2020 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. September 1-4, 2020

Implementation on benchmark of SC2LE environment with advantage actor – critic method *

Huan Hu, Qingling Wang

Abstract— **Deep reinforcement learning has already surpassed human performance in many video games, which is mainly achieved with reinforcement learning algorithms based on the actor-critic framework. With the release of PySC2 reinforcement learning environment by Google DeepMind and Blizzard Entertainment, deep reinforcement learning algorithms have attracted the attention of many AI developers on StarCraft II games. In this paper, we used the advantage actor - critic algorithm to achieve the training of agents on seven mini-maps released by DeepMind, we obtain the experimental results and compare them with DeepMind's experimental benchmark. The experimental results show that the agent trained based on the advantage actor-critic algorithm has excellent performance on SC2LE environment.**

I. Introduction

In recent years, deep learning technology has achieved great achievements in the areas of speech recognition, computer vision, and natural language processing. This has further promoted the development of deep learning technology, and also provides a powerful function approximator, that is, depth neural networks. Deep reinforcement learning algorithms and have achieved great success in some areas, such as games. Considering the role that games play in the formation of animal intelligence, it is not difficult to understand that artificial intelligence agents are developed and evaluated based on the game environment. From the first attempt of artificial intelligence in checkers [1] to today's deep reinforcement learning algorithms have exploded in some traditional games, such as classic board games based on Go [2], such as the electronic video game Atari platform [3], and Doom [4]. In particular, Google DeepMind has been actively promoting the application of machine learning in the game field, and has made great achievements in the field of Go. Later, in collaboration with Blizzard Entertainment, the StarCraft II learning environment (SC2LE) was released: an easy-to-use set of tools and libraries that can be used to connect to StarCraft II games and allow developers to research and train reinforcement learning algorithms. StarCraft II is a real-time strategy game (RTS) that combines micro-operations with high-level planning and execution requirements. Over the past 20 years, the game has attracted tens of thousands of casual and highly competitive professional players. Beating top players has long-term significance. In addition to SC2LE DeepMind also published a paper describing the structure of their RL agents. This is an end-to-end architecture. Agents learn the rules through input data. This process is similar to the perception of human players and can choose an action from the same action options as a human player [5]. In their work, the agents were required to conduct evaluation tests on a series of mini-games maps and record the experimental results as a benchmark for further research. DeepMind provides a benchmark and comparison for the evaluation and exploration of the RL algorithms.

*This work was partially supported by the National Natural Science Foundation of China (61973074), the National Key Research and Development Program of China (2018AAA0101400), the Science and Technology on Information System Engineering Laboratory (05201902), and the Fundamental Research Funds for the Central Universities.

The authors are with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: qlwang@seu.edu.cn).

In this paper, we explained the complexity of the StarCraft II environment. The huge action space and state space of the StarCraft II lead to many difficulties in training agent directly on the complete game platform, so we chose seven mini-maps released by DeepMind to test the effect of the advantage actor-critic (a2c) algorithm. we analyzed the experimental results and compared it with the DeepMind benchmark. Which proved that the learning agent can show excellent behavior under the training of a2c algorithm. In the end, we draw a conclusion of the paper and propose some future work.

II. ENVIRONMENT AND BACKGROUNDS

A. *Environment*

The StarCraft II (SC2) is a multi-agent environment. Players and players compete with each other, compete for resources, and expand their sphere of influence as much as possible, so this is different from the environment studied in previous reinforcement learning work. There are several characteristics of the game environment: First, the action space and state space are huge, and there are many types of buildings and combat units, and each unit has its own local action space, making the observations complex and diverse. Second, the problem caused by incomplete observation information is that the player can only see the local information through the local camera, but to master the global information requires moving the camera, and the player must actively explore the state of the opponent on the map other than the local camera. Finally, the reward value response can be slow, often requiring thousands of steps of long-term strategy to complete.

To be able to interact with the game environment in a programmatic way, we use the PySC2 library. This is a Python open source package specifically used to interact with SC2. It exposes a list of spatial and non-spatial features, including some actions and observation benchmarks, and it also contains several sets of mini-game maps, such as moving units to the target location to defeat enemy forces and collection of mineral resources (Figure 1). As a visualization tool, you can view the behavior of the agent, and you can view the complete specifications of input functions and operations online: https://github.com/deepmind/pysc2/blob/master/docs/environment.md

Authorized licensed use limited to: Middlesex University. Downloaded on October 20,2020 at 05:02:52 UTC from IEEE Xplore. Restrictions apply.

