Analysis of the application of Reinforcement Learning algorithms on the Starcraft II video game

Leandro Vian[1], Marcelo de Gomensoro Malheiros[2]

Abstract: In recent years Machine Learning techniques have become the driving force behind the worldwide emergence of Artificial Intelligence, producing cost-effective and precise tools for pattern recognition and data analysis. A particular approach for the training of neural networks, Reinforcement Learning (RL), achieved prominence creating almost unbeatable artificial opponents, in board games like Chess or Go and also on video games. This paper analyses the workings of Reinforcement Learning, and tests this approach against a very popular real-time strategy game, Starcraft II. Our goal is to examine the tools and algorithms readily available for RL, also addressing different scenarios where a neural network can be linked to Starcraft II to learn by itself. This work describes both the technical issues involved and the preliminary results obtained by the application of two specific training strategies, A2C and DQN.

Keywords: Artificial Intelligence. Machine Learning. Neural Networks. Reinforcement Learning. Video Games.

[1] Graduate in Software Engineering at the University of the Taquari Valley – UNIVATES. leandrovian@hotmail.com

[2] Professor at the University of the Taquari Valley – UNIVATES. PhD in Computing at UFRGS – RS, MSc in Electrical Engineering at UNICAMP – SP, graduate in Computer Engineering at UNICAMP – SP. mgm@univates.br

1 INTRODUCTION

Artificial Intelligence (AI) has become part of our life, being a frequent term when computers are involved in automated processes of decision making, advancing in areas that were previously done only by humans. The recent resurgence of Artificial Intelligence, and particularly of its Machine Learning (ML) subfield, occurred when researchers combined techniques developed decades ago with modern parallel computers and programming techniques.

Games are as old as humankind and have evolved hand in hand with civilization. With the advent of computers, they became complex pieces of software, allowing for a multitude of genres to be developed, such as role-playing, first-person shooters and strategy video games.

As many computer games offer a simulated and self-contained environment, they are very useful as targets for Artificial Intelligence research, having a variety of challenges that need to be understood and overcome to achieve victory. Those test chambers can help train and validate automated learning and decision-making strategies that would otherwise be too complex or expensive to be solvable directly in a real-world environment.

In this work, we describe the application of current algorithms of Machine Learning into the construction and validation of self-learning neural networks, by employing Reinforcement Learning algorithms.

We are particularly interested in a new environment that was created to help close the gap between video games and research: the SC2LE (Starcraft II Learning Environment), created in partnership by DeepMind and Blizzard Entertainment. This environment makes possible the development of an AI agent that plays the Starcraft II game. Such agent will then be capable of: controlling hundreds of units in cohesion, using strategy to exert influence and secure areas on the map; managing resources required to construct a base and build units; making decisions based on imprecise information offered by the game screen and mini-map, both which have a “fog-of-war” effect that prevents vision if no friendly unit is nearby an area. Moreover, maps are big and diverse, offering many obstructions and the concept of land and air units. Finally, games may last for many minutes and decisions may not show results or consequences until later on.

All points considered, SC2 games are a difficult challenge towards building better AI tools and techniques that can be later applied to many areas, raising the simulation aspect of games to a new standard and importance.

The benefits that can be achieved by merging the amount of data available with discoveries in Reinforcement Learning are vast, most areas can benefit if some manner by either have tasks being automated or simply by making better decisions through their processes. One example is in the medical area, where images, laboratory results and symptoms can go through AI systems that have been trained with millions of cases and will help doctors make decisions or maybe a simpler case where trained AI systems can help companies run more efficiently, lowering costs and improving results.

In the following sections, we describe our particular experiments, where we compare the A2C and DQN Reinforcement Learning algorithms, by creating agents that play two distinct Starcraft II mini-games. Different configurations and parameters are then analyzed, giving the basis for our results and conclusions.

2 LITERATURE REVIEW

This section presents the main concepts used during the development of this work.

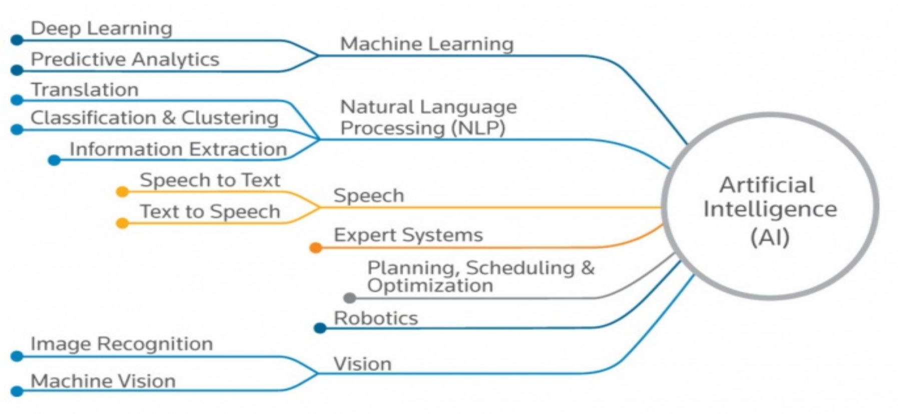
2.1 Artificial Intelligence

In the last two decades, AI has evolved a tremendous amount, stopping from being something that would be seen of specific circles of talk or science fiction movies, to becoming part of our daily routine. Things such as personal voice assistants, self-driving vehicles or even behavioral suggestive algorithms started to permeate all aspects of our life (ADAMS, 2017).

Some interesting advancements can be pointed out, such as the one that resulted in the creation of a new business model that competes with the traditional taxi companies. The Uber company offers their clients the ability to ask for a ride knowing exactly how much they will have to pay, and how long will take for the car to arrive and for the trip to be complete. This is done with the use of an AI system that incorporates specific behaviors that calculate distances, car availability, traffic and demand during specific times of the day. Another example of the regular use of AI in our routines be can be seen in the airline companies, which uses AI autopilots as a common tool and as much as possible, to many passengers surprise as noted by (NARULA, 2017).

