香　港　大　學

THE UNIVERSITY OF HONG KONG

Faculty of Engineering

Department of Computer Science

Applications of Deep Learning in Video Games

Supervisor: Dr. Dirk Schnieders

Zain Ul Abidin (3035305590)

Fawad Masood Desmukh (3035294478)

PROJECT PLAN

September 30, 2018

Contents

1 Background

Video games have always been a popular proving ground for algorithms and approaches to achieve artificial intelligence. Historically, this started with attempts to play classical board games such as chess and checkers. Alan Turing, said to be the *father* of computer science, discussed one of the first artificially intelligent approaches to play chess in his famous 1953 paper using the minimax algorithm [1]. Just a few years later, Arthur Lee Samuel, a pioneer in Machine Learning, demonstrated an early concept of artificial intelligence that is now known as reinforcement learning, when he came up with a self-learning approach to play and improve at checkers [2]. Artificial Intelligence in video games broke new ground towards the end of the century when IBM's Deep Blue managed to become the first ever computer system to defeat a reigning chess grandmaster, Garry Kasparov, in 1996 [3].

Coming up to modern day the applications of artificial intelligence have continued to evolve and grow underpinned by increasingly powerful technology and more data which can support more intensive approaches such as deep learning [4]. It was with a breakthrough approach using deep learning, that Deepmind's AlphaGo managed to defeat Lee Sedol, one of the best Go players, in 2016 [5].

Our motivation to pursue this project revolves around the following two statements:

1. *Games are good for artificial intelligence*

There are a number of reasons why games are a popular testbed for the study and benchmarking of artificial intelligence techniques and approaches:

- **Games are challenging problems to solve.** The challenge comes from the fact that the finite state spaces of the game such as the decisions and strategies the player can make are often very vast and the solution spaces are much smaller in comparison. Evaluating the effectiveness of particular strategies is often difficult due to the delayed nature of the rewards in games.
- **Games are based on well-defined rules** so approaches can take discrete actions to work towards solving a game.
- **Games offer cheaper and more convenient testbeds** than physical robotic environments. This is particularly useful for solving problems such as pathfinding and navigation.
- **Games are a popular form of entertainment.** This leads to a wide diversity of games and therefore multifarious problems to solve as the complexity and genre of games change - 2D vs 3D, Racing vs Real-time Strategy. The popularity of games also leads to a large amount of content generation to support supervised learning strategies.

- **Games present challenges for many areas of artificial intelligence** ranging from pathfinding to natural language processing. There are key historical associations between games and the development of novel approaches in AI ranging from tree searching to machine learning.

2. Artificial Intelligence is good for games

There a number of ways where the creation of better AI has been beneficial for the gaming community:

- **Create better content.** Better AI can lead to creating agents that can play the game "*well"* and the play the game *"believably"*. This can lead to an improvement in the potential enjoyment from a video game.
- **Novel approaches to designing new and interesting games** such as procedurally generated game level designs.

Recent breakthroughs in this domain have come through the application of novel approaches in deep learning such as deep reinforcement learning and evolutionary algorithms. Through our project, we hope to apply such cutting-edge techniques to explore the applications of deep learning on video games to test the limits of these approaches and make notable contributions to the wider video gaming and artificial intelligence community.

2 Objective and Scope

In this project, we will first survey the different classes of games and environments available, the challenges involved in attempting to solve these games and the existing solutions to these challenges. We will then identify a suitable choice of video game.

Due to the diverse nature of the gaming world and time constraint we will be limiting ourselves to a single game. The primary objective of the project will be to explore the new frontiers of deep learning within the scope of the chosen video game. We will focus on developing an agent to play the game by applying various deep learning approaches such as deep reinforcement learning and evolutionary algorithms. Given the choice of game and availability of supporting environment/framework, we may also have to develop a framework to allow our agent to interact with the game since not all games would have an easily available API.

After thoroughly experimenting with these approaches, we will shift our focus towards the most promising approach and work towards increasing its performance to a reasonable level. The agent we develop will eventually hope to outperform conventional scripted AI and possibly beat humans, without any prior knowledge of the game. Finally, we will report on the degree of success of our approaches after testing and experimentation.

Furthermore, if time allows we will be working towards exploring certain questions and topics of interest. A particularly interesting one is creating agents that can learn to play well not only a particular level in a game but generalize across levels and possibly even across different games and genres altogether.

