Training sparse-reward agent for first-person shooter game using DDRQN and Curriculum learning

Chengyue Jiang[1], **Jingxian Huang**[2], **Jie Min** [3],
[1] Shanghaitech SIST
[2] Shanghaitech SIST
[3] Shanghaitech SIST
jiangchy@shanghaitech.edu.cn, huangjx@shanghaitech.edu.cn, minjie@shanghaitech.edu.cn,

Abstract

Reinforcement Learning(**RL**) are widely used to solve problems associated with FPS games, but deep reinforcement learning (**DRL**) agents are more robust and feasible to problems with large state space than approximate RL methods, for they learn through train and error. Models such as **DQN** agents outperformed human player in Atari games and many other 2D scenarios. However, in 3D scenarios, DRL models become hard to train due to the incomplete observation and sparse-reward problem. In this project, we trained a 3D FPS game agent using DRL and the **ViZdoom** API. It is a doom based 3D-FPS game RL platform. We use the variance of **DRQN** model, Double DRQN to remember and make use of the experiences, and apply **curriculum learning** to reduce the sparse-reward problem. Our model performed well on simple tasks such as `Defend the center`, and be able to apply to complex map `Death match` after curriculum learning.

1 Introduction

Deep reinforcement learning(DRL) draws more and more attention recently, and have been applied to and performed well on many tasks, **DQN** model [Mnih *et al.*, 2013] outperformed human players on 49 Atari games, and Go agents trained by DRL defeated human expert to win the Go championship [Silver *et al.*, 2016]. We are interested in training DRL agents on 3D First-person shooter game, which is often partially observed and has much larger state space.

1.1 ViZdoom

In this project, we use the ViZdoom [1] [Kempka *et al.*, 2016] research platform. It provide research friendly 3D FPS game API for Deep Reinforcement Learning. The platform is based on a classical FPS game **DOOM**, in the game, player shoot and try to kill monsters, and can collect various items such as health pack, ammos, weapons to strengthen himself. We can design scenarios and maps depends on our needs. We train RL agents by only giving it raw image input (frames) and our shaped rewards, agent keep interacting with game environment in ViZdoom and gradually evolves. ViZdoom is also an AI competition in CIG [2]. There were some recent work about ViZdoom. **Arnold** [Lample and Chaplot, 2016a] is an agent trained by **DRQN** [Hausknecht and Stone, 2015], with feature augmenting. Though this agent perform well on CIG2016, but this agent use extra game feature (e.g the information of the opponents) as supervision during the training process, and switching between a navigation network and a attacking network in a game. Facebook [Wu and Tian, 2016] use **A3C** [Mnih *et al.*, 2016] and **cirriculum learning** to get the first place of CIG2016 on the limited deathmatch, IntelAct [Dosovitskiy and Koltun, 2016] models the Doom AI bot training in a supervised manner by predicting the future values of game variables (e.g., health, amount of ammo, etc) and select the actions accordingly. In this project, we use the Double DRQN architecture to predict the Q-values, and we do not use extra augmenting game features or any other cheating game variables, but use a newly designed curriculum training way.

Figure 1: DOOM

[1]https://github.com/mwydmuch/ViZDoom

[2]ViZDoom Competition at IEEE Computational Intelligence And Games (CIG) Conference, 2016 (http://vizdoom.cs.put.edu.pl/competition-cig-2016)

1.2 Solving sparse-reward problem for multi-task agents

Sparse-reward is a common but hard-to-solve problem in training DRL multi-task agents. It will significantly slower down your training efficiency. It simply means agent seldom gets a specific reward, resulting in the corresponding behaviour hard to learn. In this project, we encountered this problem many times, for example, the monsters are too strong for a agent at the exploration state(agent acts randomly), and make agent very difficult to get the kill reward, or the items such as health packs and ammos. Some methods are discussed in the next subsection to solve this problem.

Reward Shaping

Reward is important in the RL scenarios, which can significantly effect agent's manner. By shaping the reward [Ng *et al.*, 1999], such as encourage agents to shoot more by giving positive reward of using 1 ammo, can make agent easier to get kill reward in some cases. But reward-shaping is dangerous and may result in overfiting in some actions.

