times$ real time speed and it can be adjusted up to a maximum of $100\times$ real time, however in practice there was no noticeable difference (or reduction) in training times when adjusting this value; it is possible that the hardware used to train agents may not have been powerful enough to benefit from this feature. Originally during training a single network for controlling both offensive and defensive agents was attempted, however it did not appear to be learning the type of coordinated behaviours required for this project. To make training easier it was split into two separate networks; one for offensive agents and one for defensive agents. During the earlier phases of this project the intention was to train a network per individual agent role type using the single-agent PPO (Schulman *et al.*, 2017) algorithm in conjunction with a series of individually tailored curriculum's.

However as MA-POCA (Cohen *et al*., 2021) is an end-to-end algorithm specifically designed for multi-agent and team-based scenarios, it was chosen instead as a more suitable candidate.

Chapter 5

Evaluation

This chapter consists of an overview of the experimental setup used to evaluate the agent teams in section 5.1, a discussion of the results obtained is provided in section 5.2 and any notable observed behaviours highlighted. A critical appraisal is then detailed in section 5.3, covering the resulting learning environment application, the decision making rationale and the overall project processes. Finally in section 5.4 some suggestions of promising directions for future work are provided.

5.1 Experimental Setup and Results

The first set of experiments takes place in scenario one (Goal Line Offense) as described in the previous chapter. To begin with both the offensive and defensive agents were trained with MA-POCA (Cohen, 2021) only. The various other combinations of RL and IL methods were then used for the subsequent experiments. Each experiment was repeated for 100 episodes and the results are shown in table 5.1 below.

Table 5.1: Results for scenario one experiments.

Offense vs Defence	Offensive Win Rate	Defensive Win Rate	No Winner
RL vs RL	43	38	19
RL vs RL + IL	26	61	13
RL + IL vs RL	68	21	11
RL + IL vs RL + IL	53	44	3

In this scenario for the offensive team to win it needs to score a touchdown, while

the defence has to make a tackle, any other outcome is recorded as a draw. The experiments were then repeated for scenario two (Red Zone Drive) and the results are summarised in table 5.2 below:

Table 5.2: Results for scenario two experiments.

Offense vs Defence	Offensive Win Rate	Defensive Win Rate	No Winner
RL vs RL	55	45	0
RL vs RL + IL	21	79	0
RL + IL vs RL	96	4	0
RL + IL vs RL + IL	68	32	0

In this scenario anything other than the offense scoring a touchdown in four attempts or less is counted as a win for the defensive team. The two agent teams are evaluated below on their win rate and observed similarity to real world behaviours.

5.2 Discussion

The results of the experiments show that, as expected the agents training using a mixture of reinforcement learning and imitation learning methods are stronger and able to dominate those using reinforcement learning only. It should be noted that at present, even after 150 million steps of training, the system still does not appear to have learned to effectively coordinate the actions required to throw and catch the ball consistently. Instead the QB usually prefers to keep the ball and take it into the end zone itself, as noted in chapter four, section 4.4.2, passing is the much more risky choice. In scenario one when both teams are using networks trained in the same manner the outcome is roughly even but in scenario two it seems that the offensive team is able to win more consistently.

It should be noted that whether or not the observed agent behaviours appear realistic and human-like is very subjective, especially with the abstracted visuals used in this project. However some evidence of blocking being performed by the offensive agents was observed, most perceptibly for the Center agent, especially after the addition of IL methods during training. Defensively, prior to the use of imitation learning the agents would hang back in a line to protect the goal line and after combining with IL, it can be seen that they are more aggressive in coming forward towards the ball carrier.

During the course of this project it was discovered that a key element for successfully training agents in a competitive team-based environment such as SAFE is the use of Competitive Self-Play; without it the system was unable to learn to play either scenario effectively.

While this dissertation project is focused on coordination via purely emergent behaviours, in reality both the offence and defence would initially be following a predetermined play call. However some of the most exciting on-field action is when a play breaks down and the players have to improvise as they go, in a game traditional AI methods would not be able to handle this type of scenario, however it is believed that the combination of using expert demonstrations, either prerecorded from a human player or even a traditional AI method, could be then used as the input to a hybrid RL/IL system allowing an agent to deviate from its preset plan and potentially endow the agents with much more flexibility and scope for improvisation in their actions, thus providing more life-life and realistic behaviours. The next steps for this project could be to investigate a two-level hierarchical approach further.

There are a number of positives for using a combination of reinforcement and imitation learning methods such as increased flexibility and the facilitation of more natural looking emergent behaviours. The approaches used in this project are more robust to unseen situations than traditional game AI techniques and only need retrained when changes are made to the game mechanics; no additional coding is required.

A potential drawback are that training agents can be very resource intensive in terms of both wall time and compute power required. Additionally decisions made by blackbox methods using neural networks are hard to interpret or explain (Wesel and Goodloe, 2017). This is particularly an issue in cases where training never converges and agents never learn the desired behaviours. Significant time may be lost in these cases without the ability to pinpoint exactly why; its hard to discern whether the network topology is insufficient or if more training time is required or if some other factor is to blame. This might make reinforcement learning based approaches too risky for commercial projects.

5.3 Project Critical Appraisal

5.3.1 Project Processes

The project proposal was overly ambitious given the time frame available; although it had been narrowed down considerably from the initial suggestions, the agreed project proposal was too wide in its scope. The research question and objectives selected were too open-ended as there are a plethora of different ways the problem could be approached. During the main literature review process it was discovered that MADRL has a wealth of subtopics to review, all of which were potentially relevant such as curiosity, agent modelling, explicit coordination, communication and meta-RL among many others. More detail on these topics and many more can be found in a number of recent survey papers (e.g. Yang and Wang, 2020; Canese *et al.*, 2021; Li, Fussell and Komura, 2021; *Wong et al.*, 2021; Gronauer and Diepold, 2022). In retrospect it would have been better to have already identified and selected a particular focus for research much earlier in the project process.

A significant deviation from the original proposal is that as the literature review progressed the focus of this dissertation project shifted to further investigation of MA-POCA (Cohen, 2021) and *Unity Ml-Agents* (Cohen, 2021) instead of just manually re-implementing versions of existing multi-agent algorithms in Python.

5.3.2 Rationale for Key Decisions

American football was selected as the topic of interest because it is a complex team sport, not yet covered by existing MADRL simulation research. Although vision-based learning is known to provide more robust agents, this dissertation project uses a feature vector as input only as vision-based approaches are harder to train and require more resources. One of the reasons MADRL was selected for investigation is due to previous interest with RL methods and the desire to extend existing knowledge of the field to cover multi-agent settings.

