GAME AI OF STARCRAFT II BASED ON DEEP REINFORMENT LEARING

by

Junjie Luo

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

May 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Dmitri Gusev, Chair

Department of Computer and Information Technology

Dr. Dominic Kao

Department of Computer and Information Technology

Dr. David Whittinghill

Department of Computer Graphics Technology

Approved by:

Dr. ERIC MATSON

Dedicated to the grace of God

ACKNOWLEDGMENTS

This research is for master thesis from the Computer and Information Technology Program at Purdue University, with the help of Prof. Gusev, Prof. Kao and Prof. Whittinghill. The copies of references are from Purdue Library.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PySC2	Python StarCraft Learning Environment
DL	Deep Learning
ML	Machine Learning
MDP	Markov Decision Process
DRL	Deep Reinforcement Learning
DQL	Double Q-Learning
PPO	Proximal Policy Optimization

ABSTRACT

The research problem of this article is the Game AI agent of StarCraft II based on Deep Reinforcement Learning (DRL). StarCraft II is viewed as the most challenging Real-time Strategy (RTS) game for now, and it is also the most popular game where researchers are developing and improving AI agents. Building AI agents of StarCraft II can help researchers on machine learning figure out the weakness of DRL and improve this series of algorithms. In 2018, DeepMind and Blizzard developed the StarCraft II Learning Environment (PySC2) to enable researchers to promote the development of AI agents. DeepMind started to develop a new project called AlphaStar after AlphaGo based on DRL, while several laboratories also published articles about the AI agents of StarCraft II. Most of them are researching on the AI agents of Terran and Zerg, which are two of three races in StarCraft II. AI agents show high-level performance compared with most StarCraft II players. However, the performance is far from defeating E-sport players because Game AI for StarCraft II has large observation space and large action space. However, there is no publication on Protoss, which is the remaining and most complicated race to deal with (larger action space, larger observation space) for AI agents due to its characteristics. Thus, in this paper, the research question is whether the AI agent of Protoss, which is developed by the model based on DRL, for a full-length game on a particular map can defeat the high-level built-in cheating AI. The population of this research design is the StarCraft II AI agents that researchers built based on their DRL models, while the sample is the Protoss AI agent in this paper. The raw data is from the game matches between the Protoss AI agent and built-in AI agents. PySC2 can capture features and numerical variables in each match to obtain the training data. The expected outcome is the model based on DRL, which can train a Protoss AI agent to defeat high-level game AI agents with the win rate. The model includes the action space of Protoss, the observation space and the realization of DRL algorithms. Meanwhile, the model is built on PySC2 v2.0, which provides additional action functions. Due to the complexity and the unique characteristics of Protoss in StarCraft II, the model cannot be applied to other games or platforms. However, how the model trains a Protoss AI agent can show the limitation of DRL and push DRL algorithm a little forward.

Keywords: Game AI, Deep Reinforcement Learning, StarCraft

INTRODUCTION

This paper aims to use the model based on Deep Reinforcement Learning to build a Protoss (one of the Races in StarCraft II) AI agents which can defeat high-level built-in AI agents in StarCraft II.

When it comes to Game AI, people often talk about AlphaGo (D Silver et al. 2016), which was developed by DeepMind Technologies and known by the public. It is because it defeated top player of the board game Go, Lee Sedol, in 2016 and No.1 ranked Go player Ke Jie at the 2017 Future of Go Summit. This event was broadcasted by newspapers and television programs, making it extraordinarily accessible at that time. After that, increasing numbers of researchers try to use Game AI to imitate human behaviors and defeat top players based on the technologies of artificial intelligence.

In video games, artificial intelligence (AI) is not exactly what we call “AI” in the academy terms. It is not exactly Machine Learning Technologies. Defined by Wikipedia, in video games, AI is used to generate responsive, adaptive or intelligent behaviors primarily in non-player characters (NPCs) similar to human-like intelligence. The AI of most video games on the market is based on pre-set scripts instead of Machine Learning Technologies. A script-based AI agent is hard to reach the same level of an AI agent based on Machine Learning Technologies.

Meanwhile, the combination of Deep Learning and Reinforcement Learning, Deep Reinforcement Learning, which is presented by DeepMind, is also considered as an ideal system for training competitive Game AI agents, not only because of the success of AlphaGo, but also due to the first significant trial of Atari (V Mnih et al. 2015), which shows the excellent

performance of Deep Reinforcement Learning. Recently, OpenAI Five (OpenAI, 2018) which was developed by OpenAI, defeated the world champion team OG in Dota2 that is more complex than Go. A new target should be made in order to improve the technologies of Game AI. StarCraft II, as a result, becomes more and more popular in AI areas.

StarCraft II, which is a real-time strategy (RTS) game published by Blizzard in 2010, is much more complicated than Dota2, Go and Atari. Players need to make multiple decisions in a second to determine their actions, such as expanding their territories, training their army and forces, building kinds of buildings, move their army, collecting resources, interfering with the enemies' development. It is just like that players play a role as the commanders in a tiny war. AI cannot think of so many things and behave like a human for now. Besides, there are three races in the game: Terran, Zerg and Protoss. Each of these three races has its unique characteristics, and players always make tactics based on their characteristics, which means that it is challenging to create an agent that is suitable for all three races.

StarCraft II was welcomed by amateurs of Game AI at the beginning, using simple logic by Python Script. Famous gif images of StarCraft II, such as using Marines (units of Terran) to dodge the attack of Lurkers (units of Zerg), were created by those amateurs. Humans cannot do the actions in the gif images because of the harsh requirement for accuracy and speed. Indeed, AI can do many unbelievable actions to defeat humans. However, the single tactic of AI agents, which is set by programs, will be defeated by players after several trials due to the fact that players can easily find the vulnerability of the single tactic. Despite that, there are plenty of AI resources shared by those amateurs in the menu of the game. Besides, Blizzard always offers API to those players who are engaging in making special maps or agents for their games such as

StarCraft, Warcraft III, Overwatch, and StarCraft II, and most of the agents before are based on the tools Blizzard provides. That makes it popular in many professional players' forums.

