Research Article

An Efficient Agent Created In Starcraft 2 Using Pysc2

N. Meenakshi[a], Abhishek Y[b], Daniel Li James[c], Thamanraj Y[d]

[a,b,c,d] Department Of Information Technology Hindustan Institute of Technology and Science, Chennai, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: This paper introduces "Judy"(Bot/Agent that is developed using reinforcement learning algorithms using the deep mind toolset PySc2 within StarCraft II. The domain reinforcement learning is a unique uncharted field clustered with difficult problems than worked on in previously done research. By observing the actions and then associating them with reward specs within the SC2 domain we aim to supply an open Python-based GUI which consists of an In-house engine to interact with humans. Adding up to the aforementioned challenges the agent must tackle multiple game maps, using the set of mini-games that specialize in totally different parts of StarCraft II gameplay. The agent relies on game replays of humans who are skilled. By offering a starting or base result for a neural network trained by the acquired info we tend to predict the final outcomes and actions of the players. Thus, the agent experiences a brand new difficulty surrounding the environment exploring deep reinforcement learning algorithms.

Keywords:

1. Introduction

The prospect of deep-learning-based non-linear function approximation using neural networks has been attributed to many significant advances in the areas such as image processing, home automation, and others. These methodologies and approaches have been demonstrated successfully in RL [1] problems such as games like Atari, Go, 3D environments of virtuality, and simulation of robotics-related aspects. Using the benchmarking tools provided by the PySC2 deepmind's toolkit, the performance of the agent can be measured and consequently advancing deep learning and reinforcement learning (RL) testing. The video game namely Starcraft 2 focuses on real-time strategy (RTS) that demands high-level planning and execution for the fast-paced micro-actions. SCII provides an exceptional opportunity to traverse numerous challenging unexplored boundaries in RL for instance numerous participants competing for influence, authority, and resources while managing a number of units. Making use of this data toolkit will supply a standard specification to experiment not only with state-of-the-art RL algorithms but also compelling and riveting characteristics which are active areas of machine learning research. Various immersive environments are at their disposal for reinforcement learning techniques in StarCraft. This makes the StarCraft environment ideal research for exploring deep reinforcement learning algorithms with its interesting set of game-play properties and large player base.

2. Related Works

Alongside the advancement of computing, furthermore, human players who play a pivotal part of the environment associated with SC2 are Bots which are virtually intelligent agents. Yearly contests are held in the course of which many bots partake in order to determine and judge, consider the undemanding and most ideal ones. So far, the foremost outcomes have been attained by scripting bots that accomplish the scheme prepared by the programmer. Nonetheless, this modus operandi is gradually lessened owing to present-day growth and evolution in the surroundings of acquiring info with RL which has made it attainable to produce likely ideal bots such as James Bot [2]. The research and experiments led to the conclusion that a novice player can be defeated by our built-in AI-based Starcraft bots. Henceforth a plethora of breakthroughs has been made in the research of Reinforcement Learning, further advancing in the creation of efficient bots. Similarly, there are other bots mentioned in James Bot's research namely TstarBot and Alphastar which are very advanced and use concepts like reinforcement and Deep Q.

By conducting research on electronically manipulating images produced by a computer program are used in the real world to carry out simulations or theoretical scenarios. Thus, evolving artificial intelligence populating a simulated reality with agents both controlled by humans and Artificial Intelligence. By experimenting with coherent and robotic techniques on video games makes it a good middle ground. The unfathomable complexity of AI pushes researchers to test various approaches than a traditional board game(Go, Chess, Risk). The game AI's complexity can be dealt with with a feasible framework known as the Bayesian model [3]. The reactions of all the units and their controls are observed along with how objectives are taken from a tactical model. The opponent's ideas and tactics are then inferred using the knowledge of strategic prediction.