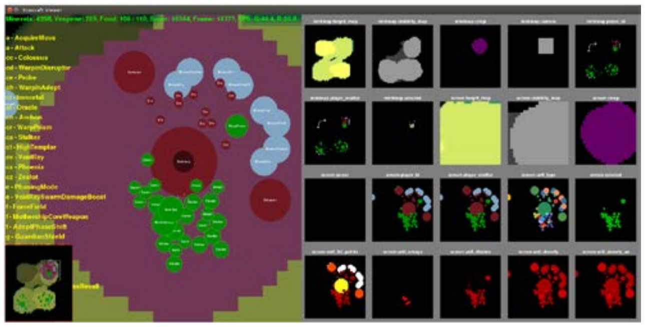

Figure. 1 An example of PySC2 environment view. On the left is a simplified game engine GUI where humans can see what is happening in the game. On the right are the actual observations received by the AI agent, including units, types, and structures.

B. *Reinforcement Learning*

The idea of reinforcement learning comes from psychology [6], that is, when an animal is stimulated in a given environment, it repeats some desired behavioral patterns. The application of this idea in computational science has formed the field of reinforcement learning [1]. Therefore, reinforcement learning can be applied to solve the multi-agent collaboration problem in StarCraft II. Reinforcement learning agents constantly interact with the environment, repeatedly trial and error to learn and converge to an optimal strategic behavior [7]. Figure. 2 depicts the classic reinforcement learning paradigm, showing the process of interaction between the agent and the environment.

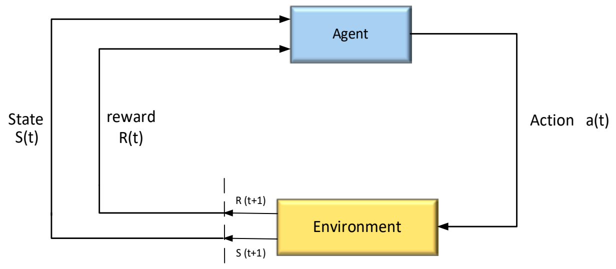

Figure. 2 Demonstration of interaction process between agent and environment in reinforcement learning.

The interaction process between agent and environment in reinforcement learning is modeled as a Markov Decision Process (MDP). The goal of reinforcement learning is to train a strategy , which can map states to executable actions, and continuously improve strategyπ by maximizing a goal reward function, which is generally set as $J(\pi)=E\left(\sum_{t=1}^{N}\gamma^{t-1}r_t \mid \pi\right)$ and called expected cumulative discount reward function, where the discount factor $\gamma \in [0,1)$, r_t is the reward the agent gets after taking action a_t at state s_t. Policies π can be either stochastic $\pi(a \mid s)$ or deterministic $\pi(s)$, where $\pi(a \mid s)$ means that each action $a \in A(s)$ an agent can perform at state s satisfies a probability distribution and $\pi(s)$ directly maps an state s to a determined action a. In general, we use gradient descent algorithm to optimize the policy weights θ. The policy $\pi(\theta)$ is commonly, a deep or shallow neural network. from [8], the most commonly used form of the gradient approximator is:

$$\hat{g} = \hat{E}\left[\nabla_\theta \log_{\pi_\theta}(a_t \mid s_t)\hat{A}_t\right] \qquad (1)$$

where $\hat{A}_t$ is called the advantage function, whose form is $\hat{A}_t(s_t,a_t) = Q(s_t,a_t) - \hat{V}(s_t)$, representing the empirical average of a limited number of samples generated during the interaction between agents and the environment at each time step t. Here $Q(s_t,a_t)$ is the state action value, which is used to measure the score value of the action a_t in the s_t state, and $\hat{V}(s_t)$ is the state value, which is used to measure the quality of the s_t state. The estimator was obtained by differentiating the objective function:

$$L^{PG}(\theta) = \hat{E}\left[\log_{\pi_\theta}(a_t \mid s_t)\hat{A}_t\right] \qquad (2)$$

This approach, which can be applied to deep learning, makes reinforcement learning tasks look more like general optimization problems [9]. In this work, we use the a2c algorithm to optimize the policy in a more stable and efficient way by seeking to improve the performance of reinforcement learning agent.

III. LEARNING MODEL AND METHOD

In our work, we use the a2c algorithm in StarCraft II environment to generate experimental results and compare with the standards given by DeepMind.

A. *Model Architecture & Challenges*

Reinforcement learning has several challenges in the application of StarCraft II game environment. First, unlike Atari games [10], which either completely use spatial features like images or completely use non-spatial features, in the StarCraft II environment, agents need to consider both spatial and non-spatial features. Second, the agent's action space in the game is huge, and each space action has thousands of basic operations. The huge state space and action space is a major challenge for the learning of agent strategy parameters.