3 Methodology

This project involves exploration of the most reasonable deep learning approaches to develop a self-learning agent to overcome the challenges in conquering a game. We realize that video games have a miscellany of challenges and plan to do a thorough research to explore these before selecting our choice of game. After our preliminary research, we have shortlisted two game genres as potential targets for our project - Arcade/Retro Games and Real-Time Strategy Games.

3.1 Game Genre

3.1.1 Arcade/Retro

This refers to the type of games found in classic arcades, home entertainment systems and consoles such as the Atari 2600 and the Nintendo Entertainment System during the 1980s and 1990s. They have classically been used as playing grounds for testing AI approaches.

Given hardware restrictions, they typically involve interactions in a 2D space (or a semi-3D isometric environment) with action-reaction being created by the collision of entities such as sprites on the screen. The movement mode can vary from continuous to discrete. These games are often characterized by the requirement to be precise and fast in responding to the changing game environment such as in *Space Invader*. Many games involve pathfinding and navigation logics as exemplified by *Pacman*. Some games like the classic game, *Breakout*, require logic that is more reactionary whereas others such as *Montezuma's Revenge* and *Super Mario Bros* need an involvement of long term planning. Therefore, the complexity of the problems varies from game to game.

Recently, there has been significant success in tackling these games using deep learning approaches. Most notably, an approach called *Deep Q-Network (DQN)* learned to play seven Atari Games in the *Atari Learning Environment* at a human-expert level [6]. This and other successful experiments have shown the suitability of applying deep reinforcement learning algorithms such as DQN and its variations to these problems. In addition, evolutionary algorithmic approaches have also showed promising results on the same Atari environment with an approach known as *Cartesian Genetic Programming* [7].

Given the extensive collection of games in this genre, we will choose a particular as-yet unsolved game that has characteristics of being dynamic and multi-agent that could prove to be challenging when we apply the above-mentioned techniques.

OpenAI has recently developed and open-sourced *OpenAI Gym Retro* - a platform to develop and compare reinforcement learning algorithms by providing a single wrapper interface to a number

of different environments such as *Atari, Nintendo, Sega* and *Flash* to turn them into "Gym" environments. This will be an important resource if our chosen game is within the Arcade/Retro genre.

3.1.2 Real Time Strategy

RTS is a genre of the game where the main objective is to conquer a conquest by collecting resources, building landmarks and units while simulating a military setting at various levels of complexity.

Unlike the classical board games such as Chess, strategy games significantly prove to be one of the harder domains as they are classified as a multi-agent problem where multiple units are required to make moves at any given time on a partially observable map. This results in a much more complex environment at hand. In addition to this, the rewards in these strategy games are based on the results at the end of the game rather than on the current moves. For these reasons, Real Time Strategy games still lack a satisfactory solution to an efficient self-learning agent to play the game and intrigues the interest of many researchers.

The StarCraft series is a good example of such a Real-Time Strategy game, which is being studied widely. The first game in the series, StarCraft has a Brood War API available, which allows interaction with the game. In addition, the game has a library, TorchCraft, built upon the API, which allows machine-learning research on this game by enabling the connection between Torch framework and the Brood War API [8].

After successfully beating Go, Deepmind has shifted its focus on now tackling StarCraft II as the next big challenge and released results of their work on StarCraft II last year [9]. Through collaboration with Blizzard Entertainment, *PySc2* and *SC2LE* were developed as environments to allow reinforcement-learning agents to interact with StarCraft II to get observations and send actions. The paper describes using a reinforcement-learning algorithm, *Asynchronous Actor-Critic Agents (A3C)* with various different deep learning architectures to try to tackle StarCraft II. None of their attempts managed to defeat the easiest in-game AI in a full game. In an attempt to simplify the situation, mini games were created which the created RL agents had good success on.

Given the complexity of the problem, our approach to simplify the problem would be as follows:

- Scale the problem down to a new mini game scenario that we create.
- Replicate the deep learning strategies as mentioned in DeepMind's research as they did not make their source code publicly available.
- Apply the strategies on the mini game and report on the results after testing and experimentation.