The present-day ultimate performing schemes and programs to play RTS games such as SC2, even so, depend for the most part on hard-coded approaches. However, it is attainable to carry out strongly or even accomplish by

winning, one of these competing events plainly by discovering a hard-coded plan of action that no other bot has a pre-planned technique to counter the same. Furthermore, skilled individual players are yet distinctly better to the foremost programmed scripts. Looking at it from an industrial perspective, a sole added challenge is to recreate the authentic behavioral patterns of a human player with an opposing bot and hence resulting in an entertaining match. The aforementioned parameters can be achieved by using a game-planning algorithm that learns to play a game by repeatedly playing with deductive reasoning and deep learning using human game replays. The FSM[4] also known as the Finite State Machines is used to break down the actions performed by the AI into different states. The FSM is controlled hierarchically into reasonable states such as an attack, repair, and boundaries that take place in the game. The use of a hard-coded approach has attained a noteworthy size of advancement, and, as we further talk about it, academically researched RTS A. I systems has also made the use of. Nonetheless, these hard-coded techniques wrestle and suffer to encode dynamic, changing adaptive behaviors, and are undoubtedly exploited by adaptive and dynamic opponents.

The reinforcement learning methods which are widely used need an extremely large dataset of samples retrieved in a specific environment. The optimal policy is only achieved by the agent performing stably in Starcraft II using the trial and error method while simultaneously computing the Huge state space and large action space. This results in the research of tackling complicated issues such as weak exploration and a multitude of oscillations. Increasing the sample efficiency while coming up against the obscured states using expert affirmation in a video game is required. Other studies have shown the benefits of expertise replay in getting higher performance, however, an additional price for storing the experiences is needed. To counter a similar work has applied an ascendable distributed off-policy method–IMPALA [5] to find out the mini-games of StarCraft II. Then by incorporating Self-Imitation Learning (SIL) into antelope to choose up past helpful experiences, that's just like knowledgeable demonstrations to realize higher exploitation, and so drive deeper exploration and improve sample potency. By the tip of it, the agents trained by antelope+SIL are anticipated to attain a similar performance because the one in every of IMPALA with fewer trajectories generated from SC2LE surroundings. Moreover, the neural spec shows its comfortable talents to capture the underlying info from the observations with fewer parameters.

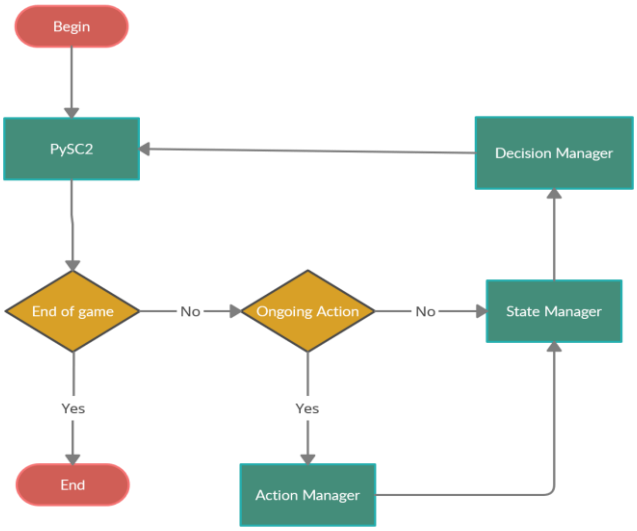


Fig.1: Block Diagram

Based upon ruled based method and learning-based method which belong to the hierarchical architecture there has been research done to create this architecture which is called Build Marines which consists of huge action spaces and observation spaces along with a long time horizon. Build Marines exhibit properties that follow the proper arch of hierarchy along with the rule-based method [6] used as per the level of work. Low-level work is being used for rule-based methods whereas learning-based has a high level. By making use of these methods, unlike other methods which have achieved a successful accomplishment in SC2, Build Marines focuses on hierarchy over a smaller scale like those of mini-games. This is done so as to inspire and encourage the exposure of micro-operation. To achieve outstanding performance in RTS the use of hierarchy at different levels plays a pivotal role.

3. Q Learning Algorithm

The domain of the agent can be portrayed as Markov Decision Process(MDP) containing all the sets of actions, states, transitions, and their reward function. The future state can be predicted and depends upon the

current state of the agent while carrying out all the prerequisites of Markov's process. The main objective of Reinforcement Learning is to identify the perfect set of tasks for the selected Markov Decision Process.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - Q(s_t, a_t) \right)}^{\text{temporal difference}}$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}} \qquad \underbrace{\qquad}_{\text{old value}}$$

Fig.2: Formula for Q Learning

Qπ(S,A) = Q(S,A)

S - Current state

A - Action performed by the agent

π - Specified strategy The expected reward policy [5] is based on the above reward function determining the action performed by the agent. The Q value obtained from the process is determined by both the state and action performed by the agent.