Aside from the rationale provided in chapter 4; another reason to use the Unity engine was existing familiarity of the C# language. Additionally, the adoption of an iterative rapid prototyping approach to the development life cycle was a good fit for the Unity editor's workflow. As it is generally desirable to keep MVP candidates as simple as possible project uses the least complex variant of the rules available so that the focus can be on the core learning elements. The selected hyperparameters and

network topology is the same as used for the *ML-Agents* dodgeball example (Berges, 2021) as they have been proven sufficient for training small groups of agents in a complex environment.

With regard to the academic and research objectives set out in the initial proposal document, this project has investigated the background theory for both single and multi-agent deep reinforcement learning and explored the key issues and challenges involved in applying deep reinforcement learning to multi-agent settings. The literature review process also examined the application of various research methods to sports and team based video games. One of the hardest aspects of undertaking this dissertation project was juggling work at the same time although this provided an opportunity to improve various project and time management skills.

The initial project timescale accounted for up to two weeks of slippage which ended up being used as training the agents took longer than expected. Using an agile approach to development tracked via *Asana* project management tool was helpful as it helped to prioritise which features are crucial to creation of a MVP and which were merely nice-to-haves. This made it easier to see which areas were on track and what was in danger of slipping and adjust accordingly.

With the benefit of hindsight it is recommended that researchers make sure to thoroughly review the codebase before beginning any agent training. Significant time was lost by training agents on a codebase that had bugs which negatively affected their behaviour. For example, there was a bug that disallowed forward passes if any agent passed the line of scrimmage, not just the Quarterback, which meant the system never attempted any passes at all.

5.3.3 Resulting Application

The resulting SAFE application is more limited in scope than was originally planned; still a work in progress it is currently more of a proof-of-concept than a fully fledged game. The application does meet the objective of allowing two teams of agents to be trained to play American football and it does allow a user to take control of any of the agents present and play against the other team. At present there are are currently only two scenarios implemented; goal line or red zone drive, where the initial plan called for a more varied collection. The creation of different formations and scenarios is currently supported but it would be better if they were updated to be GUI editor based instead of manually hand coded. The ruleset in use could also be expanded to incorporate

additional attributes such as time, penalties, and the kicking game. There are many other improvements that could be made to the game aspects of the environment such as improving the fidelity of the graphics, adding animations, audio and gamepad support. Finally, although not able to achieve a human level of performance, there is still some evidence of emergent coordinated behaviours as detailed above in section 5.2.

5.4 Potential Future Work

In addition to the potential next steps as described earlier, there has also been a trend towards graph based representations for modelling spatio-temporal relationships in team sports games (Brandt and Brefeld, 2015; Fujii, 2021; Xenopoulos and Claudio, 2021; Raabe, Nabben and Memmert, 2022), which may lead to more effective and efficient methods for training agents. The adoption of data-driven ghosting methods (Le, Yue and Carr 2017) using tracking player data from the NFL or NCAA inside the SAFE application could potentially be of use to teams or sports analysts. Alongside the imitation learning methods used in this project, other related approaches such as kickstarting agents (Schmitt *et al*., 2018) or some other transfer learning (Zhuang *et al*., 2019) based techniques may also be applicable. Additionally, as opposed to GAI (Pennachin and Goertzel, 2007) or General Video Game AI (Perez-Liebana *et al*., 2019), it is potentially worthwhile investigating a genre-based approach instead. For example, discovering a general set of reinforcement learning algorithms suitable for training a breadth of different sports game agents.

Chapter 6

Conclusion

In undertaking the task of exploring how reinforcement learning methods can be used to create more human-like and realistic game agent behaviours to answer this project's research question and objectives, this dissertation has provided an overview of the core background theory for both single and multi-agent reinforcement learning as well as some detail on imitation learning in chapter 2. Additionally this dissertation provides a rundown of the existing research relating to the application of reinforcement and imitation learning techniques to various sporting settings in chapter 3. A novel learning environment for American football based research has been introduced and described in detail alongside the methodology for training multi-agent teams used by this project in chapter 4. The new learning environment was used as the testbed for running experiments which empirically demonstrated that a combination of reinforcement learning and imitation learning speeds up the agent training process and leads to improved performance over reinforcement learning alone as detailed in chapter 5. Despite positively answering the original research question, that yes reinforcement learning is a viable approach to sports game AI, there is still much work to be done to reach the goal of realistic, human-like behaviours as would be required in a commercial product. On its own end-to-end reinforcement learning using only sparse rewards was not able to produce the type of emergent coordinated actions required to play complex team sports effectively. Hence, to finish, this dissertation has also provided some promising potential future research directions to be investigated further.

Referenced Works

Alamar, B. (2010) 'Measuring Risk in NFL Playcalling', *Journal of Quantitative Analysis in Sports*, 6, pp. 11–11. doi:10.2202/1559-0410.1235.

Alamar, B. and Weinstein-Gould, J. (2008) 'Isolating the Effect of Individual Linemen on the Passing Game in the National Football League', *Journal of Quantitative Analysis in Sports*, 4, pp. 10–10. doi:10.2202/1559-0410.1113.

Albus, J.S. (1975) 'A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)', *Journal of Dynamic Systems, Measurement, and Control*, 97(3), pp. 220–227. doi:10.1115/1.3426922.

Arora, S. and Doshi, P. (2018) 'A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress'. arXiv. doi:10.48550/ARXIV.1806.06877.

Arulkumaran, K. and Lillrank, D.O. (2021) 'A Pragmatic Look at Deep Imitation Learning', *CoRR*, abs/2108.01867. Available at: https://arxiv.org/abs/2108.01867.

Ba, J.L., Kiros, J.R. and Hinton, G.E. (2016) 'Layer Normalization'. arXiv. doi:10.48550/ARXIV.1607.06450.

Baby-boy (2021) 'Rant. It's the blocking A.I. that has ruined the game for me', Reddit. Available at: https://www.reddit.com/r/Madden/comments/qqc49l/ (Accessed: 10 September 2022).

Bai, Y. and Jin, C. (2020) 'Provable Self-Play Algorithms for Competitive Reinforcement Learning', *CoRR*, abs/2002.04017. Available at: https://arxiv.org/abs/2002.04017.

Baker, B. *et al.* (2019) 'Emergent Tool Use From Multi-Agent Autocurricula'. arXiv. doi:10.48550/ARXIV.1909.07528.

Balduzzi, D. *et al.* (2018) 'Re-evaluating evaluation', *CoRR*, abs/1806.02643. Available at: http://arxiv.org/abs/1806.02643.

Bansal, T. *et al.* (2017) 'Competitive Self-Play', *OpenAI*. Available at: https://openai.com/blog/competitive-self-play/.

Bartocci, E. *et al.* (2018) 'Introduction to Runtime Verification', in Bartocci, E. and Falcone, Y. (eds) *Lectures on Runtime Verification: Introductory and Advanced Topics*. Cham: Springer International Publishing, pp. 1–33. doi:10.1007/978-3-319-75632-5_1.

