Learning Battles in ViZDoom via Deep Reinforcement Learning

Kun Shao[†‡], Dongbin Zhao[†‡], Nannan Li[†‡], Yuanheng Zhu[†‡]
[†]The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[‡]University of Chinese Academy of Sciences, Beijing 100049, China
shaokun2014@ia.ac.cn, dongbin.zhao@ia.ac.cn, linannan2017@ia.ac.cn, yuanheng.zhu@ia.ac.cn

***Abstract*—First-person shooter (FPS) video games play an important role in game artificial intelligence (AI). In this paper, we present an effective deep reinforcement learning (DRL) method to learn battles in ViZDoom. Our approach utilizes the actor-critic with Kronecker-factored trust region (ACKTR), a sample-efficient and computationally inexpensive DRL method. We train our ACKTR agents in two battle scenarios, and compare with the advantage actor-critic (A2C) baseline agent. The experimental results demonstrate that DRL methods successfully teach agents to battle in these scenarios. In addition, the ACKTR agents significantly outperform the A2C agents in terms of all the metrics by a significant margin.**

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

Fig. 1. Game sample of the battle scenario in ViZDoom from the first-person perspective.

I. Introduction

In the last few years, we have witnessed massive progresses of game artificial intelligence (AI) with deep reinforcement learning (DRL) [1] [2]. These DRL agents have achieved impressive performances in various games, including Atari [3], Go [4], Poker [5], and StarCraft [6] [7]. It has been proven that DRL is a general and effective architecture for game AI.

In this paper, we present DRL methods to teach agents to learn battles in ViZDoom [8], which is a first-person perspective 3D environment for visual reinforcement learning research, as shown in Fig. 1. Agents in ViZDoom have many challenges. They have to navigate in high dynamic partially observable maze environment, recognize different objects, and shoot accurately at the enemy targets. All of these behaviors come from high-dimensional image inputs. We apply the state-of-the-art DRL methods to tackle these challenges in two battle scenarios. To evaluate the performance of our agent, we compare the results with the baseline method in terms of some metrics. The organization of the remaining paper is arranged as follows. In Section II, we describe the related work of ViZDoom. Then we present the DRL methods in Section III, and introduce ViZDoom battle scenarios and the learning model in Section IV. In Section V, we present the experimental setup details and the results. Finally, we draw a conclusion of our research.

This work is supported by National Natural Science Foundation of China (NSFC) under Grants No.61573353, No.61603382 and No. 61533017.

II. Related Work

In recent years, deep reinforcement learning has been widely used in game AI, showing superior performance to traditional methods. Here we focus on related work of ViZDoom with DRL methods, especially for battle scenarios.

Kempka *et al.* propose ViZDoom as a novel test-bed for reinforcement learning from visual information. There are various tasks in ViZDoom, and agents have to interact with the 3D world in a first-person perspective. In some simple tasks, e.g., the basic and the health gathering, the deep Q-Network (DQN) agent achieves acceptable performances [8]. Kulkarni *et al.* use the successor representations to decompose the value function into a reward predictor and a successor map, and generalize it in the end-to-end DRL architecture. This method has the increased sensitivity to distal reward changes and the ability to extract bottleneck states in ViZDoom [9]. Lample *et al.* present a method to augment the deep recurrent Q-network (DRQN) model to exploit game feature information during training, which improves the training speed and performance dramatically. This model bases on the proposed action-navigation architecture, and outperforms built-in agents as well as average humans in the deathmatch battle scenarios [10]. The agent, named Arnold, wins the first place for track 2 in the ViZDoom AI competition 2017. Tian *et al.* propose a framework that combines the asynchronous advantage actor-critic (A3C) model and curriculum learning [11]. This agent learns to move and shoot via playing against built-in agents in a progressive manner and wins the first place for track

1 in the ViZDoom AI competition 2016. Dosoviskiy *et al.* propose the direct future prediction (DFP), which utilizes a high-dimensional sensory stream and a lower-dimensional measurement stream as the inputs to provide a rich supervisory signal, and trains a sensorimotor control model by interacting with the environment [12]. DFP successfully generalizes across environments and goals, and outperforms state-of-the-art DRL methods on challenging deathmatch tasks. This agent wins the first place for track 2 in the ViZDoom AI competition 2016.

