Modified PPO-RND Method for Solving Sparse Reward Problem in ViZDoom

Jia-Chi Chen
Computer Science and Information Engineering
National Kaohsiung University of Science and Technology
Kaohsiung, Taiwan, R.O.C.
1104108125@nkust.edu.tw

Tao-Hsing Chang
Computer Science and Information Engineering
National Kaohsiung University of Science and Technology
Kaohsiung, Taiwan, R.O.C.
changth@nkust.edu.tw

***Abstract*—ViZDoom is an infamous first-person shooter game. Several studies have been conducted to develop agents that can automatically complete game tasks using a reinforcement learning algorithm. Although these studies yielded substantial progress, models proposed by the previous studies when applied to the "my way home" scenario in ViZDoom presented two problems. The first one is that when an agent walks into a specific room, it appears to be immobile and although it does not move until the time ends, the view constantly changes from left to right. The second problem is the slow learning speed of the model. To address these issues, a time penalty method and a modified neural network construction method are proposed in this study. The experimental results demonstrate that the addition of a time penalty improved the learning rate by 40% compared to the methods in which time penalty was not added. Moreover, the models proposed in previous studies could complete only 73% to 85% of the tasks, whereas the method proposed herein can complete 100% of the tasks.**

Index Terms*—reinforcement learning, deep learning, game AI

I. INTRODUCTION

The goal of a reinforcement learning (RL) algorithm is to allow an agent to learn a particular policy, the goal behind which is to maximize the episode rewards of the environment. In some environments, the rewards are supplied to the agent continuously[1]. Others are manually defined environment rewards; for example, calculated the walking distance of an agent as a reward provided by the environment.

However, it is unreasonable to reward each action in a real-life scenario. For example, a left-turning action does not guarantee a good or bad decision, and it is difficult to decide whether to reward or penalize an agent in such a case. This problem renders it difficult to define the environment reward manually. One of possible solutions is that a positive reward is only provided when the agent completes the goal. But, it leads to sparse rewards.

Sparse rewards can cause two problems. The first one is that the agent does not understand the type of behavior that would yield the accurate result, and the types that are unnecessary or even harmful to the completion of the goal[1]. Therefore, it is difficult to use the sparse rewards for evaluating the value of an action.

The other problem is that in an environment containing sparse reward, the agent cannot effectively explore the environment. For example, in a maze problem, we expect the agent to quickly explore different spatial domains. In the continuous environment rewards method, a penalty is directly provided to the agent when it touches the wall, and subsequently it can learn to effectively avoid the wall. Nevertheless, in the environment of sparse reward, the unreasonable behavior of an agent repeatedly colliding with a wall occurs because the agent was not penalized in time. Even if the agent accidentally reached the terminal, the final strategy that the agent learned may be that the terminal can be reached by walking along the wall. In reality, this is not a good strategy, because the correct strategy is to explore other possible methods of escape. This flawed strategy can cause task failure in environments with different features.

Pathak et al.[1] proposed an approach based on the intrinsic curiosity module (ICM)[2] to solve the sparse reward problem. The ICM calculates the novelty of the visual information to generate the exploration bonus that serves as a rewards for the agent such that the agent can develop the proper exploration strategy. However, agents in the ICM cannot implement the correct action because the novelty continues to be high. Burda et al.[3] further proposed the random network distillation (RND) method to reduce the sensitivity of the agent to novelty and avoid the shortcomings of the ICM.

However, according to our observations and experiments, agents using RND in ViZDoom still cannot act effectively at times. Two main problems are found. First, the learning time of RND remains excessively long. Next, in some scenarios, agents using RND are trapped in specific rooms. The purpose of this study is to modify the RND model such that agents can accelerate the learning rate, thus allowing the agents to act effectively in the aforementioned environments without being trapped in a specific spatial domain.

II. RELATED WORKS

The method proposed herein is to design an agent to be executed in the ViZDoom game. The basic architecture of this method is proximal policy optimization (PPO) combined with the RND method. Accordingly, this section describes the environmental characteristics of ViZDoom and the related previous studies on the development of this environment, as well as related studies on PPO and RND.