Bellemare, M.G. *et al.* (2012) 'The Arcade Learning Environment: An Evaluation Platform for General Agents', *CoRR*, abs/1207.4708. Available at: http://arxiv.org/abs/1207.4708.

Bellman, R. (1957) *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press.

Bengio, Y. *et al.* (2009) 'Curriculum Learning', in. New York, NY, USA: Association for Computing Machinery (ICML '09), pp. 41–48. doi:10.1145/1553374.1553380.

Berges, V.-P. *et al.* (2021) 'ML-Agents Plays Dodgeball', *Unity Blog*. Available at: https://blog.unity.com/technology/ml-agents-plays-dodgeball (Accessed: 10 September 2022).

Bernstein, D.S., Zilberstein, S. and Immerman, N. (2013) 'The Complexity of Decentralized Control of Markov Decision Processes', *CoRR*, abs/1301.3836. Available at: http://arxiv.org/abs/1301.3836.

Bhandari, I. *et al.* (1997) 'Advanced Scout: Data Mining and Knowledge Discovery in NBA Data', *Data Mining and Knowledge Discovery*, 1(1), pp. 121–125. doi:10.1023/A:1009782106822.

Biro, P. and Walker, S.G. (2021) 'A Reinforcement Learning Based Approach to Play Calling in Football'. arXiv. doi:10.48550/ARXIV.2103.06939.

Bishop, C.M. (1994) 'Mixture density networks'. Available at: https://publications.aston.ac.uk/id/eprint/373/1/NCRG_94_004.pdf (Accessed: 14 September 2022).

Bob Carroll, J.T., Pete Palmer (1988) *The Hidden Game of Football*. Grand Central.

Bowling, M. and Veloso, M. (2002) 'Multiagent learning using a variable learning rate', Artificial Intelligence, 136(2), pp. 215–250.

Brandt, M. and Brefeld, U. (2015) 'Graph-based Approaches for Analyzing Team Interaction on the Example of Soccer', Available at: http://ceur-ws.org/Vol-1970/paper-03.pdf

Brockman, G. *et al.* (2016) 'OpenAI Gym'. arXiv. doi:10.48550/ARXIV.1606.01540.

Burke, B. (2019) 'DeepQB: Deep Learning with Player Tracking to Quantify Quarterback Decision-Making & Performance'. in *Proceedings of the 2018 MIT Sloan Sports Analytics Conference*. Boston.

Busoniu, L., Babuska, R. and De Schutter, B. (2008) 'A Comprehensive Survey of Multiagent Reinforcement Learning', *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, IEEE Transactions on, 38, pp. 156–172. doi:10.1109/TSMCC.2007.913919.

Busoniu, L., Babuska, R. and De Schutter, B. (2010) 'Multi-agent Reinforcement Learning: An Overview', in Studies in Computational Intelligence, pp. 183–221. doi:10.1007/978-3-642-14435-6_7.

Canes (2020) 'What are your three biggest issues with the on-field AI?', *Operationsports.com*. Available at: https://forums.operationsports.com/forums/madden-nfl-football/973249-what-your-three-biggest-issues-field-ai.html (Accessed: 10 September 2022).

Canese, L. *et al.* (2021) 'Multi-Agent Reinforcement Learning: A Review of Challenges and Applications', *Applied Sciences*, 11(11). doi:10.3390/app11114948.

Cervone, D. *et al.* (2016) 'A Multiresolution Stochastic Process Model for Predicting Basketball Possession Outcomes', *Journal of the American Statistical Association*, 111(514), pp. 585–599. doi:10.1080/01621459.2016.1141685.

Claus, C. and Boutilier, C. (1998) 'The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems', in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. USA: American Association for Artificial Intelligence (AAAI '98/IAAI '98), pp. 746–752.

Cohen, A. *et al.* (2021) 'On the Use and Misuse of Absorbing States in Multi-agent Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.2111.05992.

Colledanchise, M. and Ögren, P. (2018) *Behavior Trees in Robotics and AI: An Introduction*. London, England: CRC Press.

Dcfan (2021) 'Madden 22 has some of the worst AI I've ever seen in a sports game', *Reddit*. Available at: https://www.reddit.com/r/Madden/comments/pqvlnv/ (Accessed: 10 September 2022).

Ding, Z. (2021) 'Imitation Learning', in Dong, H., Ding, Z., and Zhang, S. (eds) *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Singapore, Singapore: Springer.

Du, W. and Ding, S. (2021) 'A Survey on Multi-Agent Deep Reinforcement Learning: From the Perspective of Challenges and Applications', *Artificial Intelligence Review*, 54(5), pp. 3215–3238. doi:10.1007/s10462-020-09938-y.

Elo, A.E. (1978) *The Rating of Chessplayers, Past and Present*. New York: Arco Pub.

Espeholt, L. *et al.* (2018) 'IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures', *CoRR*, abs/1802.01561. Available at: http://arxiv.org/abs/1802.01561.

Eysenbach, B. and Levine, S. (2022) 'Maximum Entropy RL (Provably) Solves Some Robust RL Problems'. arXiv. doi:10.48550/ARXIV.2103.06257.

Fernández, J., Bornn, L. and Cervone, D. (2021) 'A Framework for the Fine-Grained Evaluation of the Instantaneous Expected Value of Soccer Possessions',*Mach. Learn.*, 110(6), pp. 1389–1427. doi:10.1007/s10994-021-05989-6.

Foerster, J. *et al.* (2017) 'Counterfactual Multi-Agent Policy Gradients'. arXiv. doi:10.48550/ARXIV.1705.08926.

Foerster, J. (2018) *Deep multi-agent reinforcement learning*. PhD Thesis. University of Oxford.

Foerster, J.N. *et al.* (2016) 'Learning to Communicate with Deep Multi-Agent Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.1605.06676.

Foerster, J.N. *et al.* (2017) 'Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning', *CoRR*, abs/1702.08887. Available at: http://arxiv.org/abs/1702.08887.

Fujii, K. (2021) 'Data-driven Analysis for Understanding Team Sports Behaviors', *CoRR*, abs/2102.07545. Available at: https://arxiv.org/abs/2102.07545.

Goldner, K. (2012) 'A Markov Model of Football: Using Stochastic Processes to Model a Football Drive', *Journal of Quantitative Analysis in Sports*, 8(1). doi:doi:10.1515/1559-0410.1400.

Goodfellow, I.J. *et al.* (2014) 'Generative Adversarial Networks'. arXiv. doi:10.48550/ARXIV.1406.2661.

Gronauer, S. and Diepold, K. (2022) 'Multi-Agent Deep Reinforcement Learning: A Survey', *Artificial Intelligence Revue*, 55(2), pp. 895–943. doi:10.1007/s10462-021-09996-w.