Curriculum Learning

Curriculum learning [Bengio *et al.*, 2009a] is a method for pushing a multitask agent optimally present the tasks during the learning procedure. The idea of using a curriculum has been explored in many prior works on supervised learning [Pentina *et al.*, 2015], and for multi-task reinforcement learning agents, curriculum learning is often hand-designed and relied on pre-specified sequences of tasks [Karpathy and Van De Panne, 2012]. More recent work use GAN [Goodfellow *et al.*, 2014] to generate learning goals [Held *et al.*, 2017a]. or construct a teacher model to automatically chooses subtasks from a given set for the Student to train on [Matiisen *et al.*, 2017], but we did not use these in our project because it's hard to choose and design a proper set of ViZdoom scenarios. In this project, we use hand designed curriculum to train the Doom agent, without game feature augmenting and other supervision.

2 Model

The up-to-date methods to solve FPS games include deep Q-Learning Network(DQN), Deep Recurrent Q-Learning Network(DRQN). And some research also use more advanced methods such as A2C[Khamassi *et al.*, 2005] and A3C[Khamassi *et al.*, 2005] but these two are not included in out discussion.

2.1 Current Model

In this part, we give a brief introduction to the current Q-Learning model on which own model is based.

Deep Q-Learning

In Q-Learning, the agent is trying to find a policy π which can maximize the expected discounted total reward, which is denoted by the Q value $Q^{\pi}(s,a)$. The optimal Q value

$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a')|s,a]$$

.

In DQN [Mnih *et al.*, 2013], we approximates the value function $Q(s,a)$ with a neural network. The output of the network is the approximate Q value Q_θ, where θ is the parameter of the network. The corresponding loss function is:

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'}[(y_t - Q_\theta(s,a))^2]$$

where y_t is fixed and $y_t = r + \gamma \max_{a'} Q_{\theta_t}(s',a')$. The pseudocode is as follows:

Algorithm 1 DQN

```
1: Initialize replay memory $D$ to capacity $N$
2: Initializeaction-value function $Q$ with random weights
3: for episode=1,M do
4:     Initialize sequence $s_1 = \{x_1\}$
5:     Initialize preprocessed sequenced $\phi_1 = \phi(s_1)$
6:     for t=1,T do
7:         With probability $\epsilon$ select a random action $a_t$
8:         Otherwise select $a_t = \max_a Q^*(\phi(s_t),a;\theta)$
9:         Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
10:        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
11:        Store transition ($\phi_t$,$a_t$,$r_t$,$\phi_{t+1}$) in $D$
12:        sample random minibatch of transitions ($\phi_t$,$a_t$,$r_t$,$\phi_{t+1}$) from $D$
13:        Set
               $y_j = \begin{cases} r_j & \text{for terminal } \phi_j \\ r_j + \gamma \max'_a Q(\phi_{j+1},a';\theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
14:        Perform a gradient descent step on $(y_j - Q(\phi_j,a_j;\theta))^2$ according to equation 3
```

Deep Recurrent Q-Learning

In FPS games, it is difficult to have a full observation, so the agent only receives partial observation but this is not sufficient to infer the actual state of the game. So Lample et al.[Lample and Chaplot, 2016b] introduced a new method called Deep Recurrent Q-Network(DRQN). In this model, instead of using fully connect network as the last layer of the network, it sends the output of the convolutional layer into a LSTM network, as shown in Fig2.

In this way, we estimate the $Q(o_t, h_{t-1}, a_t)$, where h_t is extra step returned by the previous step, which is the hidden state of the agent. Thus, the Q-value is a estimate of the states of several time steps. This efficiently increase the accuracy of the Q-value of the current state.

2.2 Our model–DRQN and Curriculum Learning

Based on the DQN and DRQN model mentioned above, we suggest our own model, Double DRQN(DDRQN) combined with curriculum training.

Though widely used, DQN and DRQN have flaws. According to Thrun et al., since action value have errors uniformly distributed within $[-\epsilon, \epsilon]$, the target value could be overestimated. The overestimate value is up to $\gamma\epsilon\frac{m-1}{m+1}$. But with Double DRQN(DDRQN), the problem can be solved.

