Final Year Project

Gamma-Star

Eoghan Hogan

Student ID: 17335293

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science with Data Science

Supervisor: Associate Professor Gianluca Pollastri

UCD School of Computer Science

University College Dublin

December 16, 2020

Table of Contents

1 Project Goals . . . 3

2 Project Specification . . . 4

2.1 Problem Statement . . . 4

2.2 Background . . . 4

2.3 Related Work . . . 4

2.4 Personal Contribution . . . 5

3 Background Work . . . 6

3.1 What is StarCraftII? . . . 7

3.2 StarCraftII's Importance in Machine Learning . . . 8

3.3 Introduction to Reinforcement Learning . . . 9

3.4 Machine Learning in Games . . . 10

3.5 Machine Learning in StarCraftII . . . 12

3.6 Sparse Rewards . . . 13

3.7 Self-Play & Game Theory . . . 13

3.8 Deep Learning . . . 14

3.9 Deep Reinforcement Learning . . . 15

3.10 Agent Evaluation . . . 16

4 Data Considerations . . . 17

5 Project Workplan . . . 18

5.1 Overview . . . 18

5.2 Breakdown of Deliverables . . . 19

Abstract

Many real-world applications require artificial agents to compete in complex environments. Significant progress in Machine Learning is influenced by deep learning, which provides a powerful toolkit for non-linear function approximation using neural networks. These techniques have also proven successful in reinforcement learning problems, yielding significant successes in many areas such as the game of Go and simulated robotics domains. The domain of StarCraftII Is a critical challenge for artificial intelligence research, owing to its status amongst the most challenging professional e-sports and its relevance to the real world in terms of complexity. Over a decade and numerous competitions, the top agents have simplified aspects of the game and utilised superhuman capabilities. Despite these advantages, previous agents have not come close to matching the skill of top StarCraftII players. I am going to address the challenge of StarCraftII using reinforcement learning.

Chapter 1: Project Goals

Core Goals

1. Review the literature on reinforcement learning.
2. Build an AWS workflow for the project.
3. Build out the python StarCraftII Learning Environment (SC2LE).
4. Train an agent to micro-manage Zerg units against a bot using the following different reinforcement learning methods.
    4.1. Q-learning
    4.2. Deep-Q-Learning
    4.3. Actor-Critic
    4.4. Deep deterministic Policy Gradient
5. Analyse and evaluate the various methods of reinforcement learning we performed.

Advanced Goals

6. Design and build an agent architecture and API for training agents to micro, based on the results we obtained.
7. Train agents to micro with all three races using self-play.
    7.1. Zerg
    7.2. Terran
    7.3. Protoss
8. Combine a trained micro agent with a macro bot script to play against scripted bots.

Chapter 2: Project Specification

2.1 Problem Statement

This Final Year Project is on the exploration of reinforcement learning. I will be utilising StarCraftII To understand and implement reinforcement learning Algorithms into StarCraftII using python. The fundamental goal is to use reinforcement learning to create an agent that can exhibit human-like behaviours within the StarCraftII environment.

I will be carrying out my aims using the python wrappers made around Blizzards' StarCraftII protocols made by DeepMind and other StarCraftII python interfaces created by Open AI, University of Oxford, Facebook AI research.

The primary challenges I will be using to train and test an agent are predefined. They will be micromanagement tasks in a pre-built environment created for Multi-Agent reinforcement learning research. The agent's tasks will be made by me using StarCraftII Multi-Agent Challenge Library.

2.2 Background

Reinforcement learning is one of the most exciting areas of current artificial intelligence Research and Development. Many real-world applications require artificial agents to compete and coordinate with other agents in complex environments.

The domain of StarCraftII has emerged as a fundamental challenge for Artificial Intelligence research, owing to its iconic and enduring status among the most demanding professional e-sports and its relevance to the real world in terms of its raw complexity and virtually Infinite ways to play.

2.3 Related Work

Note: *On arxiv alone there have already been 40+ papers referencing SC2 since the start of 2020 alone.* [1]

The most important paper that has inspired me is the AlphaStar [2] paper produced by OpenAI. The research was to build a system that could play professional StarCraftII matches and win against some of the highest skilled players. While the idea of replicating their research is something I would love to do, It is beyond my scope.

One area of particular interest is learning from "Experience," [3] which uses real gameplay from humans to train the agent. I believe replays could be very useful as the StarCraftII community has a wealth of game replays at all skill levels since it came out, giving me a veritable treasure trove of data to use.

There are papers that outline approaches to building agents that can micromanage units in narrow circumstances [4], [5], [6]. Agents that the community has made to micromanage are benchmarks I will use to compare my work.

A big part of macro-management is the build order. A build order is, simply put, a sequence of actions to be followed based on timings or supply; High-grade build orders adapt based on what is going on in the game. Learning build order alone is a challenge in itself. Professionals will spend plenty of time studying and perfecting build orders for many many situations. The build order also relies heavily on what the current meta-game is.

One key component of StarCraftII is balancing out micro and macro. While I have not outlined the problem statement to involve building a full-scale agent, I have looked at work describing ways to train full agents.

Another field of work is a hierarchical approach [7], where the hierarchy involves two levels of abstraction. One is the macro-actions extracted from expert's demonstration trajectories, which can reduce the action space by orders of magnitude yet remain effective. The other is a two-layer hierarchical architecture, which is modular and easy to scale.