Therefore, StarCraft II plays a role as an excellent platform for researchers who are working on Deep Reinforcement Learning and Game AI. Furthermore, there are no as successful agents for StarCraft II as Dota2 and AlphaGo due to its complexity.

This paper focuses on the StarCraft AI agents for a full-length game with some restrictions: Race and Map. There are only a few trials on creating an AI to defeat the built-in AI agents made by Blizzard. The agent in this paper is trained and play game matches on the map "AbyssalReefLE", and the races of the agent and the enemy are Protoss and Terran. This paper tries to train an AI agent which can defeat the built-in AI agents in a full-length game with these limitations. The next part of the thesis investigates the previous works on Deep Reinforcement Learning and Game AI agents, including agents of StarCraft II for now.

Research Justification

For now, researchers who work on DRL have not published an article about Protoss Game AI agents. Before October 2019, DeepMind was still testing AlphaStar. There are merely two articles about StarCraft II Game AI built by DRL and PySC2. Both two articles are Zerg Game AI agents. Besides, these Game AI agents cannot perform as well as AlphaGo because of larger observation space and action space. The action space of each race in StarCraft II is significantly different.

Therefore, we hope that what we work on can help understand whether DRL can improve the performance of Protoss AI agents. We also would like to build an action space for Protoss. According to Tencent AI lab's publication, the attack action space is extraordinarily broad,

which is larger than the sum of other action space. For Protoss, whose attack actions are more complex than the other two races, it is a challenge to build such large action space. We also hope that our work can help others have a better understanding of Protoss Game AI agents.

LITERATURE REVIEW

The learning model is based on Deep Reinforcement Learning (DRL). The first deep learning model which can successfully learn control policies directly from high-dimensional sensory input using reinforcement learning was presented by DeepMind in 2013, as well as the concept of DRL (V Mnih et al. 2013). The learning model is a combination of Deep Learning and Reinforcement Learning. Many famous Game AI agents that are well-known by the public, like AlphaGo, OpenAI Five, are using this model. Thus, in this section, the article discusses the related works about Deep Learning, Reinforcement Learning firstly. Then, it describes the Game agents using DRL for now and the current development of StarCraft II AI agents.

Additionally, the assumptions and tactics spread in the forums of core players are also listed in this section.

Deep Learning

Deep Learning is not a specific algorithm, but a class of algorithms which extract features from raw input, such as images and texts, by multiple levels or layers, and it is also a subclass of machine learning. Like Donald Michle described machine learning in 1968, Deep Learning also aims to let the computer learn from experience to increase their usefulness (D Michle 1994). Deep Learning is widely used in many areas such as bioinformation, medical image analysis, advertising, computer vision, natural language processing.

Deep Learning was first used in the machine learning area by Rina Dechter in 1986. In her paper, she concentrates on constraint recording in the dead-end situation of the backtrack search and treats it as the process of learning to enhance the search efficiency (R Dechter 1986).

Alexey Ivakhnenko presented this theory in 1966 in his article, in which he investigates the predicting programs that use multiple small predicting filters in the predicting process (A. G. Ivakhnenko, 1995). The prediction described in the article is based on adaptation or learning of the predicting filters, which is the original theory of deep learning.

In the book of Deep Learning published by Massachusetts Institute of Technology (I Goodfellow 2016), Deep Learning is systematically introduced by its authors, from linear algebra which is the basis of deep learning to convolutional networks. It states that the hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones. The graph of deep learning, which shows how the complicated concepts are constructed, has many layers and is deep.

In deep learning model, the features of input are extracted from the raw input, through the Neutral Network of multiple layers. In this process, the network tries to recognize the pattern of the input and then adjust the weights, which is what we call “computer learns from experience”. This paper discusses how deep learning is implemented in a specific case in the Method section.

Reinforcement Learning

Reinforcement learning is one of the areas of machine learning. Reinforcement learning, according to Csaba Szepesvári’s words, is learning to control a system in order to maximize some numerical value which represents a long-term objective (C. Szepesvári 2010). The keyword of reinforcement learning is “reward”, as shown in Figure 1. The controller or the agent receives an observation from the system. The controller chooses an action and sends the choice to the system. The system, which is also called "environment" in other material, gets the action and move to a new state. Then, the system sends the next observation, which includes the reward

to the controller. The reward can help the controller make the decisions. In the beginning, the controller usually makes decisions randomly. In this process, the rewards will help the controller make better actions as it learns from the rewards.

It is hard to find the origin of reinforcement learning, but Markov Decision Process (MDP) is the most common and basic reinforcement learning. MDP is based on a mathematic model Markov Chain presented by Andrey A. Markov in 1906 (R Bellman et al. 1957). There are five major spaces in MDP: state, action, policy, reward and return. State, action, reward have been interpreted before. The policy, which in some material is also called "possibility", is how the agent (Controller in Csaba Szepesvári's book) makes the decisions. Return is the cumulation of rewards and defines the performance of the model after the training. So, it can be expressed by the reward. These spaces control how the agent learns from the rewards.

In recent years, what the public often heard about reinforcement learning, except for AlphaGo, is Q-learning. Q-learning is an extension of MDP and is a simple way for an agent to learn how to act optimally in controlled MDP (Watkins, C. J., & Dayan, P 1992). The extensive part is the Q function. Q function uses iteration to update the rewards and the actions in the MDP during the learning process. The process, which is similar to MDP, includes initialising a zero Q table, choosing an action, acting, measuring reward, updating Q value. After the update of the Q value, the model continues to choose an action and iterate. After the training, the model can obtain a Q table which can help the agent make decisions accurately.