In DRQN, the target value y_t is denoted as $y_t = r +$

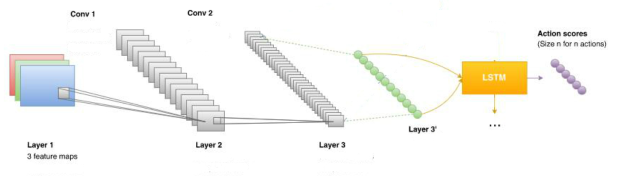

Figure 2: DRQN

$\gamma \max_{a'} Q_{\theta_t}(s', a')$. While in DDRQN, the target value is $y_t = r + \gamma Q(s', arg\max_{a'} Q_{\theta_t}(s', a'))$ and as Fig3, the network structure keeps the same.

Based on this, we add curriculum training in order to make

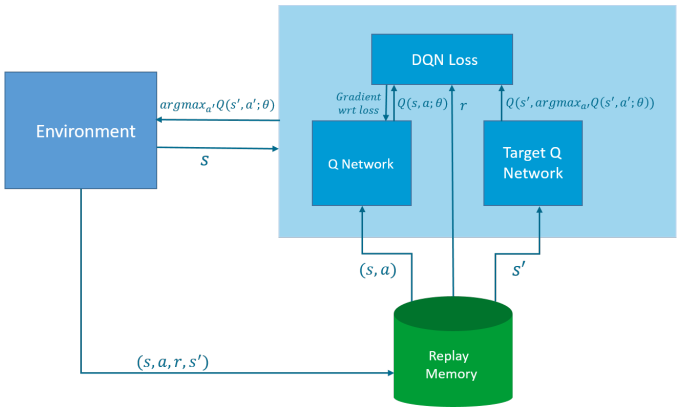

Figure 3: DDRQN

the agent learn the tasks step by step.

3 Experiments

The experiment is based on DDRQN and curriculum learning. We use Python Tensorflow and Vizdoom as tools. The input of the network is image observed in the game frame with resolution 480*640.

3.1 Test Model on Simple Scenarios

In our experiment, we first test our DDRQN model on simple games in order to verify the validity of our model. We choose the task of defend the center, the map is as Fig4. We choose this map because the shape is simple with perfect symmetry and there are only two types of monster, so the agent does not to need learn too many patterns about observation. Also, the game does not support walking, which decreases the action space and make the training easier.

We set 32 frames as a patch and send them into a CNN network and get a feature vector of length 32, and send the feature into the DDRQN network. The replay memory buffer is of size 10000 and new frames would replace old ones when the buffer is full instead of continuously be added into the buffer.

As shown in Fig5, after 1500 game episodes, the average kill rate reaches 18.172, and this surpass most of the human beings.

