Ziqi Wang

The Electric Brain: New Intelligent Agents in Esports

Master's thesis of mathematical information technology

May 3, 2022

University of Jyväskylä

Department of Mathematical Information Technology

Author: Ziqi Wang

Contact information: ziqi.ccc.wang@student.jyu.fi; 821220954@qq.com

Supervisors: Vagan Terziyan

Title: The Electric Brain: New Intelligent Agents in Esports

Työn nimi: Sähköaivot: Uusi Älykkäät Agentit E-urheilussa

Project: Master's thesis

Study line: Cognitive Computing and Collective Intelligence

Page count: 86+3 (86 pages without appendix; 3 pages of appendix)

Abstract: This thesis studies an emerging technology in artificial intelligence related to esports. Intelligent agents in video games like StarCraft and Dota 2, have become phenomenal in the game communities. Those video games are called esports because they are designed for competition, with a requirement of excellent skill. Creating electronic players in video games is a routine for game developers. However, those agent players in history never astonished human players by reaching high performance in a fair way. The recent agents are created using the same type of AI technology which gave birth to the famous Go program, AlphaGo, and they play the games showing human-like understanding and behaviors. The thesis identifies the generality of those agents by checking the fundamental requirements. The thesis also interprets the impact of these agents to the game community and studies existing cooperation between human and game agents. Two major agent creation procedures were included in comparative analysis and feedbacks to the show matches were collected for case study. The result found the boundaries of the agent's ability and the training procedure, summarized people's attitude, and proposed ways of cooperation and discussion about achieving higher generality.

Keywords: Artificial Intelligence, Deep Learning, Reinforcement Learning, Intelligent Agent, Esports, Game Community, Pro-gamer.

Glossary

AI	Artificial Intelligence
AGI	Artificial General Intelligence
IA	Intelligent Agent
RL	Reinforcement Learning
DL	Deep Learning
Esports	Electronic Sports
RTS	Real-time Strategy
MOBA	Multiplayer Online Battle Arena
FPS	First-person Shooter

List of Figures

List of Tables

Contents

1 Introduction

The past 50 years has been the age of great advancement in computing performance. The memory capacity, computational power, etc. have increased to many times than those used to be. The growing computational power has been leading the way of computer program's development. Among many types of software made for computers, Artificial Intelligence (AI) is an emerging, important, beneficial outcome. One sub-field of AI, machine learning, progressed quite a lot in the last two decades, from laboratory curiosity to a practical technology in widespread commercial use (Jordan & Mitchell, 2015). Machine learning is so versatile that it is chosen for developing practical software for speech recognition (Hannun et al., 2014; Hinton et al., 2012), natural language processing (Young et al., 2018), computer vision (Liang & Hu, 2015; Simonyan & Zisserman, 2014), robot control (Billard & Kragic, 2019), and other applications. It was predicted that deep learning, a type of machine learning, will have many more success in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data (LeCun et al., 2015).

Game playing has always been a topic of AI research. The development of video game industry expanded this concept to a broader sense. Modern video games are computer programs created with game engine coding and art packaging. Despite made for entertainment mainly, some video games turned out to be a good examination of intelligence. Most games have a fixed set of rules, where problem-solving techniques can be developed and evaluated before being applied to more complex real-world problems (Schaeffer, 2001). As Vinyals points, some games offer these following advantages to deep learning study (Vinyals, Ewalds, et al., 2017):

1. They have clear objective measures of success.
2. Computer games typically output rich streams of observational data, which are ideal inputs for deep networks.

3. They are externally defined to be difficult and interesting for a human to play. This ensures that the challenge itself is not tuned by the researcher to make the problem easier for the algorithms being developed.
4. Games are designed to be run anywhere with the same interface and game dynamics, making it easy to share a challenge precisely with other researchers.
5. In some cases a pool of avid human players exists, making it possible to benchmark against highly skilled individuals.
6. Since games are simulations, they can be controller precisely, and run at scale.

Esports are competitions using some video games where skills are required mostly. And they serve as a good multi-agent adversarial testing platform in AI research. On the other hand, AI research in esports does not concern scientists solely. Nowadays, apart from the video games themselves, esports now have become a type of commercial activity involving professional players, sponsors, organizers, commentators, audiences etc.[1]. These games have a very huge player base (Funk, 2013), so maybe the research of game AI is going to create waves of effect like a stone thrown into water. Therefore, this thesis intents to give a summary of the principle of video game AI and the effect to the player community as well as the possibilities in the upcoming future.

1.1 Research questions

For the topic, the thesis proposes the following specific research questions (RQ):

RQ1. Does the way of creating those AI players have a certain level of generality that allows the procedure to be reused, or makes the agent adaptive?

RQ2. From players' perspective, what impact does the invention of such AI players have to the gaming community?

RQ3. How possible is the cooperation between the AI and human players?

[1] A typical example of a tournament: https://liquipedia.net/leagueoflegends/LPL/2022/Spring

These questions will be traversed in the following chapters where there is sufficient information for answering them. Multiple research methods are going to be used for the questions.

For RQ1, the selected research method is comparative analysis. Currently, there are several cases where some games are researched and the special game playing AI is developed for the games. However, there are not so many compared to the number of video games created. So this is a typical "small N" topic, where N is the number of cases in the given study (Collier, 1993). By doing qualitative comparative analysis, the number of cases may be small enough to permit some degree of familiarity with each case, and large enough to warrant an interest in cross-case patterns (Ragin, 1999). Therefore, through comparative analysis, the differences in configuration of agent will be taken into consideration. This will shed a light on what are necessary in creating such agent players and how the scale of games influences their outcomes.

The second and the third question would lead to a case study based on specific esport games. Until now, esport competitive AI is not yet widespread. Even though there has been example of deploying the agent to the match making system, there has not been systematic feedback from the players' communities. The most prominent source of information related to the topic is from the researchers' reports and show matches. Therefore, case study is applied to focus on the phenomenon of the AI's invention, with the gaming communities' conventions taken into account.

1.2 Thesis Layout

The thesis is divided into 7 chapters. This is the first chapter, where the topic is introduced briefly. Chapter Two goes back to history and reviews the main AI development and video game development. It covers how those major esports were born and when the AI technologies for games are implemented. The third chapter examines the detail of those AI gaming technologies. It aims to form some concepts that could help predict whether any agent is capable of mastering a game. RQ1 is going to be made more concrete and the answer to it will be in this chapter. The fourth chapter goes over the announcements of the most

famous esports agents and focuses on the consequences. This chapter answers RQ2. Chapter Five inspects the cooperative elements in these cases, involving more about human-AI interaction in games. RQ3 is answered in Chapter Five. The sixth chapter is the discussion part of this thesis. It covers some relevant intriguing information which is not fully presented in the previous chapters. The seventh chapter is for the conclusion. Limitations of this study and possible direction for future research will be mentioned here.

2 Background

In this chapter, the development history of AI in the past decades is introduced firstly. It intends to provide an overview of how developers use AI tools to build applications requiring cognitive ability. Then the rest of this chapter explores video games and the esport scene. Furthermore, brief history of AI research in games is described.

2.1 Development History of AI

The first stage of AI development was symbolic artificial intelligence, which brought some success because early computers were seen as things that could do arithmetic and no more (Russell & Norvig, 2009). It most uses a “top-down” design method; they begin at the knowledge level and proceed downward through the symbol and implementation levels (Nilsson & Nilsson, 1998). In 1975, Newell and Simon stated a general scientific hypothesis, The Physical Symbol System Hypothesis (A. Newell & Simon, 2007): “A physical symbol system has the necessary and sufficient means for general intelligent action.” AI researchers were very optimistic and they made shocking predictions like machines will be capable of doing human's work (Simon, 1965) or even the problem of creating 'artificial intelligence' will substantially be solved' (M. L. Minsky, 1967). However, those early systems turned out to fail on the later problems with wider selections and higher difficulty because early programs did simple syntactic manipulations and knew nothing of their subject matter (Russell & Norvig, 2009). Combinatorial explosion was mentioned in the Lighthill report, which showed that in practice any solution would take too many resources to be useful (Lighthill, 1973). Intelligence was best characterized as the things that highly educated male scientists found challenging like chess, symbolic integration, proving mathematical theorems, and solving complicated word algebra problems. The things such as visually distinguishing between a coffee cup and a chair were not thought of as activities requiring intelligence (McCorduck & Cfe, 2004). It turned out that the vision is slightly more difficult than reasoning and planning (Szeliski, 2010).

Then the AI Winter came. Funding from the British government and the U.S. government was ended and no further advances were made in the following years (Haenlein

& Kaplan, 2019). The classical symbol approach of AI is also called good-old-fashioned-AI or GOFAI (Nilsson & Nilsson, 1998). At the same time, other approaches also existed such as perceptrons, a simple form of neural network (M. Minsky & Papert, 1969). However, the research funding for neural-net soon dwindled to almost nothing (Russell & Norvig, 2009).

In the 1980s, a form of AI program called "expert systems" arose using domain-specific knowledge that allows larger reasoning steps and can more easily handle typically occurring cases in narrow areas of expertise (Russell & Norvig, 2009). R1 at the Digital Equipment Corporation became the first successful commercial expert system. The program helped configure orders for new computer systems and saved the company an estimated $40 million a year by 1986 (McDermott, 1982; Russell & Norvig, 2009). Nearly every major U.S. corporation had its own AI group and was either using or investigating expert systems (Russell & Norvig, 2009). In 1981, Japan initiated the Fifth Generation Computer program to make computers that would be able to converse and understand with humans (McCorduck & Cfe, 2004). The AI industry boomed from a few million dollars in 1980 to billions of dollars in 1988 but eventually many companies failed to deliver on extravagant promises (Russell & Norvig, 2009).

In 1997, the chess machine Deep Blue developed by IBM defeated then-reigning World Chess Champion Garry Kasparov in a six-game match with a score of 3.5 to 2.5 (Campbell et al., 2002). This shocked the world and the question of, whether the computer can outperform the human at an activity that tests the intelligence, was raised (Goodman & Keene, 1997). In terms of playing chess the answer could be yes as Deep Blue did win the match, but the dedicated massively parallel search engine, and various search algorithms, determined that the system could reach the goal of design but achieve nothing more than playing chess (Hsu et al., 1995).

Figure 1. World chess champion Garry Kasparov (left) playing against IBM's supercomputer Deep Blue in 1996 during the ACM Chess Challenge in Philadelphia (Goodrich, 2021)

Even though the back-propagation learning algorithms for artificial neural networks were firstly discovered in 1969, in the mid-1980s the algorithm was reinvented by at least four different groups and caused a widespread dissemination (Bryson & Ho, 2018; Russell & Norvig, 2009). The discoveries helped to revive the field of connectionism, which was the alias for deep learning in the 1980s-1990s (Goodfellow et al., 2016). Terrence Deacon even suggested in his book that connectionism is the *defining characteristic* of humans (Deacon, 1998) because the model in deep learning, artificial neural network, is inspired by neuroscience (Marblestone et al., 2016). In deep learning, artificial neural networks do not rely on hard-coded knowledge, but extract patterns from raw data. Deep learning was greatly enhanced by the increased amount of data from the trend of society digitization (Goodfellow et al., 2016). In 2011, digital information has grown nine times in volume in just five years (X.-W. Chen & Lin, 2014). Deep learning has been successfully implemented speech recognition (Hannun et al., 2014; Hinton et al., 2012), natural language processing (Young et al., 2018), computer vision (Simonyan & Zisserman, 2014), etc.

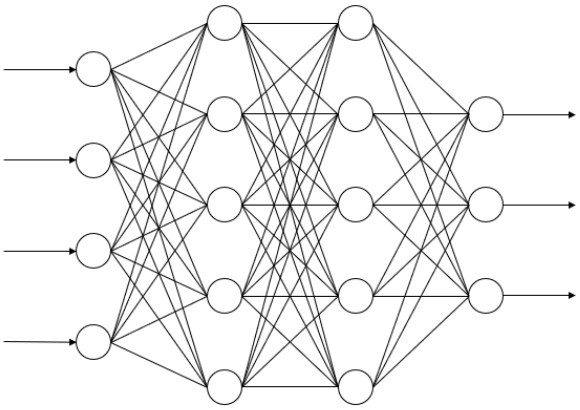

Figure 2. Example of a multi-layer perceptron neural network

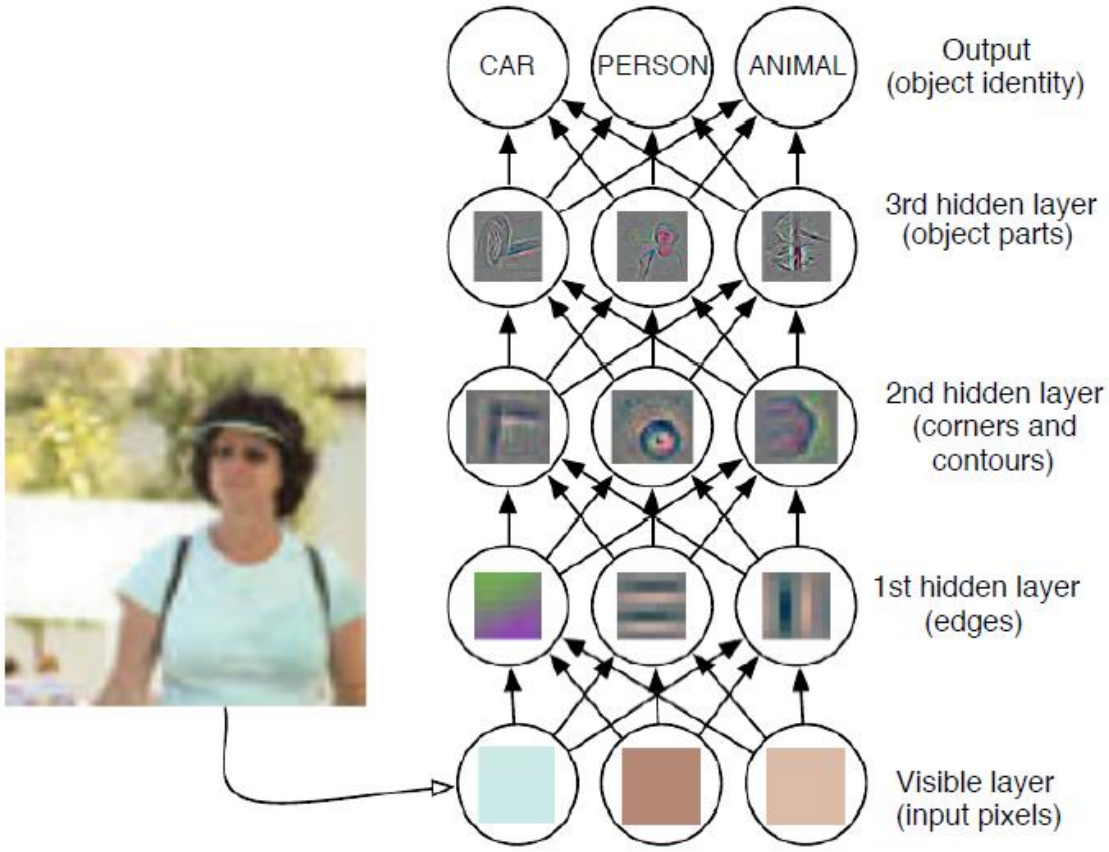

Figure 3. Illustration of a deep learning model from (Goodfellow et al., 2016)

2.2 Video games

In the last century, Moore's law predicted the exponential rate of improvement on computer's performance (Schaller, 1997). In the progress, AI was not the only field that took advantage of this increasing computational power. Video game developers had been able to realize more and more complicated idea thanks to more speedy computers. Pixel art evolved into vivid visual model. Among the games developed in the history, the fighting game *Street Fighter II* (1991) popularized the concept of direct competition between two players. Though it was possible to play *Street Fighter II* and other fighting games against the computer, but the real challenge was facing off against a skilled human opponent. The game centers were filled with teens waiting in line in front of these arcades (Loguidice & Barton, 2012). The growth of the video game industry in the 1990s gave birth to more games that later either became substantial in the esport scene or made great progress in game development. In 1993, id Software released *Doom*, a First-person Shooter (FPS) game, with a lot of innovations in the game engine such as its real time three dimensional rendering (Hall & Romero, 2011; Hutchison, 2008). Then, as the successor to *Doom* series, programmed mainly by John Carmack, *Quake* was released in 1996, offering full real-time 3D rendering and added reliable multiplayer mode over a network ("Quake (Video Game)," n.d.). The game's success leaded to some direct sequels, including *Quake III Arena*, which refined multiplayer gameplay and was used intensively in professional esport tournaments. Furthermore, the source codes of id Software's games were released under the terms of GPL-2.0-or-later (id Software, 2005). This allowed programmers to edit the engine and make new games based on the original code. One of the most well-known engine users is Valve Corporation. They developed *Half-Life* by adding heavy modification to the original Quake engines (G. Newell, 1999). This game later received a mod ("modification") called *Counter-Strike* which was later acquired by Valve and developed into a esport mainstream with many annual tournaments (Mitchell, 2018).