Kempka et al.[4] indicated that the performance of RL in two-dimensional-environment games has already surpassed human average. They reported that the next phase of RL research should be performed in a three-dimensional environment that is closer to reality, and the perspective of this environment should be the same as that observed by human eyes. ViZDoom is a classic first-person shooter (FPS) game, in which the player sees the same image as observed by real eye. Therefore, Kempka et al.[4] considered ViZDoom to be the most suitable development platform for the RL algorithm. They designed an agent to use the convolution neural network (CNN) as an encoder for an input image. This CNN converts the image into a feature space and uses a deep Q-network to

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

learn how to complete the tasks. In a ViZDoom environment involving basic scenarios, the agent could learn to shoot targets.

Henceforth, many studies [5][6][7][8] have been conducted to design agents based on various algorithms for ViZDoom. For example, Lample et al.[9] indicated that the field of view of a ViZDoom player is limited to 90° around the center of its position. This characteristic allows ViZDoom to conform to the partially observable environment proposed by Hausknecht et al.[10]. Consequently, Lample et al. applied the RNN architecture proposed by Hausknecht et al. to design an agent; in the experimental results, the agent of this method performs better in enemy detection compared to those in previous methods.

PPO is derived from the trust region policy optimization (TRPO) algorithm proposed by Schulman et al.[11]. This algorithm can determine the optimal step size to update the gradient in a RL algorithm. Schulman et al.[12] further proposed that PPO improves the complexity of the TRPO algorithm. Previously, quadratic approximation was required to solve the TRPO; whereas, PPO can be solved with only a first-order approximation.

RND is a method used to design an intrinsic module. Some researchers [13][14][15] have proposed the addition of such a module to the original architecture to solve the sparse reward problem. Agents are encouraged to explore the environment with the exploration bonus provided by count-based exploration methods.

Bellemare et al.[16] argued that although count-based exploration methods can reduce the behavioral uncertainty of agents by calculating the state counts visited, the efficiency of RL is inferior. This is because the states in some environments can rarely be accessed repeatedly, thus resulting in a severely insufficient number of state counts. Consequently, Bellemare et al. proposed the pseudo-count as a function approximation for exploration. This approximation calculates the exploration bonus through the experience reply collected by the agent.

Pathak et al.[1] proposed an approach based on the intrinsic curiosity module (ICM)[2]. Burda et al.[3] proposed the RND model to further solve the noise-TV problem encountered by ICM-based agents. The method proposed herein adopts the RND architecture and enhances the performance of agents by improving the neural network architecture of the RND.

III. METHODOLOGY

The "my way home" scenario in ViZDoom is used as an example in this study. This scenario contains nine rooms and three starting points. Each of the rooms has different wall colors, and the agent appears at one of the three starting points. The goal is to reach the terminal point. As the distance from each starting point to the terminal point is different, the further the distance, the greater is the difficulty in completing the task.

The environment provides an image after the action output of the agent is executed. Each image uses a 3 × 640 × 480 matrix to represent the image observed by the agent. When the agent views the image, it can perform any one of the three actions: move forward, turn left, or turn right. Each action is considered to be an action step. The environment does not provide any reward until the agent reaches the terminal. If the task is not completed before the allotted time, the environment provides the agent with a negative reward, and vice versa.

Our method is based on the neural network architecture proposed by Burda et al.[3], as shown in Fig. 1. This network contains three primary components: Deep reinforcement learning (DRL), RND predictor, and RND target. RL is used to train the output policy of the agent, and for the intrinsic and extrinsic values. The policy determines the action of the agent. The RND predictor and target generate a feature space each, and the exploration bonus is obtained by calculating the difference between the two spaces. The processed image is converted to an input format suitable for the three components by a preprocessing module. We modified the PPO algorithm and the neural network; the operating principle of the PPO and the reasons for the modifications are explained in the following.