HV Hasselt presents Double Q-learning (DQL) in 2010, which simplifies and accelerate the training process of Q-learning. DQL use the double estimator to Q-learning to construct Double Q-learning (Hado Hasselt 2010). Q-learning, in some situation, can overestimate the action values because Q-learning uses the maximum value as the estimated value. That can make the

performance very poor. DQL solves this problem by its double estimator for action values. Meanwhile, Deep Reinforcement Learning is based on DQL and Deep Learning (Hado Hasselt 2016).

During the period of writing, DeepMind has published their research about StarCraft II AI agents on October 30, 2019. The AI agent "AlphaStar" has defeated 99.8% of ranked human players. AlphaStar is based on Reinforcement Learning instead of Deep Reinforcement Learning. The experiment of DeepMind proved that Reinforcement Learning could also train a very high level of StarCraft II AI agents. Due to AlphaStar, Reinforcement Learning is considered a better choice for AI agents in Game Development.

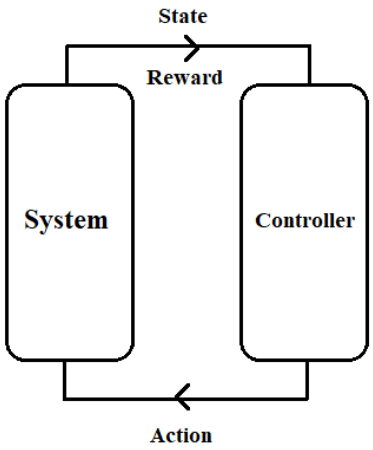

Figure 1. Reinforcement Learning scenario

Deep Reinforcement Learning

The concept of Deep Reinforcement Learning was presented by DeepMind in 2013 when DeepMind successful trained an AI agent to play Atari. Volodymyr Manih et al. do not give a precise definition of DRL. However, in their words, DRL aims to connect a reinforcement learning algorithm to a deep neural network which operates directly on RGB images and efficiently processes training data by using stochastic gradient updates. Compared to DQL, DRL

can directly extract feature from raw data. Bellman equation uses the maximum value of the next step as the estimated value, whereas DRL uses deep learning to evaluate the value of the next step.

The whole theory of DRL, as it is mentioned before, is presented by DeepMind in 2016. In that paper, the authors explained how they combine Deep Learning with DQL to train the agent of Atari. The result of the experiment shows that DRL shows excellent performance towards small-scale games. In fact, for now, DRL also shows outstanding performance when it is used to train the agents of many games, including Go, Dota2, Atari.

Well-known AI Agents

The most famous AI agent is AlphaGo by Google DeepMind. Based on DRL, DeepMind introduces 'value networks' to evaluate board positions and 'policy networks' to select moves (D silver et al. 2016). According to that paper, DeepMind trained AlphaGo by playing game matches with other Go programs. When the article was published, AlphaGo did not have such a high win rate against human players. Meanwhile, AlphaGo was also trained to play handicap games. Other Go programs can hardly defeat AlphaGo.

Another case is OpenAI Five. The system of OpenAI Five learns using a massively scaled version of Proximal Policy Optimization (J Schulman et al. 2017), which is also a kind of DRL. Dota2 is much more complicated than Go because there are two teams in a game and each team has five players. Dota2 is a special real-time strategy game, which is derived from a classic real-time strategy game: Warcraft III. Like AlphaGo, OpenAI Five was also trained to play game matches against other AI programs in 1 vs 1 at the beginning in order to let the agents learn how to play Dota2. And then, OpenAI Five was trained in a five-vs-five game match. The result of

OpenAI Five is surprising because it defeated OG Esport, which is the champion team of Ti8 and Ti9 and could be the most powerful team in recent years. However, OG did not win the championship title of Ti9 when OpenAI Five defeated it. That lets researchers make a step further and train the agents for StarCraft II, which is similar to Warcraft III and more complex than Warcraft III.

Figure 2. The moment when OpenAI Five defeated the World Champion Team, OG

StarCraft II AI Agents

Blizzard Entertainment (The company which made RTS games, such as Warcraft III, StarCraft I, StarCraft II) always opens map editors of its RTS games. Players can use this editor to develop their Agents. When StarCraft II was open to the public, Blizzard also published a Python library called “SC2” in order to let players relatively easily create their StarCraft II agents. The script-based AI agents can also defeat more than 50% of human players.

In 2018, National University of Defense Technology published a thesis about AI techniques on RTS games (Z Yang et al., 2018). UAIbertaBot, which is an AI agent for StarCraft I, is open to the public. UAIbertaBot is the prototype of the coming AI agents for StarCraft II. The structure of the program inspired many researchers like the researchers of TSTARBOT.

Google DeepMind and Blizzard Entertainment together provide an open-source Python library of StarCraft II, and it is also called "StarCraft II Learning Environment (SC2LE)". In fact, DeepMind, after publishing the library in 2017, does not present such powerful AI agents like AlphaGo. There are two agents for now which are presented by laboratories of a company and institutes.

Tencent AI lab published a paper about the Game AI agents of StarCraft II (P Sun et al. 2018). The agents are trained based on SC2LE and can only play Zerg vs Zerg game in the map "AbyssalReefLE" which is the map widely used to train AI agents in a full-length game. However, the result shows that the win rate against built-in level 10 AI is 80% if the AI agents choose to use the RUSH tactic. If the AI agents decide to use EconomyFirst tactic, the win rate is 90%. The sample that AI agents play game matches against Human players is too small, but according to their data, AI agents can hardly defeat human players of Diamond level. The AI agents can beat lower levels of players at the beginning since players do not get used to agents' tactic. Human player can learn from their experience much faster than the AI agents, and then the AI agents cannot defeat players anymore.