Gupta, J.K., Egorov, M. and Kochenderfer, M. (2017) 'Cooperative Multi-agent Control Using Deep Reinforcement Learning', in Sukthankar, G. and Rodriguez-Aguilar, J.A. (eds) *Autonomous Agents and Multiagent Systems*. Cham: Springer International Publishing, pp. 66–83.

Guss, W.H. *et al.* (2019) 'The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors', *CoRR*, abs/1904.10079. Available at: http://arxiv.org/abs/1904.10079.

Guss, W.H. *et al.* (2021) 'The MineRL 2020 Competition on Sample Efficient Reinforcement Learning using Human Priors', *CoRR*, abs/2101.11071. Available at: https://arxiv.org/abs/2101.11071.

Ha, D. and Schmidhuber, J. (2018) 'World Models', *CoRR*, abs/1803.10122. Available at: http://arxiv.org/abs/1803.10122. Haarnoja, T. *et al.* (2018) 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor', *CoRR*, abs/1801.01290. Available at: http://arxiv.org/abs/1801.01290.

Han, S. and Sung, Y. (2017) 'Multi-Batch Experience Replay for Fast Convergence of Continuous Action Control', *CoRR*, abs/1710.04423. Available at: http://arxiv.org/abs/1710.04423.

Hausknecht, M. *et al.* (2016) 'Half Field Offense: An Environment for Multiagent Learning and Ad Hoc Teamwork', in *Proceedings of the AAMAS Adaptive Learning Agents (ALA) Workshop*. Singapore. Available at: http://www.cs.utexas.edu/users/ai-lab?hausknecht:aamasws16.

Hernandez-Leal, P. *et al.* (2017) 'A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity'. arXiv. doi:10.48550/ARXIV.1707.09183.

Hernandez-Leal, P., Kartal, B. and Taylor, M. (2019) 'A survey and critique of multiagent deep reinforcement learning', *Autonomous Agents and Multi-Agent Systems*, 33. doi:10.1007/s10458-019-09421-1.

Hess, R. and Fern, A. (2009) 'Discriminatively trained particle filters for complex multi-object tracking', *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 240–247.

Hess, R., Fern, A. and Mortensen, E. (2007) 'Mixture-of-Parts Pictorial Structures for Objects with Variable Part Sets', in *Proceedings of the 2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8. doi:10.1109/ICCV.2007.4409071.

Hessel, M. *et al.* (2017) 'Rainbow: Combining Improvements in Deep Reinforcement Learning', *CoRR*, abs/1710.02298. Available at: http://arxiv.org/abs/1710.02298.

Hester, T. *et al.* (2017) 'Learning from Demonstrations for Real World Reinforcement Learning', *CoRR*, abs/1704.03732. Available at: http://arxiv.org/abs/1704.03732.

Higinbotham, W. (1958) *Tennis for Two* [Video game]. Upton, New York.

Hirotsu, N. and Wright, M. (2002) 'Using a Markov process model of an association football match to determine the optimal timing of substitution and tactical decisions', *Journal of the Operational Research Society*, 53(10), pp. 1174–1174. doi:10.1057/palgrave.jors.2601414.

Ho, J. and Ermon, S. (2016) 'Generative Adversarial Imitation Learning', *CoRR*, abs/1606.03476. Available at: http://arxiv.org/abs/1606.03476.

Hong, C. *et al.* (2021) 'AI World Cup: Robot-Soccer-Based Competitions', *IEEE Transactions on Games*, 13(4), pp. 330–341. doi:10.1109/TG.2021.3065410.

Horgan, D. *et al.* (2018) 'Distributed Prioritized Experience Replay', *CoRR*, abs/1803.00933. Available at: http://arxiv.org/abs/1803.00933.

Huang, S. *et al.* (2021) 'TiKick: Towards Playing Multi-agent Football Full Games from Single-agent Demonstrations', *CoRR*, abs/2110.04507. Available at: https://arxiv.org/abs/2110.04507.

Humongous Entertainment (1999) *Backyard Football* [Video game]. GT Interactive.

Hussein, A. *et al.* (2017) 'Imitation Learning: A Survey of Learning Methods', *ACM Comput. Surv.*, 50(2). doi:10.1145/3054912.

Hüttenrauch, M., Sosic, A. and Neumann, G. (2018) 'Deep Reinforcement Learning for Swarm Systems', *CoRR*, abs/1807.06613. Available at: http://arxiv.org/abs/1807.06613.

Intille, S.S. and Bobick, A.F. (1999) 'A Framework for Recognizing Multi-Agent Action from Visual Evidence', in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. Orlando, Florida, USA, AAAI.

Iskander, N. *et al.* (2020) 'Reinforcement Learning Agents for Ubisoft's Roller Champions', *CoRR*, abs/2012.06031. Available at: https://arxiv.org/abs/2012.06031.

Jacobs, R.A. *et al.* (1991) 'Adaptive Mixtures of Local Experts', *Neural Computation*, 3(1), pp. 79–87. doi:10.1162/neco.1991.3.1.79.

Jaderberg, M. *et al.* (2017) 'Population Based Training of Neural Networks'. arXiv. doi:10.48550/ARXIV.1711.09846.

Jaderberg, M. *et al.* (2019) 'Human-level performance in 3D multiplayer games with population-based reinforcement learning', *Science*, 364(6443), pp. 859–865. doi:10.1126/science.aau6249.

Jia, H. *et al.* (2020) 'Fever Basketball: A Complex, Flexible, and Asynchronized Sports Game Environment for Multi-agent Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.2012.03204.

Jiang, J. and Lu, Z. (2018) 'Learning Attentional Communication for Multi-Agent Cooperation', in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc. Red Hook, NY, USA. pp. 7265–7275.

Jin, J. *et al.* (2018) 'Real-Time Bidding with Multi-Agent Reinforcement Learning in Display Advertising'.

Jordan, J., Melouk, S. and Perry, M. (2009) 'Optimizing Football Game Play Calling', *Journal of Quantitative Analysis in Sports*, 5, pp. 2–2. doi:10.2202/1559-0410.1176.

Jug, M. *et al.* (2003) 'Trajectory Based Assessment of Coordinated Human Activity', in Crowley, J.L. *et al.* (eds) *Computer Vision Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 534–543.

Juliani, A. *et al.* (2018) 'Unity: A General Platform for Intelligent Agents', *CoRR*, abs/1809.02627. Available at: http://arxiv.org/abs/1809.02627.

Juliani, A. *et al.* (2019) 'Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning', *CoRR*, abs/1902.01378. Available at: http://arxiv.org/abs/1902.01378.

Justesen, N. *et al.* (2019) 'Deep Learning for Video Game Playing', *CoRR*, abs/1708.07902. Available at: http://arxiv.org/abs/1708.07902.

Kaelbling, L.P., Littman, M.L. and Cassandra, A.R. (1998) 'Planning and acting in partially observable stochastic domains', *Artificial Intelligence*, 101(1), pp. 99–134. doi:https://doi.org/10.1016/S0004-3702(98)00023-X.