III. Deep Reinforcement Learning

A. Reinforcement Learning and Actor-Critic Methods

In the reinforcement learning paradigm, an agent learns by trial and error, and determines the behavior from its own experiences with the environment [13]. Here we consider an agent interacting with a discounted Markov decision process (MDP) (S, A, γ, P, r). At time t and state s_t, the agent chooses an action a_t according to the policy $\pi(a_t|s_t)$. After receiving the action, the environment produces a reward r_{t+1} and transitions to the next state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t)$. The process continues until the agent reaches a terminal state. The goal of the agent is to maximize the expected discounted cumulative rewards under the policy π with discount factor $\gamma \in (0, 1]$

$$\mathbb{E}_\pi[R_t] = \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]. \tag{1}$$

Policy gradient methods parameterize the policy $\pi_\theta(a_t|s_t)$ directly and update parameter θ to maximize the objective function $J(\theta)$. In its general form, $J(\theta)$ is defined as

$$J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \Psi_t \log \pi_\theta(a_t|s_t)]. \tag{2}$$

Based on policy gradient methods, actor-critic reinforcement learning methods use a value function to approximate Ψ_t. Mnih *et al.* propose the asynchronous advantage actor-critic method, which asynchronously executes multiple agents on multiple instances of the environment [14]. With multiple agents playing concurrently and optimizing the networks through asynchronous gradient descent, A3C decorrelates the training data into a more stationary process and improves the performances of a number of tasks.

As the synchronous version of A3C, A2C optimizes the learning model synchronously and uses the GPU to accelerate learning process [15]. In A2C, the objective function is defined as

$$R'_t = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V_{\theta_v}(s_{t+k}), \tag{3a}$$

$$A_{\theta,\theta_v}(s_t, a_t) = R'_t - V_{\theta_v}(s_t), \tag{3b}$$

$$J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} A_{\theta,\theta_v}(s_t, a_t) \log \pi_\theta(a_t|s_t) + \beta H_\theta(\pi(s_t))]. \tag{3c}$$

θ_v are the parameters of the value network, and $H_\theta(\pi(s_t))$ is an entropy term used to encourage exploration during the training process.

B. Actor Critic using Kronecker-Factored Trust Region

A2C and other DRL methods are usually trained using simple variants of stochastic gradient descent (SGD), which are inefficient first-order methods. Natural gradient descent can perform gradient updates efficiently, which follows the steepest descent direction and uses the Fisher metric as the underlying metric. More recently, Kronecker-factored approximated curvature (K-FAC) is proposed as a scalable approximation to natural gradient, which can be used to the Fisher matrix to perform approximate natural gradient updates efficiently. Wu *et al.* combine the actor-critic, trust-region policy optimization (TRPO) and K-FAC, and introduce the actor-critic using Kronecker-factored trust region (ACKTR) [16]. ACKTR extends the framework of natural policy gradient and optimizes both the actor and the critic using K-FAC. For the actor, we use the policy distribution to define the Fisher matrix

$$F = \mathbb{E}_{p(\tau)}[\nabla \log \pi(a_t|s_t) \nabla \log \pi(a_t|s_t)^T]. \tag{4}$$

For the critic, the output of the critic v is defined to be a Gaussian distribution, and we define the Fisher matrix with respect to this Gaussian output distribution.

$$p(v|s_t) \sim \mathcal{N}(v; V(s_t), \sigma^2) \tag{5}$$

When actor and critic share lower-layer representations, we define the joint distribution as $p(a, v|s)$, and apply K-FAC to approximate the Fisher matrix to perform updates simultaneously.

$$p(a, v|s) = \pi(a|s)p(v|s) \tag{6a}$$

$$F = \mathbb{E}_{p(\tau)}[\nabla \log p(a, v|s) \nabla \log p(a, v|s)^T] \tag{6b}$$

Moreover, ACKTR adopts the trust region formulation of K-FAC, choosing the effective step size η. The natural gradient is performed with the following updates.

$$\eta = min(\eta_{max}, \sqrt{\frac{2\delta}{\Delta\theta^{\mathbf{T}} \hat{F} \Delta\theta}}) \tag{7a}$$

$$\theta \leftarrow \theta - \eta F^{-1} \nabla_\theta L \tag{7b}$$

where η_{max} is the learning rate and δ is the trust region radius.

ACKTR is the first scalable trust region natural gradient method for actor-critic DRL, and improves the sample efficiency of current methods significantly. In the following experiments, we will use this method to train our agents.

IV. Learning Model for Battles in ViZDoom

A. Battles in ViZDoom

We consider two battle scenarios in ViZDoom as the testbeds for our deep reinforcement learning agents, as shown in Fig. 2. In each scenario, the agent is armed and is under attack by enemies in a maze. The enemies spawn abundantly, move around in the environment, and shoot at our agent. In the second task, apart from enemies, health kits and ammunition