2.4 Personal Contribution

As a personal contribution, I will be building a cloud-based StarCraftII reinforcement learning solution. I will be extending StarCraftII Multi-Agent Challenge (SMAC) [8] and creating a wrapper around PySCII to ensure I can deploy my agent architectures on AWS S3 and then test and evaluate my resulting agents.

I will then be using my solution to implement a micro-management reinforcement learning agent and test various methods. The principal techniques I wish to look at in reinforcement learning are the following: Deep Reinforcement Learning [9]. Q-Learning [10], Stochastic Policy Gradient (SPG) [10], Deterministic Policy Gradient (DPG) [10], and Deep-Q-Learning [11].

I aim to compare and contrast these methods on StarCraftII using metrics such as time taken to train and results to evaluate the agents. The data should help decide which would give rise to the most capable agent and possibly which agent could lead to an agent that could play a full game.

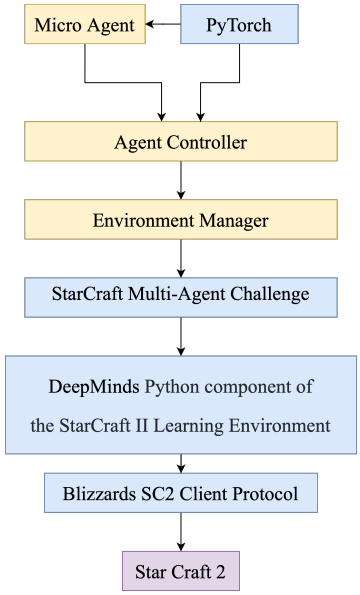


Figure 2.1: Overview of the architecture I want to use where Yellow is the work I have to do

If I can succeed in my initial tasks, I expect to tackle the challenge of hypothesising and deploying a multi-agent-based system that can play a full game of StarCraftII against the built-in scripted bot.

Chapter 3: Background Work

This project is by no means the only project tackling machine learning in StarCraftII [1]. Others have advanced the underlying goals, research teams consisting of many skilled researchers with powerful backings.

> "If I have seen a little further, it is by standing on the shoulders of Giants." [1]

As mentioned, I have been spurred on by the achievements of AlphaStar [2]. In this section, we will review a few key concepts which I believe make the foundation of this project. Firstly I will go over what StarCraftII is and why I believe it has a role in reinforcement learning. Then there will take an introductory look at reinforcement learning followed by machine learning in games. We then will take a look at work others have done in StarCraftII. We will then discuss sparse rewards, self-play, and game theory which are concepts about the gameplay. Using the knowledge up until that point, we will talk about deep learning and then deep reinforcement learning as these are the main concepts that apply to this project. Finally, there will be a quick review of evaluating reinforcement learning agents.

[1] Sir Isaac Newton

3.1 What is StarCraftII?

StarCraftII is a free to play multiplayer video game created by Blizzard Entertainment released on July 27, 2010. [12]

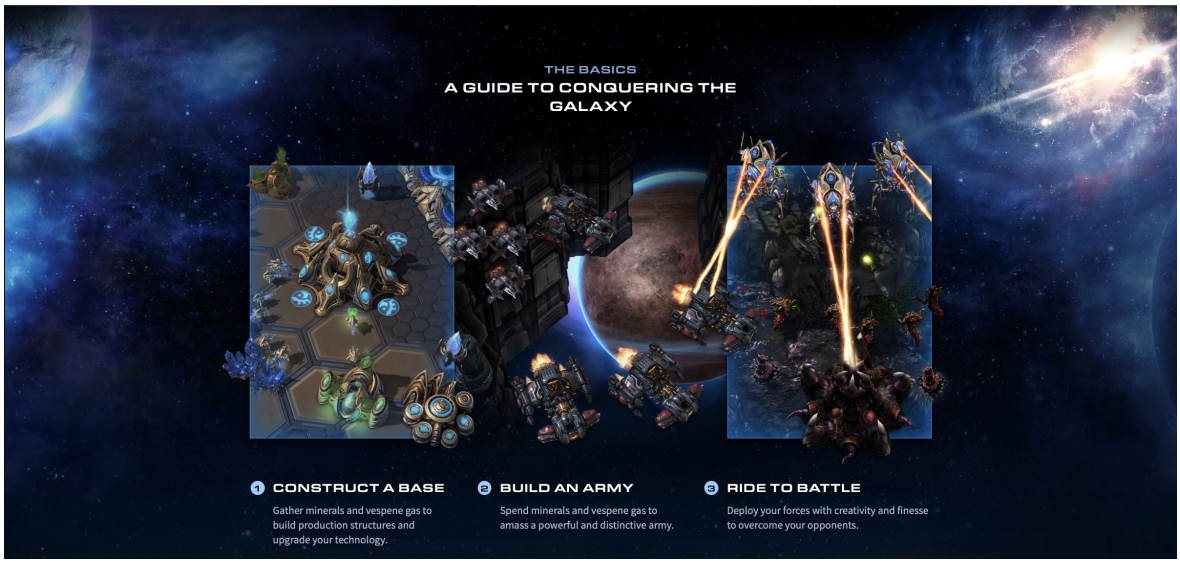


Figure 3.1: StarCraftII Basics Image, accessed 14 December, 2020, https://starcraft2.com/en-gb/game

It is a Real-Time-Strategy game with a single-player campaign where you follow the three races of the game's heads through their journeys. In which you experience playing as the three races of the game.