Kahn, J. (2003) 'Neural Network Prediction of NFL Football Games', Available at: https://docplayer.net/21763052-Neural-network-prediction-of-nfl-football-games-joshua-kahn.html (Accessed: 12 September 2022).

Kalyanakrishnan, S., Liu, Y. and Stone, P. (2007) 'Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study', in Lakemeyer, G. *et al.* (eds) *RoboCup 2006: Robot Soccer World Cup X*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 72–85.

Kanervisto, A. *et al.* (2022) 'MineRL Diamond 2021 Competition: Overview, Results, and Lessons Learned'. arXiv. doi:10.48550/ARXIV.2202.10583.

Kazak, Y. *et al.* (2019) 'Verifying Deep-RL-Driven Systems', in *Proceedings of the 2019 Workshop on Network Meets AI & ML*. New York, NY, USA: Association for Computing Machinery (NetAI'19), pp. 83–89. doi:10.1145/3341216.3342218.

Kempka, M. *et al.* (2016) 'ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning', *CoRR*, abs/1605.02097. Available at: http://arxiv.org/abs/1605.02097.

Kingma, D.P. and Ba, J. (2014) 'Adam: A Method for Stochastic Optimization'. arXiv. doi:10.48550/ARXIV.1412.6980.

Kiran, B.R. *et al.* (2020) 'Deep Reinforcement Learning for Autonomous Driving: A Survey', *CoRR*, abs/2002.00444. Available at: https://arxiv.org/abs/2002.00444.

Kitano, H. *et al.* (1995) 'RoboCup: The Robot World Cup Initiative'. Available at: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.7511

K.J Åström (1965) 'Optimal control of Markov processes with incomplete state information', *Journal of Mathematical Analysis and Applications*, 10(1), pp. 174–205. doi:https://doi.org/10.1016/0022-247X(65)90154-X.

Konda, V. and Tsitsiklis, J. (2000) 'Actor-Critic Algorithms', in *SIAM Journal on Control and Optimization*. MIT Press, pp. 1008–1014.

Kraemer, L. and Banerjee, B. (2016) 'Multi-agent reinforcement learning as a rehearsal for decentralized planning', *Neurocomputing*, 190, pp. 82–94. doi:https://doi.org/10.1016/j.neucom.2016.01.031.

Krumins, A. (2020) 'MindMaker AI Plugin for Unreal Engine 4 & 5'. Available at: https://github.com/krumiaa/MindMaker (Accessed: 4 September 2022).

Kuhlmann, G., Knox, W.B. and Stone, P. (2006) 'Know thine enemy: A champion RoboCup coach agent', in *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1463.

Kurach, K. *et al.* (2018) 'The GAN Landscape: Losses, Architectures, Regularization, and Normalization', *CoRR*, abs/1807.04720. Available at: http://arxiv.org/abs/1807.04720.

Kurach, K. *et al.* (2020) 'Google Research Football: A Novel Reinforcement Learning Environment'. arXiv. doi:10.48550/ARXIV.1907.11180.

Lanctot, M. (2019) 'Multiagent Reinforcement Learning' [Powerpoint presentation]. Available at: http://mlanctot.info/files/papers/Lanctot_MARL_RLSS2019_Lille.pdf.

Lapan, M. (2018) *Deep Reinforcement Learning Hands-On*. Birmingham, England: Packt Publishing.

Laviers, K. *et al.* (2009) 'Improving Offensive Performance Through Opponent Modeling.'. in *Proceedings of the Fith AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI Press (AIIDE'09), pp. 58–63.

Laviers, K. and Sukthankar, G. (2010) 'A Monte Carlo Approach for Football Play Generation', in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI Press (AIIDE'10), pp. 150–155.

Le, H.M., Yue, Y. and Carr, P. (2017) 'Coordinated Multi-Agent Imitation Learning', *CoRR*, abs/1703.03121. Available at: http://arxiv.org/abs/1703.03121.

Leibo, J.Z. *et al.* (2019) 'Autocurricula and the Emergence of Innovation from Social Interaction: A Manifesto for Multi-Agent Intelligence Research'. arXiv. doi:10.48550/ARXIV.1903.00742.

Li, C., Fussell, L. and Komura, T. (2021) 'Multi-Agent Reinforcement Learning For Character Control', *The Visual Computer*, 37, pp. 1–9. doi:10.1007/s00371-021-02269-1.

Li, N. *et al.* (2009) 'Constructing Game Agents from Video of Human Behavior', in *Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI Press (AIIDE'09).

Li, S. *et al.* (2019) 'Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient', in *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), pp. 4213–4220. doi:10.1609/aaai.v33i01.33014213.

Li, Y. (2018) 'Deep Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.1810.06339.

Littman, M.L. (1994) 'Markov Games as a Framework for Multi-Agent Reinforcement Learning', in In *Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, pp. 157–163.

Liu, G. *et al.* (2020) 'Deep soccer analytics: learning an action-value function for evaluating soccer players', *Data Mining and Knowledge Discovery*, 34(5), pp. 1531–1559.

Liu, G. and Schulte, O. (2018) 'Deep Reinforcement Learning in Ice Hockey for Context-Aware Player Evaluation', *CoRR*, abs/1805.11088. Available at: http://arxiv.org/abs/1805.11088.

Liu, S. *et al.* (2022) 'From motor control to team play in simulated humanoid football', *Science Robotics*, 7(69), p. eabo0235. doi:10.1126/scirobotics.abo0235.

Long, Q. *et al.* (2020) 'Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.2003.10423.

Lowe, R. *et al.* (2017) 'Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments'. arXiv. doi:10.48550/ARXIV.1706.02275.

Lowe, Z. (2013) 'Lights, Cameras, Revolution'. Available at: http://grantland.com/features/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution/ (Accessed: 11 September 2022).

Luo, Y., Schulte, O. and Poupart, P. (2021) 'Inverse Reinforcement Learning for Team Sports: Valuing Actions and Players', in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. (IJCAI'20).

Lyu, X. *et al.* (2021) 'Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning', *CoRR*, abs/2102.04402. Available at: https://arxiv.org/abs/2102.04402.

'Madden NFL' (2022) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Madden_NFL (Accessed: 20 August 2022).

Masson, W. and Konidaris, G.D. (2015) 'Reinforcement Learning with Parameterized Actions', *CoRR*, abs/1509.01644. Available at: http://arxiv.org/abs/1509.01644.

McCloskey, M. and Cohen, N.J. (1989) 'Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem', in Bower, G.H. (ed.). *Academic Press* (*Psychology of Learning and Motivation*), pp. 109–165. doi:https://doi.org/10.1016/S0079-7421(08)60536-8.

Meyes, R. *et al.* (2019) 'Ablation Studies in Artificial Neural Networks', *CoRR*, abs/1901.08644. Available at: http://arxiv.org/abs/1901.08644.