Meanwhile, apart from the FPS genre, one more game genre is awakening. In 2000, The first World Cyber Games (WCG) was held at Everland, Yongin, Korea from October 7 to 15 (WCG, 2021). The tournament featured *Age of Empires II* and *StarCraft*, which are considered representing Real-time Strategy (RTS) genre. Unlike fighting games

and FPS where every player is provided with one character in control, RTS players overlook the warfield from above and assume a god-like role. For competition, a typical RTS game has 3 core concepts: resource (also known as economy), development (a.k.a. build) and battle. Blizzard's game *StarCraft* quickly became popular in South Korea and a large quantity of tournaments were held by TV channels in the 2000s (Jin, 2010; Liquipedia, 2022). The next generation of Blizzard's RTS game was *WarCraft III*, released in 2002. It shipped a 3D graphics and an editor like its RTS predecessor, which led to lasting influences and inspired many future games. The game was so over-engineered that the systems were "capable of much more than was actually used in the game" (Moss, 2020).

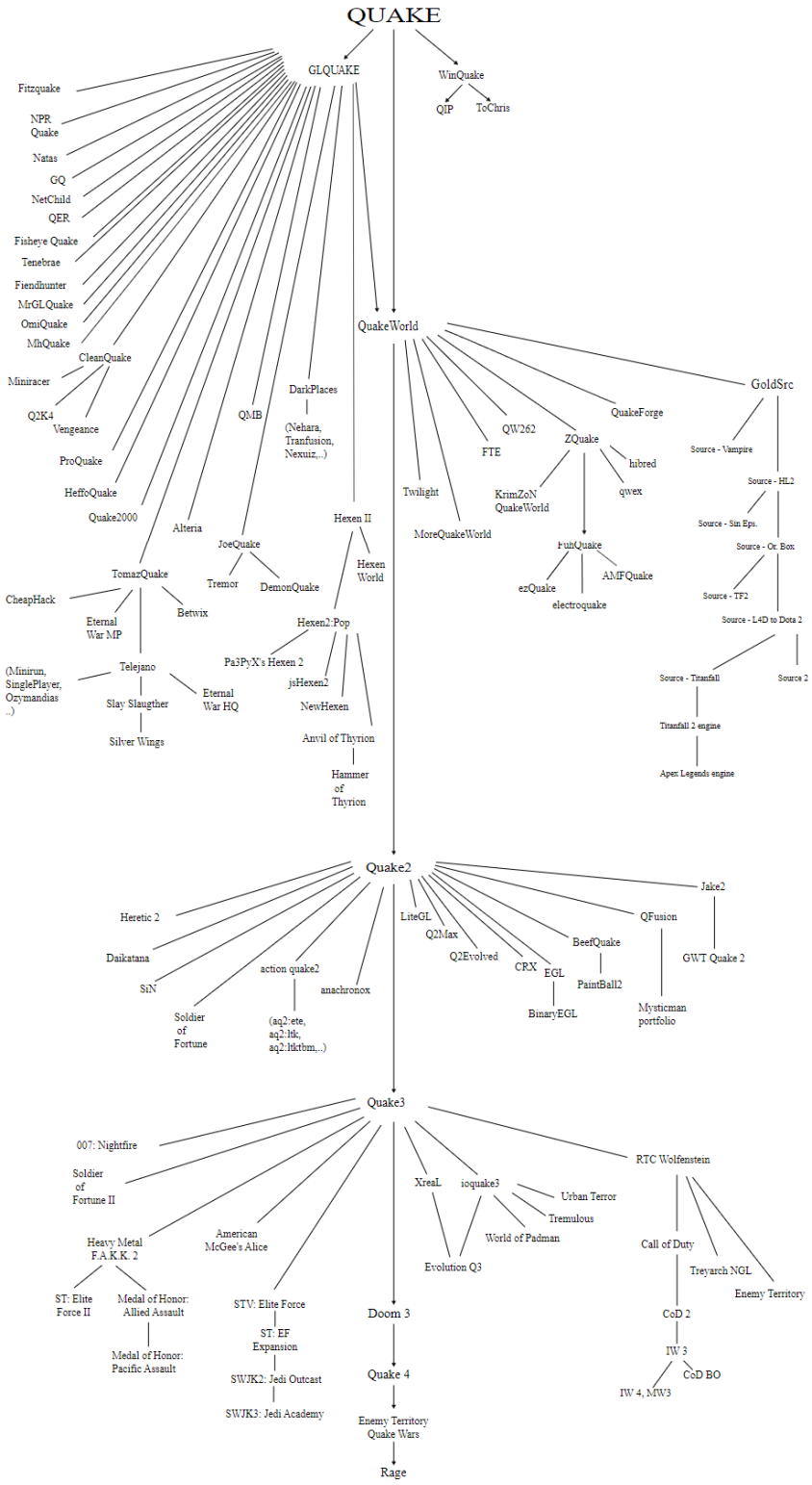

Figure 4. Game engine family tree (Tei, 2021)

The community map *Defense of the Ancients*, or *DotA* for short, from *WarCraft III*'s World Editor, increased its popularity year by year, and defined the genre Multiplayer Online Battle Arena (MOBA[2]) (Moss, 2020). In this game, two teams are given the ultimate objective to destroy their opponents' main structure, under a complicated mechanism involving some computer-controlled units. Every player controls a single character with a set of distinctive abilities. Its sequel *DotA 2* (2013), up to now (2022), ranks No. 1 in the Top Games Awarding Prize Money, with multiple Valve's annual tournament *The International* (*TI* for short) (*Top Games Awarding Prize Money*, 2020). *TI*s have crowd-funded, multi-million dollar cash prize pool and the prize money continues to grow year-after-year (Liquipedia, 2021). *DotA* and *DotA 2* are not the only thriving children in MOBA. *League of Legends* (2009) by Riot Games has a bigger userbase reaching 32 million monthly users, 12 million daily users and it also has a big prize pool and regular tournaments (Funk, 2013). *Arena of Valor* (2017) reached mobile devices while keeping the "5v5" standard MOBA game mode and adding some more game modes with unfixed number of players[3].

2.3 AI in games

Even centuries ago, Shakespeare wrote the line "And all the men and women merely players" in his comedy *As You Like It*. Games not only provide a stage for people to compete, but also have become a stepping stone in the path to real world AI solutions. Research has been done for both traditional board games such as chess and go and modern video games.

2.3.1 AI in traditional games

In 1997, IBM's Deep Blue made history as it became the first machine to beat a reigning world chess champion (Goodrich, 2021). The research did not end here as there was a much

[2] Even though *DotA* is mostly categorized as MOBA, but its official twitter (https://twitter.com/DOTA2) claims it to be a multiplayer action RTS game. The term MOBA is actually coined by the creator of *League of Legends* to escape living in the shadow of the original mod (Funk, 2013).

[3] Official page: https://www.arenaofvalor.com/web2017/gameModesList.html

more complex board game not conquered at that time. By comparison, it can be seen easily that Go is a game with much greater number of possible states. A common misconception from the chess research is that brute force techniques, utilizing good search and evaluation algorithms, is sufficient to solve any problem once it has been formally specified (Burmeister & Wiles, 1995). Most programs fall short of human performance even a beginner-intermediate level because Go has a much larger branching factor. The complexity of Go compared to chess is illustrated in the table.

Features	**Chess**	**Go**
board size	8 x 8 squares	19 x 19 grid
moves per game	~80	~300
branching factor	small (~35)	large (~200)
end of game and scoring	checkmate (simple definition – quick to identify)	counting territory (consensus by players – hard to identify)
long range effects	pieces can move long distances (e.g., queens, rooks, bishops)	stones do not move, patterns of stone have long range effects (e.g., ladders; life & death)
state of board	changes rapidly as pieces move	mostly changes incrementally (except for captures)
evaluation of board positions	good correlation with number and quality of pieces on board	poor correlation with number of stones on board or territory surrounded
programming approaches used	amenable to tree searches with good evaluation criteria	too many branches for brute force search, pruning is difficult due to lack of good evaluation measures
human lookahead	typically up to 10 moves	even beginners read up to 60 moves (deep but narrow searches e.g., ladders)

horizon effect	grandmaster level	beginner level (e.g., ladders)
human grouping processes	hierarchical grouping	stones belong to many groups simultaneously
handicap system	none	good handicap system[4]

Table 1. Comparison of features between chess and go from (Burmeister & Wiles, 1995)

Therefore, the search engine with multiple levels of parallelism from Deep Blue would not work for making a machine to play Go at high level. To master the game of Go, some other technique is required.

In 2015, the breakthrough finally arrived. AlphaGo played its first match against the reigning three-time European Champion, Fan Hui and won the game with a score of 5-0 (DeepMind, 2017a). The solution turned out to be innovative in applying a new approach. DeepMind's AlphaGo introduced a novel combination of Monte Carlo simulation and neural networks. There are two kinds of networks which are value and policy networks. The value network evaluates board positions and the policy networks select moves. Those networks are deep convolutional neural network and the policy networks are trained using supervised learning firstly, reinforcement learning afterwards while the value network is trained only using reinforcement learning method (Silver et al., 2016). DeepMind later made some upgrades to the AlphaGo to facilitate the performance. Eventually, The Future of Go Summit – a five day festival of Go and artificial intelligence in the game's birthplace, China, held several matches to uncover more strategies with a variety of game formats including pair Go, team Go, and a 1:1 match with the world's number one player Ke Jie (DeepMind, 2017c). The matches not only proved that AlphaGo was arguably the strongest Go player in history in a series of matches against human players, but also provided a glimpse of human-AI cooperation in the future (DeepMind, 2017c).

[4] Usually by means of extra stones at the beginning of the game or compensation points for the weaker player.

2.3.2 AI in video games

Even though embedding general AI applications such as natural language processing into a video game and implementing language-based control is a part of the tasks for AI (Cavazza et al., 1999), the majority of research in this field focuses on creating an autonomous agent to achieve human's performance of intelligence or to make the game objects behave in a proper manner. For most of the video games, a typical usage of agent is making the computer-controlled enemies respond and conduct attack to the player. This work belongs to the level design in game development and is usually done with some easy hard-coded rules. Mostly found in single player or cooperative games, those agents are so unnoticeable that their response is also described as "logic" sometimes.

Figure 5. Screenshot from Neo Geo arcade game *Blazing Star*. The stage boss on the right controlled by computer is trying to attack the player's machine with purple laser on screen as well as some other weapons. It is quite common to see enemies in many games to have the ability of locating the player and attacking just like this, which can be considered as perception and action of agent.

Since the main purpose of these agents is to make the game playable and entertaining, these agents are not the most complicated gaming AI system ever created. The researchers tend to pay attention to the player side more. In 2013, Marc and others introduced the Arcade Learning Environment (ALE), a platform providing an interface to hundreds of Atari 2600 game environments (Bellemare et al., 2013). ALE is a software framework for

interfacing with emulated Atari 2600 game environments, a second generation game console originally released in 1977. Atari 2600 games are simpler than the modern console or PC games. ALE allows testing the agents on the games other than the training game, in order to check overfitting. Since the release, there have been multiple solutions from different groups of researchers, including Matthew's neuroevolution algorithm approach (Hausknecht et al., 2014), and DeepMind's reinforcement learning approach (Mnih et al., 2013).

Among all of video games, competitive games are the most challenging to human players as they need to face some other players of various skill levels. These games usually have complex design requiring deep understanding, quick reaction, good foresight, if any player wants to compete at the highest level. *StarCraft* and *StarCraft II* are famous for being the representative esport in the RTS genre where the intelligence is challenged by many ways. This led to the advent of The Brood War API (BWAPI, 2022) in 2009. After that, people started to develop intelligent agents with the API and soon, the first StarCraft AI competition was organized in 2010 (Buro & Churchill, 2012). Since then, the competition is held annually (Churchill, 2022)[5]. The 2011 Man Versus Machine Match saw human player easily defeat Skynet, the winner of the computer competition (Buro & Churchill, 2012). However this time, unlike previous board human vs machine games (chess), human player's victory is not that critical anymore. Players have already realized that, in theory, by issuing commands to units at an impossibly high rate for human, machine players are able to gain a huge advantage in the game. Despite the computer agents could break the limit of human body to issue orders to the units at an incredible speed, people still expect to see an AI player playing the game in the way of human player – a mixture of economic planning, high-level strategy, and medium level of unit micromanagement for combat. Until 2013, in the integration architectures of these bots, there are two main tools being used. One is abstraction, the other one is divide and conquer (Ontañón et al., 2013).

Years later, the mature agent was born in 2019. DeepMind streamed an exhibition game of their StarCraft II intelligent agent AlphaStar. The agents were trained to play the game in Protoss vs Protoss match-up on the map CatalystLE. The exhibition game

[5] From the official website, the bots can be acquired together with the source code.

invited professional players Dario "TLO" Wünsch and Grzegorz "MaNa" Komincz. This time, AlphaStar had an average APM of around 280, significantly lower than the professional players, but they interacted with the game engine directly via its raw interface (Vinyals, Babuschkin, Chung, et al., 2019). The professional players lost all 10 games to AlphaStar, but they did not feel as unfair as playing against superhuman. Dario "TLO" Wünsch stated “While AlphaStar has excellent and precise control, it doesn’t feel superhuman – certainly not on a level that a human couldn’t theoretically achieve” in the summary (Vinyals, Babuschkin, Czarnecki, et al., 2019).

Just like how DeepMind created AlphaGo using artificial neural network, in AlphaStar, DeepMind once again chose neural network accompanied by self-play via reinforcement learning, multi-agent learning and imitation learning. Apart from self-play, anonymised human games released by Blizzard allowed AlphaStar to learn the basic micro and macro-strategies used by players on the StarCraft ladder (Vinyals, Babuschkin, Chung, et al., 2019).

The power of reinforcement learning with deep neural network is shown in AlphaStar. However, they are not the first to adopt such learning-based system. Early in 1992, researchers at IBM developed a self-teaching backgammon program, TD-Gammon, achieving a strong level of play. Similarly, it was a neural network based on reinforcement learning algorithm (Tesauro, 1994). The program was tested by several highly rated human players at that time, and the results were all negative for the program player. Moreover, the stochastic feature was emphasized in the summary. The dice rolls stochastically force the system into some state in the game space, which has the effect of a certain minimum amount of exploration for the agent.

DeepMind also knocked the door of shooter game as they presented reinforcement learning in the game of Quake III Arena Capture the Flag. Capture the Flag is a team-based mode where two teams are placed in a world with terrain and obstacles. For both teams, the objective is rushing to the opponents’ base, stealing their flag back while avoiding being knocked down. Scoring needs both enemies’ flag and own flag at the base. Capture the Flag included team strategy while not neglecting personal play skill. This is a

complex, multi-agent environment and it is about not missing shots on the enemies and cooperating with the teammates. The team-based gameplay, the changing maps and the raw observation raised more challenges to the learning method (Jaderberg, Czarnecki, Dunning, Marris, et al., 2019). Again, due to human's slower biological signaling, those agents are modified to have an inbuilt delay to compensate the reaction time. Surprisingly, the trained agents discovered some strategic behaviors to cooperate in a complementary manner (Jaderberg, Czarnecki, Dunning, Graepel, et al., 2019).

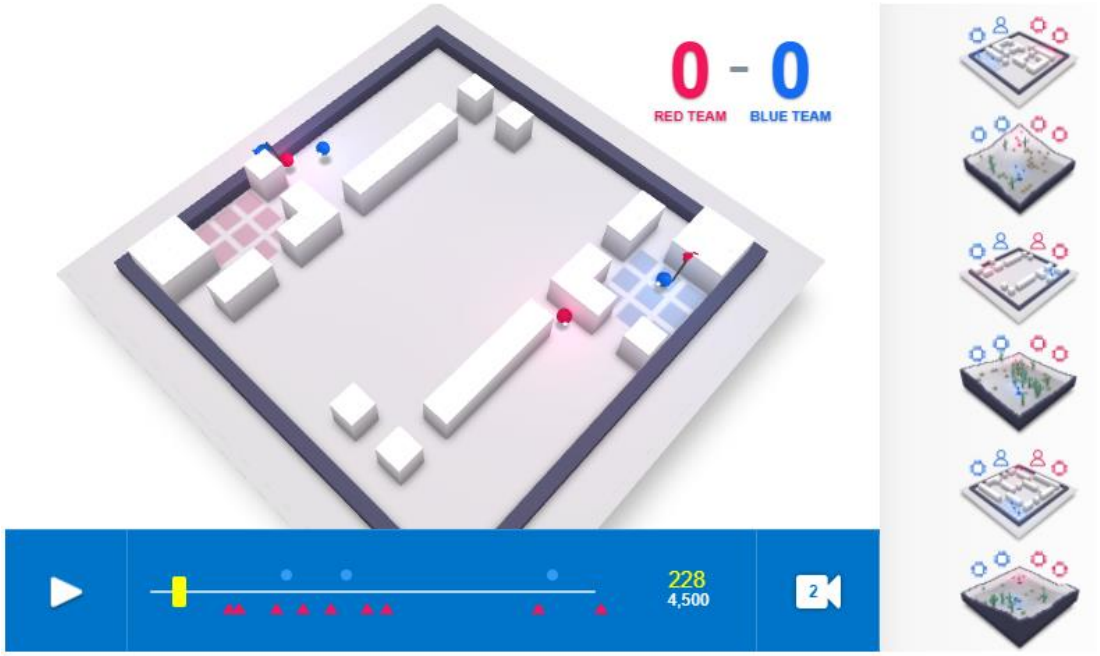

Figure 6. Screenshot of interactive CTF game explorer from (Jaderberg, Czarnecki, Dunning, Graepel, et al., 2019). This is a very interesting situation. Both teams' flags are almost brought to the opponents' bases so none of them can score. The best hope of carriers is that the teammate takes out the enemy carrier so the flag may be returned immediately and it may score. However if both carriers are tagged in a short time then both flags may be returned before any team scores. Or the teammates can be taken down and the carriers meet them at their own base without own flag.