In another paper in which researchers also make AI agents of Protoss, the win rate against human players and built-in AI are not improved (ZJ Pang et al. 2019). However, it enhances the training process. The researchers use two-layer hierarchical architecture to accelerate the training process compared to the AI agents by Tencent AI lab.

In October 2019, DeepMind published their new AI agents of StarCraft II, AlphaStar (O Vinyals et al. 2019). AlphaStar has reached a grandmaster level in StarCraft II. Initially, DeepMind used Deep Reinforcement Learning to train their agents. When the paper is published, AlphaStar is based on Reinforcement Learning instead of Deep Reinforcement Learning. Compared to TSTARBOT from Tencent AI lab, AlphaStar moves a giant step forward. We do not know the reason why researchers from DeepMind used Reinforcement Learning instead, but this decision makes a difference. Although AlphaStar cannot reach the same level as OpenAI in Dota2, it is the most advanced AI agents for StarCraft II for now. AlphaStar is not a single StarCraft II AI agent, but a set of AI agents. It includes AI agents of three races in StarCraft II, and each of them can reach the grandmaster level.

In DeepMind's paper, they present a new phrase to describe how many operations an AI agent executes in a minute, effective actions per minute (EPM). Traditionally, players of RTS games use actions per minute (APM) to describe how many actions a player can perform in a minute. In other words, APM is the number of how many times a player clicks the mouse and the keyboard in a minute. A player's APM is always associated with the player's skill. However, APM cannot be used to accurately describe the ability of AI Agents, which are made by PySC2. According to PySC2's function, each action, for AI Agents, in StarCraft II is done directly when the command is executed, whereas a human player needs to click more than one times to do the same thing.

Meanwhile, it is a habit that most human players will click empty places in StarCraft II or click "1" or "2" on their keyboards frequently to keep their figures in good status when they play StarCraft II. DeepMind does not mention this in their paper. DeepMind limited EPM of their AI Agents to try to let AlphaStar can make better decisions than human players.

Gaps

For now, many unknown fields are still not explored by researchers, such as how to improve the win rate, how to make AI agents nimbler to the tactic, how to repeat the training on other two races. This paper tries to use DRL and SC2LE to train the AI agents of Protoss against Terran built-in level 10 AI and investigate the tactic problems of Protoss.

As is mentioned before, the tactic for each race is significantly different. Tencent AI lab builds two agents which use various tactics because it is hard to include two different tactics in an agent. Due to the unique characteristics of three races, the agents for Zerg cannot apply to Protoss. The RUSH tactic of Zerg is Zergling Rush, while the RUSH tactic of Protoss is four Gateway RUSH and is much more powerful and a little slower than Zergling Rush. However, for Terran which does not need to build their building near Pylon of Protoss or Lair of Zerg, it has significantly different tactics. How to react to these tactics is also what the agents need to play during the training.

Meanwhile, DeepMind also pointed out that though the best AI agents “AlphaStar” can defeat 99.8% players of StarCraft II, there is still a weakness. By watching the replays which are provided in their publication, we can observe that AlphaStar is still a precise, effective “machine”. For example, in the game match of “AlphaStar vs MaNa”, AlphaStar did not know how to utilize the advantages of architecture. It also did not show excellent performance in tactics. However, its actions are more precise and faster than MaNa, who is a famous professional player. This phenomenon indicates that Machine Learning, for now, fail to teach AI agents how to learn tactics from human players.

Aim and Objective

The objective is that the model which is based on Deep Reinforcement Learning can help us build an AI agent that can defeat built-in cheating AI as how TSTARBOT and AlphaStar do. Our perspective is that DRL can improve the performance of AI agents which can have a better performance than random agents and agents created by hard code (built-in AI agents).

METHOD

In this section, we describe the framework of the design and how to implement the StarCraft II AI agents in detail. Firstly, the framework of the design will be introduced, which describes how the design is constructed by layers and how the program is divided. Secondly, we will introduce each layer of the program, including inputs, outputs, and how to process the data in each layer. They will be divided into four parts: game operations, deep learning, reinforcement learning and training procedures. Thirdly, we will summarize the results and describe the weaknesses and strengths of the method.

Framework

This part will mainly focus on the construction of the design. In the beginning, we ignore all the methods and suppose that the computer is a real player who is learning how to play StarCraft II. The player needs to finish the tutorial of the game. In this stage, the player starts to understand how to operate each unit in StarCraft II. However, the player does not know any advanced knowledge which can help the player have a better understanding of the game, such as skills of units and tactics. It is the first stage, in which the computer needs to know how to operate the units in the game, such as attacking, building an architecture, training army.

In the second stage, the computer needs to grasp information of itself and its enemy. For example, a new player who is learning how to play the game needs to know the information he/she sees on the screen. The player can know how much minerals, vespene and food he/she has now from the top-right corner of the screen. The player is also able to know where his/her base is located from the mini-map. Besides, the player can count how many workers or other units

he/she has. The computer is not as smart as human beings. Therefore, all the information should be recorded in a special mini-map. The data is the input of deep learning methods.

In the next stage, the player needs to know what can help him/her win the game and what can make him/her lose the game. Our brains can do it so that we can learn from our experience. The computer also needs to learn what help it win based on the information. This stage can be described as reinforcement learning.

Thus, the computer becomes more and more powerful as it learns from its experience. The win rate of the AI agents could increase to 100% if the built-in AI agent is very weak. Then, the level of the built-in AI agent will increase and repeat the learning process.

Figure 3 shows the construction of the framework.