Despite the numerous advancements and researches released making huge headlines on news around the Internet, most of the time they are referred to as AI for the sake of simplicity and to help readers associate the information with the overall area of expertise. AI is a huge area of study which encompasses many subfields (ATTICK, 2016). A representation of AI and its subfields can be seen in Figure 1.

Figure 1: Breakdown of AI and its subfields.



Source: (ATTICK, 2016)

2.2 Machine Learning

Machine Learning is a field that is attracting a lot of currently, being responsible for many new advancements, the term was coined by Arthur Samuel in 1959 to whom was attributed the creation of the world first self-learning program. The program he developed played checkers and used a search tree of the board to determine the possible moves based on the state of the board. The idea behind ML is to have the program teach itself by iterating its logic using all the data available t him as input, each iteration results and some knowledge that will help it build its neural network (RUSSEL, 1995).

The modern society produces an incredible amount of data each day, being it originated from smartphones and computers, applications and social media or even antennas and satellites. With an unending stream of data to be feed, ML systems are discovering new ways of sell and advertise products, helping business owners to target audiences, helping doctors making decisions and even helping with the test of new medicines (DOMINGOS, 2015).

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a subarea of Machine Learning that is focused on how to teach agents the best actions to take on a specific scenario, this is done by letting the agent take an action and then analyzing the outcome and comparing that to what is considered a good outcome. The idea is that with each cycle of action, reward and analyzes, the agent will gradually understand what actions produce the best outcomes and will steer toward those actions.

Reinforcement Learning was responsible for many great feasts in the latest years, being considered a buzzword in the IT community, this influx of interest also attracted many new followers, researchers and investors to the area, looking forward to discoveries and new opportunities (SHAIKH, 2017).

2.4 Artificial Intelligence in video games

Artificial Intelligence is an area inside the game industry responsible for making the games look smart, being it with enemies that behave with human nature, make reasonable pathfinding decisions or simply choose the best tool/weapon for a specific situation. All games have some sort of AI responsible for all manner of jobs, but before 1990 all of them used similar techniques which relied on hard-coded choices based on if-else decisions that determined the next action. During the '90s games did still relied on defined states but started to introduce AI improving techniques, such as sense simulation, which allowed enemies to notice things such as dead friends and react accordingly (MILLINGTON, 2009).

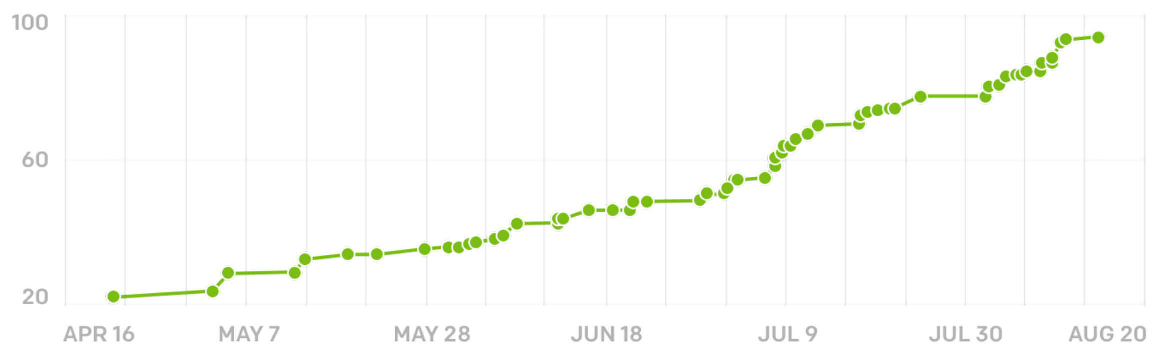
During the next years, other advancements were made, such as strategy games introducing noticeable AI pathfinding for units and formation motion for groups, while in the 2000s some games presented things such as neural network-based units. Some other topics still were not solved, like RPGs using tree-based dialog systems or sports games having trouble with some dynamical calculation required for sports simulation.

While the need for complex AI depends on each game, with some requiring an advanced AI to be enjoyable, others don’t see much improvement by implementing a complex system, relying on basic solutions. The common use for AI in most modern games is described by (MILLINGTON, 2009): “The AI in most modern games address three basic needs: the ability to move characters, the ability to make decisions about where to move, and the ability to think tactically or strategically.”

One example of RL being at the spotlight was seen during a professional gaming tournament called The International, where a team of players fought each other competing for prizes of a total of $24,787,916. During the tournament a presentation was made, putting an agent trained with RL to play Dota 2 against the best Dota 2 players in the world, the result left the crowd ecstatic as the agent won easily against all human opponents (SHAIKH, 2017).

The agent was created by the nonprofit research company OpenAI, which after developing the agent, trained it for over four months against itself in countless games where it learned on its own how to play the game. During those four months, the bot went from a completely clueless agent that didn't know how to navigate the map, to be able to beat the top world player handily. In Figure 2 we can see the evolution of the agent matchmaking rating (MMR) evolution during the four-month training period before the tournament. The timeline represents that 15% of players are below 1.5k MMR, 58% of players are below 3k and 99% are below 7.5k (OPENAI, 2017).

Figure 2: Agent evolution graph.



Source: (OPENAI, 2017)

3 DEVELOPMENT

This section presents details relative to the tests, research, programming, and results obtained during the experimentation phase. A special focus is given to neural network paradigms and sets of algorithms and their characteristics related to the problems in discussion. Also, the tools that were used are discussed, such as Starcraft II, SC2LE, and TensorFlow. All are either at the core of the work or present significant benefits that justify their use at one of the later stages.

3.1 Game environment

Starcraft II is the second game of the Starcraft franchise developed by Blizzard Entertainment and released in 2007.

The first game of the franchise was released in 1998, being highly successful. During its period it developed a community of developers who built scripted bots to fight each other in competitions to see which was the best. The second game was also considered a success, which now offers the following challenges to RL agents:

- Two resources that need to be mined and managed (minerals and gas).
- Construction of production buildings.
- Different units and buildings, each with their actions and options.
- Completely different races.
- Large maps with different terrain levels.
- Land and air units.
- Training of an army.
- Management of hundreds of units during battle.
- Imperfect information due to the fog-of-war effect.
- Decisions that may only see their consequences later in the game.