We test our model that is trained on Defend the Center on the

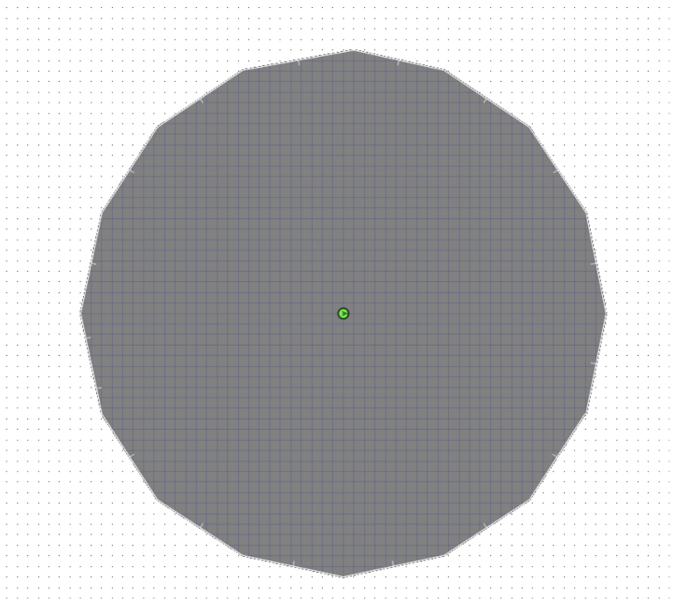

Figure 4: Defend the Center Map

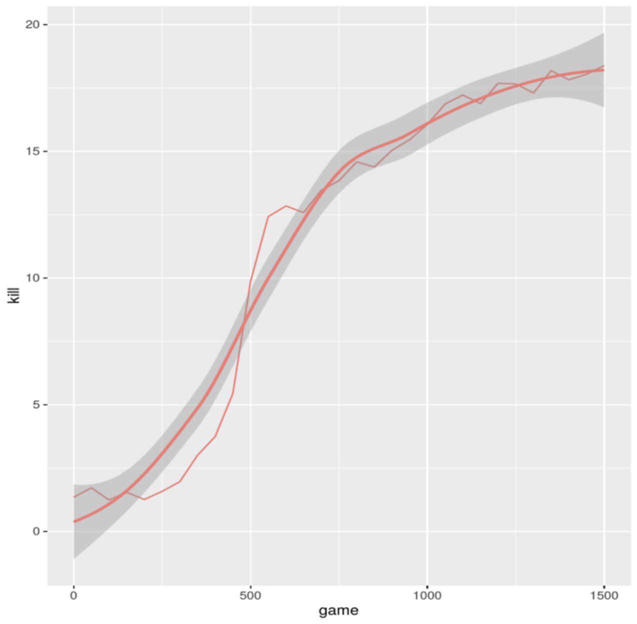

Figure 5: Training curve

game mode Defend the Line, in which the agent is at the back of a wall and try to shoot the monsters across the room. In this game, the room is a rectangle but there are still only two types of monsters and no supplements. We run 500 games and the average kill is 18.574, according to Fig6.

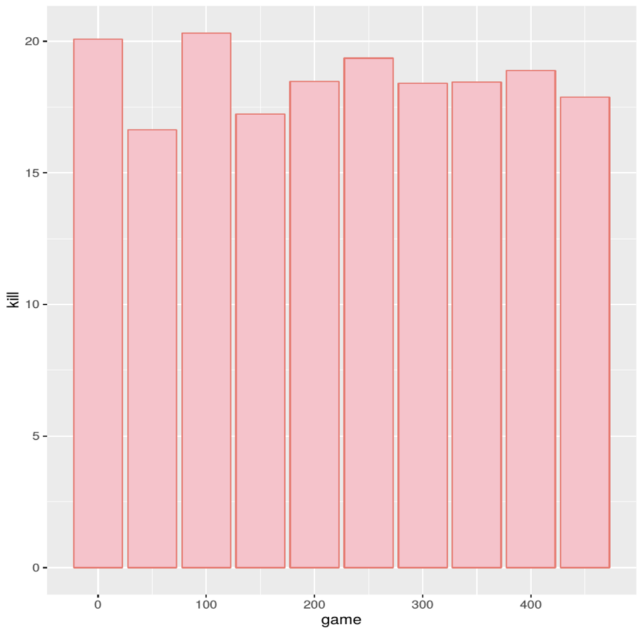

Figure 6: Testing on Defend the Line

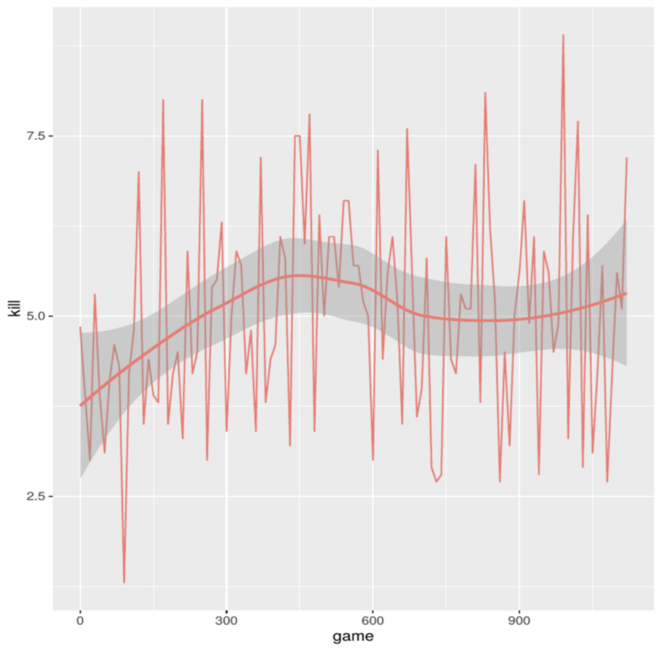

Figure 7: Test on more complicated game.