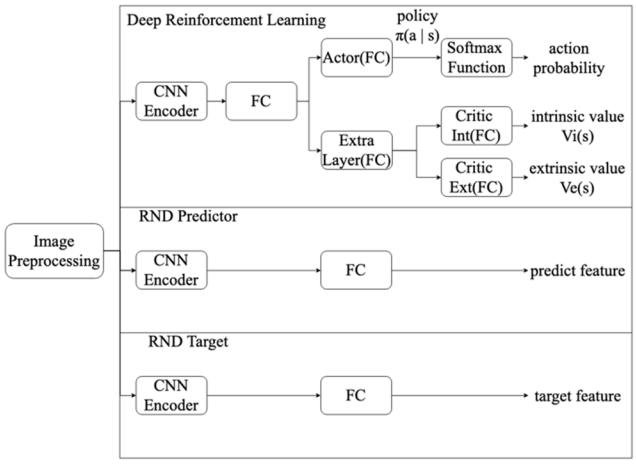

Fig. 1. Neural network architecture proposed by Burda et al.[3]

This architecture uses the CNN as an encoder in the DRL model. The CNN first converts an image to a feature space; subsequently, the feature space is input to a fully connected network (FCN). Finally, the two FCNs, actor, and additional layer are input. The actor outputs a policy matrix to a softmax function, which determines the action of the agent. The additional layer outputs a feature matrix for the intrinsic and extrinsic critics. The two FCNs' critic output $V(S_t)$ of the intrinsic value and the extrinsic value, where s_t represents the state of time t.

PPO calculates the temporal-difference error δt through the environment reward r_t and the next state, as shown in (1)

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{1}$$

where γ is the learning rate that controls the magnitude of the gradient in the update. PPO subsequently uses δ_t to calculate the advantage function $\hat{A}_t$, as shown in (2)

$$\widehat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{2}$$

where λ is an exponentially weighted value, and both λ and γ are used to control the learning rate.

The PPO algorithm[12] uses the updated output $\pi_\theta(a_t|s_t)$ and the old output $\pi_{\theta_{old}}(a_t|s_t)$ at time t of the actor of the neural network architecture to calculate the ratio $r_t(\theta)$. This is

applied as the parameter for the updated magnitude of PPO, as shown in (3)

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{3}$$

where θ is the weight matrix of the neural network, a_t is the action, and s_t is the state. Finally, the loss function of PPO $L^{CPIP}(\theta)$ can be calculated from (4)

$$L^{CPIP}(\theta) = \hat{E}_t\left[min\left(\left(r_t(\theta)\hat{A}_t,\ clip(r_t(\theta),\ 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{4}$$

where $\hat{\mathbb{E}}_t$ is the expected value of time t, and the clip function $r_t(\theta)$ is a function that constraints the updated magnitude. If $r_t(\theta)$ is less than 1-ϵ, the clip function outputs $1-\epsilon$; if $r_t(\theta)$ is greater than $1+\epsilon$, the clip function outputs $1+\epsilon$; the remaining output $r_t(\theta)$.

We discovered that in the sparse reward problem, the agent did not receive an environment reward in most cases, leading to the calculation of r_t of δ_t in (1) to be nearly zero. Therefore, we modified (1) to (5)

$$\delta_t = r_t + p + \gamma V(s_{t+1}) - V(s_t) \tag{5}$$

where p is the time penalty calculated using (6)

$$p = \frac{n_f}{n} \cdot \frac{step_{max} - step_{min}}{step_{max}} \tag{6}$$

where n is the number of samples, n_f is the number of times in the episodes the samples failed to reach the terminal, and $step_{max}$ and $step_{min}$ are the maximum and minimum numbers of action steps in the episode taken to reach the terminal, respectively. p is close to unity at the beginning of the training, because the training sample of the agent is insufficient, thereby resulting in a high number of failures n_f. The difference in the number of action steps required for each episode to reach the terminal is significant. In theory, after the model is trained, the number of failure decreases, and the difference in the number of action steps required for different episodes to reach the terminal is also reduced, which causes p to decrease continuously.

We believe that adding a time penalty offers two advantages to the training agents: first, the original environment reward is no longer sparse, and the agent can better judge the value of each behavior, thus enabling it to improve policy learning efficiency. Additionally, using time penalty, the agent can better comprehend the advantages and disadvantages of a policy in an episode. If the agent reaches the terminal faster, the rewards in (5) are higher, and vice versa.