The agent model structure in this paper mainly refers to the FullyConv network structure in the SC2LE paper, as shown in figure 3 below. FullyConv model has the characteristics of full convolution and maintaining resolution. The StarCraft II game mainly relies on the mouse to click on the floor to control the directional behavior of the agent. The input of this model is three independent modules: screen spatial data and mini-map spatial data. These two input modules mainly contain information seen on the screen or mini map (such as unit type and health status). The third module is non-spatial information, which is additionally available to the agent, such as the number of enemies. Then these three input modules are

Authorized licensed use limited to: Middlesex University. Downloaded on October 20,2020 at 05:02:52 UTC from IEEE Xplore. Restrictions apply.

integrated into a three-dimensional tensor to represent the current state, and then transformed into three output modules via a neural network, which are spatial strategy, value estimation and non-spatial strategy. The spatial strategy is obtained from the current three-dimensional tensor through the 1 □ 1 convolution layer and the output filter. The spatial strategy indicates where the action should be performed on the map. The non-spatial strategy and state value estimation part is that the 3D tensor first passes through a common fully connected layer (FC) network, outputs the state value estimation, and then passes through a fully connected layer network to output the non-spatial strategy. The non-spatial strategy indicates what actions the agent should take. This means that to properly select the strategy parameters, the input status should contain enough information.

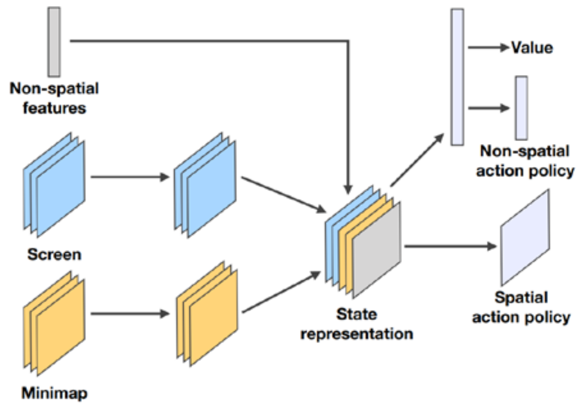

Figure. 3 FullyConv architecture as described and illustrated in [5].

B. *Reward Structure & Observations*

- **Reward structure:** The agents to be trained play against the built-in agents in the environment. The built-in agent is written based on the rules and has ten difficulty levels. The trained agent needs to meet several requirements to win the built-in agent. First, the agent needs to accumulate various resources (such as gas and crystal ore), then construct buildings, produce various arms and troops, and finally destroy opponent's buildings and barracks. There will be two reward value mechanisms in the game. One is a ternary signal (+1 (victory), 0 (draw), -1 (defeat)) at the end of the game. It will feed back to the agent. The other is the Blizzard score that is closely related to the performance of the agent in the game. For example, the agent collects more resources to build more buildings, which will increase the Blizzard score. On the contrary, the more the building is destroyed by the enemy, the lower the Blizzard score is. The two different reward mechanisms have different effects on the training process of the agent. The ternary signal is relatively sparse, and the Blizzard score can be used as a continuous feedback of the agent's performance.

- **Observations:** Observations are mainly composed of feature layers. These feature maps are represented with m□ n pixels (m and n are configurable) and contain specific information, such as building types and map visibility. There are two types of feature maps, one is a mini-map, which is a rough rendering of the overall map environment, and the other is screen information, which is a specific game information presentation in the field of view of the camera, including building types and numbers. DeepMind has released seven kinds of mini-maps for testing agents to perform different tasks. At the same time, the observation also contains non-spatial information such as the amount of resources and the set of executable actions.

C. *Action Policies*

The space for action in the StarCraft II game is huge, including left mouse button and right mouse click, left click to select the unit type, right click on the floor to select the map location. At each time step, the agent will face an action list, which is an action identifier and parameters provided by the game environment to the agent. For example, clicking the left mouse button on the screen is the action identifier so that the coordinates clicked on the map can be used as parameters. There are 524 action identifiers and 13 parameter types.

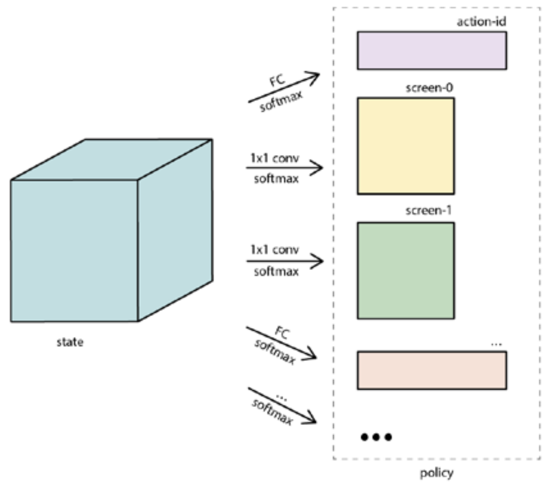

Figure. 4 Action selection process. The status module is transformed into spatial strategy output and non-spatial strategy output. The output is a probability distribution, which contains the action ID and the pixel position to be clicked.