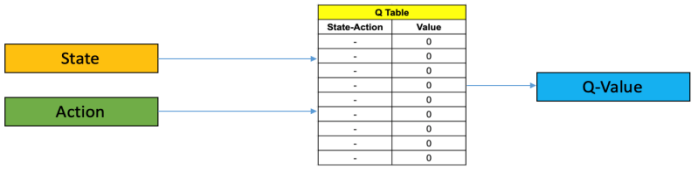
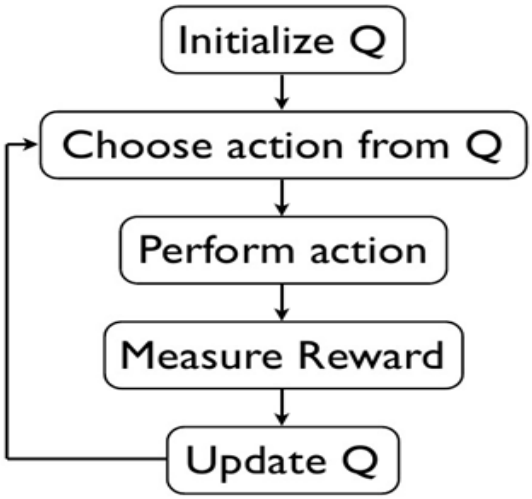


Fig 3: Q Learning

To specify the reward policy Q, one can make use of the Q-Learning Algorithm while performing multiple ilterations while initializing the parameter of the state and action of the training agent.



We create a Q-table for an environment with the dimensions SxA, where S and A are the numbers of states and actions, respectively. There are actions for each state, and the likelihood of selecting a specific action is determined by the values in the Q-table known as the state-action value. The values of the Q-table are initially set to 0. For each state, an action is selected. The Q-value for the state-action is increased if that action provides a good reward for the next state; otherwise, the Q-value is reduced.

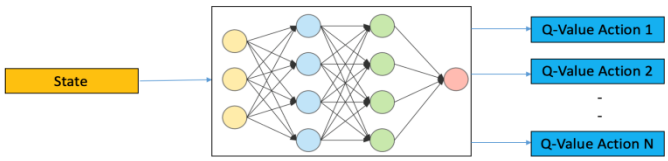


Fig 4: Deep Q Learning

The value of Q(S,A) ought to be concurrent with the desired and expected value, i.e. the agent needs to have a plan of action that should approach the optimal and ideal one, with the play of events that are a finite sequence in the Markov Decision Process.

The bot was programmed in a computing environment that included the Python programming language and the PySC2 library, similar to the StarCraft game.

II Learning Environment.

Fig 5: Starcraft 2

The bot was plotted and sketched in form of remote elements and subsystems that attain separate functions:

1) For processing the input data or IP we use State Manager or SM,

2) The selection of its state is established on the ground by the state of S along with its function which is Q(S, A) that is responsible to develop and progresses in the phase of learning. This is where the Decision Manager or DM acts by carrying out the functions.

3)The decisions taken in Decision Manager are processed by Action Manager or AM for environmentally understandable action in SC2. This grading and scale clearly stipulate the discrete phases of the entire procedure or operation. During the emergence of the environment of StarCraft2 by implementing its algorithm that resettles authority to its agent. Furthermore, two conditions are examined. The initial condition examines if the environment has landed in its final state. In the final state, every action has already been implemented by it. It carries onto the secondary conditions where they examine the bot for its performance in the selected action manager. Here the bot is passing the control either to the Action manager or MS if it is not transferring.

Fig 6: Training Judy

If there emerges a circumstance wherein no action or steps are implemented by the bot, the monitored information from the environment is then obtained and forwarded to the SM module[6]. Followed by which particulars filtered are sent to the DM or Decision Manager from the subsystem. The Decision Manager or DM proceeds to then carry out the Reinforcement Learning algorithm by reconditioning the functions in it and also selecting new actions. The most ideal steps are then selected and are further carried out by Action Manager or AM, and then orders are sent to the environment.

Other works carried out on generating optimal algorithms to simulate fights in SC2 have been done with multiple objectives in the plan[7]. The absence of a free application programming interface that is available to control units in the game directly had raised the norm to come up with an alternative. Parameter values to be found that control the units of each individual to maximize the damage caused to two players who are opposite each other in a competitive match.