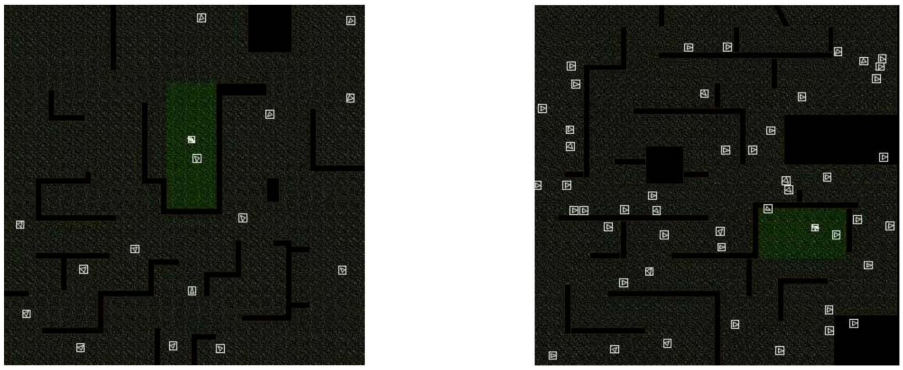

Fig. 2. Examples of the ViZDoom battle maps in the experiment, left: the map of the first battle; right: the map of the second battle.

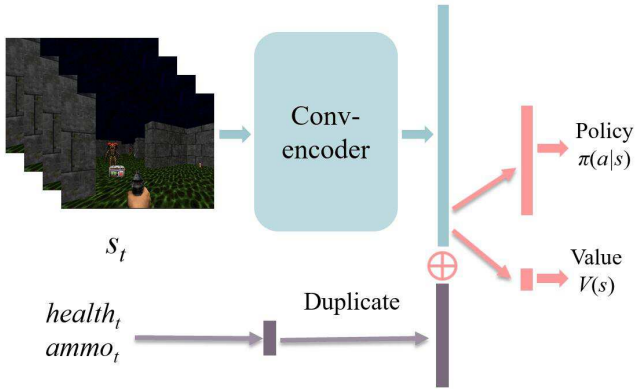

Fig. 3. The architecture of the actor-critic DRL model for battle scenarios in ViZDoom. The model takes in recent four frames visual inputs s_t, game variables $health_t$, $ammo_t$, and produces policy $\pi(a|s)$, value function $V(s)$.

are sporadically distributed throughout the environment and can be collected by the agent.

We will train the agent to learn battles with complex visual inputs in these two scenarios. The agent has seven actions: move forward, move backward, move left, move right, turn left, turn right, and attack. An episode comes to an end when the agent is dead or reaches the maximum game steps. The measurements that the agent can access in these scenarios are the health point, the ammo, and the kill count. The objective of the agent is to kill as many enemies as possible.

We use the reward shaping method in these scenarios. Basically, when the agent is dead, it will receive a penalty of -100. Furthermore, to accelerate the learning process, we give additional rewards for gathering a health kit and a ammo kit, which are set to +10. If our agent is attacked by the enemies, it will receive a penalty of -10. And we add a penalty for wasting ammo, which is set to -2. In the training process, we reshape the total reward and divide it by 100. Each episode finishes after 4200 steps.

B. Learning model for Battles

We follow the architecture of the Facebook F1 bot that uses the actor-critic deep reinforcement learning model to tackle the battle scenarios in ViZDoom. The actor-critic method used in our experiment learns both a policy $\pi_\theta(a_t|s_t)$ and a value function $V_{\theta_V}(s_t)$. The model receives a state observation s_t, game variables $health_t$, $ammo_t$, and uses deep convolutional neural networks as the encoder. The resolution of the original grey-scale visual image is 160×120. We stack four recent observations as the inputs to track historical traces of the agent. In the data preprocessing, we resize the visual observation to 84×84.

The neural network used in the experiment follows a similar architecture in the DQN paper. The network uses a convolutional layer with 32 filters of size 8×8 with stride 4, followed by a convolutional layer with 64 filters of size 4×4 with stride 2, followed by a convolutional layer with 64 filters of size 3×3 with stride 1, followed by a fully connected layer with 512 hidden units. All four hidden layers are followed by a rectifier nonlinearity. Thereafter, we concatenate the game variables tensor and use the softmax function to output the policy $\pi_\theta(a_t|s_t)$, and use one linear output for the value function $V_{\theta_V}(s_t)$. We share all the non-output layers in the actor network and the critic network. This layer-sharing mechanism can stabilize the training process. To balance exploration and exploitation in reinforcement learning, we use the Boltzmann exploration method to select actions. The details of the actor-critic DRL model for ViZDoom battle scenarios are depicted in Fig. 3.

V. Experiments

A. Experimental Setup

The experiments are performed on the two presented ViZDoom battle scenarios with the following setup. We refer to the hyperparameters of DRL model in the OpenAI baseline [17]. In each experiment, we open 128 processes running on a single machine with one Nvidia P100 GPU. The DRL methods perform updates after every 20 action steps. The discount factor is set to 0.99. We use the standard non-centered RMSProp optimizer to update the A2C agent, and the K-FAC optimizer to update the ACKTR agent.