With the release of SC2, the developers behind the game decided to offer tools to help the community build their bots, partnering with Deepmind, an Artificial Intelligence research company (VINYALS, 2017).

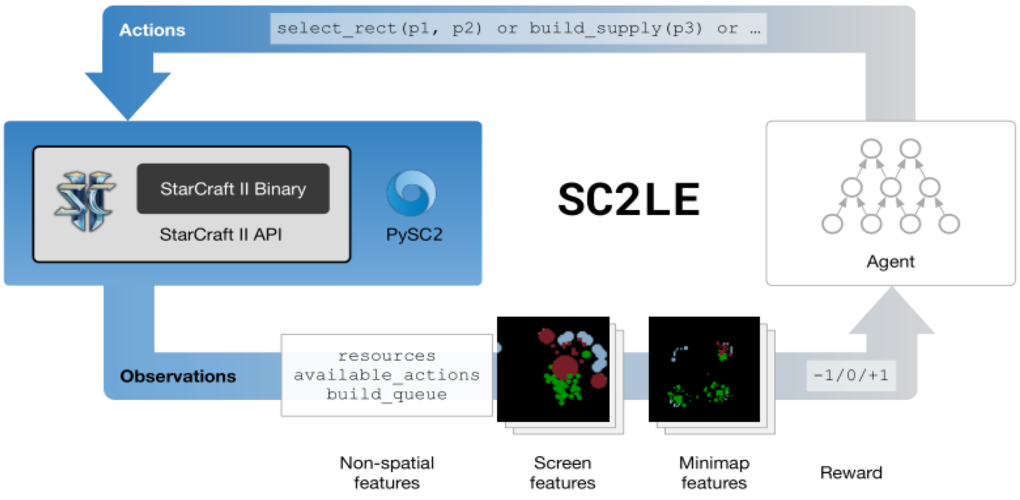
3.1.1 SC2LE

Starcraft II Learning Environment (SC2LE) is a set of tools created to allow the community of AI research using the Starcraft II game.

It is composed of the SC2client-proto, a Machine Learning API created by Blizzard that allows direct control of the game. It also contains the PySC2, which is a toolset written in Python that facilitates the use of the Blizzard API with the agents. There are also packs of replays from SC2, with more than 500 thousand replays to be used for RL. Finally, there are mini-games to help with the research of specific areas of the game, allowing for the training of specific areas of the game, removing complexity involved with the need for the agents to tackle the whole game from start.

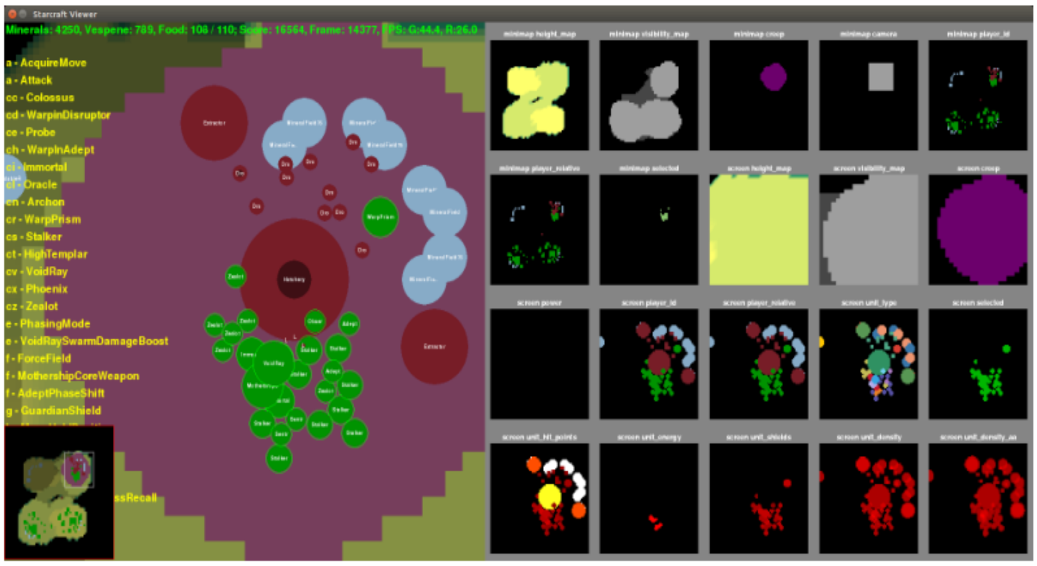
Figure 3 shows a representation of the SC2LE components during its use by an agent, demonstrating the input, decision making, and reward analyze flow. Figure 4 shows a representation of the image of the game broken down into layers that related to the various kinds of information necessary for the agents to make decisions.

Figure 3: Representation of SC2LE flow.



Source: (VINYALS, 2017)

Figure 4: Depiction of the different information layers.



Source: (VINYALS, 2017)