One more important attempt is related to *Dota 2*. *Dota 2* is by far the most complex esport game on public scene in terms of the design. The total amount of heroes in *Dota 2* exceeds the number of units (for combat) from all three races in *StarCraft II*. Each hero is unique and has no less than 4 abilities and some other attributes. On April 13th, 2019, OpenAI Five defeated the world champions Team OG. OpenAI Five was also opened to the game community for competitive play and it won 99.4% of over 7000 games (OpenAI et al., 2019). It was not a surprise that OpenAI Five used reinforcement learning techniques and self-play training as well. Despite engineered with powerful techniques, the game could be played only under a set of restrictions for those agents (OpenAI, 2018a).

2.4 Summary

In this chapter, the history of AI, the history of video games, and those solutions of AI used in either video games or esports are reviewed and explained briefly. The whole pictures of both scenes are shown.

The first section narrated the development history of AI. The earliest AI products are mostly based on symbolic approach (Russell & Norvig, 2009). They are implemented to integrate knowledge in computer programs, so they ignored many intuitive aspects of human cognition. This led to the AI winter and researches were stopped for a long time due to lack of funding (Haenlein & Kaplan, 2019). After the AI winter, in 1990s, those problems were resolved by deep learning, and in the last decade, deep learning was enhanced greatly by "Big Data" (Goodfellow et al., 2016).

The second section presented the history of video games. Video games from the categories of fighting, FPS, RTS, MOBA are explored mainly in respective time orders. The relations between some games are also illustrated. These genres have the most players and the most awarding prize money (*Top Games Awarding Prize Money*, 2020).

The third section covered the use of AI in games. For traditional board games, intelligent agent is created to act as a genuine player. AI had two influential battle against human players in chess and go (DeepMind, 2017a; Goodrich, 2021). Despite both battles emphasized the high level of play that artificial player achieved, their methodologies are

essentially different. In the video games, intelligent agents are commonly embedded in game characters. Those competitive games not only host the tournaments for best human players, but also act as a testbed for intelligent agents. Intelligent agents in *StarCraft*, *StarCraft II*, *Quake III Arena*, *Dota 2* are mentioned in this part. The common attribute of these agents is that most of them use a reinforcement learning setting, so the neural network inside could learn the game through self-play or multi-agent training.

3 Game playing AI technology

From now on, the focus is the AI technologies applied in video games, especially agent technology. In artificial intelligence, the term 'agent' has multiple versions of definition. Since here the context is video game, the term could be specified within game environment. One 'agent' or 'intelligent agent' (IA) refers to the partial computer program that can act as one player in multiplayer modes, as known as 'bot'. Some equivalent computer-controlled characters in single-player are also included in the concept due to their similarity.

3.1 Different Targets

3.1.1 Developer's Side

Unlike the research being conducted by scientists in the recent years, AI has been developed and used by the game developers for a long time. Those agents are put in the games mostly because the vacant role of player needs to be filled in order to make the game enjoyable. A lot of games marked as esports nowadays have built-in agents to achieve this target. This greatly alleviates the pain for the newcomers to start learning the game as they do not need to face an unknown player from random skill level, thus reducing the randomness of gameplay and possibility of coping with bad manners in game. Designing agents is not always the same concept as designing AI system for games. In the book *Artificial Intelligence for Games*, Ian and John divided AI in games into 3 categories (Millington & Funge, 2018):

1. Hacks (ad hoc solutions and neat effects)
2. Heuristics (rules of thumb that only work in most, but not all cases)
3. Algorithms (the "proper" stuff)

The method of hacks does not care the nature of reality or mind. It only wants the characters to look right. This method correlates behaviorism the most, as the early psychological approach of AI is making a machine to replicate the human intelligence by

hard coding. One good example of this type is the "gesture" command in *Quake II*[6] as the gesture can flip their enemy off so the characters feel emotional, but it has nothing to do with the gameplay. Hack is not constrained, so it is not common to find it in competitive agents. Triggering behavior beyond the rule can be disastrous for game experience. The notorious *Street Fighter II* AI is a representative of such situation. The agent in the game cheats in 5 ways according to a video[7] where all these cheats are presented, or human players fail if they try to reproduce the behavior. A research into the disassembly of the source byte code reveals that the moves of agents are simply executed as instructions, without making the input like how human players do (Sf2platinum, 2017). The moves of computer-controlled character skip the game rule, thus causing unpleasant results to many players.

Ian and John wrote that "A heuristic is a rule of thumb, an approximate solution that might work in many situations but is unlikely to work in all" (Millington & Funge, 2018). Heuristics are general principles found in the past, often related to pathfinding or goal-oriented behaviors. In terms of esports, one simple and clear target is to play and win the game against any kinds of opponent. Since the agents could not guarantee 100% victory for every game, by this definition, those agents fall into this category. However, it is clear that both game developers' agents and scientists' agents are created based on simple guideline, achieving victory. Both sides have specific evaluation for agent players but in general those agents are not supposed to break the rule, suppressing human players in every single game.

Algorithms are the once-and-for-all solution, but most of them are designed for smaller tasks such as movement, pathfinding, decision making and so on. They do have a serious impact on traditional board games such as *Renju*[8] where the first player has an overwhelming advantage (van den Herik et al., 2002). It is verified that small scale traditional board games are vulnerable to search algorithms. The situation is predicted early in 1913 by E. Zermelo, a German mathematician. Zermelo's conclusion is that, in two-

[6] https://en.wikipedia.org/wiki/Quake_II

[7] How the CPU cheated you in Street Fighter 2: https://www.youtube.com/watch?v=laUAgEUunsI

[8] A board game where the target is to place five stones in a row vertically, horizontally or diagonally.

person zero-sum games with perfect information, if a player is in a winning position, then that player can always force a win no matter what the other player may employ (Schwalbe & Walker, 2001). On the contrary, there is no silver bullet for most video game playing agents because they run on fast simulation and have too broad legal moves.

For game developers, designing agent players requires the game context. Therefore, only after the game is defined, the bots are created to fit in the game. The Quake III Arena Bot is a well-designed, well-documented example from the last century for the competitive classic. There is a detailed description of how the bots are supposed to be by the developer:

> *"... As such, the bot should be hard to distinguish from a human player... The bot also has to navigate through the environment in a life-like manner, pick up items and handle weapons just like human players do... To make the game more enjoyable and more versatile, there has to be a range of different bot characters that each play the game in their own style, and provide different challenges for the human player. The bot has to be a fair opponent and should in no event cheat. The communication with a bot should also be hard to distinguish from communication with human players. The bot has to be able to chat with other players. The bot also needs to communicate with team mates in the team based game modes like CTF... with both human players and other bots.*
>
> *The bot also has to meet a number of requirements on a technical level. The bot has to be resource efficient, both CPU and memory usage have to be low...*
>
> *Third party developers have to be able to easily modify and customize the bot AI code to make the bots work with new game variants and modifications...."* (Van Waveren, 2001)

These words contained a very comprehensive list of requirements, covering the elements of navigation, item interaction, personality, communication, technical efficiency, customization, etc. In fact, some elements are redundant and missing in some agents from

other video games such as communication, customization. It is also rare for human players to chat with text during the game, because players are usually busy in doing the actions one by one without idling. The rest qualities are shared throughout many games even in other types. The requirement of those qualities is brought by the game itself. If the game defines a space for players, then there needs a hint for the locations. If the game defines an item, then the tutorial teaches how to use the item. Even though the methods are different for human and computer, playing a game is mostly guided by the game itself.

Figure 7. Sign and waypoints in Counter-Strike. On the left a sign is painted for human to navigate to the site A. But bots can 'see' those extra green, red, purple waypoints which are invisible for human and their move is strictly from one waypoint to another close one. Image on the right from https://gamebanana.com/mods/40119.

3.1.2 Scientists' Side

Programming a computer to play games has always been a goal of the AI research, because in common sense playing games require a normal level of intelligence. In 1950, Claude Shannon published paper developing a computer program to play a tolerably good game of chess (Shannon, 1993). In 1951, Alan Turing also did a similar job, creating an algorithm for chess which lost to a weak human player (Turing, 1953). Even after that for a long time, chess was considered as a medium of human intelligence and no computer program is able to defeat the best player in the world. While researchers tried their best in the following years to make a better chess playing computer, there was, at the same time, some debate around

whether computers really ‘think’ or not (Turing, 1951). People often described computers as mechanical brains, but some scientists knowing what computers are had the opposite point of view. For this question, Turing invented the famous Turing Test, a.k.a. the Imitation Game (Turing, 1950). In a Turing Test, player C is unable to see either player A or player B and can communicate with them only through written notes. Player C tries to figure out which is the human and which is the machine merely with their answers. The Turing Test is highly influential due to its simplicity of how the definitions of intelligence and thinking are defined, while it is criticized for being too subjective and not an indication of machine intelligence (Marcus, 2014). Nevertheless, Turing Test is widely recognized method and customizable. The games can also use Turing Test as a criterion. It only requires a change from writing answers to playing games in this case. People debated whether this judgement is too arbitrary or not. But the debate had not been the main focus for the future of research.

In the history before 2000, the biggest advances in computer playing had come as a result of work done on the alpha-beta search algorithm (Schaeffer, 2001). In game AI taxonomy, it is categorized as algorithm. If the game states are small enough to be traversed, the algorithm type AI solves the problem completely. Deep Blue by IBM reached a new high altitude of this approach. Deep Blue is a beast of lightning speed. It searches more than hundreds of millions chess positions per second. But this is enabled by special-purpose chess chips, meaning that they are not capable of tasks other than chess simulation (Campbell et al., 2002). In 1997 Deep Blue defeated Garry Kasparov by a score of 3.5-2.5. It is marked as a milestone in the history of computing science. But the end of the match left Kasparov unsatisfied. Kasparov demanded a rematch, but IBM refused the rematch, and retired Deep Blue soon after the match (McCorduck & Cfe, 2004; Schaeffer, 2001). Up to this, artificial intelligence was acting as the opponent of human’s intelligence.

Search algorithm showed its limit when coping with the game of Go. Meanwhile, around 2010, deep learning and reinforcement learning technology has gradually become useful as many domains including speech recognition, visual object recognition were dramatically improved by those technologies (LeCun et al., 2015). Researchers found new possibility of mastering the game of Go with these promising technologies. This time, a new kind of challenge arose as the learning technologies are not the same as those used in Deep

Blue. The nature of deep learning is not encoding the knowledge or the strategy in machines anymore. What deep learning does is reading the datasets and using the backpropagation algorithm to indicate how a neural network should change its internal parameters that are used to compute from the input layer to the output layer (LeCun et al., 2015). This essential property of deep learning shows great potential of handling all kinds of problems with such a general solution of learning instead of traditional specific problem-solving engineering. The game of Go has become an experiment to prove that potential.

The first man to get a taste from the Go playing machine in matches is Fan Hui, the European champion. He had 5 formal games and 5 informal games. AlphaGo won these games 5-0 and 3-2 respectively (Silver et al., 2016). Fan Hui firstly felt very strange and unhappy for losing, but he was also happy to witness this in the history, then he joined the research group (DeepMind, 2020). To prove that AlphaGo had really mastered Go to the high level, DeepMind found Lee Sedol, a 9 dan Korean player, and agreed to have a match, broadcast live, with AlphaGo. The match was under spotlight. Lee Sedol was originally confident of beating AlphaGo by a score of 5–0, but he changed his mind when he saw the creative move by AlphaGo and he lost the series 1-4 (DeepMind, 2020).

The challenge continued and eventually the research team reached China, the birthplace of Go. Apart from the match between the number one Go player Ke Jie and AlphaGo, the event ‘Future of Go Summit’ featured some other cooperative game formats (DeepMind, 2017a). From the event, there is an emphasis from the researchers that the event’s target was not only beating the best players in the world, but also discovering the undiscovered strategies and possibilities of collaboration between human and artificial intelligence (DeepMind, 2017b).

The difference between board game and video game does not result in a full change of techniques used in creating artificial players by researchers. Reinforcement learning with neural networks works well with video game as it was demonstrated that the network has outperformed expert human players on basic video games of Atari (Hausknecht

et al., 2014; Mnih et al., 2013). However, as video game has an alias, 'the ninth art'[9], meaning novel, highly compound art, it is reasonable to think that the gameplay elements, the aesthetic elements, and the sport elements of new era's video games would bring some new questions to AI, even though at the moment, the research is still following 'the extended old path', overcoming the complexity of problem and creating a capable agent. In this way, esports are chosen to be the testbed.

When the researchers started to dig into the problem, they soon found several important distinctions resulted by the mechanism of games. For example, real-time strategy (RTS) game is a simulation of economy, military and combat, having the following differences compared to chess:

- They are "real-time" simultaneous move games. The games have such kind of characteristic that is found in sports. Just like how a flying ball is pulled down towards the ground by the gravity in the game of football, there is change of the game state in esports games regardless of players' interference. This flow of state changes can be influenced by the players in the way they want by taking actions at any time before the victorious is decided.
- In most RTS games, there is a concept called 'fog of war' that makes the situation only partially observable. Therefore, unlike chess and go, players in RTS games do not always have the perfect information like that in board game.
- There is little random factor in RTS. Usually, there is a small random delay in some actions before they take effect. But in general, given all players' input into the game, the result is predictable.
- Finally, the complexity of video games is basically nonenumerable. Esports video games are like most of video games, designed for human. They run on concepts built for human to understand and play the game. Human can summarize the state of the game with them, but the raw space of possible states is so large that there is

[9] A concept in Chinese video game community. Explanation from encyclopedia sites (in Chinese): https://zh.wikipedia.org/wiki/%E7%94%B5%E5%AD%90%E6%B8%B8%E6%88%8F and https://zh.moegirl.org.cn/index.php?title=%E7%AC%AC%E4%B9%9D%E8%89%BA%E6%9C%AF

no way for any computer to enumerate and simulate all of them. The state space of *StarCraft* in a typical map is estimated to be many orders of magnitude larger than that of Go (Ontañón et al., 2013).

Among these features, the first one is the most impactful to human players. Human playing esports is under the physical limit of body. No matter how fast the best player tries to issue command, it is still too slow to match up the speed of scripted actions. The same situation applies to reaction. The average response time of human reacting to a simple signal is around 250ms while that for machine could be less than the interval of two refreshes on screen. At those simple tasks, machine overwhelmingly suppress human[10]. There are two directions out. One is to force the agent to play fairly, acting in human speed. This is the choice of researchers who want result which is closer to human intelligence, such as DeepMind. The other one opens a new competition for all kinds of AI using extraordinary playstyle. The platforms for *StarCraft* AI and *StarCraft II* AI are AIIDE StarCraft AI Competition (Churchill, 2022) and SC2 AI Arena[11] respectively.

For the legitimate side, the achievement is delightful. A showcase live of AlphaStar is joined by commentators and professional *StarCraft II* players[12]. The learning based AlphaStar was recognized by the pro-gamers, even though they are defeated by the AlphaStar in most games. They thought that they played a fair game (Vinyals, Babuschkin, Chung, et al., 2019). So far, a limit on agent player is added to prevent the overgrowth of it, in order to make it behave like human. The attention paid to applying proper limitation is unprecedented this time.

From the history, it can be seen that the games being chosen for research have become more and more complicated. Correspondingly, the solutions are prepared and

[10] Automaton 2000 Micro Bot is scripted agent with insane unit control. Early in 2011, no longer after the release of *StarCraft II*, it proved that using computer control of units can gain a huge advantage of combat. Demo link: https://www.youtube.com/watch?v=IKVFZ28ybQs

[11] https://sc2ai.net/

[12] https://www.youtube.com/watch?v=cUTMhmVh1qs

modified for the different challenges. There are certain types of qualities that researchers would like to see in the agent. The table below shows a summary of the games and AI solutions.