3.2 Curriculum Learning

We then try to test DDRQN on more complicated games with various of supplement as well as more types of monsters. We add up the action space(TURN_LEFT, TURN_RIGNT, MOVE_FORWARD, MOVE_LEFT, MOVE_RIGHT, MOVE_BACKWARD, ATTACK). We also increase the enemy types into 4. The items are randomly distributed, including Health Pack, Weapons, Ammos. However, the traing curve is not ideal. Instead of getting better, the performance decrease, see Fig7.

To solve this problem, we survey from relative papers [Bengio *et al.*, 2009b] [Florensa *et al.*, 2017] [Held *et al.*, 2017b] and then find curriculum training might be a solution. Curriculum training [Bengio *et al.*, 2009b] divide training process to multiple part, from easy tasks to difficult tasks step by step, based on the 'training result on easy tasks, it can handle more difficult tasks.

Phase One

We designed a map as shown in Fig8. The yellow dots on the top represent weapons(two types), the dark dots on the right represents ammos(two types), and the purple dots on the left are medication(one type). Players start position is in the middle of the circle. There are two types of enemies but no too strong and as can be seen, the objects are aligned in order. But after the same iteration as the defend the center experiment, but the agent cannot learn to pick up supplements because the reward for that is too sparse. So we transfer the model in to Fig9, so that the reward is denser.

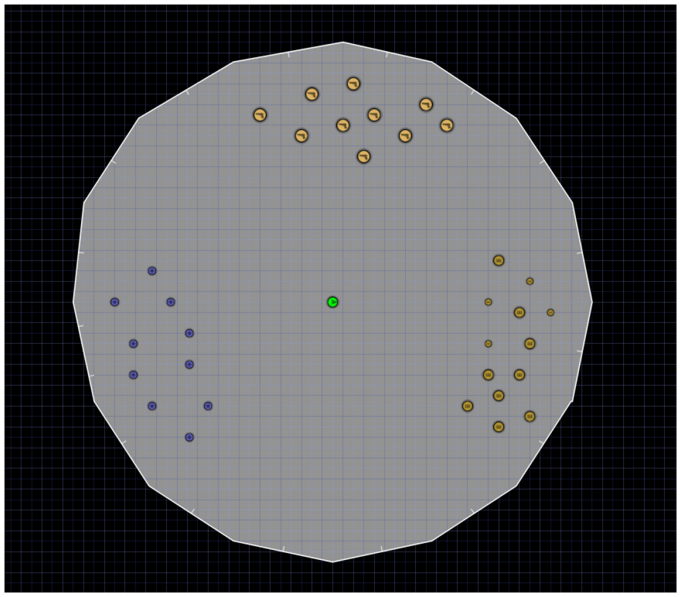

Figure 8: Sparse reward

Phase two

The reward funtion is as Table 1. However, after training, the agents only pick up supplements instead of kill enemies. So we increase the reward for killing monsters and use ammo and decrease the reward for picking supplement, as shown in Table 2.

After map shaping and reward shaping, the result is shown in Fig10. After 2300 episodes, the agent learns a clear pattern to kill the enemies. The learning curve performance really