In StarCraft II, the actions that can be performed in different screen states are different. For example, after selecting a unit, only the move instruction is valid. In order to simplify the difficulty, we consider the action identifier and parameter selection to be an action step, and then select the action through a joint probability distribution. The merged state tensor outputs the spatial strategy after passing through the convolution layer, and the non-spatial strategy is obtained after the FC fully connected layer. Finally, the model output is converted into a probability distribution through a softmax layer, as shown in Figure 4. The agent's action selection can be obtained from the probability distribution of the result. After training with the neural network, the agent remembers the actions that need to be collected, discards the unavailable actions, and sets the collection probability of the unreasonable actions to zero. Finally, we then re-normalize the policy probabilities so that the sum of the probabilities is 1.

Authorized licensed use limited to: Middlesex University. Downloaded on October 20,2020 at 05:02:52 UTC from IEEE Xplore. Restrictions apply.

D. Advantage Actor-Critic Algorithm

A2C is a classic actor-critic algorithm, in which strategy and value estimation are trained in a self-looping manner. In the a2c algorithm, the advantage function is used instead of the value function to evaluate the critic part, and then the actor part is updated according to the advantage function of critic. The advantage function is updated and changed by the data obtained by the interaction between the agent and the environment. It uses a neural network as a non-linear function approximator for strategy and state values, which significantly improves the learning ability to complex environments. In the a2c algorithm, the model parameter content is stored on the server side, while the proxy client only interacts with the environment to collect training data. Because the model can be calculated on the GPU hardware device, the a2c calculation speed is very prominent. The use of gradient descent to introduce a separate strategy entropy and maximize it, the specific loss function is as follows:

$$J(\theta) = E_\tau\left[\log \pi\,(a_t|s_t;\theta)\hat{A}\left(s_t, a_t + \left(R_t - \hat{V}(s_t;\theta)\right)^2 - \pi(a_t|s_t;\theta)\,\log\left(\pi(a_t|s_t;\theta)\right)\right)\right]$$

Algorithm 1 ***Advantage Actor-Critic algorithm***

```
Input: learning rate $\alpha$, number of updates $T_{\max}$,
number of n-steps $t_{\max}$
    while $T < T_{\max}$ do:
        $t \leftarrow 0$
        Get $s_0$ state
        while $t < t_{\max}$ do:
            Perform $a_t \sim \pi(\cdot \mid s_t;\theta)$
            Get $r_t$ reward and $s_{t+1}$ state
            $t \leftarrow t+1$
        if $s_{t-1}$ is not terminal then:
            $R_{t_{\max}} \leftarrow \hat{V}(s_{t-1};\theta)$
        else
            $R_{t_{\max}} \leftarrow 0$
        for all $i \in \{t_{\max}-1,...,0\}$ do:
            $R_i \leftarrow r_i + \gamma R_{i+1}$
        $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$
        $T \leftarrow T+1$
```

IV. EVALUATION AND RESULTS

In this section, our agent will realize various tasks, these tasks are reflected in different mini-maps, and we list our experimental results and compare them with the results of DeepMind.

A. Tasks

The following section will introduce 7 types of mini maps developed by DeepMind, which specialize in the application of reinforcement learning in StarCraft II. The agent can complete the corresponding tasks on each mini-map.

- **MoveToBeacon:** In this map, the agent needs to find a beacon. The position of the beacon appears randomly somewhere on the map, and it is indicated by a green ring. This map tests the agent's navigation ability.
- **DefeatRoaches:** In this map, the agent needs to fight against the enemy army, the enemy army is intact, and it is different from the agent, so the agent needs to find the best attack strategy to defeat the built-in army while only controlling its own units.
- **BuildMarines:** In this map, the agent's task is to build as many marines as possible. This requires the agent to complete a lot of preparatory work, so this map tests the long-term planning capabilities of the agent.
- **CollectMineralShards:** There are two agents in this map. The agents need to cooperate to complete the mineral shards collection task. The best strategy is to drive the two agents to maintain the same action at the same time.
- **CollectMineralsAndGas:** In this map, the agent needs to collect as much minerals and gas as possible, which tests the economic reasoning ability of the agent. For example, the agent can build more workers to collect resources. This can increase the resource collection rate.
- **DefeatBanelingsAndZerglings:** In this map, the agent has to fight two kinds of hostile units, namely banelings and zerglings. Among them, the banelings will explode when they encounter the agent, so the agents need to avoid poisonous explosions during the fight against insects.
- **FindAndDefeatZerglings:** This map tests the agent's navigation capabilities in a partial environment. Agent finds and defeats enemy.