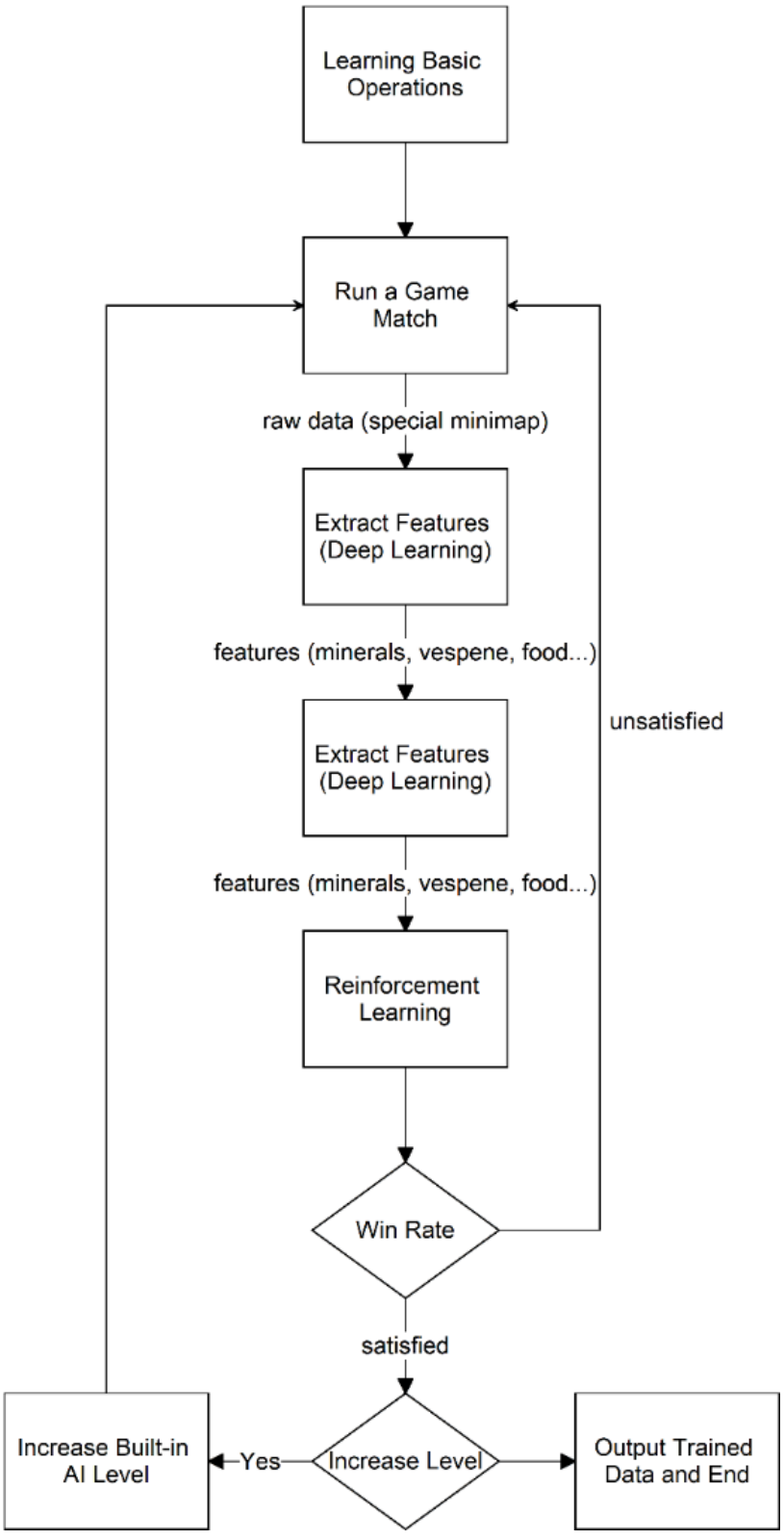

Figure 3. The framework of the training process

Basic Action

When a player is playing a game match, the keyboard and mouse of the computer get input from the player and send it to the computer. So, the computer can render the action on the screen. In this part, the requirement is that the computer just does one thing in every step. In 2019, DeepMind updated raw functions (do an action based on mini-map location instead of the location of the current screen), and there are 563 kinds of actions in each step. For each kind of action, the computer needs to know which unit it is operating, which action a unit should do, towards which unit a unit should do action, and when and where it performs the unit. Although there are 563 kinds of actions, each of Race in StarCraft II has its unique actions, and most of the actions are unique actions. For Protoss which is the Race of the agent in this paper, it has less than 200 kinds of actions, and we need to define less than 30 functions to implement basic operations in order to imitate a player's operations.

$$Action(actionID, unitTag, queued, targetTag, Location)$$

This function shows the basic format of functions:

1. "actionID" is the parameter which indicates what action a unit should perform;
2. "unitTag" is the parameter that tells the current agent which unit should do an action;
3. The parameter "queued" has two kinds of input, "now" and "queued". "now" means that the unit should stop what it is doing now and then do the action, while "queued" means that the unit will do the action after it finishes the action for now.
4. The parameter "targetTag" means towards which target the unit do the action;
5. "Location" is the coordinator of the place where the unit does the action.

DeepMind presents "EPM" to take the place of "APM" in its paper about AlphaStar. How an AI agent executes an action is different from a human player. Take building a Pylon as

an example. A human player is required to select a worker, click “build basic buildings”, click “Pylon” and select an empty place to complete building a Pylon. There are four actions. Nevertheless, an AI agent just needs to execute one action to finish building a Pylon.

Which Unit

For each action, a player needs to select a unit, like Nexus, Pylon, Zealot. Similarly, an AI agent also needs to know which unit it should operate. Compared to a human player, an AI agent can directly operate a unit instead of clicking a keyboard or mouse to operate it.

What to do

Each unit has one or more skills. For example, a zealot can attack, and a worker can build buildings. In the script, an AI agent needs to know which skills they should use. “Move” is also one of the skills for every unit.

Where/Target

After a unit knows what to do, it also needs to know the target. For instance, if a worker is going to build a Nexus, it needs to know where to build it. The location is the “Target”. Likewise, for “attack” action, a unit needs to know which unit it should attack.