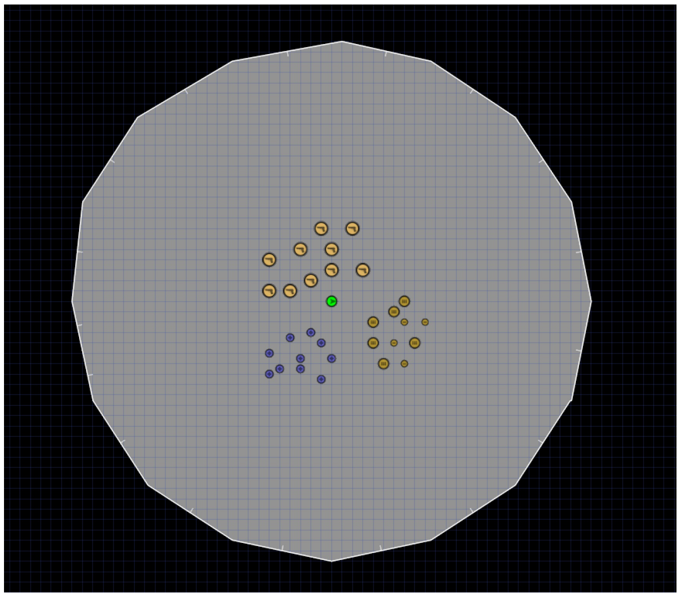
Figure 9: Dense reward

Events	Rewards	Remark
Kill enemy	+1	
Use Ammo	[-0.4, 0.0]	different reward for different ammo
Pick up ammo	[0.0, 0.4]	different reward for different ammo
Pick up new weapon	+1	weapon number increase/no punch
Lose Health	-0.3	
Picke up health pack	1	

Table 1: Wrong reward shaping

Events	Rewards	Remark
Kill enemy	+1.8	
Use Ammo	[-0.3, 0.0]	different reward for different ammo
Pick up ammo	[0.0, 0.2]	different reward for different ammo
Pick up new weapon	+0.3	weapon number increase/no punch
Lose Health	-0.3	
Picke up health pack	+0.2	

Table 2: Correct reward shaping

well, however, agent have not learned to pick up items in the map. We analyze this data curve, one possible reason is our agent could use the default gun to kill almost all enemy in one game (up to 25 kills / 30 bullets), so the agent might not try to pick up stronger weapon or healthy pick to add its health.

Phase Three

Because the previous results didn't show pattern about picking guns and healthy pick, so we want to apply the model

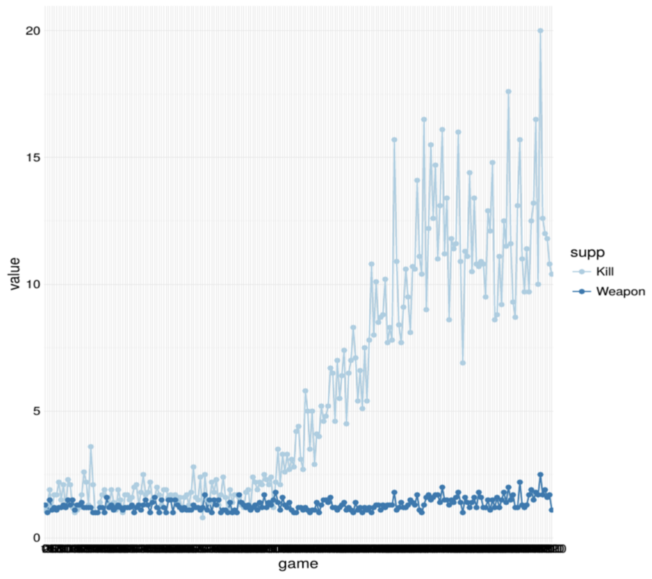

Figure 10: Correct reward shaping and map shaping

trained the into more complicated maps, agent can not use the default gun to fight against all enemy anymore, so the agent might force to pick up healthy pick and stronger guns. So we randomly arrange the supplements, as in fig11 Also, there are 4 types of monsters and we change the probability so that stronger monsters have a higher probability to appear.

Because this is one step in the whole curriculum training,

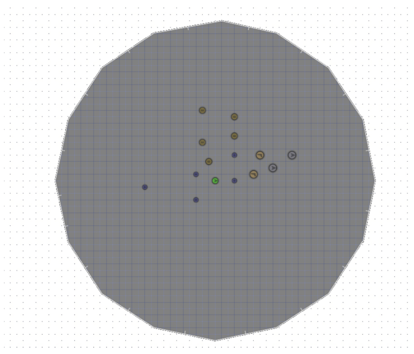
Figure 11: Harder level

so we take the model weights in last phase as initial weights in this phase. We pick weights in the meddle of last process, in order to avoid over-fitting. After 2300 game episodes, the training curve is as fig13. We can see that upon learning to kill, the agent can learn to pick up health pack as well as weapons. So it acquires the three skills.