Game	**Year**	**Name**	**Target**	**Approach**	**Significance**
Chess	1997	Deep Blue	Beating the best player in the world	Search algorithm	First time a chess world champion lost to a computer program.
Go	2016	AlphaGo	Winning against the best player in the world, cooperative showmatch	Reinforcement learning, Monte Carlo method	AI faced the previously impossible challenge with new method, showed possible collaboration with human.
StarCraft II	2019	AlphaStar	Grandmaster level in the game of unprecedented complexity	Reinforcement learning	AI applied to a real-time simulation under critical restriction

Table 2. Some games solved by AI in the past.

3.2 Agent detail

Since the agents are made for different games, and distinct in terms of design, the details of agents are going to be explained in 3 steps. The first step is the agent's interaction interface. This concerns how the agents see the situation and take actions in the game. The second step is to study the core action generator, which represents how the agent thinks and responses.

The third step is the discussion on how differently the agent is operating from the way human does. This comparison will include both the interface and the learning.

3.2.1 AlphaStar

AlphaStar is DeepMind's *StarCraft II* playing agent. AlphaStar uses a deep neural network that is trained from raw game data by supervised learning and reinforcement learning (Vinyals, Babuschkin, Chung, et al., 2019). The training process is enabled by PySC2 (Vinyals, Gaffney, et al., 2017). PySC2 is a part of the tool set, SC2LE, released by DeepMind together with the partner Blizzard Entertainment, the creator of *StarCraft II*. The release of SC2LE contains the following items (Vinyals, Gaffney, et al., 2017):

- An application programming interface (API) as a client for full external control of *StarCraft II*, StarCraft II Client (s2client-proto)[13].
- A dataset of anonymized game replays. There are more than 65k of game replays.
- An open-source version of DeepMind's toolset, PySC2, to allow researchers to easily use Blizzard's feature-layer API with their agents[14].
- A series of simple RL mini-games. They are some simple maps created to test the performance of agents on specific small tasks such as sole mining.
- A joint paper that outlines the environment and reports initial baseline results on the mini-games, supervised learning from replays and the full 1v1 ladder game against the built-in AI.

StarCraft II has a very rich action and observation space. For Observation in game, *StarCraft II* contains both a spatial space camera, and an extra UI around the main screen showing a lot of information. See Figure 8 for a view of human's observation and explanations of items.

[13] Github repository of s2client-proto: https://github.com/Blizzard/s2client-proto

[14] Github repository of PySC2: https://github.com/deepmind/pysc2

Figure 8. Human's observation in *StarCraft II*.

For the agents, the RGB pixels are available for only the main screen as well as for the minimap. This enables the agent to see like a human, but does not include all the UI elements in human's screen like the command card, unit status. They are exposed as rgb_screen and rgb_minimap (DeepMind, 2019).

The minimap is a view of the entire battlefield. It shows everything from the player's all units' visions, but it is rendered in low resolution. So, it does not provide all the details, but it is useful for checking the enemy's move around before the enemy launches a surprise attack (DeepMind, 2019).

The screen is a higher resolution view of the selected part of the battlefield. It occupies the most area in Figure 8, containing view of many units. It requires certain level of familiarity with the game for a human player to read information quickly from the screen, as there could be lots of units distributed all over the screen. For the agent, it is rendered from a top-down orthogonal camera, as opposed to a perspective camera that a human use (DeepMind, 2019).

Moreover, the game exposes feature layers from the RGB pixels. They are unavailable to human's normal play but they represent the decomposed, structured information. The feature layers are functioning like computer vision technology but they offer faster, resource-saving analysis results by omitting visual neural network. The minimap feature layers are terrain levels, visibility, creep, camera position, visible units' owner id (also a layer telling whether they are friendly or hostile), and selected units. Since the screen is on a specific area of interest, the screen feature layers show everything of the area which is readable from the minimap feature layers. Additionally, the screen feature layers present more information about unit – unit type, hit points, energy, shields, unit density (how many units are in this pixel), unit_density_aa (An anti-aliased version of unit_density with a maximum of 16 per unit per pixel. For the agent, it plays in a much lower screen (feature layer) resolution. So if many units are overlapping, the information would be inaccurate. It helps the agent understand how many units are actually in a small area.). The screen and the minimap together with their feature layers offer a basic visual ability, as the observation, to the agent. See Figure 9 for a visualized representation of feature layers (DeepMind, 2019).

For most of the non-spatial data, the game delivers them to the agents as tensors. They mainly mirror the information shown in the bottom part of human interface except the minimap in Figure 8.

Content	**Tensor shape**	**Details**
General player information	(11)	player_id, minerals, vespene gas, supply used, max supply, army supply, worker supply, idle worker count, army count, warp gate count (for protoss), larva count (for zerg)
Control groups	(10, 2)	showing the (unit leader type and count) for each of the 10 control groups
Single select	(7)	unit type, player_relative, health, shields, energy, cargo size, build progress as a percentage if it is still being built
Multi select	(n, 7)	the same as single select but for all n selected units
Cargo	(n, 7)	similar to single select but for all units in a transport
Build Queue	(n, 7)	similar to single select but for all units being built
Available actions	(n)	all the actions ids that are available at the time of this observation (command card)
Last actions	(n)	all the actions ids that were made successfully since the last observation
Action result	(n) (usually 1)	the result of the action
Alerts	(n) (max 2)	when being attacked in a major way

Table 3. Non-spatial observation data tensors of PySC2.

Figure 9. Demo of the feature layers on the right for agents.

The SC2 action space is very big. In addition to selection on screen, all possible moves which could be issued by the player to the unit are shown in the command card. There are totally 15 slots in the command card for either clickable button or passive icon. Many actions are combinatorial, taking a point in screen or minimap, or a specific unit in the game. If the action space is flattened into a single dimension, there would be millions or even billions of possible actions, which is not ideal for agent program to operate. Instead, PySC2 created function actions that are rich enough to give composability, without the complexity of an arbitrary hierarchy. Each action is a single FunctionCall in pysc2.lib.actions with all its arguments filled. The full set of types and functions are defined in pysc2.lib.actions. The set of functions is hard coded and limited to just the actions that humans have taken in competitive games (DeepMind, 2019).

To see all existing actions run:

```
python -m pysc2.bin.valid_actions
```

This prints a long list like:

```
...
```

```
345/Rally_CommandCenter_screen
(3/queued [2]; 0/screen [84, 84])

346/Rally_CommandCenter_minimap
(3/queued [2]; 1/minimap [64, 64])

...

352/Research_AdvancedBallistics_quick
(3/queued [2])

353/Research_BansheeCloakingField_quick
(3/queued [2])

...
```

The meanings of the functions are indicated by their names. For example, Rally_CommandCenter_minimap takes one argument named minimap, which requires two integers in the range [0, 64), representing the coordinates on the minimap.

The random agent is a code example of issuing random commands and consumes ValidActions and fill FunctionCalls randomly, but generates mostly useless commands. It is implemented as follows (DeepMind, 2019):

```
import numpy

from pysc2.agents import base_agent

from pysc2.lib import actions

class RandomAgent(base_agent.BaseAgent):

  """A random agent for starcraft."""

  def step(self, obs):

    super(RandomAgent, self).step(obs)
```

```
    function_id =
numpy.random.choice(obs.observation.available_actions)

    args = [[numpy.random.randint(0, size) for size in
arg.sizes]

            for arg in
self.action_spec.functions[function_id].args]

    return actions.FunctionCall(function_id, args)
```

AlphaStar's behavior is generated by a deep neural network of complex architecture that receives all observation from the start of the game as inputs, and selects actions as outputs. The agent architecture consists of general-purpose neural network components that handle StarCraft's raw complexity. Observations are processed using a self-attention mechanism (Vaswani et al., 2017). Scatter connections are included to integrate spatial and non-spatial information. To deal with fog of war, a deep long short-term memory (LSTM)(Hochreiter & Schmidhuber, 1997) system is at the core of the network, allowing the agent to remember some information. An auto-regressive policy (Vinyals, Ewalds, et al., 2017) and recurrent pointer network (Vinyals et al., 2015) are utilized to manage the combinatorial action space (Vinyals, Babuschkin, Czarnecki, et al., 2019).

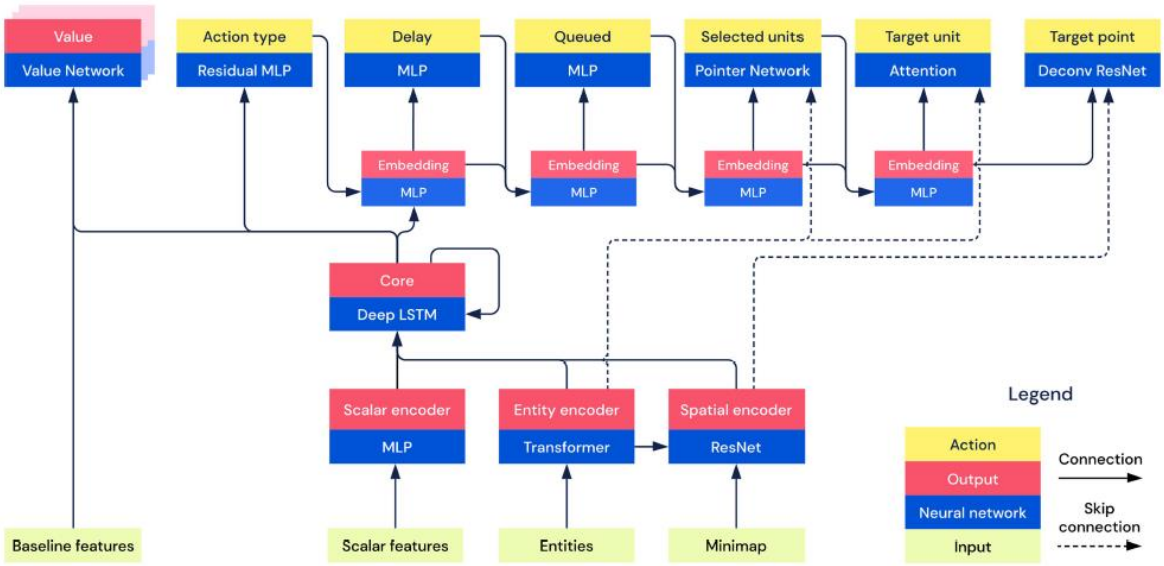

Figure 10. Overview of the architecture of AlphaStar from (Vinyals, Babuschkin, Czarnecki, et al., 2019).

AlphaStar is trained by a hybrid of supervised learning and reinforcement learning. Agent parameters were firstly adjusted by supervised learning with a public human replay dataset. The network was trained to predict actions given all the observations and summary of strategies. The procedure created a diverse set of agents adopting human play modes (Vinyals, Babuschkin, Czarnecki, et al., 2019).

Then the agents enter the reinforcement learning stage, aiming at maximizing the win rate. The reinforcement training for AlphaStar is off-policy learning, updating the agent's policy from experience generated by a previous policy. A combination of temporal difference learning (TD(λ))(Sutton, 1988), clipped importance sampling (V-trace)(Espeholt et al., 2018), and a new self-imitation algorithm (UPGO)(Oh et al., 2018) are used to make the agents learn effectively (Vinyals, Babuschkin, Czarnecki, et al., 2019).

The reinforcement training is conducted in a continuous league, a system similar to the StarCraft ladder. The league functions as a match-making manager, so the agents could play as competitors against some other agents. By selecting the opponent mixture, agents get their own learning objective: beating other competitors with certain style of playing. New competitors are added to the league, by branching from the existing one, while some are

frozen. This creates diversity of strategies in the league. After training using Google's v3 TPUs for 14 days, each agent has experienced up to 200 years of SCII play, and the final AlphaStar agent is discovered from the most effective mixture of strategies, that could run on a single desktop GPU (Vinyals, Babuschkin, Chung, et al., 2019).

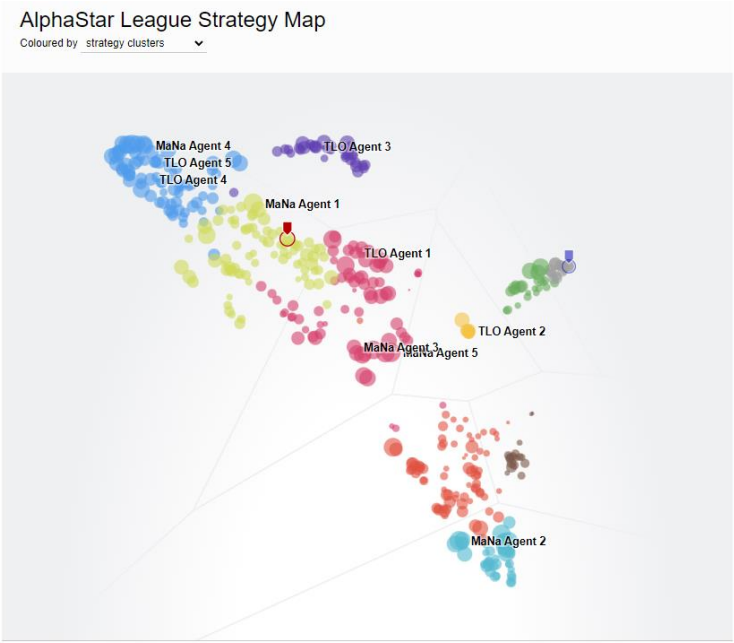

Figure 11. AlphaStar Strategy Map from (Vinyals, Babuschkin, Chung, et al., 2019). Colors represent strategies categorized by average units built in game. The yellow part has agents using primarily a composition of ground units while the agents of grey part on the right use void ray strategy.

In general, PySC2, the foundational interface of AlphaStar is based on imitation of human player's observation. Although the agent's observation differs from human seeing the game through a monitor, it is reasonable to exclude the trouble of graphical computation for the sake of both efficiency and avoiding confusion of game settings. Primary observation is still made through a camera with multiple layers of features. There are two other special characteristics for the agent's camera only. First, it is a top-down orthogonal camera, as opposed to a normal camera that has an angle to the ground. It is more like a scanner rather than a real camera that could show 3D shape when filming from multiple angles. Second,

the default resolution is only 84*84 big, which is very small compared to the common monitors that are 1920 pixels wide (DeepMind, 2019). Though the camera is supplemented by feature layers, from the replays it is observable that the agents make mistakes related to spatial clicking such as pulling extra workers to the gas field. However, in scalar parts of observation, the agent has more information than human player. For example, supply used by workers is not available to human interface, as well as the build queue. Human can infer these information from information taken seconds or minutes ago during playing, but it would require some attention, which is limited as human has to manage the dire battlefield.

One primary concern of human players is that the agent could issue commands at an impossibly high speed to gain advantage. Luckily, APM (actions per minute) limit can be applied to the agents during training. Apart from that, the commands issued by the agent are different in a minor way. Agent does not have direct access to the command button, actions are done by invoking functions. Since the functions are hard-coded, any unprepared action is not available to the agent at all. Besides that, as mentioned before, there is inaccuracy in actions by the agent due to the small screen resolution and delay.

AlphaStar's agents are 'initialized' by supervised learning with human replays and strengthened by reinforcement learning of self-play (multi-agent league). The training to the best agent costs over 44 days of training time (Vinyals, Babuschkin, Czarnecki, et al., 2019). While this is a standard procedure for reinforcement learning, it is quite a long time of pure competition from human's perspective, because human's learning consists of not only trial-and-error. *StarCraft II* is a computer game so there are many textual, graphical, and auditory elements in the game to guide and amuse people. One typical experience route is playing the single player campaign mode (story mode) first, then to the competitive multiplayer mode. Players who have finished the campaign mode are mostly able to play against some other player since there is huge similarity in the modes, even though probably not at a high skill level. Missions from the campaign mode mostly have objectives other than destroying an enemy, so the abilities of 3 phases, which are mining, building, combat, may be yielded in a distinct sequence from that of agent (Figure 12).

Moreover, human could read the description of any unit and ability in text so that player could know the units and speculate how the unit does. AlphaStar is programmed only to accept replays as an external learning source, certainly not having full intelligence of human.