B. Setup

For the map environment used, we set the screen and mini-map resolutions to 64px. This is mainly to keep the resolution set by DeepMind. The weights are all initialized by He initialization [11]. This has become a standard practice in modern machine learning. The main parameters are shown in TABLE I.

Authorized licensed use limited to: Middlesex University. Downloaded on October 20,2020 at 05:02:52 UTC from IEEE Xplore. Restrictions apply.

TABLE I. TRAINING PARAMETERS

Parameters	Value
Learning Rate	5 * 10^{-4}
Discount Factor	0.99
Screen resolution	64 × 64
Minimap resolution	64 × 64
Frequency of agent's action	1 per 8 steps of sc2bot
Exploration strategy	Epsilon greedy

C. *Results*

We let the training agent converge to a stable state, and then use the trained agent to test on different mini maps. In order to ensure that the agent can eventually converge, we train the agent multiple times and take different random seeds. We recorded the final average score of the model and compared it with the benchmark given by DeepMind. Specific training details are shown in TABLE II and the results are shown in TABLE III.

TABLE II. TRAINING DETAILS

Map Name	Samples	Episodes
MoveToBeacon	563,200	2,304
DefeatRoaches	172,800,000	1,609,211
CollectMineralShards	74,752,000	311,426
CollectMineralsAndGas	16,864,000	20,544
FindAndDefeatZerglings	29,760,000	89,654
DefeatBanelingsAndZerglings	10,496,000	273,463

In the table above, "Samples" refer to total number of observe→step→reward chains in one environment and "Episodes" refer to total number of rounds of agent training. The entire process is completed on four 2080TI GPUs to complete the training.

TABLE III. COMPARISON OF EXPERIMENTAL RESULTS

Map Name	A2C Agent	Deep Mind	Human
MoveToBeacon	21.3	26	28
DefeatRoaches	72.5	100	215
BuildMarines	0.55	3	133
CollectMineralShards	81	103	177
CollectMineralsAndGas	3320	3978	7566
FindAndDefeatZerglings	22.1	45	61
DefeatBanelingsAndZerglings	56.8	62	727

The agent which has been trained by a2c algorithm tests for 100 episodes and we calculate the average reward value of the 100 rounds. The "A2C Agent" in the above table is the result of our experiment. "DeepMind" represents the benchmark provided by DeepMind's baseline FullyConv results and "Human" Represents the score gathered by DeepMind from a GrandMaster level player.

V. CONCLUSION

We use the advantage actor - critic algorithm to implement the application of reinforcement learning in seven game mini-maps. Then we evaluated the learning ability of the agent and compared it with the benchmark provided by DeepMind. This proves that the learning agent can perform excellently under the training of a2c algorithm. Future work may focus on further exploration of the algorithm and network architecture, and then place the implementation of the algorithm in a real StarCraft II environment instead of mini-maps.

REFERENCES

[1] A. L. Samuel, "some studies in machine learning using the game of checkers," IBM Journal of Research and Development, 1959, 3(3):210-229.

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

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran,D. Wierstra, S. Legg, and D. Hassabis, "Human-level control throughdeep reinforcement learning," Nature, 518(7540):529–533. 2015.

[4] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, ViZDoom: "A doom-based AI research platform for visual reinforcement learning," In 2016 IEEE Conference on Computational Intelligence and Games (CIG). IEEE. 2016.

[5] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, Sasha A. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van 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 e-prints, 2017.

[6] E. Thorndike, "SOME EXPERIMENTS ON ANIMAL INTELLIGENCE," Science, 7(181):818–824, 1898.

[7] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 1998.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347v2 [cs.LG], 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[9] J. Schulman, "Optimizing expectations: From deep reinforcement learning to stochastic computation graphs," Ph.D. dissertation, UC Berkeley, 2016.

[10] V. Mnih, Adrià Puigdomènech Badia, M. Mirza, A. Graves, Timothy P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. ICML 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In 2015 IEEE International Conference on Computer Vision (ICCV). IEEE. 2015.

Authorized licensed use limited to: Middlesex University. Downloaded on October 20,2020 at 05:02:52 UTC from IEEE Xplore. Restrictions apply.