3.1.2 Agents

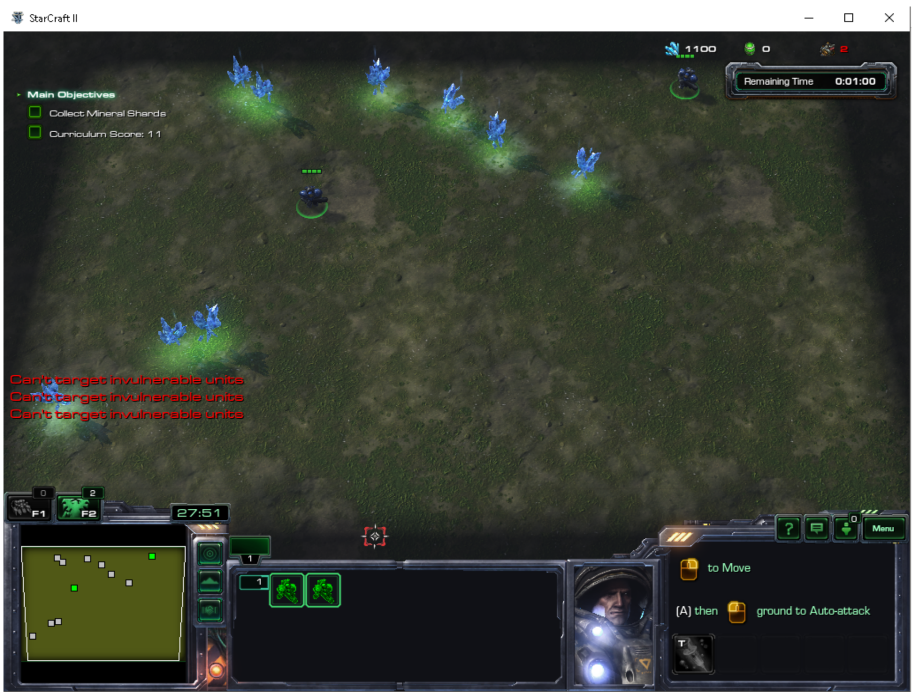
The SC2LE agents consist of stages that are similar no matter which mini-game or RL algorithm is being used. The process starts reading the current game state and doing an analysis of the rewards received from the last episode, which then is used in conjunction with the current neural network to choose the next action that’s going to be taken.

The agent in this work was built using the Python language and packages available for its environment, such as the OpenAI Baselines. This structure allowed the agent to be created with a lean structure that only diverged when defining the RL algorithm that was selected for the agent to be run with. That is, most of the code was generic enough to provide a testbed for a comparison of distinct RL algorithms.

3.1.3 Mini-games

SC2LE mini-games consist of small Starcraft II maps that isolate elements of the full game, allowing for agents to be created and tested in a much simpler environment and with a smaller degree of complexity. Those mini-games break the full game into smaller tasks such as moving into specific locations, collecting resources such as minerals and gas, building units, defeating enemies and finding the enemies themselves.

Figure 5: Mini-game being played.



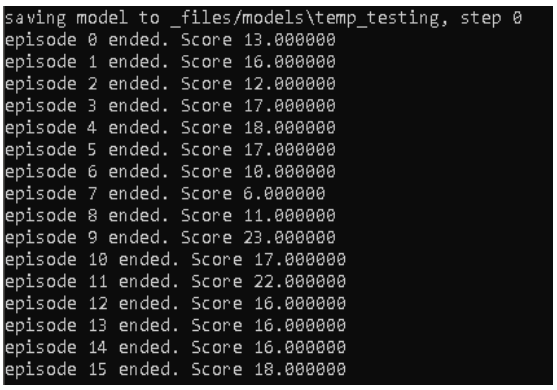
Can't target invulnerable units
Can't target invulnerable units
Can't target invulnerable units

Source: from the authors (2019)

The first mini-game used was the **Collect Minerals**, which is one of the most basic tasks required to master to successfully play the game. This mini-game challenges the agent to collect the game resource as efficient as possible controlling two individual units at the same time. In this map, the agent starts with two marine units and must use them to collect all 20 mineral shards available on the map.

Each episode of the mini-game lasts for 120 seconds, a time during which the agent will try to collect as many minerals as possible. If all the initial 20 minerals are collected, a new set of 20 minerals is spawned. An example of the mini-game session can be seen in Figure 5. The final score of the agent is based on the number of minerals collected during each episode and can be seen on the top left corner of the mini-game screen or printed to the console, as shown in Figure 6.

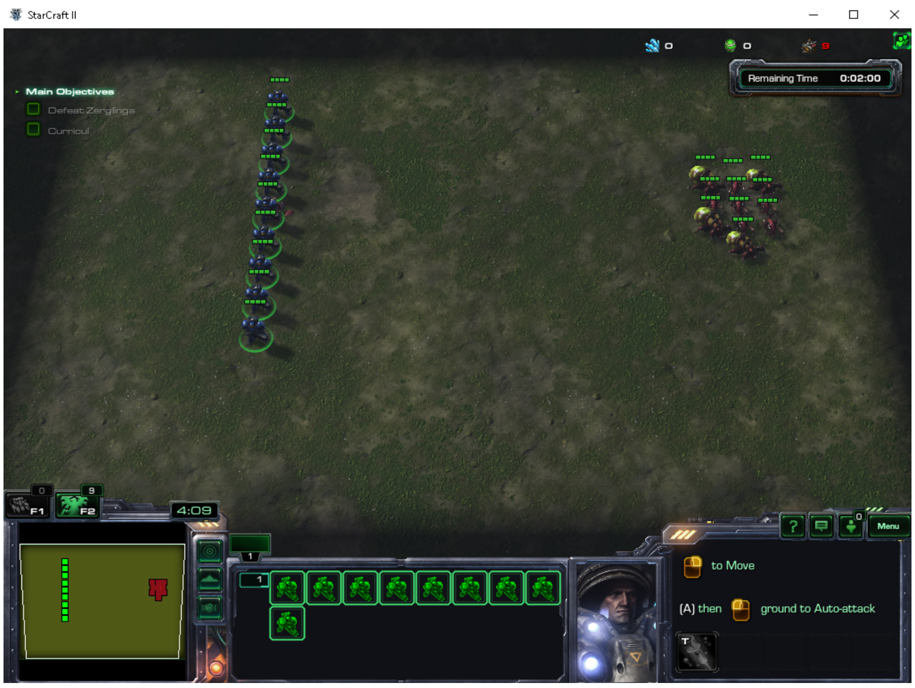
Figure 6: Scores by episode.



```
saving model to _files/models\temp_testing, step 0
episode 0 ended. Score 13.000000
episode 1 ended. Score 16.000000
episode 2 ended. Score 12.000000
episode 3 ended. Score 17.000000
episode 4 ended. Score 18.000000
episode 5 ended. Score 17.000000
episode 6 ended. Score 10.000000
episode 7 ended. Score 6.000000
episode 8 ended. Score 11.000000
episode 9 ended. Score 23.000000
episode 10 ended. Score 17.000000
episode 11 ended. Score 22.000000
episode 12 ended. Score 16.000000
episode 13 ended. Score 16.000000
episode 14 ended. Score 16.000000
episode 15 ended. Score 18.000000
```

Source: from the authors (2019)

Figure 7: Mini-game being played.