As mentioned in section 1, the method shown in Fig. 1 in the "my way home" scenario of ViZDoom resulted in a problem, i.e., when the agent walks into a specific room, it appears to be immobile and although it does not move until the time ends, the view changes from left to right constantly. We believe that this problem is because the neural network architecture proposed by Burda et al.[3] is better suited for addressing more complex environments. For example, the environment "Montezuma's revenge" originally proposed by Burda et al.[3] contains a large number of rooms and the agent has more varied action options. We found that the original neural network construction method is not suited to a simple environment such as ViZDoom, which is possibly because the agents that are adapted to a complex environment are prone to the problem of overfitting in a simple environment. If the abovementioned assumption is correct, then the neural network construction method proposed by Pathak et al.[1] is a reasonable design to be applied to each module in Fig. 1 because it was proved to be effective in the "my way home" scenario. The experiments in the next section verify if such a design can solve the task incompletion problem.

IV. Experimental Results

The environment for this experiment is ViZDoom. ViZDoom provides an image each time the agent executes an action. The size of each image is adjusted to 42 × 42 pixels and a gray-scale image in the preprocessing module. The stack method is subsequently used to merge four images into an image as the input data for the method proposed herein.

The first experiment compares the difference in learning rates between the time penalty method proposed herein and that of Burda et al.[3]. The former requires the value of n to be 100 when calculating the time penalty, which are obtained from the episodes of 500 agent trainings. The experimental results are shown in Fig. 2. In addition, Fig. 2 illustrates that the method used by Burda et al.[3] requires a training of 2500 episodes for the action steps to converge to less than 250 steps; the method of adding a time penalty requires training of only 1200 episodes to achieve the same result. Therefore, adding a time penalty increases the learning rate by 40% compared to the method when time penalty is not added.

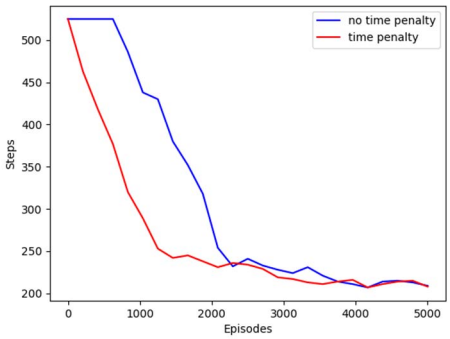

Fig. 2. Time penalty comparison

The second experiment compares the difference in task completion ratios between the method of Burda et al.[3] and the neural network construction method proposed herein. Table I shows the DRL and RND neural network architectures proposed herein. The representations of the symbols in the "neural network architecture" field in the table are as follows: xCySzPr is the convolutional layer, x is the number of filters, y is the kernel size, z is the stride size, and r is padding size; xFC represents the fully connected layer with x being the number of neurons, and ELU is the activation function selected for the module. xN indicates that the output of the module is x neurons.

Fig. 3 illustrates the success rate of completing a task of the two methods at different numbers of training episodes. As the environment contains three starting points, and both methods had completed all the tasks for the two points closer to the terminal, only the task completion ratio for the farthest starting point is compared in Fig. 3. Additionally, Fig. 3

demonstrates that after training 2500 episodes, the construction of Burda et al.[3] had completed only a ratio of 73% to 85% of the tasks, while the method proposed herein completed 100% of the tasks. In addition, even when trained to 20000 times, the task completion ratio of the construction of Burda et al.[3] did not increase significantly. The method proposed herein completed 100% of the tasks after training 2500 episodes. Therefore, the neural network construction proposed herein is an effective solution to the problem of task incompletion encountered by Burda et al.[3].

TABLE I. CONSTRUCTION METHOD OF DRL NEURAL NETWORK PROPOSED HEREIN

Components	Module	Method
DRL	Conv-Encoder	42x42x4-32C3S2P1(ELU)32C3S2P1(ELU)-32C3S2P1(ELU)-32C3S2P1(ELU)-288N
	FC	288-256FC(ELU)-256N
	Actor(FC)	256-3FC-3N
	Extra Layer(FC)	256-256FC(ELU)-256N
	Critic Int(FC)	256-1FC-1N
	Critic Ext(FC)	256-1FC-1N
RND	Predict Conv-Encoder	42x42x1-32C3S2P1(ELU)-32C3S2P1(ELU)-32C3S2P1(ELU)-32C3S2P1(ELU)-288N
	Predict FC	288-256FC-256N
	Target Conv-Encoder	42x42x1-32C3S2P1(ELU)-32C3S2P1(ELU)-32C3S2P1(ELU)-32C3S2P1(ELU)-288N
	Target FC	288-256FC-256N