When to do

Each unit in StarCraft II has a “First in First Out” action queue. Human player can press “shift” to add action into a unit’s action queue. Otherwise, the unit will clear its action queue and execute the new action. For AI agents, it is also required to tell units to execute the action directly or add the action into their queues.

StarCraft II has 564 kinds of basic actions. We cannot put every action into Q-learning because too many actions will make it hard to train the AI agent. Thus, we need to group several related actions as an action. The actions of the research center in StarCraft II can be grouped into two actions.

Extracting Features

A player can only see the picture like Figure 4 when he/she is playing the game. DeepMind builds up StarCraft II Learning Environment (SC2LE), and they have integrated their feature extractors into the environment, as what we can see in Figure 4. These feature extractors are based on the neural network, which can break down the information on the screen into the features we see in Figure 5. The input of the input layers is the left side of Figure 4, and the output of the output layer is the right side. Researchers can get access to these features by an observer in SC2LE. These features can help the computer know the situation it is facing.

Figure 4. What a player can see on screen in a StarCraft II game match

Figure 5. The features that the agent can see

Deep Reinforcement Learning

In the literature review, we introduce the fundamental of Reinforcement Learning used by DeepMind is Q learning. The researchers of DeepMind improve Q-learning and introduce double Q learning to reduce error.

The double Q-learning error can be written as:

$$Y_t^{DoubleQ} \equiv R_{t+1} + \gamma Q(S_{t+1}, argmaxQ(S_{t+1}, a; \theta_t); \theta'_t)$$

Double Q-learning uses θ' to replace θ to solve the overoptimism due to the estimation errors.

Furthermore, deep double Q-learning introduces experience replay and target network to increase the effectiveness of the algorithms significantly. DeepMind uses the online network to execute a greedy policy and use the target network to estimate the values:

$$Y_t^{DoubleDQN} \equiv R_{t+1} + \gamma Q(S_{t+1}, argmaxQ(S_{t+1}, a; \theta_t^-); \theta'_t)$$

Q table will update based on each action. In the beginning, the AI agent will randomly choose the action in each step. Based on the information in the features, the Q table will be gradually built up, and the AI agent knows how to choose the best actions for the current state.

Training Procedures

In the beginning, because the AI agent randomly chooses the actions, the enemy level is 0, which means very easy. Level 10 built-in AI is a cheating AI which has double minerals and vespene, and it does not have war fog which means that it can know where its enemy's base is located (other information is useless for level 10 AI). When the win rate of AI agents does not change any more, the training process is completed.

Expected Outcome

The expected outcome is the model which can fully train Protoss AI agent that can defeat level 10 built-in AI agent. If DRL is not able to help the Protoss AI agent defeat high-level AI agents, we will figure out the problems and list the reasons for that.

Summary

Strengths

Deep Reinforcement Learning helps the agent learn how to choose the best option better than Q-learning. Based on DRL, the AI agent can perform much better than the original AI.

Weakness

The weakness for now is the basic operations. The computer needs to know the coordination in basic operations, but how to build architecture is extraordinary complex,

especially for Protoss. Because of the characteristics of Protoss, architecture plays a significant role in its tactics. However, the location of architecture is the weakness of this algorithm. It is also the reason why DeepMind starts research from Terran instead of Zerg and Protoss. The architectures of Terran can be operated as a unit.

Evaluation Plan

The AI agent trained by the model based on DRL is evaluated by the win rate against built-in AI agents and which level of built-in AI agent it can defeat. More specially, the AI agent should have a win rate which is higher than 99% against level 1-8 built-in AI agents which cannot cheat. On account of the level 7 agent which is made by hard code, DRL can improve the performance of Protoss AI agents if they cannot defeat level 7 or lower-level built-in AI agents, while it should have a win rate which is higher than 80% against level 8-10 built-in AI agents which can cheat during a game match.

The evaluation plan is based on the win rate of TSTARBOT created by Tencent AI lab since there is no other data about the win rate of Game AI agents of StarCraft II based on DRL.

EXPERIMENT

In this section, the training process and the win rate against StarCraft II AI agents will be introduced to show the performance of the Protoss AI agent based on Deep Reinforcement Learning.

Training Process

Table 1. The accumulative win rate during the training process

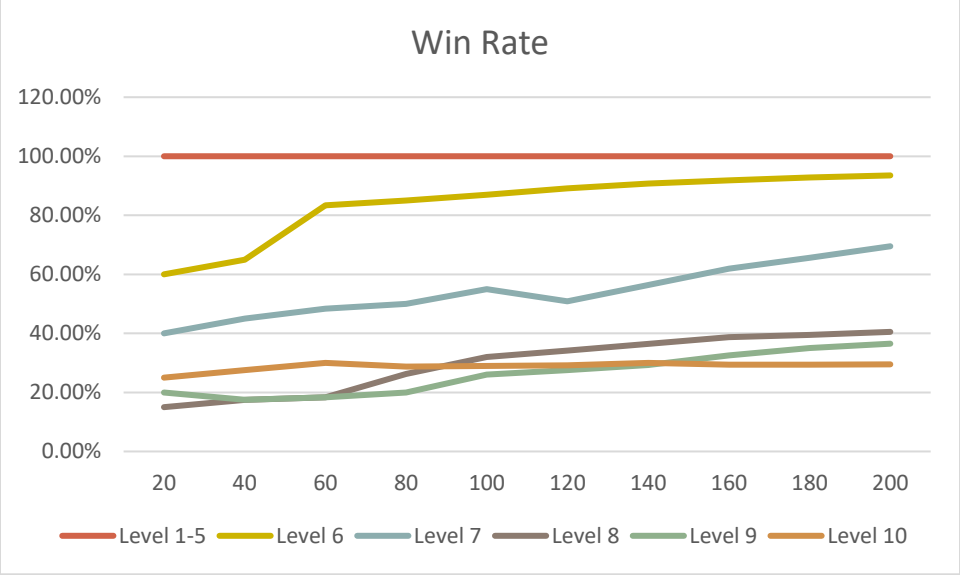

Table 1 shows the accumulative win rate during the training process. Initially, the Protoss AI agent is a random agent, which means that the AI agent chooses the action from the action pool randomly. The built-in AI agents from level 1 to level 5 cannot defeat the agent. The Protoss AI agent will play 200 game matches with the same level AI agent. The agent shows an excellent learning performance during the game matches with the built-in AI agents, which are below level 8. Level 8, Level 9 and Level 10 are cheating AI agents. The Protoss AI agent does not show an excellent performance against these agents.