Source: from the authors (2019)

The second mini-game used was the **Defeat Zerglings and Banelings**, which mimics the challenge the agent with the combat mechanic from the game, requiring the agent to adeptly use the units available to defeat the enemies. In this map, the agent starts with nine marine units and must defeat six zerglings units and four banelings units, as in Figure 7.

Each episode of the mini-game lasts for 120 seconds, the time during which the agent will try to kill as many enemies as possible. If all the enemies are defeated, a new set of six zerglings and four banelings are spawned and the agent receives four extra marine units at full health. The final score of the agent is based on the number of enemies killed during the episode and can be seen on the top left corner of the mini-game screen or likewise printed to the console.

3.2 Machine learning tools

TensorFlow is a software library for numerical computation through graphs. Nodes represent mathematical operations and the edges represent the data arrays communicated between different nodes. It was created by the Google Brain project, and since then it proved to be general enough to be applied in a wide range of cases and domains.

The OpenAI Baselines is a collection of high-quality Reinforcement Learning algorithms, already being vastly used in many academic and commercial projects and studies. From the available implementations, the Advantage Actor-Critic (A2C) and the Deep Queue Network(DQN) were selected for being the most commonly used in academic projects.

3.2.1 Advantage Actor Critic – A2C

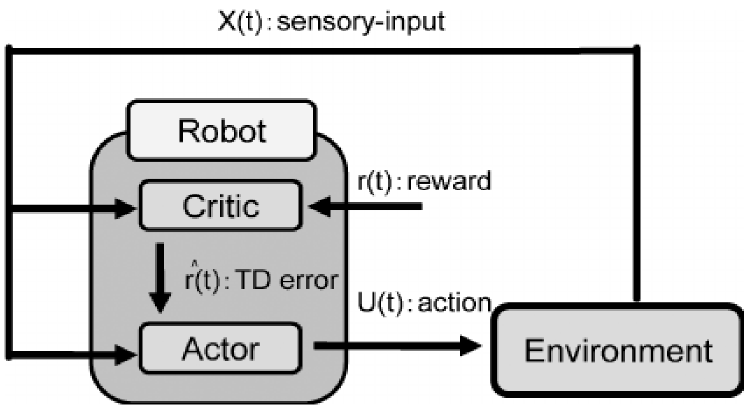
In general, the idea for Reinforcement Learning algorithms is to have an agent that receives the state of an environment and then takes actions while trying to maximize its rewards. The Advantage Actor Critic fulfills that premise in the following steps.

First, it receives the current state of the environment and uses it to generate two outputs. One output is the estimate of rewards he expects to find ahead, which is called ‘state value’ and is meant to be the ‘critic’. The second output is a recommendation of what action to take, called ‘policy’, and is meant to be ‘actor’.

Each step of state-action-reward results in the recording of the state, reward expected, action taken and reward found. After collecting the information of three iterations, the agent adjusts its critic based on the estimates he had before and the results collected, calculating the difference between what he expected, what he received and what lead him to take the action.

The algorithm also makes sure to ensure that the agent won't always choose the safest option but that has a low potential for rewards. That is done by subtracting a value called ‘entropy’ from the loss function, which is responsible for measuring the performance of the actions. A schematic implementation of the A2C model can be seen in Figure 8, using a robot as an example scenario.

Figure 8: Actor – Critic Model.



Source: (MAO, 2017)

3.2.2 Deep Queue Learning – DQN

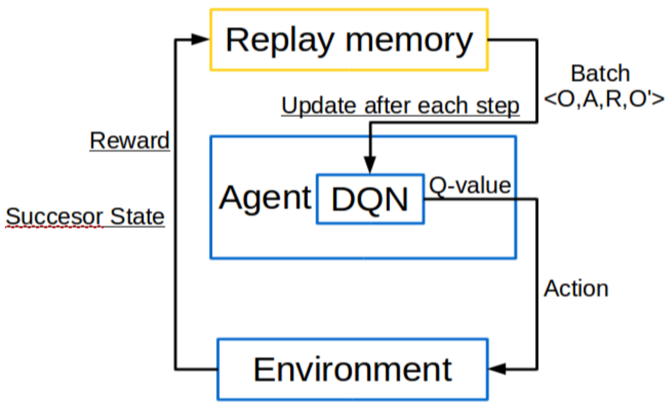
The Deep Queue Learning algorithm is a direct evolution from the Q-Learning algorithm. It was created to solve the lack of generality, which is a weakness present in the Q-Learning implementations. So even though Q-Learning is a very good RL algorithm, that lack of generality makes it unable to estimate the outcome values for states that it has not yet seen.

DQN improves the Q-Learning algorithm by introducing a neural network to estimate the reward values, giving it the ability to store the results already obtained by specific action-state decisions taken. Another improvement is the fact that the DQN implementation uses randomly picked batches of experiences from its pool, to help the network to develop itself with a broad range of experiences.

The third addition to the Q-Learning implementation is the use of a second network during the training process, responsible for reward estimates that are used in the loss function during the training process. By having separate networks for each area, DQN reduces the risk of estimations values spiraling out of control, resulting in a feedback loop of state, reward estimates and actions are taken.

A schematic implementation of the DQN model can be seen in Figure 9.

Figure 9: Deep Queue Network Model.



Source: (LEE, 2018)

3.3 Communication

The setup necessary for the training requires that multiple separate parts work together, such as the communication between the agent and the game, which allows the agent to send commands to the game and receive the current game state.