- Terrans are the descendants of a disastrous colonisation expedition launched from Earth centuries ago.
- The Protoss are an ancient race with highly advanced technology and potent psionic abilities. (Psionics are a form of supernatural powers)
- The Zerg are a biologically focused race composed of many different species integrated into the Swarm via Zerg infestation.

The main attraction of StarCraftII is its professional Player-VS-Player scene in which the top players globally play against each other, testing their reflexes, strategy, and endurance. It is played professionally throughout the world, though much like its predecessor StarCraft Brood War's professional competition, the highest level of play has historically been in South Korea. StarCraftII is the largest e-sports globally and has brought e-sports to the rest of the world in the same way the original brought it to South Korea. Since its release, it has experienced a decline and a more recent resurgence following its transition to a free-to-play business model. Now the top players are in all areas of the world.

3.2 StarCraftII's Importance in Machine Learning

In various ways, games have been a significant way to test and evaluate artificial intelligence systems' performance. As capabilities have improved, the research community has explored games with increasing complexity that capture various skills required to solve scientific and real-world problems.

To win a game of StarCraftII, a player must correctly decide on long term administration of their economy - known as macro - along with low-level command of their units - known as micro. The need to balance short and long-term goals, and adapt to unpredictable conditions, poses a substantial difficulty for intelligent systems that have often tended to be inflexible.

Challenges for intelligent systems to overcome to play a game of StarCraftII includes the following:

- **Game theory knowledge:** There is no single best approach for a player to play; instead, they need to show strategic knowledge.
- **Imperfect information:** Unlike games like chess or Go, where players can see everything, crucial details on their opponents are hidden and this requires players to gather intelligence on their opponent via scouting.
- **Long term planning:** Like many real-world problems, cause-and-effect is not immediate. Games can also take anywhere from less than ten minutes up to one hour to complete. As professionals will tell you: actions taken early in the game are crucial in the long term.
- **Real-Time:** In StarCraftII, players do not alternate between turns; instead, players must perform actions continually as the game progresses. Inaction at any moment can be detrimental.
- **Action Space:** At every moment of play, a player can have anywhere between a few dozen up to over a thousand possible actions to take.

Due to the above points, StarCraftII has emerged as a "grand challenge" for AI research. Not only does an agent need to compete in the game itself, but StarCraftII also has an ever-changing meta-game. The meta-game of StarCraftII is concerned with how playable maps, unit effectiveness and counter units change and how playable strategies change with it. In the professional scene, the players continuously work to figure out new strategies and how to defeat or defend against old and current ones.

In November 2016, Alphabet's DeepMind branch announced a collaboration with Blizzard to create "a useful testing platform for the wider AI research community." In December 2018, DeepMind's StarCraftII agent, AlphaStar, defeated a top StarCraftII player 5-0, albeit under conditions some deemed unfair. A fairer version of AlphaStar attained Grandmaster status in August 2019; an accomplishment called a "landmark achievement" for the field of artificial intelligence. [13]

3.3 Introduction to Reinforcement Learning

In as few words as possible, what is reinforcement learning? [14] Well, if we start with the idea that biological systems appear to be learning from interaction, that is, infants, learn by exploration and have no tutor telling them what to do.

Animals (humans included) can make observations about their environments, and when they exist in their environment, they receive feedback continuously, which drives their understanding of their world. Biological systems also have a notion of actions which do not provide an immediate payoff but in the long term produce a substantial success.

Reinforcement learning is, then, the computational approach of intelligent agents learning from the interaction, modelling how they might explore and exploit an environment to maximise a reward along with how we might teach an agent to understand delayed reward. Reinforcement learning is both a class of solutions and the field which studies problems and its solution methods.

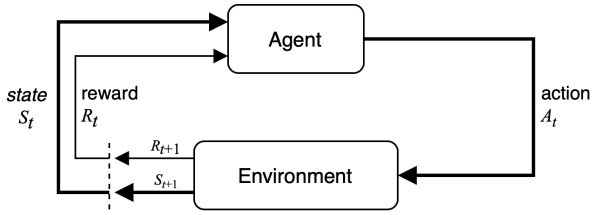


Figure 3.2: The Diagram of Reinforcement

Reinforcement learning is not supervised learning [15]. In learning from interaction, it is not optimal to generate examples/labels, using experts, of desired behaviours that correspond to all situations an agent might encounter. Agents must instead learn from experience and generalise their responses to environmental states. Reinforcement learning is also not unsupervised learning. Typically unsupervised learning is about finding the underlying structure of data which may appear unstructured and has no labelling. I was once often confused when talking to colleagues, considering reinforcement learning and unsupervised learning as synonyms; however, they are not the same thing. Reinforcement learning is the process of learning correct behaviours to maximise a reward function. While understanding the structure of agents experience during its learning via an unsupervised learning method is, by all means practical, that is uncovering structure in the data that the agent is being fed and what the agent is producing. However, unsupervised learning does not sufficiently address the challenges of reinforcement learning. Therefore we are making the distinction.