3.2 Techniques/Algorithms

Through our preliminary research, we have identified the following two approaches for further exploration:

3.2.1 Deep Reinforcement Learning

In reinforcement learning, an agent interacts with the environment and learns a particular behavior through this interaction. A video game can be suitably modeled as a reinforcement-learning problem with the player being the agent interacting with the game environment by choosing an action from a finite action space and these actions then lead to success or failure. The rewards for an action can be captured from the game, which may be immediate such as a change in score or delayed as in a win or loss of the game.

Various Deep reinforcement-learning algorithms such as Deep Q Network and A3C/A2C have been successfully applied in video games and we look into using these in our project. In particular, the OpenAI baseline reinforcement learning algorithm implementations will be an important resource [10].

3.2.2 Evolutionary Approaches

These approaches involving neuroevolution can be used to optimize and adjust a neural networks' weights as well as topology. Various algorithms such as *Cartesian Genetic Programming* and *NeuroEvolution of Augmenting Topologies (NEAT)* [11] have been successfully applied in video games and we look into using these in our project.

4 Project Schedule and Milestones

Time	Milestone
2018	
September	• Project research. • Review of existing work. • Phase One Deliverable - Detailed Project Plan and Project Web Page
October 1 - 14	• Research on various classes of games and approaches to apply deep learning. • Choose Game
October 15 – 31	• Setup environment with chosen game
November	• In-depth research on deep reinforcement learning and evolutionary approaches
December	• Implement researched approaches onto chosen game
2019	
January	• Evaluation of implementation • Iterate and improve on results • First Presentation (7 - 11) • Phase Two Deliverable - Preliminary implementation and Detailed interim report (20)
February - March	• Explore alternative approaches if initial approaches fail • Testing and experimentation
April	• Final Cleanup • Report Writing • Phase Three Deliverable - Finalized Tested Implementation & Final Report (14) • Final Presentation (15-17) • Project exhibition (29)

5 Materials Required

Looking at the challenge that we have on our hand and the huge domain of the games we will require a GPU for training the self-learning agent.

6 Separation of Responsibilities

The group undertaking this project consists of two members. For a thorough and complex project as this, we feel it is imperative that both members of the group contribute equally towards the completion of project. After a mutual consensus, both group members realize that a fair distribution of responsibility and consistent sharing of knowledge will be required for the success of the project. As of now, there is no specific set of duties assigned to the individual members. However, the group will be clearer about the upcoming responsibilities as the project progresses.

7 References

[1] Alan M. Turing. Digital computers applied to games. Faster than thought, 101, 1953.

[2] Arthur L. Samuel, "Some studies in machine learning using the game of Checkers," *IBM Journal of research and development*, vol. 3, no. 3 pp. 210 - 229, July 1959.

[3] History.com Editors (2009, November 16). Deep Blue defeats Garry Kasparov in chess match [Online]. Available: https://www.history.com/this-day-in-history/deep-blue-defeats-garry-kasparov-in-chess-match. [Accessed: 2018, September 29].

[4] Buchanan, Bruce G., "A (very) brief history of artificial intelligence," *AI Magazine*, vol. 26, no. 4, Winter 2005

[5] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature.*, vol. 529, no. 7587, pp. 484 - 489, January 2016.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint* arXiv:1312.5602, December 2013.

[7] D. G Wilson, S. Cussat-Blanc, H. Luga and J. F Miller, Evolving simple programs for playing Atari games., *arXiv preprint* arXiv:1806.05695, June 2018.

[8] G. Synnaeve, N. Nardelli, A. Auvolat, S. Chintala, T. Lacroix, Z. Lin, F. Richoux, and N. Usunier., Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint* arXiv:1611.00625, November 2016.

[9] O. Vinyals, T. Ewalds, S. Bartunov, A. S. Georgiev, Petko Vezhnevets, M. Yeo, A. Makhzani, H. Kuttler, J. Agapiou, J. Schrittwieser, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. v. Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing., Starcraft II: A new challenge for reinforcement learning. *arXiv preprint* arXiv:1708.04782, August 2017.

[10] GitHub. (n.d.). OpenAI Baselines: high-quality implementations of reinforcement learning algorithms. [Online]. Available: https://github.com/openai/baselines. [Accessed: Sep. 29, 2018].

[11] Kenneth O. Stanley, and Risto Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *MIT Press*, vol. 10, no. 2, pp. 99-127, 2002