The SC2client-proto is a Machine Learning API created by the SC2 game creator Blizzard, being an interface that offers full control of the game, exposing all the necessary commands and information for agents, bots and replay analyzers to work.

To facilitate the task of creating agents that play SC2, DeepMind created the PySC2, which consists of a set of tools that exposes SC2client-proto API as a Python RL environment package. It provides an interface for developers to create the agents without the need to use direct commands to the game, offering an easy-to-use wrapper for the RL agents to interact with SC2.

The mini-games being used for the agents training consist of SC2 custom maps that were created with specific sets of rules, and can only be run using the SC2 game client. SC2 maps are created using the Blizzard Map creator tool and all the data related to each map is stored in a '.SC2Map' format file.

4 TRAINING

The training was divided into two groups, defined by the mini-game being played, with each group of tests being composed by sessions of the agent playing the mini-game with both RL algorithms and the learning parameters being tweaked.

Each training session was limited to 100 episodes of the mini-game being played by the agent, focusing the attention on the score value obtained by each set of algorithm and parameters during the session. Each training session data generated was collected as well as the trained model, which allowed the analysis of the score by episode, as well as the use of the pre-trained agent model on side-by-side tests with other agents.

4.1 Agent Setup

The agent has a set of parameters that defines some aspects of its behavior during training, those parameters being independent of the RL algorithm being used. The relevant parameters are the following:

- Discount: reward discount for the agent, can be tweaked to test the reward influence during training;
- Loss Value Weight: how much a loss weight, can be tweaked to test the influence of a loss during the training;
- Entropy: correspond to the spread of action probabilities, low entropy means one dominant action, while high entropy means multiple actions with similar probability. It can be tweaked to test how it affects the agent’s exploration of new actions or strategies.

The agent setup for the training sessions was defined as one setup with default values for all described parameters for both algorithms and one setup with an altered entropy value from 1e-6 to 1e-5 for both algorithms, described in Table 1.

Table 1: Agent Parameters Setup Table

	Default Setup	**Custom Setup**
Discount value	1	1
Loss Weight value	1	1
Entropy value	1e-6	1e-5

Source: from the authors (2019)

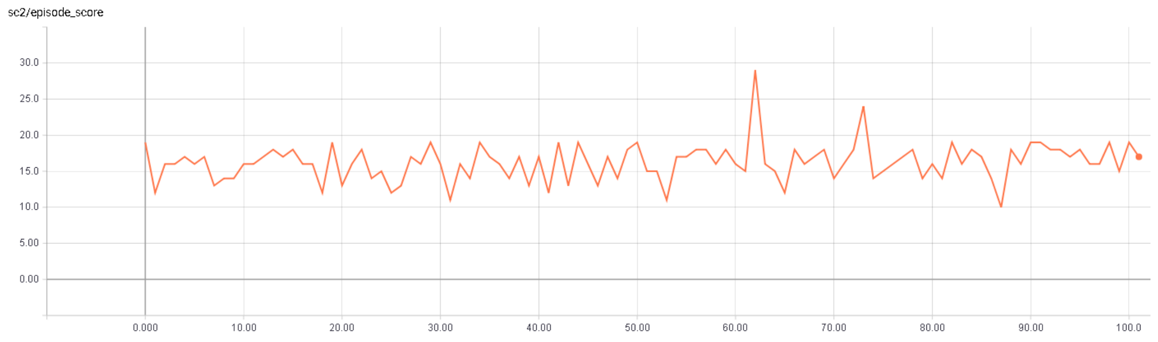
4.2 Results

This section presents all the results collected during the tests, demonstrating the most significant values in each test, resulting in graphs and a table of values for comparison.

4.2.1 Collect Minerals mini-game

The first test was conducted using the A2C algorithm with default parameters. The scores obtained by the agent ranged from 10 to 29, resulting in the graph shown in Figure 10.

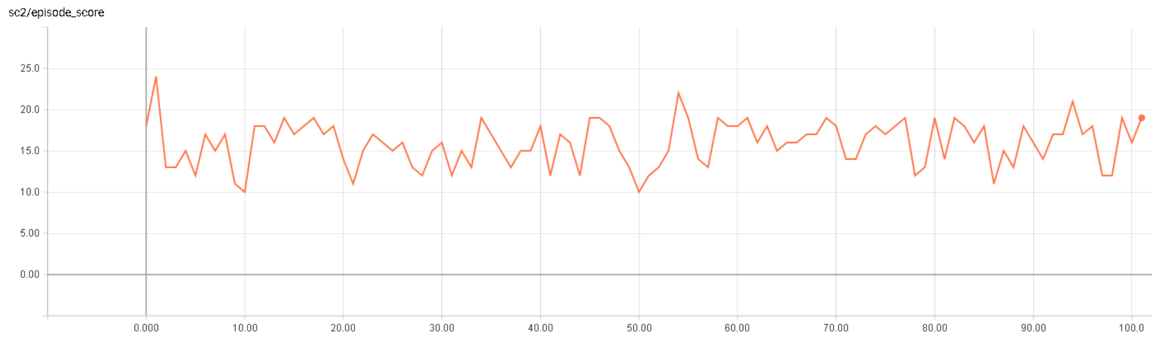
Figure 10: A2C With Default Parameters Results For Collect Minerals.



Source: from the authors (2019)

The second test was conducted using the DQN algorithm with default parameters. The scores obtained by the agent ranged from 10 to 24, resulting in the graph seen in Figure 11.