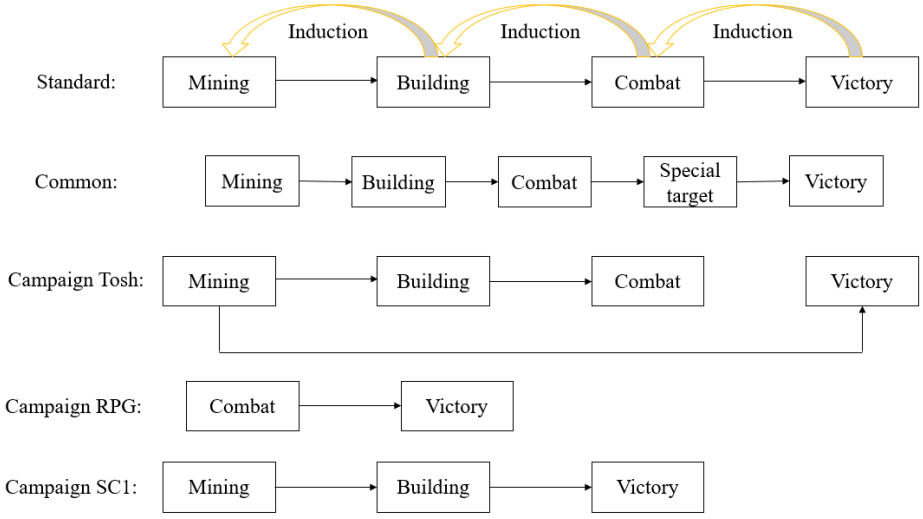

Figure 12. Ability chain for different game modes in *StarCraft* I and II. Black arrow means the previous ability enables ability being pointed, or achieve the goal. Given that victory is the biggest and the ultimate reward defined by the game, it is used to induce the associated abilities for agents (Silver et al., 2021). This creates some sub-goals and those goals are supposed to contribute to achieving the final goal. However, the victory condition is not simply beating the enemy in most campaign missions. Human shows high versatility by either combining the puzzle piece of abilities or understanding the requirement through language.

3.2.2 OpenAI Five

OpenAI Five is OpenAI's bots playing the competitive 5 vs 5 game *Dota 2*. It uses a deep reinforcement learning method, which is also marked as a scaled-up version of Proximal Policy Optimization. It plays the game with some additional restrictions (OpenAI, 2018a).

Dota 2 is the sequel of MOBA's pioneer, the original *Dota*. It has the highest prize pool in esports scene and the most amount of game concepts. Therefore, *Dota 2* has a very huge observation space. The *Dota 2* player interface shares similarity with RTS games due to the relation, but it focuses more on the attributes and the items of a single unit, especially player-controlled heroes.

Figure 13. *Dota 2* game screenshot.

An OpenAI Five bot sees the current game state with Valve's Bot API, which is the programming interface provided by the game's developer (OpenAI, 2018a). One of the agents at each step observes around 16,000 inputs from the game state. The observation is mostly real numbers plus some integer categorical data. This makes up an array as the observations. Generally, the agent's observation is more semantic than human's indirect observation on screen. And the AI program skips visual processing, so all observations are done instantly, whereas a human needs to click into various menus and options to get these data (OpenAI et al., 2019). See Table 4 for the full list of observations and Figure 14 for a schematic outline.

Global data	22
time since game started	1
is it day or night? time to next day/night change	2
time to next spawn: creep, neutral, bounty, runes	4
time since seen enemy courier is that > 40 seconds?	2
min&max time to Rosh spawn	2
Roshan's current max hp	1
is Roshan definitely alive?	1
is Roshan definitely dead?	1
Next Roshan drops cheese?	1
Next Roshan Drops refresher?	1
Roshan health randomization	1
Glyph cooldown (both teams)	2
Stock counts (gem, smoke of deciept, observer ward, infused raindrop)	4

Per-unit (189 units)	43
position (x, y, z)	3
facing angle (cos, sin)	2
currently attacking? time since last attack	2
max health last 16 timesteps' hit points	17
attack damage, attack speed	2
physical resistance	1
invulnerable due to glyph? glyph timer	2
movement speed	1
on my team? neutral?	2
animation cycle time	1
eta of incoming ranged & tower creep projectile (if any) # melee creeps atking this unit	3
[Shrine only] shrine cooldown	1
vector to me (dx, dy, length)	3
am I attacking this unit? is this unit attacking me? eta projectile from unit to me	3
unit type	1
current animation	1

Per-hero add'l (10 heroes)	25
is currently alive?	1
number of deaths	1
hero currently in sight? time since this hero last seen	2
hero currently teleporting? if so, target coordinates (x, y) time they've been channeling	4
respawn time	1
current gold (allies only)	1
level	1
mana: max, current, & regen	3
health regen rate	1
magic resistance	1
strength, agility, intelligence	3
currently invisible?	1
is using ability?	1
# allied/ enemy creeps/heroes in line btwn me and this hero	4

Per-allied-hero additional (5 allied heroes)	211
Scripted purchasing settings	7
Buyback: has?, cost, cooldown	3
Empty inventory & backpack slots	2
Lane Assignments	3
Flattened nearby terrain: 14x14 grid of passable/impassable?	196
scripted build id next item to purchase	2

Nearby map (8x8)	6
terrain: elevation, passable?	2
allied & enemy creep density	2
area of effect spells in effect.	2
area of effect spells in effect.	2

Previous Sampled Action	310
Offset? (Regular, Caster, Ward)	3x2x9
Unit Target's Embedding	128
Primary Action's Embedding	128

Per-modifier (10 heroes x 10 modifiers & 179 non-heroers x 2 modifiers)	2
remaining duration	1
stack count	1
modifier name	1

Per-item (10 heroes x 16 items)	13
location one-hot (inventory/backpack/stash)	3
charges	1
is on cooldown? cooldown time	2
is disabled by recent swap? item swap cooldown	2
toggled state	1
special Power Treads one-hot (str/agi/int/none)	4
item name	1

Per-ability (10 heroes x 6 abilities)	7
cooldown time	1
in use?	1
castable	1
Level 1/2/3/4 unlocked?	4
ability name	1

Per-pickup (6 pickups)	15
status one-hot (present/not present/unknown)	3
location (x, y)	2
distance from all 10 heroes	10
pickup name	1

Minimap (10 tiles x 10 tiles)	9
fraction of tile visible	1
# allied & enemy creeps	2
# allied & enemy wards	2
# enemy heroes	1
cell (x, y id)	3

Table 4. OpenAI Five Full Observation Space at each time step from (OpenAI et al., 2019). Blue entries are categorical data. Value availabilities vary according to the team, fog of war, etc.

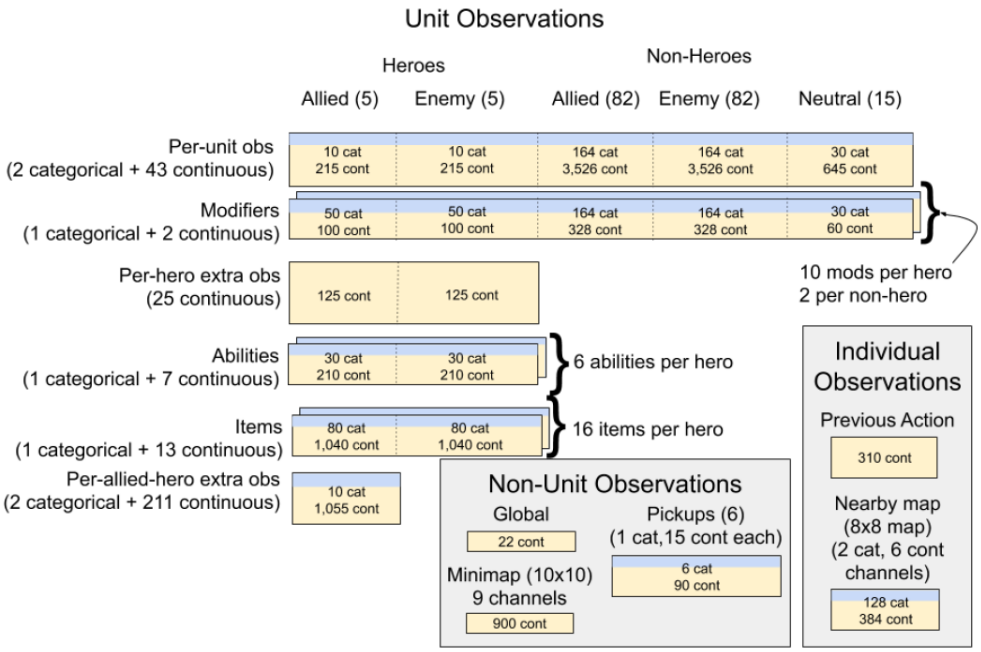

Figure 14. Observation Space Overview from (OpenAI et al., 2019).

The main feature of OpenAI Five's observation is unit-centrality. Depending on what unit is, the amount of detailed properties about the unit varies. Only few observations are not tied to any unit. Even though the number of units changes during the game, there is a policy for observation of a category, e.g., allied non-hero units. If the number of visible units in the category is less than the allocated number, the rest observation values are zero. If more, only the closest units to allied heroes are observed, meaning that there is a priority (OpenAI et al., 2019).

In *Dota 2*, all effective actions are related to own hero. Normally, player uses a mouse and keyboard to click the buttons and the screen to either issue actions to the hero or select unit for observation. Without the peripherals, the action space presented to the artificial agents, is divided into two parts. One part is represented as "primary action", which is controlled by the deep reinforcement learning network. The other part of actions is scripted, which is defined by a rules-based system to handle the decisions. Primary actions are the majority of the actions in a game, which are mostly combinations of high-level command and a target. Typical high-level actions are attacking, using a certain spell, or activating a certain item. For those actions to work, some need a target point on the map or a target unit.

Therefore, a primary action is joined by a set of parameters, which are delay, unit selection and offset. The number of available primary actions varies from time step to time step, so is that of the parameters (OpenAI et al., 2019).

Action Target Type	**Example**	**Parameters**
No Target	Power Treads	Delay
Point Target	Move	Delay, Offset (Caster)
Unit Target	Attack	Delay, Unit Selection (Regular)
Unit Offset Target	Sniper's Shrapnel	Delay, Unit Selection (Regular), Offset (Regular)
Teleport Target	Town Portal Scroll	Delay, Unit Selection (Teleport), Offset (Regular)
Ward Target	Place Observer Ward	Delay, Offset (Ward)

Table 5. Action Target Types from (OpenAI et al., 2019).

(a) **Delay:** An integer from 0 to 3 indicating which frame during the next frameskip (4 frames) to take the action on.

(b) **Unit Selection:** One of the 189 visible units in the observation.

(c) **Offset:** A 2D (x, y) coordinate indicating a spatial offset, used for abilities which target a location on the map.

Figure 15. Action Parameters from (OpenAI et al., 2019).

At the start of OpenAI Five, the scripted actions were more than how there remain at the end. The scripted logic takes in charge of actions which are less frequent during a game but decisive. The full set of remaining scripted actions is (OpenAI et al., 2019):

1. **Ability Builds:** Each hero can learn four spell abilities and upgrade them as the hero level increases. When the hero levels up, a chance of upgrading one ability is granted and a decision should be made. There are schedules for builds and randomly accepted in matches.
2. **Item Purchasing:** The heroes earn gold which can be used for purchasing items. Items are divided into consumables and everything else. The logics for consumable items and other items are different. The schedules for purchasing are also randomly applied for robustness.
3. **Item Swap:** Inventory holds only 6 active items while 3 items in the backpack would be inactive. A better strategy is to keep the most valuable items in the inventory, so they contribute more to the battle.
4. **Courier Control:** A courier is a non-armed item carrier from shop to the hero controlled by the player.

Dota 2 is famous for its immense dynamics. There are over 100 heroes for choosing and each team has 5 unique heroes in a match. There are also hundreds of items to buy. All these result in a super big possibility of this game. However for OpenAI Five, it does not handle all the situations as it would cost tremendous amount of effort to either train the agent or build the interface. Matches with OpenAI Five's participation are played under a set of restrictions, and the restriction changed over time. In 2018, the game is played under a wide set of restrictions, which makes the game quite different from how it is played at high level (See Table 5 for the full list of early restrictions). The restrictions were later lifted to some extent, especially adding more heroes, allowing hero pick-ups. Before facing world champion team, OG, the AI system eventually included 17 heroes, as Lich was removed from the pool because his abilities were changed significantly in *Dota* version 7.20 (OpenAI, 2019).

Game element	Settings used by OpenAI Five
Heroes	Initial: Mirror match of Necrophos, Sniper, Viper, Crystal Maiden, Lich After July 18, 2018: Pool of 18 heroes (Axe, Crystal Maiden, Death Prophet, Earthshaker, Gyrocopter, Lich, Lion, Necrophos, Queen of Pain, Razor, Riki, Shadow Fiend, Slark, Sniper, Sven, Tidehunter, Viper, or Witch Doctor)
Warding	No*
Roshan	No*
Invisibility (consumables and relevant items)	No*
Summons/illusions	No
Shop items	Divine Rapier, Bottle, Quelling Blade*, Boots of Travel*, Tome of Knowledge*, Infused Raindrop* are not available.
Courier	5 invulnerable couriers, no exploiting by scouting or tanking
Scan	No

Table 6. Restrictions used by OpenAI Five. Contents marked with an asterisk are no longer in the restriction list after the benchmark (July 18, 2018)(OpenAI, 2018b, 2018a).

The core of OpenAI Five program is a recurrent neural network acting as a policy function from the history of observations to a probability distribution over actions. The neural network consisted of a single-layer 1024-unit LSTM (OpenAI, 2018a), later extended to 4096-unit (OpenAI, 2019). The game passes the current observation to the network as input and the network samples an action from the output distribution at each timestep. At a high level, the network has 3 parts. The first part processes and pools the observation into a

single vector summarizing the state. The second part is a single-layer large LSTM processor. The final part receives the output of LSTM and produce action using linear projections.

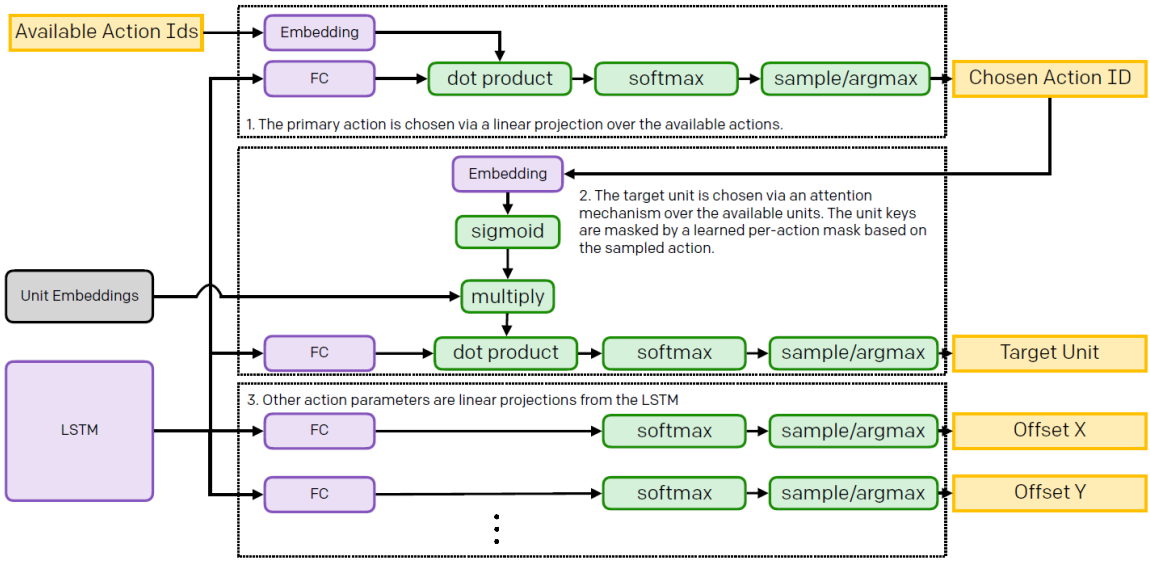

Figure 16. The action generator part of the network from (OpenAI et al., 2019).

Though the ultimate goal is to win the game, the single winning reward would be very inefficient in training. Therefore, OpenAI Five uses a reward function which includes additional signals such as characters dying, collecting resources, etc., because most of the time, they do not conflict with the final goal and represent advantage in the game. These additional signals became important for successful training (OpenAI et al., 2019).

The policy is trained using Proximal Policy Optimization, a new class of reinforcement learning algorithms by OpenAI (Schulman et al., 2017). The optimization algorithm uses Generalized Advantage Estimation to stabilize and accelerate training (Schulman et al., 2015). The policy is trained using collected self-play experience in a system including 4 types of machines (OpenAI et al., 2019):

1. Forward Pass GPUs hold the recurrent neural network and open its port for receiving game observation and producing action output.
2. Rollout workers run multiple *Dota 2* games on CPUs and communicate with Forward Pass GPUs, which sample actions.

3. Optimizers run gradient updates based on the game results received from rollout workers. So new parameters for the neural network are generated.
4. Controller works as a storage, receiving the updated parameters from the optimizers. It also provides the new parameters with the Forward Pass CPUs when they pull the latest parameter version.