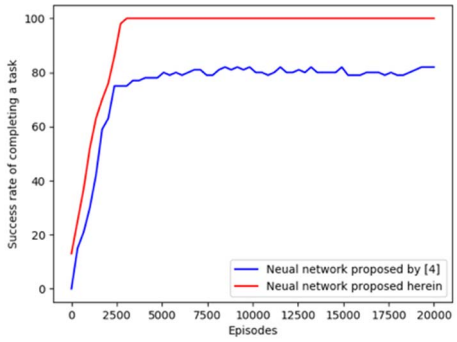

Fig. 3. Probability of agent successfully reaching the end point before and after neural network modification

V. CONCLUSION AND FUTURE WORKS

Effective solutions to the two problems encountered in previous studies in ViZDoom "my way home" scenario was proposed herein. First, a time penalty was applied to accelerate the learning speed of the model. Next, the task incompletion problem was eliminated with the use of the proposed neural network construction. The proposed method demonstrated excellent performance, and several directions can be undertaken for future research.

In this paper, only a preliminary experiment was conducted on the "my way home" scenario. In future experiments, however, the effectiveness of the proposed method must be verified in other scenarios and environments. Additionally, the neural network construction used in this study was originally designed to function effectively in ViZDoom. Although the current experiment confirmed that the design allowed for PPO-RND, which was not specifically designed for ViZDoom, to operate in a new environment, the optimal neural network construction in the environment thereof remains a challenge and further research is warranted.

ACKNOWLEDGMENTS

This study was partially supported by the Ministry of Science and Technology, under the grant 107-2511-H-992-001-MY3, and the University Sprout Project - Chinese Language and Technology Center of National Taiwan Normal University, sponsored by the Ministry of Education, Taiwan.

REFERENCES

[1] D. Pathak, P. Agrawal, A. A. Efros, T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," *The IEEE Conference on Computer Visison and Pattern Recognition(CVPR) Workshops*, 2017, pp. 16–17.

[2] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation," *IEEE Transactions on Autonomous Mental Development*, 1990–2010, pp. 230–247.

[3] Y. Burda, H. Edwards, A. Storkey, O. Klimov, "Exploration by Random Network Distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[4] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jakowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," *IEEE Conference on Computational Intelligence and Games (CIG)*, 2017, pp. 1–8.

[5] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, S. Gelly, "Episodic curiosity through reachability," *arXiv preprint arXiv:1810.02274*, 2018.

[6] M. Wydmuch, M. Kempka, W. Jaskowski, "ViZDoom Competitions: Playing Doom from Pixels," *IEEE Transactions on Games*, 2018

[7] C. Schulze, M. Schulze, "ViZDoom: DRQN with Prioritized Experience Replay, Double-Q Learning, & Snapshot Ensembling," *Proceedings of SAI Intelligent Systems Conference*, 2018, pp. 1–17.

[8] N. Justesen, S. Risi, "Automated Curriculum Learning by Rewarding Temporally Rare Events," *IEEE Conference on Computational Intelligence and Games(CIG)*, 2018.

[9] G. Lample, D. S. Chaplot, "Playing FPS games with deep reinforcement learning," *Thirty-First AAAI Conference on Artifical Intelligence(AAAI)*, 2016, pp. 2140–2146.

[10] M. Hausknecht, P. Stone, "Deep recurrent q-learning for partially observable mdps," *AAAI Fall Symposium Series(AAAI)*, 2015, pp. 29–37.

[11] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel, "Trust Region Policy Optimization," *Journal of Machine Learning Research(JMLR)*, 2015.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[13] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, R. Fergus, "Intrinsic motivation and atuomatic curricula via asymmetric self-play," *arXiv preprint arXiv:1703.05407*, 2018.

[14] A. S. Klyubin , D. Polani, C. L. Nehaniv, "Empowerment: A universal agentcentric measure of control," *IEEE Congress on Evolutionary Computation*, 2005.

[15] H. Tang, R. Houthooft, D. Foote , A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, P. Abbeel, "Exploration: A study of count-based exploration for deep reinforcement learning," *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[16] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, "Unifying Count-based exploration and intrinsic motivation," *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.