Due to the limitations of computing resources (See Appendix) and time, the number of game matches during the training process is 200 for each level. The win rate can be a little improved after 200 game matches, but much more game matches are required to execute to improve the performance of Protoss AI agent.

Performance

Table 2. The win rate of agents against built-in AI agents

Agents	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8	Level 9	Level 10
Random Agent	100.00%	100.00%	100.00%	67.50%	13.50%	0.00%	0.00%	0.00%	0.00%	0.00%
DRL Agent	100.00%	100.00%	100.00%	100.00%	100.00%	99.00%	93.00%	49.50%	51.00%	30.50%
Level 10 Agent	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	98.00%	66.50%	71.00%	52.50%

Table 2 describes the win rate of agents against built-in AI agents after the training process. We tested 200 game matches for each situation in the table.

Random Agent

Random Agent cannot defeat the built-in AI agent, which is higher than level 5. Due to the fact that the random agent selects the action randomly, the random agent cannot have a good performance against a built-in AI agent. However, a random agent is the initial AI agent, which means that it can be used to compare with the trained agent (DRL Agent in Table 2).

Level 10 Agent

Level 10 Agent is a cheating AI agent, which means that it can gain double resource and the information of the whole map. Script-based AI agent cannot defeat a Level 10 agent because it is cheating.

DRL Agent

DRL Agent can easily defeat the built-in AI agents which are below Level 8. However, when it plays game matches against cheating agents, the learning speed and win rates are significantly lower than before. The disadvantages of scouting and resources make DRL Agent easily lose game matches. Meanwhile, the limitation of EPM of DRL agent is 480. Among 480 actions within a minute, there are 50% of actions which are “no action”. The EPM of DRL Agent is not as high as it is set in the program.

DISCUSSION AND CONCLUSION

By watching the replay and analyze the data from the replay, the Protoss AI Agent based on DRL can reach a level which is higher than script-based AI agents (Level 1-7 built-in AI agents). However, it cannot easily defeat cheating AI agents (Level 8-10) after 2000 training game matches. We will analyze the reasons and the phenomena in the following content.

Additionally, the difference between other smart AI agents, such as AlphaStar, will also be listed.

Game Replay

When the machine learning is not widely used in the area of Game AI, SB Stene presents his paper to point out that the performance of AI agents in RTS game are associated with tactics AI agents use (SB Stene, 2006). DeepMind did not point out the weakness of AI Agents based on Machine Learning Technology. Some researchers focus on the information in the statistics data and ignore the information in the RTS games. However, according to the replays Blizzard offer and the history of RTS games, a tactic is the most important factors which can let a professional player win a championship.

By watching the replays of AI agents against Professional Players (The only resource is from DeepMind) and the replays of the Protoss AI agents in this paper, it is easy to find out that AI agents know nothing about tactics. TSTARBOT, which is the only one Zerg-vs-Zerg AI agent, places architecture randomly. In its paper, the authors state that Where to build buildings is trivial. AlphaStar also shows the same property as TSTARBOT – Where to construct buildings is trivial. The professional player MaNa from Team Liquid placed his Gateways on the entrance

to his main base during his first game match against AlphaStar. The locations where architectures are placed are critical in some versions.

In the first version of StarCraft II, if Terran or Zerg did not place their buildings to block the entrance to their bases, Protoss's tactic "4BG", which means that Protoss builds 4 Gateways at the beginning of a game match, is unstoppable. Architecture is an important part of many tactics.

Unfortunately, for now, there is no good method to introduce tactics and architecture locations into the training process.

Statistics

Win Rate

The win rate shows that Deep Reinforcement Learning can help a Protoss AI agent perform better than script-based AI Agents. Table 2 shows that only when script-based AI agents cheat can they defeat the Protoss AI agent based on Deep Reinforcement Learning.

APM and EPM

There is a weakness which must be pointed out. Many researchers limit the EPM of AI agents within the average APM of human players. It is not accurate and precise. According to the replays of StarCraft II, most human players have a habit of clicking empty places and useless keys on the keyboard. This seemly unnecessary process is called "warm-up" by human players. It is used to keep players in a highly active status. Meanwhile, AI agents can do the same things with fewer actions than human players, because AI agents did not need to select units, skills and targets by input devices. The EPM of Protoss AI agent in this paper is set lower than human players. However, the more highly trained AI agents are, the more precise actions they execute.

In the experiment section, the experiment shows that the Protoss AI Agents contain 50% “no action”. The real EPM of the agent is lower than 240.

How to win

Timing

There is a word “Timing” in RTS game. “Timing” is the time when the enemy does not have enough resources or army to fight back the player. All the AI agents based on Machine Learning show a good performance of catching Timing. They often attack their enemy during Timing. AI agents can precisely forecast when the enemy is weak if the anti-scout strategy of the enemy is bad.

Precise Action

AI agents’ EPM is much higher than human players, which means that AI agents can do more thing in one minute. It is a great advantage during combat. AI agents can move their units precisely to avoid attacking and attack their enemies. The action pool of TSTARBOT also focuses on combating actions. According to the replays, AI agents usually have an advantage over human players during combat.