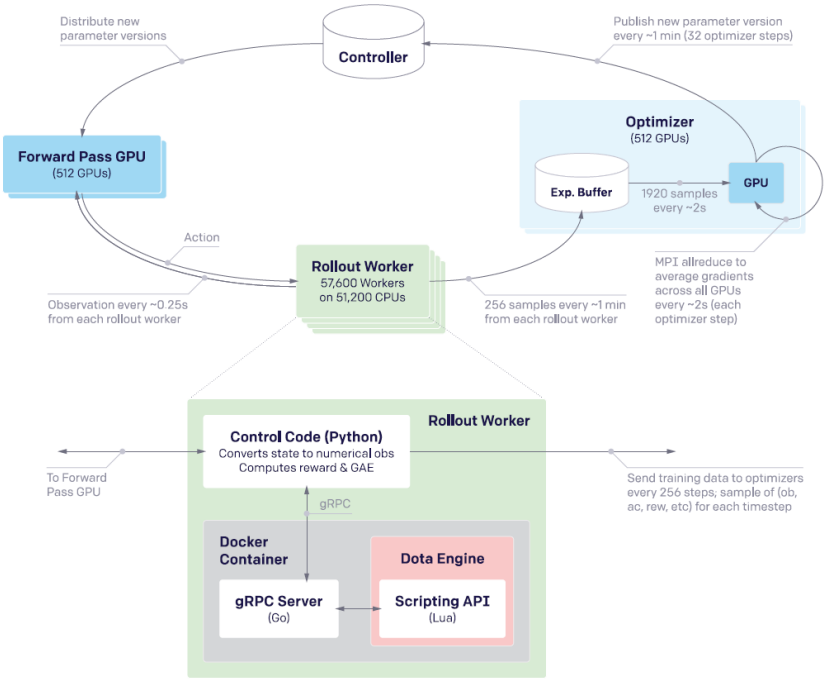

Figure 17. Open AI Five training system overview from (OpenAI et al., 2019).

The training environment was not changeless during the project. First, the changes to the training process (reward structure, observations, etc.), including the architecture of the policy neural network were implemented. Second, since the game played by the agents was a modified, customized version where some mechanism was removed, the researchers added complexity back piece by piece over time, towards perfection. Third, Valve occasionally publishes new *Dota 2* versions which change the game details, for example, heroes, items, maps. For compatibility with human players, the agent is adjusted to suit the up-to-date patch. The OpenAI Five team invented “surgery” and performed it to the old model to obtain a new model compatible with the new environment. This contains Net2Net operations, that is transferring knowledge contained in one neural network to

another neural network (T. Chen et al., 2015). By doing “surgery”, the new model inherits the high skill level instead of being trained from scratch, though complete restart of training is a safe and common practice in this kind of situation (OpenAI et al., 2019).

Compared to the approach of mouse, keyboard and monitor, OpenAI Five plays *Dota 2* in a way that is quite unusual. The use of observation array does not conform to the standard observation of human. Despite agents do not see some part of game like shrapnel zones (areas where some effect is applied on units)(OpenAI, 2018a), the unit-centered observation removes the necessity to move camera around the map to get a view of the situation. In other words, the agents are given most legitimate valuable information about the players from the game state, gaining an advantage. Human players receive a voice announcement and short log when a key event happens during the game, for example, when a hero is killed. Any other players except the ones having a battle are able to choose to either intervene in the battle against the enemies or stay away and keep accumulate power only before the announcement is made. Therefore, it is very common for human players to ask teammates for assist in voice chat. This channel of communication fills the gap of partial observation, but it is based on trust among teammates. Neither bad teammate nor zero attention on teammates help win the game. For OpenAI Five, there is no such a communication channel. They are five autonomous agents, made from a copy of itself (OpenAI, 2019). Though the five agents run five neural networks, they have almost same observations and share similar processing units, so they can easily come to the same conclusion, being ‘selfless’.

The actions by OpenAI Five agent are also a little different than human’s possible actions. Just like the observation, the action implementation is also a unit-centered approach. Many abilities and items in game have a point target, requiring selecting a point on a map. The interface simplified this selection by binding it to a unit with an offset (Unit Offset Target). This transformation works most of the time as most target points are near a unit, but has a potential problem with some minor abilities, for example, scouting ability whose target is a point in fog of war. The limited hero pool avoided this problem but when the agent is more generalized, the problem would possibly occur. Moreover, some actions are scripted, making the move by the agents more predictable, especially item purchasing.

Learning to play *Dota 2* is not easy for human. So is that for agents. The restrictions made the game easier, but there is still a lot to remember for players. *Dota 2* is famous for its numerous heroes and items, and it is a big playground for exploration. Therefore, it is very interesting to see how humans and agents learn all these heroes and items. Humans learn those in many ways like reading the description, trying out in game, watching how others use them, listening to others' recommendation. People generalize the use after getting familiar with some of them. OpenAI Five agents, however, purely rely on self-play to gradually acquire knowledge about them. The training had very little slowdown when going from less to more regarding those heroes (OpenAI, 2019). Another one important point in learning is the cooperation. A variable called Team Spirit recalculates the training reward for the individual agents. If team spirit is 0, then every hero receives reward for themselves. If team spirit is 1, then every reward is split equally among all five agents (OpenAI et al., 2019). Team-based game can be frustrating due to an individual's selfish play, especially when gameplay involves asymmetrical character design. There is such a credit assignment problem so professional teams playing *Dota 2* have another role of coach in addition to the players.

3.3 Comparison of agents

The text above listed two esports game playing agents created by using state-of-the-art technologies. These agents showed many similarities, but they are not exactly the same. This part aims to point out the necessary components in the projects to find the minimal requirement for creating champion level artificial players in esports.

3.3.1 Interface for agent's observation and action

To allow an agent to play as a player, a programming interface supporting observation and action is necessary. Both *StarCraft II* and *Dota 2* have an official interface for agent to participate in the games. Interestingly, none of AlphaStar and OpenAI Five used the native graphical interface for observation. This alternative simplifies the observation, but there are reasons beyond that. First, games have optional graphical settings, which could make the

game look different. It also takes up plenty of unnecessary computational power to render on screen. Second, by skipping the graphics, the game could be run at a higher speed than normal playing. More frames can be simulated per second in a customized setting of game, which could accelerate the training process.

However, due to the essential difference between two games, two customized approaches are applied for observation. AlphaStar preserved the notion of screen but added some semantic supplement layers. OpenAI Five created an array gathering most available information. Since both agents achieved high skill level recognized by professional players, both approaches are relatively fair and demonstrated to be successful. However, the setting could have an indirect influence on the agent's learning. A screen operates at a lower interaction level (see Table 7, level 2) and requires the player to move it around to see everything. One hypothesis could be that, as long as the agent is provided the same level of information as human receives, plus the information is converted to another semantic equivalent form, the agent's observation is enough to induce intelligence in learning. The fact showed that the agent could be even more robust than that: the OpenAI Five agent's observation has a missing piece of visual effect that is related to an areal damage, but the agent learned to walk out of the damage zone (OpenAI, 2018a). Because of neural network's "black box" nature, the internal logic and workings are hidden to users, thus it could not be explained explicitly how the agent learned to do so (Carvalho et al., 2019). This shows the sign of the existence of machine inference, as clues could be gathered from the relevant observations such as heroes and their abilities.

The actions are not issued with mouse and keyboard either. The 'cockpit' design is game oriented. The action interface needs to connect the agent to the game and map the output to the valid actions. The way of action is affected by observation: The unit-centered observation caused the action to be some unit-centered (see Table 7, level 3). Though camera-based observation did not restrict the target clicking, action in PySC2 is done with prior knowledge of abilities (see Table 7, level 4). There is another one important game type in esports not sampled here. First-person 3D vision creates a totally new action space, but it belongs to shooter (FPS) games in esports. Most of these games contain an action system using the mouse to rotate the camera. However, the move of mouse is considered as

continuous analog signal for human (though actually not). Either any imitation does not look like human behavior, or it could easily outperform human with super reaction and precision. This action interface issue makes FPS harder for creating acceptable IA.

3.3.2 Learning system

Coincidentally, both AlphaStar and OpenAI Five contain LSTM units at the core of their neural networks, and both of them use 'self-play' for training the network. In terms of game design, deep reinforcement learning is not a prerequisite, but in order to allow agent to achieve high skill in esports while minimizing the additional interactive advantage, there is no other choice for managing the mapping from the input to the output. Neural network structures are different, but still, more details in both designs that are not identical.

AlphaStar has an exclusive bootstrap period when agents are trained in supervised learning of replays (Vinyals, Babuschkin, Czarnecki, et al., 2019). This process does not exist in OpenAI Five's document. Even though supervised learning helps the agent to start playing the game, it does not make the agents reach the skill level of the winner in the replay. The agents reach professional level only after trial-and-error in the games. The reason including this process is obvious, as the default random agent in PySC2 almost never generates any meaningful action, showing that agents need an initialization.

OpenAI Five, has a reward system which gives clearer instruction of objectives than that of AlphaStar. AlphaStar's reward is simply dependent on the outcome of game in the reinforcement learning process, win, tie or lose. OpenAI Five learning reward copies some concepts from *Dota 2*, for example, earning gold, destroying towers and barracks, and is normalized by weights (OpenAI et al., 2019). These events happen regularly during a game and are considered as important sub-goals. Thus, the reward could give dynamic evaluation of agent's performance in a short period of game. There is a contradictory reward in the list. Destroying barracks has a reward of 6, while winning the game, destroying the ancient building, only has a reward of 5. Though from game experience, it could be understood that the barracks can hardly be skipped before taking down the main ancient

building due to the buildings' positions and importance of barracks, this may lead to a potential weak point in last moments of few fierce battles.

In the reinforcement learning of those agents, the concept "self-play" is not fully loyal to its literal meaning of playing with itself. In fact, the so called "self-play" involves multiple agents with possibly different neural network parameters. The agents are copied in training, matched up against both newer agents and older agents. However, not all agents eventually become the best one and many of them developed a playstyle that is not optimized (see Figure 11). Some agents keep their playstyle to the end of training and fail to achieve the highest level. The playstyle can be considered as the ability that arises from the pursuit of the goal of winning, thus maximizing the reward drives that behavior (Silver et al., 2021). But at the same time, uneven performances of agents show that it is not guaranteed that all agents find the best answer to the problem. The league of "self-play" has become a selective process, hinting that learning does not start with a tabula rasa, a blank state, but with a set of representational commitments (Roitblat, 2021).

The "surgery" tools have been utilized for handling changes to the environment, model architecture, observation space, and action space in OpenAI Five. On the contrary, AlphaStar did not have any major change of those during the project. This is a meaningful step towards generality. General intelligence may be defined as the ability to flexibly achieve a variety of goals in different contexts (Roitblat, 2021). Neural networks in reinforcement learning could be complicated, and it could be very costly to train them. Unfortunately, the networks are bound with specific tasks and their customized input and output. Currently, it is still impossible for the network itself to realize the changes. Manual intervention permits features to be added in the iteration, instead of everything at the beginning, thus making the process look more feasible. One alternative is to build the agent on a more primitive interaction level (in Table 7, lower number of interaction level), so that the agent would at the beginning learn how to use more general tool, just like using a pen instead of typing with keyboard. However, this would create a longer ability chain (Figure 12), certainly resulting in slower training, which could be perhaps unacceptable.

Interaction level	Object	Type
1	peripherals (mouse keyboard)	physical
2	user interface (UI) elements (cursor, buttons)	on screen surface
3	game objects under control (units, items, etc.)	selective
4	things in game (terrain, space, units, spells...)	gameplay
5 (optional)	things in game	gameplay

Table 7. Interactive chain. The levels are in fact tree structure, and one chain describes an action by human player, and is a linked nodes from the level 1 to highest level, which could range from 2 to more than 5, depending on game design. For example, a camera movement can be a level 2 action, involving mouse and UI element. Ranged attack and many point target spell are level 6 chains because it requires level 3,4,5,6 objects to be respectively own unit, the concept of attack or spell, space, enemy unit. In this case, lower levels carry the contextual data.

3.4 Generality problems

There are two types of generality problems. The first question is how general the training process is. The second is how general the agent is. These questions will be examined when some scenarios are provided later. These questions together are equivalent to RQ1, answering the question of generality.

3.4.1 Patches

Patches are updates to games released by game officials regularly. Patches may include game balance updates, bug fixes, improvements, and so on. For modern video game, patches are fetched via the internet from the game servers. The updates are downloaded before the game runs and there is now game version check between the online server and player’s client for

preventing clients with mismatching versions from entering the game. Mostly, the game developers are in charge of releasing new game patches. So are the esports games. Game balance is maintained by them to keep a fair environment for human players selecting different characters, factions to play and compete. Patches come usually together with documents explaining what kinds of changes they have done to the game, just like how end-user documents are provided in software engineering convention.

However, game agents do not read documents. They are created using machine learning technologies. This raises the question whether those agents could keep playing the game after the patches change the game. According to the introduction to the structure of agents, they are not able to realize that the game unit has changed. Thus, the change could possibly cause two problems to the agent:

1. Pure numerical change (e.g., skill damage, hit points) does not change the interface, so the agent could still invoke the function to issue that action. However, it would be confusing to one agent that the number has changed so the unit may not be best utilized. However, since agents are copied and they do not have the same neural network parameters in the end, it is possible that some agent originally weaker becomes stronger after such patch due to lucky adaptation during training selection.
2. Redesign could change the action thoroughly. It could give the ability a new name (id) or a new target type or a new effect. So, the original action call would lead to invalid action. This would disable the action without notifying the agent. From the random agent of PySC2, it could be known that game could receive invalid action and generates a warning but execute no action. This is similar to the exception handling in programming, evading program crash. If the action being changed was vital before patch, this would bring down the agent's skill level pretty much.

Since pure numerical change does not change the interface, the agent could be trained again. The training can once again adjust the parameters of the neural network, make the agent adapt to the new settings. Redesign tends to be more catastrophic. To cope with

that, OpenAI team's surgery is a successful example of handling changes. A drawback is that surgery requires involvement in training process and is not an attribute of agent.

3.4.2 Custom arcade games

Arcade games are some community-created games included in the arcade mode of those games. They are not considered as official esports but some of them have the potential to become a new genre of esports, e.g., the original *Dota*. Both standard melee and arcade games rely on a dataset, which is called mod. But arcade games could have a map and some other components. The gameplay of some arcade games can be quite far from the standard games indicated by the names such as *StarCraft* and *Dota*, however the content creators still need the game engine to host their games. The creativity is enabled by the development tools of games, and the distribution of the tools does not follow a single standard. Some games themselves contain editors which could be access in game, for example, *Age of Empires*. Some games are bundled with additional editor program during retail, for example, Blizzard's games. Some provide source code of the game engine, e.g., id Software. Some others require external editor and provide tutorials like Valve. There are also opposites who are very conservative about their game. Fortunately, *Dota 2* and *StarCraft II* allow flexible extension under the restriction of end-user license agreement (EULA).

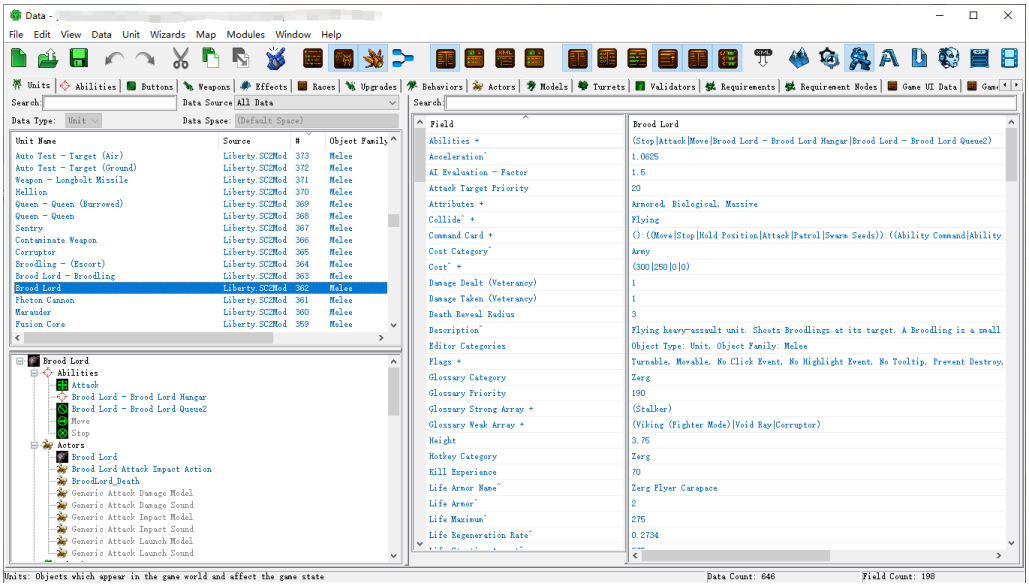

Figure 18. Data view of StarCraft II editor, as known as Galaxy Editor.