In reinforcement learning, we have a big challenge between trade-offs, short-term gain, versus long-term gain and exploration versus exploitation. These two are big dilemmas in the field and researchers still tackle them. Also, these two trade-offs tie into each other, an agent that focuses too much on exploitation will, by default, fall into the trap of maximising short-term gains. The crux of the issue is that an agent cannot focus on just exploration as it will not learn any methods of exploitation. Conversely, an agent that only exploits its environment will end up not exploring actions that may be beneficial to it in the long run.

3.4 Machine Learning in Games

Games, as mentioned previously, make for a suitable environment for researching reinforcement learning Models [16], [17]. How can we train agents to act smartly in a given scenario while working with sparse rewards?

3.4.1 Board Games

When we first started creating intelligent systems chess was the golden standard, but then On May 11, 1997, IBM's Deep Blue played Garry Kasparov and won which was the first time a machine won a chess game against a reigning world champion [18].

Figure 3.3: Gray Kasparov vs Deepblue, Accessed 15 December, 2020, https://www.pri.org/stories/2018-01-05/garry-kasparov-and-game-artificial-intelligence

Since then beating humans at chess has become a relatively trivial task. Chess is "solved", this is what we say when machines are much better than humans at a challenge. From then on the next board game for machine learning researchers to tackle was Go. Go is a board game that is over two thousand five hundred years old. Chess has about 10 to the power of 120 possible games whereas Go has approximately 10 to the power of 170 possible games. And In October 2015, AlphaGo [19], [20] played its first match against the reigning three-time European Champion, Mr Fan Hui. AlphaGo, a reinforcement learning agent, won the first-ever game against a Go professional with a score of 5-0.

Figure 3.4: Lee Sedol vs alphago. Photographer: Lee Jin-man/AP, Accessed 15 December, 2020, https://www.theguardian.com/technology/2016/mar/15/alphago-what-does-google-advanced-software-go-next

3.4.2 Video Games

Researches have expanded their horizons when looking for ideal ways to train and evaluate agents, and it is no surprise that video games have become the standard. Video games proved controllable and reproducible environments. Furthermore, video games offer parallelism and automatically labelled data.

To further normalise experimentation in the field, researchers introduced the Arcade Learning Environment (ALE) [21] which laid out a standard API for agents to interface with 500 Atari games. The ALE has allowed for many researchers to carry out reinforcement learning experiments while allowing for a similar protocol so that others may reproduce their work.

Facebook research then built on top of the ALE and provide what they call the Extensive Lightweight, and Flexible (ELF) [22] platform which adds Real-Time Strategy games and board games and their goal were to minimise computational power so that researchers could benchmark state-of-the-art end-to-end agents on laptop hardware.

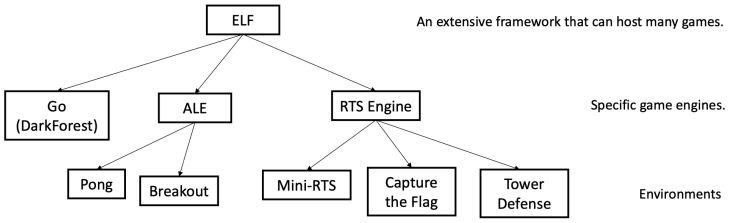


Figure 3.5: List of Games in ELF

Another method that has been built on to offer a standardised method for conducting reinforcement learning experiments is Open Ai's gym [23], [24]. The gym library is a collection of test problems that you can use to work out your reinforcement learning algorithms. These environments have a shared interface, allowing you to write general algorithms.

OpenAI also is the powerhouse of reinforcement learning research dominating human experts on video games. In April of 2019 OpenAI Five [25], [26] became the first machine learning system to defeat professional Dota 2 Players.

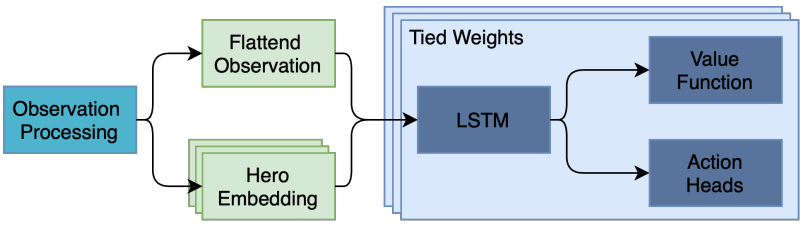


Figure 3.6: The Architecture of OpenAI Five

OpenAI Five was a reinforcement learning research project which involved large scale, thousands of GPU, time-consuming, multiple months, training.

Lastly, there is another platform for reinforcement learning research with video games called DeepMind lab [27]. DeepMind Lab is a first-person 3D game platform designed for research. As per DeepMinds paper, the API was designed and built with reinforcement learning in mind. DeepMind developed this reinforcement learning API that was constructed on top of the Quake game engine to provide agents with complex observations and a diverse set of actions they can carry out.

3.5 Machine Learning in StarCraftII

StarCraftII is the focus of this paper. For interacting with StarCraftII DeepMind built PySC2 [28], [29] which is a Python library that is a wrapper around Blizzard Entertainment's StarCraftII machine learning Protocol [30]. The PySC2 Python library was designed and produced as a product of a collaboration between DeepMind and Blizzard. The idea behind the partnership was to build StarCraftII into an environment that could be a gold standard for reinforcement learning research. The PySC2 Library allows for agents to make observations from the game and send actions. This library is what I will be using in my project.