Fig 7: Agent performing tasks

III Defining Reward

The concept of rewards is introduced to the agent determined by the task performed. A lot of RL systems restrict the learning of systems according to their rewards action to 1 and 0 for winning and losing respectfully. Any action can be rewarded as per our will. Implementing a reward system for killing units or destroying buildings is introduced.

KILL_BUILDING_REWARD = 0.5

KILL_UNIT_REWARD = 0.2

So as to achieve a confirmation about the successful killing of a unit or its building we make use of a scoring system that is cumulative in nature. Since the scoring system is gradually increasing and step by step there arises a need to continuously track and note the previous step values in order to differentiate with the currently calculated values. An increase in values indicates the killing of entities.

4. Simulations and Analysis

Upon analyzing the outcome of various micromanagement stages and scenarios we are able to make certain conclusions and discussions about the RL model and its execution. Considering the leftover scenarios the initial scenario is taken up as the starting point of all units to train them and in the further structure, we make the use of the DQL method which is the Deep Q method implemented to measure the fight and combat in substantial scenarios.|| Purpose and target of SC2 micro-management consists of conquering the enemies in various difficult plots and frameworks. Understanding the entire scheme better by analyzing the ratios and percentages of winning and losing the number of steps made by episodes, and the avg. rewards achieved during the implementation at training including all the new strategies learned.

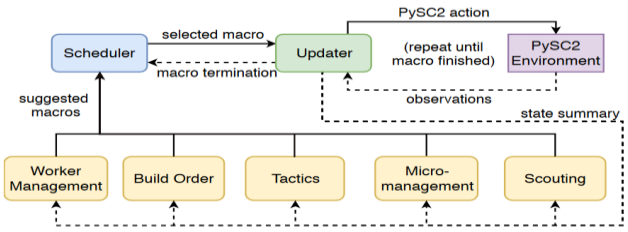


Fig 8: Micromanagement

We have the ability to choose an action of the bot or its gameplay regardless of the actions performed on the battlefield with equivalent probability. The end goal for the random bot to achieve a win was not possible as the strategy ended up being very simple which caused the bot 20% and 90% of draws and losses respectively. After completion of approx. 200 episodes the testing of the game was stopped and was checked for effectiveness where it was found to have stopped changing.

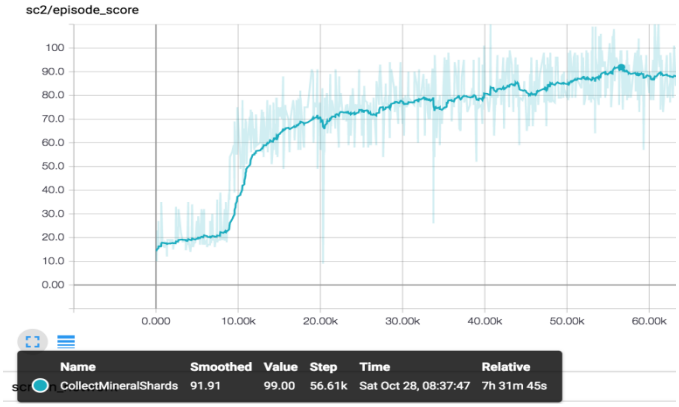


Fig 9: CollectMineralShards

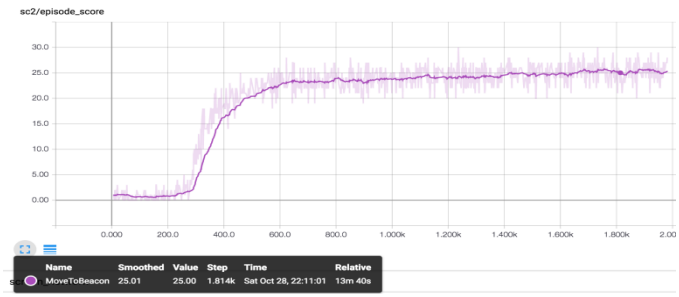


Fig 10: MoveToBeacon

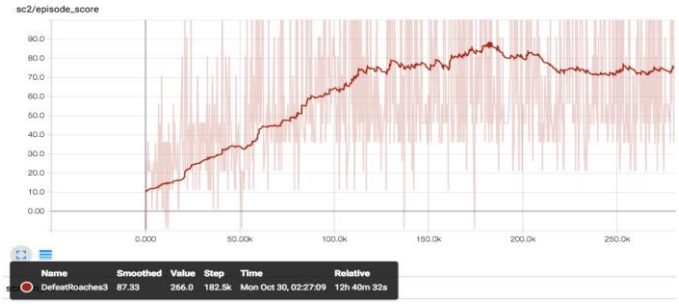


Fig 11: DefeatZergs

Taking the repo into consideration we make use of the default guidelines and parameters other than:

- DefeatRoaches, DefeatZerglinsAndBanelings entropy_weights 1e-4/1e-4, n_steps_per_batch 5 Number of envs 32 or 64
- From the Fullyconv strategy and policy, the DeepMind scores are exhibited in the release paper to check differences, in contrast, and compare.
- The aim of the model to learn successfully collecting minerals and gas or to build marines was a failure.

5. Conclusion

Using the Q-learning technique, Judy Bot is capable of acquiring superiority above a reasonably undemanding random automated bot. It was also able to achieve a reasonably good performance in terms of it being able to show its capability and winning skills. Q Learning technique with currently available avant-garde is fairly effortless to carry out in different sectors of languages. The demand required for resources is very less along with computing energy and hence it also paves the way for the usage of less memory

References

1. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT press, 2018.
2. Z. Kowalczuk, J. Cybulski and M. Czubenko, "JamesBot - an intelligent agent playing StarCraft II," 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 2019, pp. 105-110, doi: 10.1109/MMAR.2019.8864611.

3. G. Synnaeve and P. Bessière, "Multiscale Bayesian Modeling for RTS Games: An Application to StarCraft AI," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 8, no. 4, pp. 338-350, Dec. 2016, doi: 10.1109/TCIAIG.2015.2487743.
4. S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill and M. Preuss, "A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 5, no. 4, pp. 293-311, Dec. 2013, doi: 10.1109/TCIAIG.2013.2286295.
5. Z. Hu and T. Kaneko, "Application of Deep-RL with Sample-Efficient Method in Mini-games of StarCraft II," 2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), Kaohsiung, Taiwan, 2019, pp. 1-6, doi: 10.1109/TAAI48200.2019.8959866.
6. T. Liu, X. Wu and D. Luo, "A Hierarchical Model for StarCraft II Mini-Game," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 2019, pp. 222-227, doi: 10.1109/ICMLA.2019.00042.
7. Schmitt, Jonas & Kostler, Harald. (2016).“A multi-objective genetic algorithm for simulating optimal fights in StarCraft II”. 1-8. 10.1109/CIG.2016.7860422.
8. Richoux, F., Uriarte, A., & Baffier, J.-F. (2016). ghost: A Combinatorial Optimization Framework for Real-Time Problems. IEEE Transactions on Computational Intelligence and AI in Games, 8(4), 377–388. doi:10.1109/tciaig.2016.2573199
9. Certicky, M., Sarnovsky, M., & Varga, T. (2018). Use of Machine Learning Techniques in Real-Time Strategy Games. 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA). doi:10.1109/disa.2018.8490528
10. Andersen, P.-A., Goodwin, M., & Granmo, O.-C. (2018). Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games. 2018 IEEE Conference on Computational Intelligence and Games (CIG). doi:10.1109/cig.2018.8490409
11. Hu, H., & Wang, Q. (2020). Implementation on benchmark of SC2LE environment with advantage actor – critic method*. 2020 International Conference on Unmanned Aircraft Systems (ICUAS). doi:10.1109/icuas48674.2020.9214032
12. Hsieh, J.-L., & Sun, C.-T. (2008). Building a player strategy model by analyzing replays of real-time strategy games. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). doi:10.1109/ijcnn.2008.4634237
13. CHEN, L., LIU, T., & LIU, Y. (2020). Research on the StarCraft ii Decision Method Based on Hierarchical Reinforcement Learning. 2020 Chinese Control And Decision Conference (CCDC). doi:10.1109/ccdc49329.2020.9164140
14. Michail Tsikerdekis ; Sean Barret; Raleigh Hansen, Efficient Deep Learning Bot Detection in Games Using Time Windows and Long Short-Term Memory (LSTM) , IEEE Access , DOI: 10.1109/ACCESS.2020.3033725
15. X. Fang, P. Cui and Q. Wang, "Multiple agents cooperative control based on QMIX algorithm in SC2LE environment," 2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS), Guangzhou, China, 2020, pp. 435-439, doi: 10.1109/ICCSS52145.2020.9336865..