Snowball Effect

AI agents have a better performance of catching Timing than human players. Thus, AI agents plunder resources and destroy enemies’ workers and buildings. By this way, AI agents can lead in the resources, which means that they can create more army and collect more resources. Then, AI agents will catch the next Timing and end the game match in case enemies

get over lacking resources. Some players of the Dota2 Forum states that OpenAI Five can always catch Timing and end the game match in an early game.

Difference between AI Agents

Algorithms

The Protoss AI agent in this thesis is built on the structure of TSTARBOT1. TSTARBOT project created two AI agents. The first one is based on classic MDP and Deep Double Q-learning. In this thesis, we remake the action part and train the Protoss AI agent instead of Zerg AI agent in TSTARBOT project. AlphaStar, however, uses an auto-regressive policy (O Vinyals et al., 2017) and recurrent pointer network (O Vinyals et al., 2015) instead of DQL. The researchers classify their algorithms as Reinforcement Learning rather than Deep Reinforcement Learning. AlphaStar shows a better performance than the Protoss AI agent in this paper.

Computing Recourse

DeepMind does not point out the experiment equipment it uses to train the AI agents of AlphaStar, but the researchers state that they input more than 971, 000 replays to train their AI agents. According to the data, the computing recourses DeepMind could access are much more than ours. The computing resource we had access to is a personal computer.

Conclusions

By the advantages of EPM and catching Timing, The Protoss AI agent based on Deep Reinforcement Learning can defeat the enemies that do worse in EPM than it. Because the Protoss AI agent is not trained enough, it cannot defeat cheating AI. However, it can win a game matches against script-based AI agents in a landslide. According to the AI agent TSTARBOT,

which has the same structure, the Protoss AI agent in this paper can defeat a Level 10 AI agent if it is trained enough.

According to the replays, whether an AI agent can win a game match firstly depends on Precise Action, catching Timing and Snowball Effect. Although the agent in this paper can execute more precise actions than built-in AI agents, cheating agents have different Timing, and the cheating behaviors make it difficult for the Protoss Agent to lead a Snowball Effect. More training processes need to be done to make the AI agent get used to cheating agents and defeat them.

Future Work

AI agents based on Machine Learning Technology need a large number of replays to make it find out how to win a game match. The AI agent in this paper is not fully trained. The computing resources are vital because StarCraft II requires both CPU and GPU computing resources.

Meanwhile, StarCraft II AI agents, including the agent in this paper, is “barbaric”, which means that they know nothing about tactics and use overwhelming resources to win a game. Human players’ tactics cannot be reflected by statistics data for now. Perhaps the AI agents based on Machine Learning Technology can be improved by researching on some complex Strategy Games, in which faster and more precise actions are useless.

As a part of tactics, architecture in StarCraft II can be used to help AI agents defeat stronger enemies. Although it takes too much computing resources to train an AI agent where it should place its architecture, for now, architecture can significantly improve the ability to attack and defend. Perhaps we can investigate how to introduce architecture location into the training process.

APPENDIX A. INSTRUMENTS

Personal Computer

OS	Microsoft Windows 10 Pro (64 bit)
CPU	Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz(3600 MHz)
Motherboard	ROG MAXIMUS XI EXTREME
RAM	32.00 GB (3200 MHz)
DISK	510 GB
GPU	NVIDIA GeForce RTX 2080 Ti (11264MB)
Monitor	Acer X34 P 60Hz
Audio	Realtek High Definition Audio
NIC	Intel(R) Wireless-AC 9560 160MHz

APPENDIX B. BUDGET

PC	Paid before
Game match replay	Open Source
StarCraft II Learning Environment	Open Source
Python Standard Library	Open Source

APPENDIX C. DATA STRUCTURE

- Data
    - 1 iteration
        - Replay (*.SC2Replay)
        - Training_Data
            - 1 (1, 2, 3, …, 200) (*.png)
            - 2(1, 2, 3, …, 200) (*.png)
            - …
            - Y (1, 2, 3, …, 200) (*.png)
        - Trained_Data (Customize files)
        - Victory_Record (*.csv)
    - X(1, 2, 3) iteration
- Src
- Doc
- Bin
- Lib

REFERENCES

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484.

Ivakhnenko, A. G., & Ivakhnenko, G. A. (1995). The review of problems solvable by algorithms of the group method of data handling (GMDH). *Pattern Recognition And Image Analysis C/C Of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, *5*, 527-535.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... & Oh, J. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), 350-354.

OpenAI five. https://blog.openai.com/openai-five/. Accessed August 30, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv *preprint arXiv:1312.5602*.

Dechter, R. (1986). Learning while searching in constraint-satisfaction problems. University of California, Computer Science Department, Cognitive Systems Laboratory.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine learning. *Neural and Statistical Classification, 13*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*.

Bellman, R. (1957). A Markovian decision process. *Journal of mathematics and mechanics*, 679-684.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4), 279-292.

Hasselt, H. V. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems* (pp. 2613-2621).

Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Zhen, Y., Wanpeng, Z., & Hongfu, L. (2018, December). Artificial Intelligence Techniques on Real-time Strategy Games. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence (pp. 11-21).

Sun, P., Sun, X., Han, L., Xiong, J., Wang, Q., Li, B., ... & Zhang, T. (2018). Tstarbots: Defeating the cheating level builtin ai in starcraft ii in the full game. *arXiv preprint arXiv:1809.07193*.

Pang, Z. J., Liu, R. Z., Meng, Z. Y., Zhang, Y., Yu, Y., & Lu, T. (2019, July). On reinforcement learning for full-length game of starcraft. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 4691-4698).

Stene, S. B. (2006). Artificial intelligence techniques in real-time strategy games-architecture and combat behavior (Master's thesis, Institutt for datateknikk og informasjonsvitenskap).

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... & Quan, J. (2017). Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems* (pp. 2692-2700).