AlphaStar [2] is my source of inspiration. DeepMind Collaborated with Blizzard as we said above to build PySC2. DeepMind used PySC2 and the most extensive set of anonymised game replays ever released to train the breakthrough agent, AlphaStar. AlphaStars Architecture is Massive. [2]

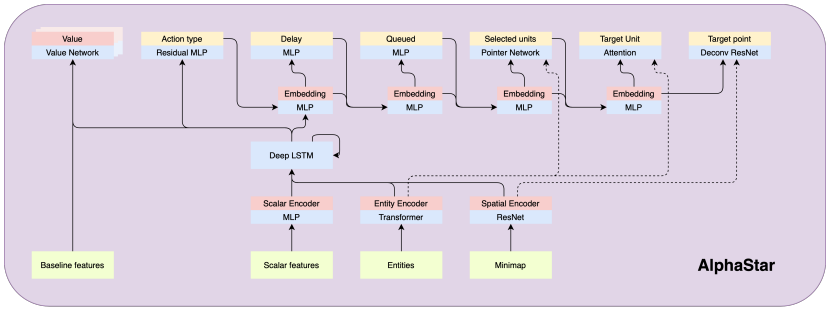


Figure 3.7: AlphaStar Model Architecture

AlphaStar had a complex training process. [2] The research team used both Supervised and reinforcement learning to train the final pool of agents. They used supervised learning to Initialise the models, and then reinforcement learning is used to upskill them. There were multiple reinforcement learning methods used, actor-critic [31], temporal difference [32], Vtrace, and their self imitation algorithm. OpenAI applied these four algorithms along with self-play [33] to train the agents in a league environment where DeepMind played against each other. Along with this, the researchers added new agents and froze old ones in what is known as fictitious self-play. This process took a total of 44 days.

DeepMind tested the final agents that were the most successful in the training league. The top agents that DeepMind trained were assigned a skill level, on the open multiplayer ladder, above 99.8% of ranked players in StarCraftII. This ranking means that AlphaStar has achieved the highest level of play for an agent in any widespread e-sport without simplifying the game.

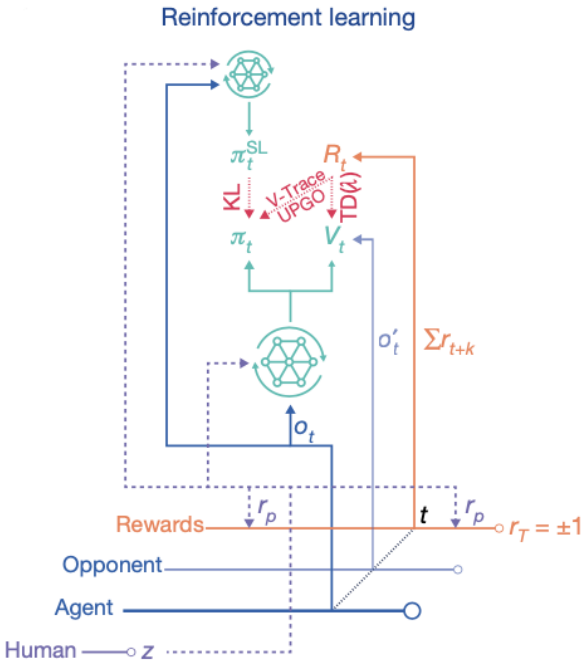


Figure 3.8: AlphaStars Reinforcement Learning Architecture

3.6 Sparse Rewards

In reinforcement learning, the agents need to learn to take advantage of the reward systems in different games. If playing at a professional level players are expected to know how their play benefits them. Problems arise due to the reward systems games use. Some games may continually provide an agent with positive feedback, non-sparse rewards, which an agent can use to learn. However, lots of video games, including all professional e-sports, feature a sparse reward system that does not provide agents with a wealth of information about their play.

To help overcome the problem of sparse rewards, researchers have developed novel methods [34]. In the training of AlphaStar [2], they used three different types of reinforcement learning Methods, as mentioned above. Other researchers have proposed to leverage demonstrations to help Deep Deterministic Policy Gradient methods [35] of overcoming the problems of sparse rewards. The demonstrations are of the form of reinforcement learning transitions.

3.7 Self-Play & Game Theory

Self-Play [33] is, for the sake of simplicity, when agents play games against "themselves". The idea of self-play is that it allows agents to train themselves magnitudes of times faster than if they were to play against humans. This type of learning, self-play, has been used in all significant reinforcement learning breakthroughs in games including AlphaGo [19], AlphaStar [13] and OpenAI-Five [26].