Arcade games can be considered as a more intensive redesign to the game. Some of them could be almost brand-new games to players without competitive element. So it is reasonable that the agents do not support their actions at all, based on how actions are defined. This question expands the topic to the larger space of more games, which is quite challenging. However, cross game support existed in a simpler level. One deep reinforcement learning agent could play multiple Atari game, using screen as input (Hausknecht et al., 2014; Mnih et al., 2013). The complexity of those old games is not on the same level as today's ones. The adversarial attribute of the games makes porting even less possible to adapt.

3.4.3 Training cost

Deep learning technology is notorious for its high cost of computational resources. Suggest that one more competitive game wants a similar agent system, the question is how much time and resources it would cost.

AlphaStar run the training in more than 40 continuous days. Every training agent runs 16000 concurrent matches and 16 actor tasks (each using a tensor processing unit (TPU) v3 device with eight TPU cores) to perform inference. The game instances cost equivalent to 150 processors with 28 physical cores each. Actors send sequences of observations, actions and rewards to a central 128-core TPU learner worker. The research group instantiates 12 separate copies of this actor-learner setup (Vinyals, Babuschkin, Czarnecki, et al., 2019).

For OpenAI Five, from Figure 17 above, it could be seen that tens of thousands of CPUs and thousands of GPUs are used in the training system. Moreover, OpenAI Five is a long project that last 1 year with many times of surgeries and rerun. Roughly 30% of computation is optimization GPU machine. The same number goes to forward passes for the rollout workers and actual rollouts CPUs running the self-play games. Other infrastructure costs about 10% (OpenAI et al., 2019).

This information shows that all the computation involved in the project would require research group with financial background. Not the only agent that shows up in the match costs resource. The high demand of computation requires the training to be

implemented on computing service provider, such as a cloud platform. However, one agent in the end could run on a high-end consumer GPU, which only does network forwarding for the agent. Compared to the higher cost of creating the agent, the deployment of the agent is very cheap.

3.5 Summary

This chapter reviews the technologies that enabled esports games playing agent. Traditional rule-based agent is widely used by game developers. On the contrary, scientists try to achieve high level performance using deep reinforcement learning technology. Two outstanding projects that created professional level agents are AlphaStar and OpenAI Five. Their implementations are briefly introduced. The similarities and differences are discussed after the introduction, obtaining some basic requirements of building such agents. The next step is setting three scenarios that concerning the agent and the training process. These questions are sub-questions of RQ1, raised by the common events and phenomena that happen to the games. The next paragraph summarizes the answer to RQ1:

The IA itself created for certain kind of esport game has only limited generality that only allows it to handle minor changes to the game. It can be enhanced by manual intervention to solve some observation and actions problems caused by changes. Those special agents do not support other community-made arcade game using the game engine. For the reinforcement learning of them, many patterns are alike, including recurrent neural network, league settings. But the interface cannot be reused because it is shaped by the game and needs to be set at a semantic level. The creation of the agents for certain game is beyond personal level in terms of cost, but suitable for a research group.

Therefore, the power of deep reinforcement learning is demonstrated, but it still lacks the ability to be highly general and flexible. Intelligence about the game can be acquired by agent accompanied by some assist from human.

4 Impact to community and feedback

Inventions change the world. These new technologies are revolutionary breakthrough in esports. So, the consequences can be studied after the release conference of their agents. In esports, there are many professional players fighting for the victory in tournaments and many join the professional scene while some leave at a certain point of their competitive career. However, there has never been a non-human player participating in any major esports team, and it is not likely that any computer agent would participate. These days, players can easily get to know that there exist such agents via game community, social platforms. They could also easily provide feedback and comments. Moreover, this could have an impact on players' opinions about the game, leading to some unusual outcome beyond playing the game.

4.1 Conference video feedback

OpenAI Five's video and AlphaStar's video attract plenty of viewers on Youtube and many of them left a comment. Comments could be liked and the number of likes could show how much people agree with a serious comment, or how interesting and hilarious the comment is. Comments with would be sampled using a method (see Appendix). Explanation and analysis would be given according to the comments.

4.1.1 OpenAI Five

The first video (https://www.youtube.com/watch?v=l92J1UvHf6M), titled OpenAI + Dota 2, is the oldest video about OpenAI Five and attracted more than 1 million views. It introduces *Dota 2* as the background, and the goal of building general artificial intelligence. It is uploaded on 12 Aug 2017, around the time the research project started. The most-liked comment in the video has 1.1K likes:

"The creep block is honestly beyond human comprehension.

The zoning is crazy good.

The cs is just insane.

But can it type "?" in all chat?"

The comment is a joke. The first three lines review the agent's excellent performance regarding game techniques. They appreciate how good the agent could handle the game, though the video only shows very little part of the agent's play. The last line is a contrast. It talks about the bad manner in game. Typing question marks in game when the player gets a score or escapes an almost successful assault is generally considered as a bad manner in *Dota 2*, questioning the motivation of the opponents' actions. Normally, one encounters such bad manners from random players, but this assumes OpenAI Five is surely not going to support this functionality. In conclusion, this comment mentioned the emotional toxicity of *Dota 2* with some humor.

The second liked comment in the same video is by another user. The content is:

I'm sure this is VAC ban worthy

Normally, players may be banned due to an inappropriate behavior by Valve Anti-Cheat (VAC) system[15]. It does not make sense that an agent of research playing this game should be banned. However, if the condition is that one player uses it to substitute itself to be good or the agents create an unfair advantage in normal games, the behavior can be considered as abusing, thus leading to a ban.

Another important Open AI Five video is released when they have created the standard 5 vs 5 agents (https://www.youtube.com/watch?v=eHipy_j29Xw). The top comment's opinion is very clear here. It attracted 1.8K likes and reads:

"WE ARE DOOMED"

The user is a *Dota 2* video maker with plenty of subscribers. The comment describes that human players would never win against best AI players. Despite this negative view, OpenAI Five had not played the show match against *Dota 2* champions at this moment and did not play the full game of *Dota 2*. It had not played the full version either eventually.

[15] This wiki page explains more on how players may be banned from VAC: https://dota2.fandom.com/wiki/Ban

4.1.2 AlphaStar

AlphaStar is exhibited to the world in the live stream DeepMind StarCraft II Demonstration (https://www.youtube.com/watch?v=cUTMhmVh1qs) on 24 Jan 2019. The live stream briefly introduced the technologies, reviewed the test matches, and had an exhibition live match of Mana and AlphaStar. Surprisingly the most liked comment came from the official account of League of Legends, the popular MOBA game. The comment has 4.3K likes:

"let's see your fancy robot try to win a 1v1 against our boy imaqtpie"

Here, imaqtpie is an American League of Legends player. This comment invites such research straightforwardly. In last chapter it is demonstrated that creating such agents is not a small amount of workload.

The second high like numbers (1.6K) belong to two comments. One is a collection of timestamps of the parts in the video. So, it is ignored for containing no valuable information. The other one is:

"Props to deepmind for enduring 200 years of pure protoss vs protoss."

For normal *StarCraft II* games, playing mirroring race can be thought as tedious and monotonous due to lack of different race features. For unknown reason, the live stream did not have any other matchup than protoss vs protoss, and only one map, even though from the published article it can be seen that all three races are tested (Vinyals, Babuschkin, Czarnecki, et al., 2019). Agent training time can be scaled so it could be impossibly long compared to human experience. Even in esports, games are built partially for entertainment, not for pure competition fairness. Entertainment remains a big reason for people to play any game.

The third top comment has 1.3K likes. Its content is:

I've been oversaturating my mineral line for the past 6 years and I've never been more validated.

Oversaturating mineral line is a technique in *StarCraft II*. Mineral line is the area that contains 8 mineral fields and the convention is to put 16 workers to gather the resources.

However, the video showed that AlphaStar put more than 16 workers on one mineral line, which is called "oversaturating". Oversaturating causes a decline in income/worker ratio due to the occupation of a mineral field[16]. Oversaturating mineral line means some more reserved workers with less efficiency but is more resistant to loss of workers caused by the opponent because backups would fill the loss of mining speed, resulting in less loss of mineral overtime. In conclusion, it is a proof of the saying "time is money". The commentor feels recognized due to the similar behavior of AlphaStar in the video.

4.2 Deployment

Both AlphaStar and OpenAI Five were deployed to the official game platform during a period of time for the public. AlphaStar was on Blizzard's matchmaking system for evaluation, when the game was on patch 4.9.3 (Vinyals, Babuschkin, Czarnecki, et al., 2019). OpenAI opened OpenAI Five Arena during April 18-21, 2019, and played total 7257 games, and winning 99.4% (OpenAI et al., 2019).

Despite there is worry mentioned above that such technology would create a disturbance to the game environment due to abuse, the deployment of their agents did not last long. The agents' neural networks are not made available to the public. OpenAI Five attended The International (TI), *Dota 2*'s official annual tournament, and no decrease of the prize pool is observed after that (OpenAI, 2019; *Top Games Awarding Prize Money*, 2020). *StarCraft II* had a decline in the annual tournaments' prize money, but in the last years (2019, 2020, 2021), game developer Blizzard had big personnel changes, and faced plenty of controversies, causing huge damage to the games ("Blizzard Entertainment," 2022). These incidents are hardly relevant to the AI developed for *StarCraft II* and had a major effect on the esports.

[16] See this wiki page for more detail. Mining Minerals: https://liquipedia.net/starcraft2/Mining_Minerals

4.3 Successors

Since PySC2 is made publicly available on coding platform Github, there have emerged some successors to AlphaStar project. Inspir.ai has also made *StarCraft II* playing agent (Inspir.ai, 2021)[17]. DI-Star is born under the supervision of retied zerg player from the Chinese AI company SenseTime[18]. DI-Star is a reimplementation of AlphaStar and it focuses on zerg vs zerg matchup, providing a set of pre-trained models that could run on personal computers (Contributors, 2021).

4.4 Summary

This chapter explores the consequences of the invention of such agent players from player community's perspective, in order to answer RQ2. The main outcome of those projects to the gaming community is the announcements by the researchers that make people realize that such agents exist now. They did not bring big changes to the esports environment. Apart from the time-limited deployment, those agents did not stay in touch with players for long. According to the feedback, attitudes towards the agents vary. There is no unified welcome or opposition among players. Some comments joke around and do not even focus on the skill of agents, which is the researchers' main objective. In conclusion, the impact to gamers is not very significant. It only left some new ways to think about the game and provided an approach to create new type of bots.

[17] For more detail check video (in Chinese): https://www.bilibili.com/video/BV1HA411i7pD

[18] Link to the video by Harstem about DI-Star: https://www.youtube.com/watch?v=fvQF-24IpXs

5 Cooperation opportunity

Competitive sports are only winning and losing. What is not beneficial to one player hurts the player in the end. The IAs, born to master the game at the highest level, must fight against one another and human to achieve the designer's goal, which sounds threatening and causes the players' worries mentioned in the last chapter. Though it does not rise to the level of killer robots, it lets people think about the question: How can the gaming IAs do good things on the human's side, or cooperate with human and assist people? Some examples have been given in the show matches.

5.1 Examples

When AlphaGo first came out, as the first modern reinforcement learning game playing agent, it already raised a concern about the humanity vs machine. Then, in The Future of Go Summit, the format of Pair Go showed a feasible way to combine human intelligence with machine intelligence. Top Chinese professionals Gu Li and Lian Xiao are paired with their own AlphaGo teammate, and both human and machines make decisions on the next moves. Often, AlphaGo and the human player think alike about the move, but sometimes they can surprise each other (DeepMind, 2017c). Go is a turn-based board game, and for every move there is time to think and examine. Therefore, the cooperation could be implemented in a simple way. The machine suggests the move every time and the human decides whether to adopt the move or choose an alternative. Though Go is not an esport, some esports are slow paced or de facto ported board game.

In OpenAI Five Finals, a special match was hosted including human and agents on both sides. OpenAI Five's ability to play together with human is tested in this cooperative mode. One notable point is that OpenAI Five is trained to play on the side with full OpenAI Five agents, but it is general enough to participate in a team with some human players in the team (OpenAI, 2019). The testers had been chatting during the match without the team barrier and it was a fun experience. The bot teammates adapted themselves well in the game environment and performed their own duties, though they did not response to the human's ping message, and they did few weird actions. In team-based esports, a method of grouping

human players and AI agents is to fill the remaining slots of players in the team with agent players.

5.2 Obstacle to cooperation

Soon after the invention of computers, people started to take advantage of them and program them to make benefits. Back in 1960s, a branch of AI applications called expert systems were developed (Shu-Hsien Liao, 2005), and were considered as one of the first successful forms of AI software. However, the main idea of expert systems is to store human knowledge in the computer programs so that the computer could make inferences and answer specific questions. Game playing agents are, to some extent, expert systems in the domain of video game. Nevertheless, neither old knowledge-based systems nor modern game agents could be in communication with human just like how people can exchange information and interact. These machines do not respond emotionally and game playing agents keep the trick to victory as a secret.

Deep neural network is the major contributor to the high performance in game playing. However deep neural network has the limitation of lack of transparency (Carvalho et al., 2019). It is impossible to extract knowledge from the networks' parameters to the form of rule described in language. This makes game agents not only uninterpretable but also uncontrollable. Agents take actions fully autonomously towards their reward, planning the moves according to the observable game state. There is no external port on the agent for issuing an order to change its playstyle to a safer one or a more aggressive one. The agent learned the game through its experience, and that is the only thing which could shape its playstyle.

Therefore, to have cooperation with the agent players, the environment is the basic key, not the IAs themselves. *StarCraft II*'s official competitive format is 1 vs 1, leaving no space for the IA to help any teammate, though it is possible to play the game in team modes. In current esports, team is a necessary element in enabling in-game cooperation, and that is the same for IA. Games may define interaction with teammates, for example, the items that could be used for teammates in *Dota 2*. After the game defined those cooperative actions,

and the agent output supported them, it is then possible to see them triggered by the agent's judgement in some situations.

5.3 Indirect help

Game replays and spectating matches are a useful resource for both enjoying the game and learning the game from others. In training of agents, not all researcher groups applied the same policy on using game replays. AlphaStar utilized more than thousands of replays, while OpenAI Five did not use any replays for learning at all. Replay could help make a clearer thinking after the match. Learning from replays requires one's initiation, so it is not the agent assisting the player directly in game. However, this can be still inspiring, as the professional player reported the agent's play style unimaginably unusual (Vinyals, Babuschkin, Chung, et al., 2019). The agents can act as a strategy explorer, substituting human players, and generate replays for both people and agents to watch.

Practice makes perfect. Playing against an opponent of similar skill level is a universal way to improve in many kinds of esports. Now, the option of opponent is joined by IA. After training, mature IA is capable of responding to most kinds of game situations with some human-like actions. Compared to the built-in rule-based script used by games, it could avoid the scripts' flaws about some situations that human players may intentionally create. IA by reinforcement training could get non-allergic to some anti-computer tactics unlike script bot. Therefore, one IA is closer to a real player in performance, and represents a more stable skill level. IAs become very good training opponents with perfect manner, never talking rudely in any game. It is reported that some professionals used the RL agents for training, proving their value as opponents to human (OpenAI, 2018a).

5.4 Summary

This chapter aims at escaping from the pressure caused the IA's gaming skill, listing some of the methods to take advantage of the IAs. Cooperation between human and IA is possible, but there are obstacles preventing IA from co-working with human in esports, including lack of communication, agent's autonomy and environment settings. Despite having these

problems, IAs can still be useful to human as explorers or opponents. That is the answer to RQ3 about cooperation between AI and human players.

6 Discussion

6.1 Dummy System or Smart Core

The thesis mainly pays attention to the novation of new fully autonomous IAs in esports, which could be named "Dummy System"[19]. As an alternative, AI has another form to participate in video games, called "Smart Core"[20]. Smart Core, here in definition, is an assistant ability to the players enabled by external software running with the game program at the same time, as known as "hacking". Hacking does not take over all the control of the player, just assisting at specific time to gain an advantage, triggered by some actions in game. First person shooter games are the main victims of hacking because mostly perfect aiming action requires super-human reaction. Normally, hacking is detected by running a guarding program that monitors the process list of the computer. This traditional signature-based detection is challenged by a new method using computer vision. It uses external video capture to find human-shaped enemies in the image, and calculates the mouse movement needed to aim at them, sending the result to an input-passthrough device (Orland, 2021). The toolchain completely evades the traditional detection method, and it is hard to differentiate between the external emulated inputs and the legitimate pro-gamers' actions. Hacking can be catastrophic to game environment and this certainly makes it worse. Although hackers must have an external input-passthrough device, which is not widespread among gamers, it is only a matter of time that more hackers possess those devices. This is a mini version of failure case in AI ethics. Computer vision technology and the tools themselves do not have tendency to break the rules, but eventually they are used by evil minds for their selfishness.

[19] Dummy System is, in anime Neon Genesis Evangelion, an operating device acts as an artificial pilot, locking out the human pilot's control.

[20] Smart Core is the core ability for a titan in FPS game Titanfall 2. When activated, it provides unlimited ammo, and enables auto aiming, locking to the targets.