Michel, O. (2004) 'Webots: Professional Mobile Robot Simulation', *Journal of Advanced Robotics Systems*, 1(1), pp. 39–42.

Min, B. *et al.* (2008) 'A compound framework for sports results prediction: A football case study', *Knowledge-Based Systems*, 21(7), pp. 551–562. doi:https://doi.org/10.1016/j.knosys.2008.03.016.

Minsky, M. (1961) 'Steps toward Artificial Intelligence', *Proceedings of the IRE*, 49(1), pp. 8–30. doi:10.1109/JRPROC.1961.287775.

Mnih, V. *et al.* (2013) 'Playing Atari with Deep Reinforcement Learning', *CoRR*, abs/1312.5602. Available at: http://arxiv.org/abs/1312.5602.

Mnih, V. *et al.* (2015) 'Human-level control through deep reinforcement learning', *Nature*, 518, pp. 529–533.

Molineaux, M., Aha, D.W. and Sukthankar, G.R. (2009) 'Beating the Defense: Using Plan Recognition to Inform Learning Agents', in *FLAIRS Conference*.

Morales, M. (2021) *Grokking Deep Reinforcement Learning*. New York, NY: Manning Publications.

Mozgovoy, M., Preuss, M. and Bidarra, R. (2021) 'Team Sports for Game AI Benchmarking Revisited', *International Journal of Computer Games Technology*, 2021, p. 5521877:1-5521877:9.

Narvekar, S. *et al.* (2020) 'Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey'. doi:10.48550/ARXIV.2003.04960.

Neto, G. (2005) 'From Single-Agent to Multi-Agent Reinforcement Learning: Foundational Concepts and Methods'.

NFL. (2022) 'The Digital Athlete and How it's Revolutionizing Player Health & Safety'. Available at: https://www.nfl.com/playerhealthandsafety/equipment-and-innovation/aws-partnership/digital-athlete-spot (Accessed: 14 September 2022).

Nguyen, T.T., Nguyen, N.D. and Nahavandi, S. (2020) 'Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications', IEEE Transactions on Cybernetics, 50(9), pp. 3826–3839. doi:10.1109/tcyb.2020.2977374.

Nichol, A. *et al.* (2018) 'Gotta Learn Fast: A New Benchmark for Generalization in RL', *CoRR*, abs/1804.03720. Available at: http://arxiv.org/abs/1804.03720.

Oliehoek, F.A. and Amato, C. (2016) 'A Concise Introduction to Decentralized POMDPs'. Springer Publishing Company, Incorporated.

Omidshafiei, S. *et al.* (2019) 'Learning to Teach in Cooperative Multiagent Reinforcement Learning', in *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), pp. 6128–6136. doi:10.1609/aaai.v33i01.33016128.

OpenAI (2018) 'Part 2: Kinds of RL Algorithms', *https://spinningup.openai.com/*. Available at: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html (Accessed: 11 August 2022).

OpenAI *et al.* (2019) 'Dota 2 with Large Scale Deep Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.1912.06680.

Oroojlooyjadid, A. and Hajinezhad, D. (2019) 'A Review of Cooperative Multi-Agent Deep Reinforcement Learning', *CoRR*, abs/1908.03963. Available at: http://arxiv.org/abs/1908.03963.

Osa, T. *et al.* (2018) 'An Algorithmic Perspective on Imitation Learning', *CoRR*, abs/1811.06711. Available at: http://arxiv.org/abs/1811.06711.

Papoudakis, G. *et al.* (2019) 'Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning'. arXiv. doi:10.48550/ARXIV.1906.04737.

Patek, S.D. and Bertsekas, D.P. (1998) 'Play Selection in American Football: A Case Study in Neuro-Dynamic Programming', in *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*. Boston: Springer, pp. 189–213.

Peng, P. *et al.* (2017) 'Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games'. arXiv. doi:10.48550/ARXIV.1703.10069.

Pennachin, C. and Goertzel, B. (2007) 'Contemporary Approaches to Artificial General Intelligence', in Goertzel, B. and Pennachin, C. (eds) *Artificial General Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–30. doi:10.1007/978-3-540-68677-4_1.

Perez-Liebana, D. *et al.* (2019) *General Video Game Artificial Intelligence*. Morgan & Claypool Publishers (2).

Plaat, A. (2021) *Learning To Play: Reinforcement Learning and Games*. Cham, Switzerland: Springer Nature.

Plaat, A. (2022) *Deep Reinforcement Learning, a textbook*. arXiv. doi:10.48550/ARXIV.2201.02135.

Pomerleau, D. (1989) 'ALVINN: An Autonomous Land Vehicle In a Neural Network', in Touretzky, D.S. (ed.) *Proceedings of (NeurIPS) Neural Information Processing Systems*. Morgan Kaufmann, pp. 305–313.

Pomerleau, D.A. (1991) 'Efficient Training of Artificial Neural Networks for Autonomous Navigation', *Neural Computation*, 3(1), pp. 88–97. doi:10.1162/neco.1991.3.1.88.

Pullum, L.L. *et al.* (2018) 'Mathematically Rigorous Verification and Validation of Scientific Machine Learning', in *Proceedings of the 2018 DOE Scientific Machine Learning Workshop*.

Purucker, M.C. (1996) 'Neural network quarterbacking', *IEEE Potentials*, 15(3), pp. 9–15. doi:10.1109/45.535226.

Raabe, D., Nabben, R. and Memmert, D. (2022) 'Graph representations for the analysis of multi-agent spatiotemporal sports data', *Applied Intelligence*, pp. 1–21. doi:10.1007/s10489-022-03631-z.

Racanière, S. *et al.* (2017) 'Imagination-Augmented Agents for Deep Reinforcement Learning', in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc. (NIPS'17), pp. 5694–5705.

Rashid, T. *et al.* (2018) 'QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning', *CoRR*, abs/1803.11485. Available at: http://arxiv.org/abs/1803.11485.

Riley, P. and Veloso, M. (2002) 'Planning for Distributed Execution through Use of Probabilistic Opponent Models', in *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*. AAAI Press (AIPS'02), pp. 72–81.

Roberts, A. (2022) *Biographia Literaria by Samuel Taylor Coleridge*. Edinburgh: Edinburgh University Press. doi:doi:10.1515/9780748692095.

Romer, D. (2006) 'Do Firms Maximize? Evidence from Professional Football', Journal of Political Economy, 114, pp. 340–365. doi:10.1086/501171.

Ross, S. and Bagnell, J.A. (2010) 'Efficient Reductions for Imitation Learning', in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.

Ross, S., Gordon, G.J. and Bagnell, J.A. (2010) 'No-Regret Reductions for Imitation Learning and Structured Prediction', *CoRR*, abs/1011.0686. Available at: http://arxiv.org/abs/1011.0686.