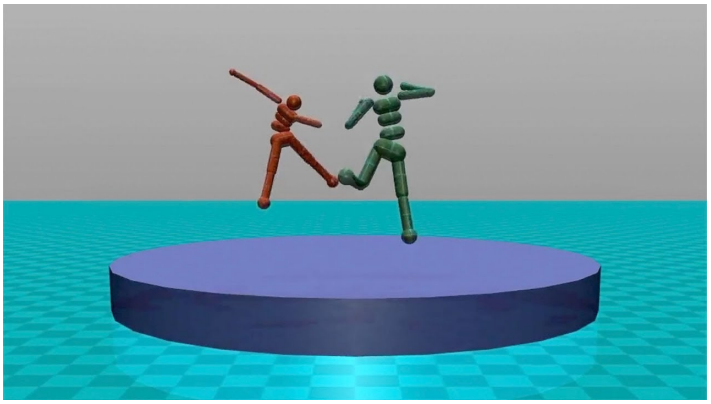


Figure 3.9: Example of Self Play in fighting characters

Game theory [36] has to do with the study of interactions between rational agents. Game theory is involved when studying StarCraftII. An agent-versus-agent game is an interaction between two rational agents, and thus we must concern ourselves with the concepts involved.

A problem that emerges when training agents using self-play is that agents naturally fall into an equilibrium in which one agent, A, beats agent, B, in one game and then the reverse happens in the next game. Or when using more than two agents cycles may appear where agent A beats agent B, agent B beats agent C and agent C beats agent A. In these equilibrium scenarios, no new information is getting generated - when the agents play this way their understanding of the games plateaus. Researchers are interested in pushing back this equilibrium, known as Nash equilibrium [37] so that agents don't fall into this tit-for-tat state until they have reached full potential.

3.8 Deep Learning

Deep learning [38] has been the reason for many advancements in machine learning lately. It is the proverbial tree that keeps on giving. Deep learning essentially allows giant models that are composed of multiple Neural Layers to learn data representation at various levels of abstraction. Most modern deep learning models are artificial neural networks with just many hidden layers, but there are other architectures as well. The type of deep learning models we see most often are convolutional neural networks [39], recurrent neural networks [40], and deep neural networks. However, there are other forms of deep learning models which include latent variables organised layer-wise in deep generative models such as the nodes in deep belief networks [41] and deep Boltzmann machines [42].

Deep learning [38] has dramatically improved machine learning and has achieved state-of-the-art performance in almost every task where researchers have used it as a solution method. Deep learning has enabled breakthroughs in speech recognition, object-recognition, drug discovery, genomics, text-to-speech, and of course, video gameplay.

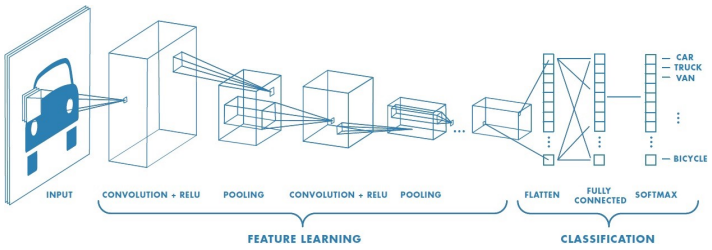


Figure 3.10: Convolutional Neural Network

To note on one of the applications stated above, convolutional neural networks [39] have caused some of the most significant publicity due to object recognition and the advent of autonomous vehicles.

Here we can see where the term "Deep" gets its name. There are many, many Layers involved in this type of network and the update rules and optimisation methods are vast.

3.9 Deep Reinforcement Learning

Deep learning and reinforcement learning [11], [10] have gone together considerably well. Deep learning [38] has been helping reinforcement learning make many breakthroughs because it enables automatic feature engineering and end-to-end learning through gradient descent. Feature engineering usually takes time and is not fully complete; however, deep learning enables the exploitation of the hierarchical composition of data and can help combat the curse of dimensionality.

Deep reinforcement learning such as deep-Q-learning (DQN) [11] and deep deterministic policy gradient (DDPG) [43] are just some of the methods used since around 2015 to reach superhuman levels of play in many games and far outclassed the reinforcement learning methods that came before them.

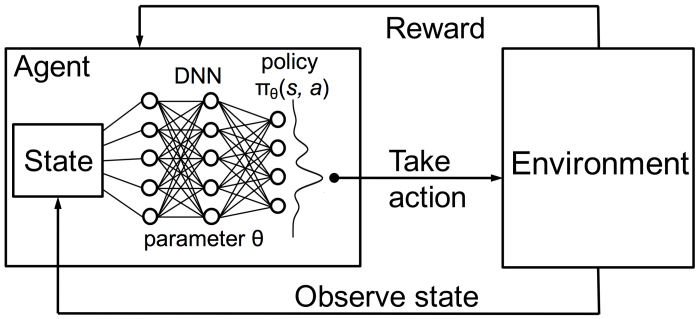


Figure 3.11: The Reinforcement Learning cycle for Deep Reinforcement Learning

Deep reinforcement learning is currently, and has been, used to play StarCraftII multiple times now [1]. As has been noted several times now, the DeepMind team created an agent, called AlphaStar [2], which was a strong enough player that it got placed in the same rank as the top 0.2% of players. Other researchers have achieved a 93% win rate against the most challenging built-in bot using hierarchical reinforcement learning with deep networks [44]. A team of students in Hong Kong were able to train an agent to micro-manage like a human on the Python StarCraftII learning environment using Neuro-Evolution and deep reinforcement learning [5].

3.10 Agent Evaluation

Agent evaluation is an exciting topic. [45] in reinforcement learning. The big projects we have looked at up until this point were AlphaGo [20], AlphaStar [2] and OpenAI-five [25]. These groundbreaking achievements were all evaluated against professional players in their respective domain. This method of using professionals is not the only way to assess an agents performance.