6.2 Generality on a higher level

Building game playing IA is treated as a stepstone in the path to artificial general intelligence (AGI). Despite some outstanding achievements in contemporary esports IA development, some still argue that these solutions are not general enough. The challenge of games, computer vision, pure language processing are the benchmarks on narrow tasks, which on the contrary shows that these activities do not require general intelligence at all (Shevlin et al., 2019). DeepMind proposed that reward can be sufficient for inducing all kinds of intelligence (Silver et al., 2021), but as long as the dependency on humans to structure the problem is not removed, some do not recognize this hypothesis (Roitblat, 2021). At the moment, IAs playing *StarCraft* and *Dota 2* received far more human's involvement than that could be described as almost independent. AGI is still an unreachable conceptual perfection that represents all of human's abilities.

A similar concept in computer science is virtualization. Virtualization creates some IT services with the actual resource for a better management or without the resource at all. Memory virtualization allocates contiguous memory to apps but on hardware memory it is not contiguous. Virtual machine runs like a real computer but it is simulated by software fully. Virtual reality creates a world in the machine and gives people experience by taking over the vision. The ultimate goal of AGI could be thought as "creating virtual intelligence" without human in the software. Creating virtual hardware in computer world is much easier as the real hardware is designed and known by human. As neuroscience has not completely discovered the working principle of brain, replicating a human brain functionality is not feasible yet. However, the achievement of making such IAs is another story. Artificial neural networks have been proven to be a tool showing stable intelligent performance, without biological infrastructure. This diversion from studying "brain virtualization" emphasizes more on application. It could not be yet determined which path leads to mature AGI and it could show in the future whether biological structure is unique in intelligence.

7 Conclusion

The thesis studied the new esports intelligent agents by using comparative analysis and case studies to two main agent systems. To answer the question of agent generality, impact to the game community and possible cooperation between AI and human, the thesis examined the agents' design, conference show matches, feedback to the showcases and some consequences. The findings are as follows: The training procedure and the agent only have limited generality. Training requires some human guidance for adaptation and the agent handles only specific game version. Impact from the new agents is limited. They did not revolutionize the game community, but just provided a new way of thinking and opened a new field of creating bots. Intelligent agents can play with human as a way of cooperation, but they are less cooperative than human. Apart from team-based in-game cooperation, they can inspire spectators and are good opponents for solo training. These findings are not only an evaluation of generality of modern reinforcement learning, but also a review of whether and how the agents helped people in game community.

A limitation in this thesis is that it did not try to reimplement the training procedure because of the cost and the difficulty. Some successors managed to create newer agents but the result is mostly as predicted. Therefore, for future research, creating more general solutions operating on lower level while keeping the complexity of game would be a meaningful task. Human-agent interaction is another beneficial topic, which is not covered in this thesis.

Bibliography

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Billard, A., & Kragic, D. (2019). Trends and challenges in robot manipulation. *Science*, *364*(6446), eaat8414. https://doi.org/10.1126/science.aat8414

Blizzard Entertainment. (2022). In *Wikipedia*. https://en.wikipedia.org/wiki/Blizzard_Entertainment

Bryson, A. E., & Ho, Y.-C. (2018). *Applied optimal control: optimization, estimation, and control*. Routledge.

Burmeister, J., & Wiles, J. (1995). The challenge of Go as a domain for AI research: a comparison between Go and chess. *Proceedings of Third Australian and New Zealand Conference on Intelligent Information Systems. ANZIIS-95*, 181–186. https://doi.org/10.1109/ANZIIS.1995.705737

Buro, M., & Churchill, D. (2012). Real-Time Strategy Game Competitions. *AI Magazine*, *33*(3), 106. https://doi.org/10.1609/aimag.v33i3.2419

BWAPI. (2022). *The Brood War API*. https://bwapi.github.io/

Campbell, M., Hoane, A. J., & Hsu, F. (2002). Deep Blue. *Artificial Intelligence*, *134*(1), 57–83. https://doi.org/https://doi.org/10.1016/S0004-3702(01)00129-1

Carvalho, D. V, Pereira, E. M., & Cardoso, J. S. (2019). Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics*, *8*(8). https://doi.org/10.3390/electronics8080832

Cavazza, M., Bandi, S., & Palmer, I. (1999). Situated AI in Video Games: Integrating NLP, Path Planning, and 3D Animation. *AAAI 1999 Spring Symposium on Artificial Intelligence and Computer Games*, 6–12.

Chen, T., Goodfellow, I., & Shlens, J. (2015). *Net2Net: Accelerating Learning via Knowledge Transfer*. arXiv. https://doi.org/10.48550/ARXIV.1511.05641

Chen, X.-W., & Lin, X. (2014). Big Data Deep Learning: Challenges and Perspectives. *IEEE Access*, *2*, 514–525. https://doi.org/10.1109/ACCESS.2014.2325029

Churchill, D. (2022). *AIIDE StarCraft AI Competition*. https://www.cs.mun.ca/~dchurchill/starcraftaicomp/index.shtml

Collier, D. (1993). The comparative method. *Political Science: The State of Discipline II, Ada W. Finifter, Ed., American Political Science Association*.

Contributors, D. (2021). *DI-star: An Open-sourse Reinforcement Learning Framework for StarCraftII*. GitHub.

Deacon, T. W. (1998). *The symbolic species: The co-evolution of language and the brain* (Issue 202). WW Norton \& Company.

DeepMind. (2017a). *AlphaGo*. https://deepmind.com/research/case-studies/alphago-the-story-so-far

DeepMind. (2017b). *The Future of Go Summit: AlphaGo Pair Go & Team Go*. https://www.youtube.com/watch?v=V-_Cu6Hwp5U

DeepMind. (2017c). *The Future of Go Summit*. https://www.deepmind.com/research/highlighted-research/alphago/the-future-of-go-summit

DeepMind. (2019). *StarCraft II Learning Environment*. Github. https://github.com/deepmind/pysc2

DeepMind. (2020). *AlphaGo - The Movie*. https://www.youtube.com/watch?v=WXuK6gekU1Y

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In J.

Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 1407–1416). PMLR. https://proceedings.mlr.press/v80/espeholt18a.html

Funk, J. (2013). *MOBA, DOTA, ARTS: A brief introduction to gaming's biggest, most impenetrable genre*. Polygon. https://www.polygon.com/2013/9/2/4672920/moba-dota-arts-a-brief-introduction-to-gamings-biggest-most

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, Issue 2). MIT press Cambridge.

Goodman, D., & Keene, R. (1997). Man versus machine: Kasparov versus deep blue. *ICGA Journal*, *20*(3), 186–187.

Goodrich, J. (2021). How IBM's Deep Blue Beat World Champion Chess Player Garry Kasparov. *IEEE Spectrum*. https://spectrum.ieee.org/how-ibms-deep-blue-beat-world-champion-chess-player-garry-kasparov

Haenlein, M., & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, *61*(4), 5–14.

Hall, T., & Romero, J. (2011). *Classic Game Postmortem - DOOM*. Game Developers Conference. https://www.gdcvault.com/play/1014627/Classic-Game-Postmortem

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. Y. (2014). *Deep Speech: Scaling up end-to-end speech recognition*.

Hausknecht, M., Lehman, J., Miikkulainen, R., & Stone, P. (2014). A Neuroevolution Approach to General Atari Game Playing. *IEEE Transactions on Computational Intelligence and AI in Games*, *6*(4), 355–366. https://doi.org/10.1109/TCIAIG.2013.2294713

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B. (2012). Deep Neural Networks for

Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hsu, F., Campbell, M. S., & Hoane Jr, A. J. (1995). Deep Blue system overview. *Proceedings of the 9th International Conference on Supercomputing*, 240–244.

Hutchison, A. (2008). Making the water move: techno-historic limits in the game aesthetics of Myst and Doom. *Game Studies: The International Journal of Computer Game Research*, *8*(1).

id Software. (2005). *Quake III Arena GPL source release*.

Inspir.ai. (2021). *Inspir.ai*. https://www.inspirai.com/company?language=en

Jaderberg, M., Czarnecki, W. M., Dunning, I., Graepel, T., & Marris, L. (2019). *Capture the Flag: the emergence of complex cooperative agents*. DeepMind. https://deepmind.com/blog/article/capture-the-flag-science

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., & Graepel, T. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, *364*(6443), 859–865. https://doi.org/10.1126/science.aau6249

Jin, D. Y. (2010). *Korea's online gaming empire*. The MIT Press.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260. https://doi.org/10.1126/science.aaa8415

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3367–3375.

Lighthill, J. (1973). *Artificial Intelligence: A General Survey*. http://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/p001.htm

Liquipedia. (2021). The International. In *Liquipedia*. https://liquipedia.net/dota2/The_International

Liquipedia. (2022). Tournaments. In *Liquipedia*. https://liquipedia.net/starcraft/Leagues

Loguidice, B., & Barton, M. (2012). *Vintage games: an insider look at the history of Grand Theft Auto, Super Mario, and the most influential games of all time*. Routledge.

Marblestone, A. H., Wayne, G., & Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, *10*, 94.

Marcus, G. (2014). *What Comes After the Turing Test?* The New Yorker. https://www.newyorker.com/tech/annals-of-technology/what-comes-after-the-turing-test

McCorduck, P., & Cfe, C. (2004). *Machines who think: A personal inquiry into the history and prospects of artificial intelligence*. CRC Press.

McDermott, J. (1982). R1: A rule-based configurer of computer systems. *Artificial Intelligence*, *19*(1), 39–88.

Millington, I., & Funge, J. (2018). *Artificial intelligence for games*. CRC Press.

Minsky, M. L. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall Englewood Cliffs.

Minsky, M., & Papert, S. (1969). *Perceptrons*.

Mitchell, F. (2018). Esports Essentials: The Legacy of Counter-Strike. *The Esports Observer*. https://archive.esportsobserver.com/how-esports-works-counter-strike/

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*.

Moss, R. (2020). *How Warcraft III birthed a genre, changed a franchise, and earned a Reforge-ing*. Ars Technica. https://arstechnica.com/features/2020/01/how-warcraft-iii-birthed-a-genre-changed-a-franchise-and-earned-a-reforge-ing/

Newell, A., & Simon, H. A. (2007). Computer science as empirical inquiry: Symbols and search. In *ACM Turing award lectures* (p. 1975).

Newell, G. (1999). *Half Life: Interview With Gabe Newell*. GameSpot UK. https://web.archive.org/web/20010723160349/http://extra.gamespot.co.uk/pc.gamespot/features/halflife_uk/02.html

Nilsson, N. J., & Nilsson, N. J. (1998). *Artificial intelligence: a new synthesis*. Morgan Kaufmann.

Oh, J., Guo, Y., Singh, S., & Lee, H. (2018). Self-Imitation Learning. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 3878–3887). PMLR. https://proceedings.mlr.press/v80/oh18b.html

Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013). A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games*, *5*(4), 293–311. https://doi.org/10.1109/TCIAIG.2013.2286295

OpenAI. (2018a). *OpenAI Five*.

OpenAI. (2018b). *OpenAI Five Benchmark*. https://openai.com/blog/openai-five-benchmark/

OpenAI. (2019). *OpenAI Five Defeats Dota 2 World Champions*. https://openai.com/blog/openai-five-defeats-dota-2-world-champions/

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki,

J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., … Zhang, S. (2019). *Dota 2 with Large Scale Deep Reinforcement Learning*. https://arxiv.org/abs/1912.06680

Orland, K. (2021). *Cheat-maker brags of computer-vision auto-aim that works on "any game."* Ars Technica. https://arstechnica.com/gaming/2021/07/cheat-maker-brags-of-computer-vision-auto-aim-that-works-on-any-game/

Quake (video game). (n.d.). In *Wikipedia*. https://en.wikipedia.org/wiki/Quake_(video_game)

Ragin, C. C. (1999). Using qualitative comparative analysis to study causal complexity. *Health Services Research*, *34*(5 Pt 2), 1225.

Roitblat, H. (2021). *Building artificial intelligence: Reward is not enough*. TechTalks. https://bdtechtalks.com/2021/07/07/ai-reward-is-not-enough-herbert-roitblat/

Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach. In *Prentice Hall* (3rd ed.).

Schaeffer, J. (2001). A Gamut of Games. *AI Magazine*, *22*(3), 29. https://doi.org/10.1609/aimag.v22i3.1570

Schaller, R. R. (1997). Moore's law: past, present and future. *IEEE Spectrum*, *34*(6), 52–59. https://doi.org/10.1109/6.591665

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. arXiv. https://doi.org/10.48550/ARXIV.1506.02438

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv. https://doi.org/10.48550/ARXIV.1707.06347

Schwalbe, U., & Walker, P. (2001). Zermelo and the Early History of Game Theory. *Games and Economic Behavior*, *34*(1), 123–137.

Sf2platinum. (2017). *The AI Engine*. https://sf2platinum.wordpress.com/2017/01/20/the-ai-engine/

Shannon, C. E. (1993). Programming a computer for playing chess. *First Presented at the National IRE Convention, March 9, 1949, and Also in Claude Elwood Shannon Collected Papers*, 637–656.

Shevlin, H., Vold, K., Crosby, M., & Halina, M. (2019). The limits of machine intelligence: Despite progress in machine intelligence, artificial general intelligence is still a major challenge. *EMBO Reports*, *20*(10), e49177–e49177. https://doi.org/10.15252/embr.201949177

Shu-Hsien Liao. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications*, *28*(1), 93–103. https://doi.org/https://doi.org/10.1016/j.eswa.2004.08.003

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Silver, D., Singh, S., Precup, D., & Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, *299*, 103535. https://doi.org/https://doi.org/10.1016/j.artint.2021.103535

Simon, H. A. (1965). *The shape of automation for men and management* (Vol. 13). Harper \& Row New York.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. https://doi.org/10.1007/BF00115009

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science \& Business Media.

Tei, B. (2021). *Quake - family tree*. Wikipedia. https://commons.wikimedia.org/wiki/File:Quake_-_family_tree.svg

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, *6*(2), 215–219.

Top Games Awarding Prize Money. (2020). Esports Earnings. https://www.esportsearnings.com/games

Turing, A. (1950). *Computing Machinery and Intelligence*.

Turing, A. (1951). Can digital computers think? *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, 111–116.

Turing, A. (1953). Digital computers applied to games. *Faster than Thought*.

van den Herik, H. J., Uiterwijk, J. W. H. M., & van Rijswijck, J. (2002). Games solved: Now and in the future. *Artificial Intelligence*, *134*(1), 277–311. https://doi.org/https://doi.org/10.1016/S0004-3702(01)00152-7

Van Waveren, J. M. P. (2001). The Quake III Arena Bot. *University of Technology Delft*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., … Silver, D. (2019). *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., & others. (2017). Starcraft ii: A new challenge for reinforcement learning. *ArXiv Preprint ArXiv:1708.04782*.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf

Vinyals, O., Gaffney, S., & Ewalds, T. (2017). *DeepMind and Blizzard open StarCraft II as an AI research environment*. DeepMind. https://deepmind.com/blog/announcements/deepmind-and-blizzard-open-starcraft-ii-ai-research-environment

WCG. (2021). *WCG history*. https://www.wcg.com/2021/history/historyList?lang=en

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, *13*(3), 55–75. https://doi.org/10.1109/MCI.2018.2840738

Appendix

A Simple Youtube top comments finding method

Youtube video page has top comment option but it does not sort the comments simply based on like numbers. Some rules are hidden therefore a simple method for finding the most-liked comments.

1. Go to the video page and scroll down to the end of comments, or where the comments have very low like numbers. This ensures that most-liked comments are loaded into the local browser.
2. Apply the following JavaScript code in developer console. Youtube weirdly uses an identical id for comments tags. This reads all like numbers and sort them.

```
let done = 0; let likes = []

while (!done) {

  let two = document.getElementById("vote-count-middle")

  if (two) {

    if (two.innerText)

      likes.push(two.innerText)

    two.id = "vote-count-middle2"

  }

  else

    done = 1

}


const TtN = (text) => {
```

```
  if (text.endsWith('K'))
    return Number(text.substring(0, text.length - 1)) *
1000
  else if (text.endsWith('M'))
    return Number(text.substring(0, text.length - 1)) *
1000000
  else if (text.endsWith('B'))
    return Number(text.substring(0, text.length - 1)) *
1000000000
  else {
    const a = Number(text)
    return Number.isNaN(a) ? 0 : a
  }
}
const TtP = (text) => {
  return {
    text: text,
    loves: TtN(text)
  }
}
let unique = new Set(likes)
let sorted = [...unique].map(TtP).sort((a, b) => a.loves
- b.loves)
```

```
let result = sorted.map(i=>i.text)

result
```

3. Since “result” is the last line in the code, it should show the result of likes. If it does not show then type “result” and press enter.
4. Use browser’s find (Ctrl + F) and type the highest like number text (e.g., “2.9K”) to find the top comments. One like number text may point to two or more comments.