Routley, K. and Schulte, O. (2015) 'A Markov Game Model for Valuing Player Actions in Ice Hockey', in *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. Arlington, Virginia, USA: AUAI Press (UAI'15), pp. 782–791.

Rummery, G. and Niranjan, M. (1994) 'On-Line Q-Learning Using Connectionist Systems', Technical Report CUED/F-INFENG/TR 166 [Preprint]. Available at: http://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf (Accessed: 14 September 2022).

Sandholtz, N. and Bornn, L. (2020) 'Markov decision processes with dynamic transition probabilities: An analysis of shooting strategies in basketball', *The Annals of Applied Statistics*, 14(3). doi:10.1214/20-aoas1348.

Schmitt, S. *et al.* (2018) 'Kickstarting Deep Reinforcement Learning', *CoRR*, abs/1803.03835. Available at: http://arxiv.org/abs/1803.03835.

Schrittwieser, J. *et al.* (2019) 'Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model', *CoRR*, abs/1911.08265. Available at: http://arxiv.org/abs/1911.08265.

Schulman, J. *et al.* (2015) 'Trust Region Policy Optimization'. arXiv. doi:10.48550/ARXIV.1502.05477.

Schulman, J. *et al.* (2017) 'Proximal Policy Optimization Algorithms', *CoRR*, abs/1707.06347. Available at: http://arxiv.org/abs/1707.06347.

Seidl, T. *et al.* (2018) 'Bhostgusters: Realtime Interactive Play Sketching with Synthesized NBA Defenses', in *Proceedings of the 2018 MIT Sloan Sports Analytics Conference*. Boston.

Shao, K. *et al.* (2019) 'A Survey of Deep Reinforcement Learning in Video Games', *CoRR*, abs/1912.10944. Available at: http://arxiv.org/abs/1912.10944.

Shapley, L.S. (1953) 'Stochastic Games', Proceedings of the National Academy of Sciences, 39(10), pp. 1095–1100. doi:10.1073/pnas.39.10.1095.

Sharbaugh, S. (2021) 'AI's impact on video game outcomes frustrates gamers', Sportslitigationalert.com. Available at: https://sportslitigationalert.com/ais-impact-on-video-game-outcomes-frustrates-gamers/ (Accessed: 4 April 2022).

Shoham, Y., Powers, R. and Grenager, T. (2007) 'If multi-agent learning is the answer, what is the question?', *Artificial Intelligence*, 171(7), pp. 365–377. doi:https://doi.org/10.1016/j.artint.2006.02.006.

Sicart, M. (2013) 'A tale of two games: Football and FIFA 12', in *Sports Videogames*, pp. 32–49. doi:10.4324/9780203084496.

Silva, F. and Costa, A. (2019) 'A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems', *Journal of Artificial Intelligence Research*, 64. doi:10.1613/jair.1.11396.

Silva, F.L.D., Taylor, M.E. and Costa, A.H.R. (2018) 'Autonomously Reusing Knowledge in Multiagent Reinforcement Learning', in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, pp. 5487–5493. doi:10.24963/ijcai.2018/774.

Silver, D. *et al.* (2016) 'Mastering the game of Go with deep neural networks and tree search', *Nature*, 529, pp. 484–489. doi:10.1038/nature16961.

Silver, D. *et al.* (2018) 'A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play', *Science*, 362(6419), pp. 1140–1144. doi:10.1126/science.aar6404.

Silver, D. *et al.* (2021) 'Reward is enough', *Artificial Intelligence*, 299, p. 103535. doi:https://doi.org/10.1016/j.artint.2021.103535.

Sin, B. (2013) 'Artificial intelligence in sports video games is frustrating. It doesn't have to be', Sports Illustrated. Available at: https://www.si.com/extra-mustard/2013/09/20/artificial-intelligence-in-sports-video-games-is-frustrating-it-doesnt-have-to-be (Accessed: 4 April 2022).

Singh, A., Jain, T. and Sukhbaatar, S. (2018) 'Learning when to Communicate at Scale in Multiagent Cooperative and Competitive Tasks'. arXiv. doi:10.48550/ARXIV.1812.09755.

Song, Y. *et al.* (2019) 'Arena: A General Evaluation Platform and Building Toolkit for Multi-Agent Intelligence', *CoRR*, abs/1905.08085. Available at: http://arxiv.org/abs/1905.08085.

Stracuzzi, D.J. *et al.* (2011) 'An Application of Transfer to American Football: From Observation of Raw Video to Control in a Simulated Environment', *AI Mag.*, 32(2), pp. 107–125. doi:10.1609/aimag.v32i2.2336.

Sukhbaatar, S., Szlam, A. and Fergus, R. (2016) 'Learning Multiagent Communication with Backpropagation'. arXiv. doi:10.48550/ARXIV.1605.07736.

Sunehag, P. *et al.* (2017) 'Value-Decomposition Networks For Cooperative Multi-Agent Learning', *CoRR*, abs/1706.05296. Available at: http://arxiv.org/abs/1706.05296.

Sutton, R.S. and Barto, A.G. (2018) *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book.

Takayanagi, R. *et al.* (2022) 'Decision Making in American Football under State Uncertainty by Stochastic Inverse Reinforcement Leaning', *Bulletin of Networking, Computing, Systems, and Software*, 11(1), pp. 25–29.

Tan, M. (1993) 'Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents', in *In Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann, pp. 330–337.

Tan, T. *et al.* (2020) 'Cooperative Deep Reinforcement Learning for Large-Scale Traffic Grid Signal Control', *IEEE Transactions on Cybernetics*, 50(6), pp. 2687–2700. doi:10.1109/TCYB.2019.2904742.

Taylor, M.E. and Stone, P. (2009) 'Transfer Learning for Reinforcement Learning Domains: A Survey', *J. Mach. Learn. Res.*, 10, pp. 1633–1685.

Tom Tango, M.L. and Dolphin, A. (2007) *The Book: Playing the Percentages in Baseball*. Potomac Books.

Tuyls, K. *et al.* (2020) 'Game Plan: What AI can do for Football, and What Football can do for AI', *CoRR*, abs/2011.09192. Available at: https://arxiv.org/abs/2011.09192.

Unity Technologies (2021) 'ML-Agents Toolkit Overview', GitHub. Available at: https://github.com/Unity-Technologies/ml-agents/blob/main/docs/ML-Agents-Overview.md (Accessed: 10 September 2022).

Vaswani, A. *et al.* (2017) 'Attention Is All You Need'. arXiv. doi:10.48550/ARXIV.1706.03762.

Vinyals, O. *et al.* (2019) 'Grandmaster level in StarCraft II using multi-agent reinforcement learning', *Nature*, 575. doi:10.1038/s41586-019-1724-z.

Virgil, C. and Machol, R.E. (1971) 'Operations Research on Football', *Operations Research*, 19(2), pp. 541–544.