Seeing as it is generally much too hard for researchers to develop agents like AlphaStar due to time and resource limitations, there have been other methods proposed to measure agent performance rather than playing against professional players. Here we introduce the StarCraftII Multi-Agent Challenge (SMAC) [8], [46]. The StarCraftII Multi-Agent Challenge is a Python library [46] that provides us with multiple handcrafted scenarios in StarCraftII. All the scenarios provided are comprised of micro-management tasks; fortunately, this is our goal.

While Facebook designed SMAC [8]for multi-agent research in mind, we can still take a general idea and use it for ourselves. Furthermore, the StarCraftII Learning environment provides us with a handful of mini-games. These mini-games consist of balanced scenarios where a model must defeat the opponent but is not an entire full-length game. The idea behind these mini-games is to simplify the game. From my exploration, it appears that you can build your custom mini-games using the StarCraftII map builder and a simple scripted bot.

Chapter 4: Data Considerations

For this project, there are some considerations to make around training data. As we have seen DeepMind initialised their models via supervised learning, and if I were to undertake the same plan, I would need data from starcraft games.

Thankfully there are multiple websites where you can freely download game replays. The wealth of data is so vast that Facebook researchers decided to aggregate data into a research dataset which is available also [47]

I have spent the time searching for replay data, and the challenge is deciding which replays you want. There are more replays freely available than I could ever want, but it is challenging to know what exactly you want from a replay. We may say that we will use Stochastic gradient descent to train an agent on the end-to-end task of playing a full-length game, but this is infeasible given the timeframe of the project and the resources available.

What this project requires is examples of micro-management in starcraft. Every replay contains micro-management, and the PySC2 [28], [48] library allows us to tap into these replays and extract the actions that a human was performing at a given time segment.

Chapter 5: Project Workplan

The project has a fourteen-week timeframe. To complete the core goals of the project, I have developed a rigorous timeline with specific deliverables in place.

5.1 Overview

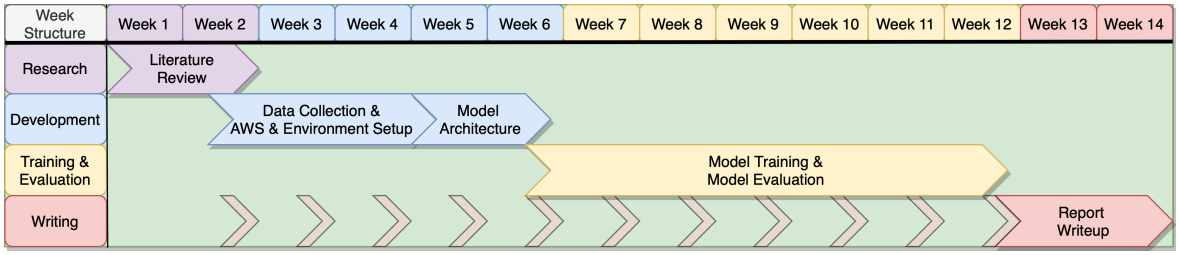
As a bird's eye view of the tasks that I need complete, I have created a brief project Timeline. There are four central states of the project which we can see in the project chart.

A literature review is where I will gather all relevant knowledge. Knowledge gathering is a crucial stage, and the rest of the project depends on it. Fully understanding the literature and organising all learning resources I can find will be critical in the long run as it will save time in the rest of the project.

The Development stage encompasses all non-model related development. Development involves the tasks of making sure all my libraries work together. A significant step in this is making sure I can get all the components to run on AWS and that I can quickly get the data back to my local machine.

The Training and Evaluation steps are self-explanatory. Training will involve much waiting, so in this section, I will also be working on the report while waiting for models to train.

The Report Write up stage is the final hurdle of the project and will involve the completion of the final deliverable, the report.



5.2 Breakdown of Deliverables

1. **Literature Review**
    1.1. **Week 1** - Review works in Deep Reinforcement Learning and the algorithms involved.
    1.2. **Week 2** - Review reinforcement learning in StarCraftII.
2. **Development (Week 3 to Week 6)**
    2.1. **Data Collection** - If data needs to be collected this is the time that I should find, scrape, review and store it.
    2.2. **Python Library Wrappers** - Build out the Libraries I will be using and make sure everything works.
    2.3. **AWS Setup and Scripts** - This is the time I have allocated to ensure that I can deploy and train models in the cloud. It will involve the need of ensuring everything can be worked on and monitored through AWS.
    2.4. **Model Architecture** - This is the time Allocated to building the Architecture I will be using for training and building the actual model itself in PyTorch.
3. **Training and Evaluation (week 6 to Week 12)**
    3.1. **Training** - The training phase is where I will train and test the Agents using the methods I proposed in my core goals.
    3.2. **Evaluation** - Evaluation will be the process of collecting data based on the training and testing and organizing it into a meaningful graphical explanation. Evaluation will also consist of seeing how well the Agents generalized.
4. **Report Write Up (Week 13 & Week 14)**
    - Allocated time to fully complete the report write up. As seen from the diagram above I also intend to spend at least a few hours a week on the report throughout the duration of the project.