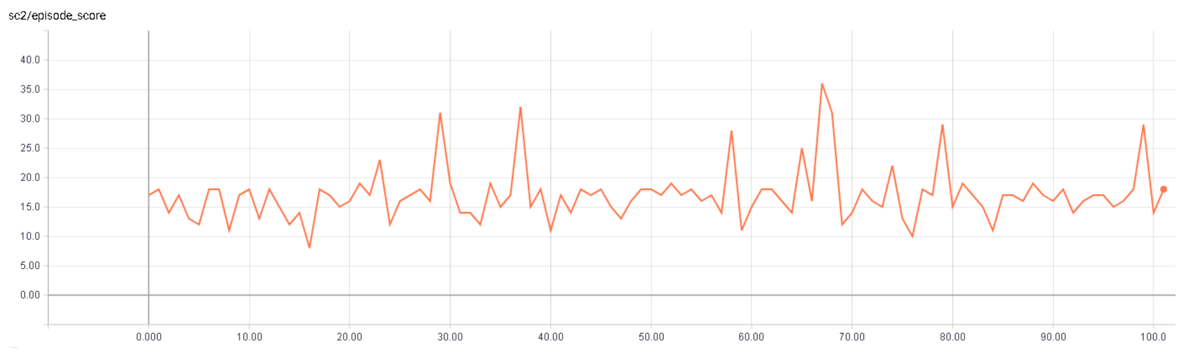
Figure 11: DQN With Default Parameters Results For Collect Minerals.



Source: from the authors (2019)

The third test conducted was the ‘Collect Minerals’ mini-game using the A2C algorithm with custom parameters. The scores obtained by the agent ranged from 8 to 36, resulting in the graph seen in Figure 12.

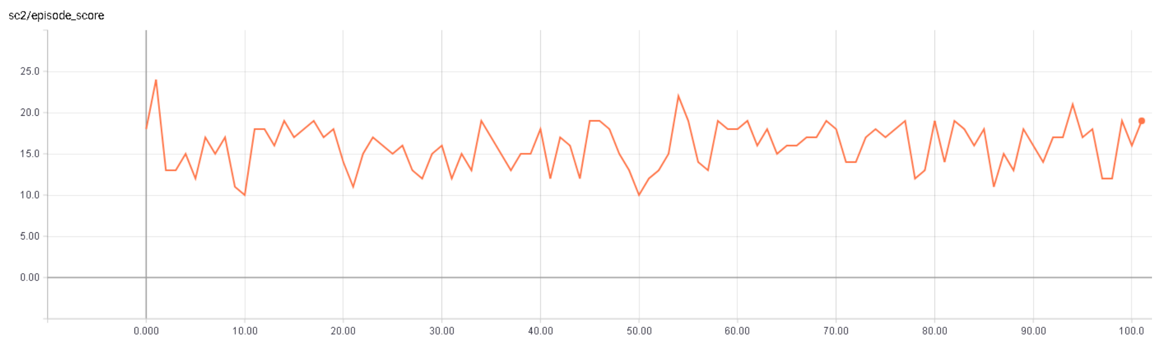
Figure 12: A2C With Custom Parameters Results For Collect Minerals.



Source: from the authors (2019)

The fourth test conducted was the ‘Collect Minerals’ mini-game using the A2C algorithm with custom value parameters. The scores obtained by the agent ranged from 7 to 33, resulting in the graph seen in Figure 13.

Figure 13: DQN With Custom Parameters Results For Collect Minerals.



Source: from the authors (2019)

In Table 2 are presented the lowest, highest and average values obtained by each algorithm during the ‘Collect Mineral’ mini-game.

Table 2: Collect Mineral Shards Results Table.

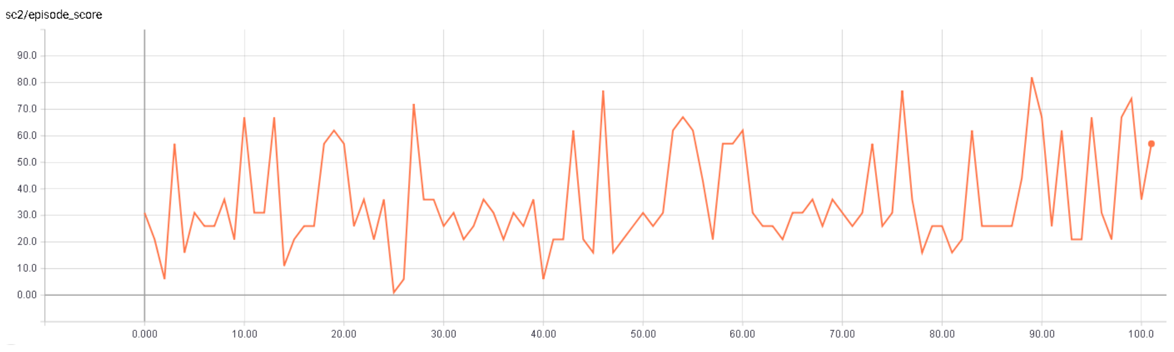
	A2C	**DQN**	**Custom A2C**	**Custom DQN**
Lowest value	10	10	8	7
Highest value	29	24	36	33
Average value	16,18	15,88	17,07	16,65

Source: from the authors (2019)

4.2.2 Defeat Zerglings mini-game

The first test was conducted using the A2C algorithm with default parameters. The scores obtained by the agent ranged from 6 to 108, resulting in the graph seen in Figure 14.

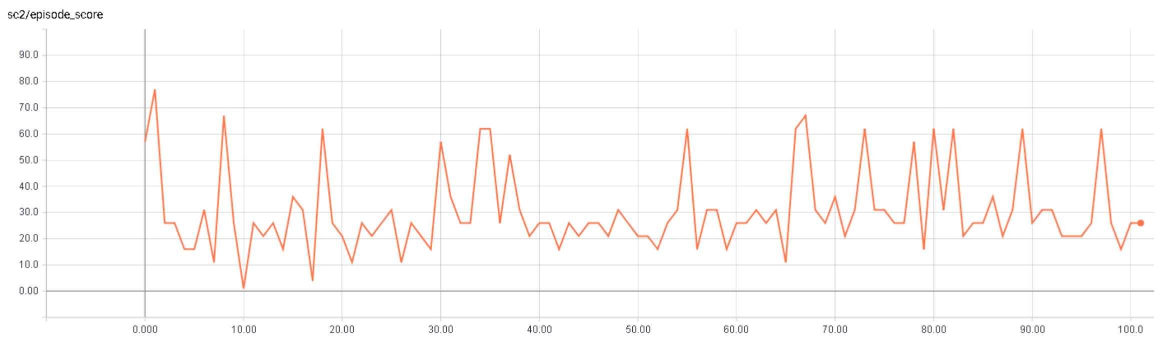
Figure 14: A2C Results For Defeat Zerglings.