B. Experimental Results

We present the experimental results of our ACKTR agent and the A2C agent in Fig. 4. In both battle scenarios, the

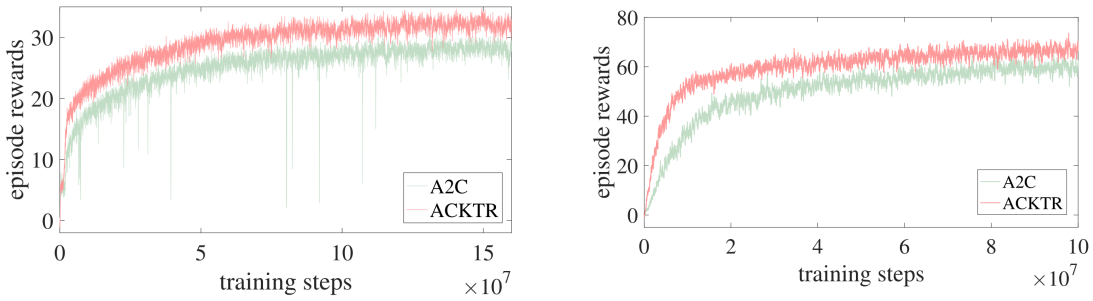

Fig. 4. Episode rewards achieved by DRL agents in the two battle scenarios during training. These rewards are averaged among 128 processes.

TABLE I
TEST RESULTS OF DRL AGENTS IN TWO BATTLE SCENARIOS.

Battles	Agents	Rewards	LivingSteps	KillCounts
Battle-1	A2C	27.8±6.9	1287.7 ±530.1	23.3±5.6
Battle-1	ACKTR	**32.1** ± 7.2	**1303.5**±514.2	**26.2**±5.7
Battle-2	A2C	60.7±19.8	3758.5.7±890.8	20±7.6
Battle-2	ACKTR	**66.1**±21.6	**3975.7**±580.1	**24**±6.6

curves of episode rewards increase very fast in the early stage, and the models converge at the end of training. In the first battle scenario, we train the agents for 150 million steps. In the second battle scenario, the training process ends after 100 million steps.

After training, we test the agents in each battle scenario for 100 games. We record the episode rewards, the living steps and the killcount, which are presented in Table I. From the killcount results, we can see that both DRL agents can successfully learn how to battle in these scenarios. Compared with the A2C agent, the ACKTR agent has better performances in terms of all the metrics. In particular, the ACKTR agent achieves 12.4% and 20% higher killcount scores than the A2C agent in the two battles. Moreover, the ACKTR agent significantly outperforms the A2C agent in terms of sample efficiency by a significant margin. And the performance of ACKTR agent is more stable than the A2C agent, since the A2C agent has several obvious declines during training.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present the efficient ACKTR deep reinforcement learning method to learn battles in ViZDoom. The experimental results show that the DRL agents can successfully tackle these battle scenarios after training. The ACKTR agent significantly outperforms the A2C agent in terms of all the metrics. In the future, we will introduce auxiliary tasks to improve the performance and generalization of our model. And we will train an agent to play the full deathmatch scenario with DRL methods.

REFERENCES

[1] D. Zhao, K. Shao, Y. Zhu, D. Li, Y. Chen, H. Wang, D. Liu, T. Zhou, and C. Wang, "Review of deep reinforcement learning and discussions on the development of computer Go," *Control Theory and Applications*, vol. 33, no. 6, pp. 701–717, 2016.

[2] Z. Tang, K. Shao, D. Zhao, and Y. Zhu, "Recent progress of deep reinforcement learning: from AlphaGo to AlphaGo Zero," *Control Theory and Applications*, vol. 34, no. 12, pp. 1529–1546, 2017.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, d. D. G. Van, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] M. Moravik, M. Schmid, N. Burch, V. Lisy, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[6] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Kttler, J. Agapiou, and J. Schrittwieser, "StarCraft II: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[7] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, DOI:10.1109/TETCI.2018.2823329, 2018.

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

[9] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. Gershman, "Deep successor reinforcement learning," *arXiv preprint arXiv:1606.02396*, 2016.

[10] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2016, pp. 2140–2146.

[11] Y. Wu and Y. Tian, "Training agent for first-person shooter game with actor-critic curriculum learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[12] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," in *International Conference on Learning Representations (ICLR)*, 2017.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[15] C. Alfredo, C. Humberto, and C. Arjun, "Efficient parallel methods for deep reinforcement learning," in *The Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2017, pp. 1–6.

[16] W. Yuhuai, M. Elman, L. Shun, G. Roger, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 1285–5294.

[17] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Openai baselines," https://github.com/openai/baselines, 2017.