Bibliography

1. Arxiv. *All Starcraft Papers* https://arxiv.org/search/?query=starcraft&searchtype=all&source=header.
2. Vinyals, O. *et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575,** 350–354 (2019).
3. Zhao, D., Wang, H., Shao, K. & Zhu, Y. *Deep reinforcement learning with experience replay based on SARSA* in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016), 1–6.
4. Shao, K., Zhu, Y. & Zhao, D. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence* **3,** 73–84 (2018).
5. Desmukh, F. M. Evolving Human-like Micromanagement in StarCraft II with Neuro-Evolution and Reinforcement learning (2019).
6. Shantia, A., Begue, E. & Wiering, M. *Connectionist reinforcement learning for intelligent unit micro management in starcraft* in *The 2011 International Joint Conference on Neural Networks* (2011), 1794–1801.
7. Barto, A. G. & Mahadevan, S. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems* **13,** 41–77 (2003).
8. Samvelyan, M. *et al.* The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).
9. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *nature* **518,** 529–533 (2015).
10. Li, Y. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017).
11. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G. & Pineau, J. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560* (2018).
12. Blizzard. *StarCraftII Website* https://starcraft2.com.
13. Vinyals, O. *et al. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II* https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/. 2019.
14. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).
15. Cunningham, P., Cord, M. & Delany, S. J. in *Machine learning techniques for multimedia* 21–49 (Springer, 2008).
16. Robertson, G. & Watson, I. A review of real-time strategy game AI. *Ai Magazine* **35,** 75–104 (2014).
17. Risi, S. & Preuss, M. From Chess and Atari to StarCraft and Beyond: How Game AI is Driving the World of AI. *KI-Künstliche Intelligenz* **34,** 7–17 (2020).
18. Campbell, M., Hoane Jr, A. J. & Hsu, F.-h. Deep blue. *Artificial intelligence* **134,** 57–83 (2002).
19. Silver, D. *et al. AlphaGo the story so far* https://deepmind.com/research/case-studies/alphago-the-story-so-far.
20. Silver, D. *et al.* Mastering the game of go without human knowledge. *nature* **550,** 354–359 (2017).

21. Bellemare, M. G., Naddaf, Y., Veness, J. & Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* **47,** 253–279 (2013).
22. Tian, Y., Gong, Q., Shang, W., Wu, Y. & Zitnick, C. L. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. *Advances in Neural Information Processing Systems* **30,** 2659–2669 (2017).
23. Brockman, G. *et al.* Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
24. OpenAI. *Gym* https://gym.openai.com.
25. Berner, C. *et al.* Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).
26. OpenAI. *OpenAI Five Defeats Dota 2 World Champions* https://openai.com/blog/openai-five-defeats-dota-2-world-champions/.
27. Beattie, C. *et al.* Deepmind lab. *arXiv preprint arXiv:1612.03801* (2016).
28. Vinyals, O. *et al.* Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).
29. DeepMind. *PySC2 - StarCraft II Learning Environment* https://github.com/deepmind/pysc2.
30. Blizzaed. *s2client-proto* https://github.com/Blizzard/s2client-proto.
31. Konda, V. R. & Tsitsiklis, J. N. *Actor-critic algorithms* in *Advances in neural information processing systems* (2000), 1008–1014.
32. Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning* **3,** 9–44 (1988).
33. Oh, J., Guo, Y., Singh, S. & Lee, H. Self-imitation learning. *arXiv preprint arXiv:1806.05635* (2018).
34. Gaina, R. D., Lucas, S. M. & Pérez-Liébana, D. *Tackling sparse rewards in real-time games with statistical forward planning methods* in *Proceedings of the AAAI Conference on Artificial Intelligence* **33** (2019), 1691–1698.
35. Vecerik, M. *et al.* Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* (2017).
36. Leyton-Brown, K. & Shoham, Y. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis lectures on artificial intelligence and machine learning* **2,** 1–88 (2008).
37. Holt, C. A. & Roth, A. E. The Nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences* **101,** 3999–4002 (2004).
38. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521,** 436–444 (2015).
39. Yan, L., Yoshua, B. & Geoffrey, H. Deep learning. *nature* **521,** 436–444 (2015).
40. Miao, Y., Gowayyed, M. & Metze, F. *EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding* in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (2015), 167–174.
41. Hinton, G. E. Deep belief networks. *Scholarpedia* **4,** 5947 (2009).
42. Salakhutdinov, R. & Hinton, G. *Deep boltzmann machines* in *Artificial intelligence and statistics* (2009), 448–455.
43. Lillicrap, T. P. *et al.* Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
44. Pang, Z.-J. *et al. On reinforcement learning for full-length game of starcraft* in *Proceedings of the AAAI Conference on Artificial Intelligence* **33** (2019), 4691–4698.
45. Hingston, P. A turing test for computer game bots. *IEEE Transactions on Computational Intelligence and AI in Games* **1,** 169–186 (2009).

46. Samvelyan, M. *et al. Python MARL framework* https://github.com/oxwhirl/smac.

47. Lin, Z., Gehring, J., Khalidov, V. & Synnaeve, G. Stardata: A starcraft ai research dataset. *arXiv preprint arXiv:1708.02139* (2017).

48. DeepMind. *liquipedia Replay Websites* https://github.com/deepmind/pysc2.