Watkins, C. and Dayan, P. (1992) 'Technical Note: Q-Learning', *Machine Learning*, 8, pp. 279–292. doi:10.1007/BF00992698.

Watkins, C.J.C.H. (1989) *Learning from Delayed Rewards*. PhD Thesis. King's College, Oxford.

Weiss, G. (1995) 'Distributed reinforcement learning', *Robotics Autonomous. Systems*, 15, pp. 135–142.

Wesel, P. van and Goodloe, A. (2017) 'Challenges in the Verification of Reinforcement Learning Algorithms'. Available at: https://ntrs.nasa.gov/api/citations/20170007190/downloads/20170007190.pdf (Accessed: 14 September 2022).

Wijmans, E. *et al.* (2019) 'Decentralized Distributed PPO: Solving PointGoal Navigation', *CoRR*, abs/1911.00357. Available at: http://arxiv.org/abs/1911.00357.

Wolpert, D.H. and Tumer, K. (1999) 'An Introduction to Collective Intelligence'. arXiv. doi:10.48550/ARXIV.CS/9908014.

Wolpert, D.H. and Tumer, K. (2001) 'Optimal Payoff Functions For Members Of Collectives', *Advances in Complex Systems*, 04(02n03), pp. 265–279. doi:10.1142/S0219525901000188.

Won, J., Gopinath, D. and Hodgins, J. (2021) 'Control Strategies for Physically Simulated Characters Performing Two-Player Competitive Sports', *ACM Trans. Graph.*, 40(4). doi:10.1145/3450626.3459761.

Wong, A. *et al.* (2021) 'Multiagent Deep Reinforcement Learning: Challenges and Directions Towards Human-Like Approaches', *CoRR*, abs/2106.15691. Available at: https://arxiv.org/abs/2106.15691.

Wydmuch, M., Kempka, M. and Jaskowski, W. (2018) 'ViZDoom Competitions: Playing Doom from Pixels', *CoRR*, abs/1809.03470. Available at: http://arxiv.org/abs/1809.03470.

Xenopoulos, P. and Silva, C. (2021) 'Graph Neural Networks to Predict Sports Outcomes', in *Proceedings of the 2021 IEEE International Conference on Big Data*, pp. 1757–1763. doi:10.1109/BigData52589.2021.9671833.

Xi, L. *et al.* (2018) 'Smart generation control based on multi-agent reinforcement learning with the idea of the time tunnel', *Energy*, 153, pp. 977–987. doi:https://doi.org/10.1016/j.energy.2018.04.042.

Yam, D. and Lopez, M. (2019) 'What was lost? A causal estimate of fourth down behavior in the National Football League', *Journal of Sports Analytics*, 5, pp. 1–15. doi:10.3233/JSA-190294.

Yang, Y. *et al.* (2020) 'Mean Field Multi-Agent Reinforcement Learning'. arXiv. Available at: https://arxiv.org/abs/1802.05438.

Yang, Y. and Wang, J. (2020) 'An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective', *CoRR*, abs/2011.00583. Available at: https://arxiv.org/abs/2011.00583.

Yannakakis, G.N. and Togelius, J. (2018) *Artificial Intelligence and Games*. Cham, Switzerland: Springer.

Yu, C. *et al.* (2021) 'The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games', *CoRR*, abs/2103.01955. Available at: https://arxiv.org/abs/2103.01955.

Yurko, R., Ventura, S. and Horowitz, M. (2018) 'nflWAR: A Reproducible Method for Offensive Player Evaluation in Football'. arXiv. doi:10.48550/ARXIV.1802.00998.

Zai, A. and Brown, B. (2020) *Deep Reinforcement Learning in Action*. New York, NY: Manning Publications.

Zhang, K., Yang, Z. and Basar, T. (2019) 'Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms', *CoRR*, abs/1911.10635. Available at: http://arxiv.org/abs/1911.10635.

Zhao, Y. *et al.* (2019) 'On Multi-Agent Learning in Team Sports Games', *CoRR*, abs/1906.10124. Available at: http://arxiv.org/abs/1906.10124.

Zhong, Y., Zhou, Y. and Peng, J. (2020) 'Efficient Competitive Self-Play Policy Optimization', *CoRR*, abs/2009.06086. Available at: https://arxiv.org/abs/2009.06086.

Zhuang, F. *et al.* (2019) 'A Comprehensive Survey on Transfer Learning', *CoRR*, abs/1911.02685. Available at: http://arxiv.org/abs/1911.02685.

Ziebart, B.D. *et al.* (2008) 'Maximum Entropy Inverse Reinforcement Learning', in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI Press (AAAI'08), pp. 1433–1438.

Appendix A

Software and Tools

A list of software and tools used to support the development process:

- Unity Personal: 2020.325f1
  - Unity MLAgents Toolkit: 2.1.0-exp.1
  - Barracuda Inference Library: 2.0.0-pre.3
- Anaconda3: 2.1.1
- Python: 3.9.7
- Python MLAgents: 0.29.0
- Tensorboard: 2.9.1
- PyTorch 1.11.0
- CudaToolKit: 11.3.1
- Microsoft Visual Studio Community Ed: 2019 v16.11.8
- Blender: 3.2.0
- Windows 10 home 21H2
- Notepad++
- Google Docs
- Overleaf online latex editor

- GitHub Desktop
- Zotero
- Asana

Training Machine Specification:

- Lenovo Legion T5 28IMB05
- Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
- 16.0GB RAM
- Nvidia GeForce RTX 2070 Super

Appendix B

Hyperparameters

Reinforcement Learning:

```
behaviors:
  FootballOffense:
    trainer_type: poca
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: constant
    network_settings:
      normalize: false
      hidden_units: 512
      num_layers: 2
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 5
    max_steps: 200000000
```

```
    time_horizon: 1000
    summary_freq: 10000
    self_play:
      save_steps: 50000
      team_change: 200000
      swap_steps: 1000
      window: 10
      play_against_latest_model_ratio: 0.5
      initial_elo: 1200.0
  FootballDefense:
    trainer_type: poca
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: constant
    network_settings:
      normalize: false
      hidden_units: 512
      num_layers: 2
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 5
    max_steps: 200000000
    time_horizon: 1000
    summary_freq: 10000
    self_play:
      save_steps: 50000
```

```
      team_change: 200000
      swap_steps: 1000
      window: 10
      play_against_latest_model_ratio: 0.5
      initial_elo: 1200.0

# Enable when training from a build or in the cloud.
env_settings:
  base_port: 5004
  num_envs: 5
```

Additional Imitation Learning parameters used for both networks.

```
    gail:
        strength: 0.01
        gamma: 0.99
        demo_path: Assets/Demos/Offense/
    behavioral_cloning:
        demo_path: Assets/Demos/Offense/
        strength: 0.025
        steps: 2500000
```