Source: from the authors (2019)

The second test was conducted using the DQN algorithm with default parameters. The scores obtained by the agent ranged from 1 to 77, resulting in the graph seen in Figure 15.

Figure 15: DQN Results For Defeat Zerglings.



Source: from the authors (2019)

The third test was conducted using the A2C algorithm with custom parameters. The scores obtained by the agent ranged from 1 to 77. The fourth test was conducted using the DQN algorithm with custom parameters. The scores obtained by the agent ranged from 6 to 93. Both graphs are not shown for brevity.

In Table 4 are presented the lowest, highest and average of values obtained by each algorithm during the ‘Defeat Zerglings’ mini-game.

Table 3: Defeat Zerglings Results Table.

	A2C	**DQN**	**Custom A2C**	**Custom DQN**
Lowest value	1	1	1	6
Highest value	82	77	77	93
Average value	35,21	30,6	28,59	29,69

Source: from the authors (2019)

4.3 Analysis

The results from the Collect Minerals mini-game showed that the A2C algorithm fared better than the DQN on both the default and custom parameters tests. The entropy parameter changes seemed to have influenced the resulting values, resulting in a bigger difference range between the lowest and highest scores, which also resulted in higher score average values.

The results from the “Defeat Zerglings’ mini-game showed similar results for the A2C and DQN algorithm, with the A2C faring better with default value parameters while the DQN did better on the custom ones. The entropy parameter changes drove the DQN scores up and the A2C down, resulting in a bigger difference range between lowest and highest scores, which resulted in lower score average values.

Considering the results obtained, the A2C seems to offer the best results when used for Reinforcement Learning agents running Starcraft II mini-games. The parameter changes during the ‘Defeat Zerglings’ that resulted in the DQN doing better at that test, point to the notion that a finer tweak in the parameters can influence the final results, for best or worst.

The test consisted of a total of eight training sessions, four on each mini-game, being two with each algorithm, one with default parameters and one with custom parameters. Each Collect Minerals training session took close to one hour, while the Defeat Zerglings varied between 30 to 50 minutes because the episodes could end sooner if the agent lost all its units.

The machine used for the tests was a low-end desktop configuration, with an AMD Phenom II 955 running on 3.200MHz clock and using 8Gb of DDR3 ram running on 1333Mhz. Taking into consideration the computer specs, it’s reasonable to affirm that the time taken for the test could be considerably lower using a more computationally powerful machine.

5 CONCLUSION

This work has briefly analyzed the current state of two Reinforcement Learning algorithms applied to a complex and modern computer strategy game. Tests were conducted using default and custom agent parameters, offering some preliminary answers about how each algorithm would achieve success on each mini-game and how the parameters might affect them.

While each test session only amounted for a small number of episodes, we confirmed that changes in the parameters and the algorithm being used on each mini-game do significantly affect the results, even without the agent achieving a convergence point.

The entropy being selected to be the changed parameter showed us that it can have a heavy influence on the results, leaving open the question about how the other parameters would affect the tests.

Related works focus on convergence, comparing algorithms and the time required for each to achieve the convergence point, while this work focused more on fine comparison of algorithm results and parameters on a smaller episode range of episodes.

5.1 Future Work

This work used a reduced amount of episodes for each training session, that decision forces the analyses of the results to look for sharp changes in behavior based on the parameters and algorithm being tested.

Future works that enjoy the benefits of dedicated servers with hardware-specific for mathematical calculations could extend the training sessions by a large amount, being able to achieve convergence in results as the agents learn the best strategies for each mini-map

Another interesting task is the study of Reinforcement Learning algorithms that were not tested in this work, followed by the challenge of training agents to play the whole game, which would require it master all of the tasks previous seen individually in the mini-games.

REFERENCES

ADAMS, R. **Ten powerful examples of Artificial Intelligence in use today.** Available at: <https://www.forbes.com/sites/robertadams/2017/01/10/10-powerful-examples-of-artificial-intelligence-in-use-today/>. Accessed on October 23 2017.

ATTICK, R. **Intelligent Things:** It's all about machine learning. Available at: <https://www.linkedin.com/pulse/intelligent-things-its-all-machine-learning-roger-attick//>. Accessed on October 22, 2017.

DOMINGOS, P. **The Master Algorithm:** How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books, 2015.

LEE, W. **Deep Q Networks.** Available at: <https://dnddnjs.gitbooks.io/rl/content/deep_q_networks.html>. Accessed on June 14, 2018.

MAO, H. **Reinforcement Learning using Asynchronous Advantage Actor Critic.** 2017. Available at: <https://medium.com/@henrymao/reinforcement-learning-using-asynchronous-advantage-actor-critic-704147f91686>. Accessed on June 14, 2018.

MILLINGTON, I. **Artificial Intelligence for Games.** Morgan Kaufmann Publishers, 2009.

NARULA, G. **Everyday Examples of Artificial Intelligence and Machine Learning.** Available at: <https://www.techemergence.com/everyday-examples-of-ai/>. Accessed on October 22, 2017.

NIELSEN, M. **Using neural nets to recognize handwritten digits.** Available at: <http://neuralnetworksanddeeplearning.com/chap1.html>. Accessed on October 25, 2017.

OPENAI. **GYM: A toolkit for developing and comparing reinforcement learning algorithms.** Available at: <https://github.com/openai/gym/>. Accessed on October 25, 2017.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence – a modern approach**. Prentice-Hall, New Jersey, 1995.

SHAIKH, F. **Simple Beginner's guide to Reinforcement Learning & its implementation.** Available at: <https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/>**.** Accessed on October 25, 2017.

VINYALS, O., et al. **Starcraft II: A new challenge for reinforcement learning.** arXiv preprint arXiv:1708.04782 (2017).