Phase Four

The last curriculum training phase we experiment is on a random distributed items map. With more remote and stronger enemy, game become very difficult. Because this is one step in the whole curriculum training, also we take the model weights in last phase as initial weights in this phase. We pick

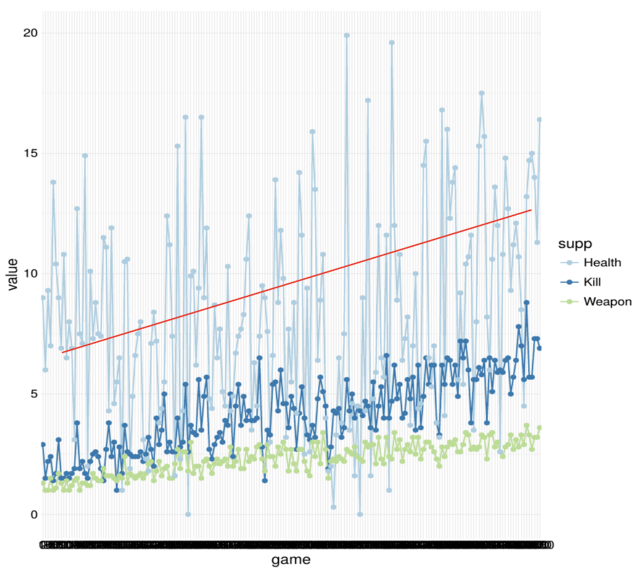

Figure 12: Training curve of phase 3

weights in the meddle of last process, in order to avoid overfitting. You can see the curves of this experiment in Fig , the number of agent pick up healthy pick, pick up stronger weapons, kill counts, all keep incresing. Also we can observe that those three datas show positive correlations, agent start to handle other tasks. This pattern didn't show in the previous works.

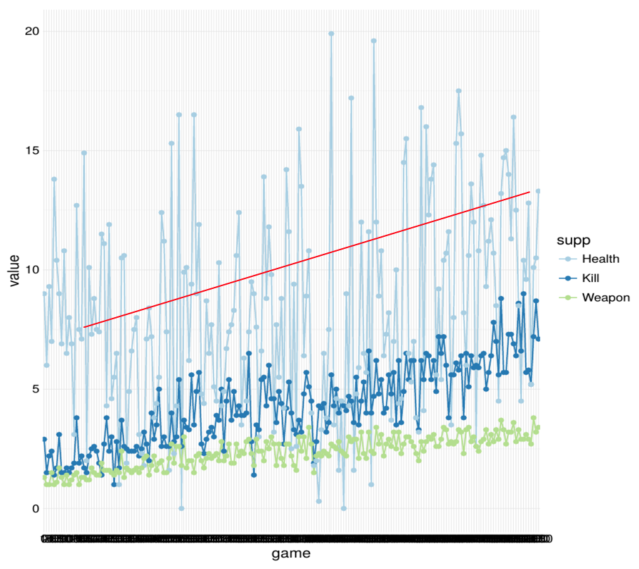

Figure 13: Training curve of phase 4

3.3 Evaluation

In ViZdoom games, the game environment vary in different experiments, which includes the shape of the map, the color of the decorations and the limitation of the numbers of supplements available. So, it is hard to find a commonly applied baseline except the CIG competition track. However, the tracks in CIG match often constrained the AI to just use 1 weapon, which is inconsistent with our aim of learning to collect the items. In our curriculum phase III, our data tells us that our agent not only learning to kill, but also learns to pick up weapons to strengthen them self. We trained our model on 1 TitanX gpu, the longest training time is 36 hours, much smaller than some other implementations (Arnold, Facebook F1 bot, several days), but still shows its capability to perform well on complex tasks. In this project, we just use human player as the baseline for evaluation, we need a better evaluation, so our future work is to train and transfer our model to the more recent works and to have a better evaluation.

4 Conclusion

3D FPS game seem hard to learn option policy in previous works, we should overcomes large state space and particle observation in 3D FPS games. We use Vizdoom as our simulation environment, Vizdoom is an open source FPS learning environment used by many previous relative works. Apart from previous work, we use DDRQN (Double Deep Recurrent Q-Network) in this paper, however, in our training experiments, we failed in two experiments. Sparse reward might cause agent hard to learn and get reward.After survey, we decided to use curriculum method to reduce the influence of sparse reward faced in task training. We did over 6 experiments among several tasks, analyze it's performance with baseline works and average human stat.We show that the proposed architecture substantially outperforms of this game as well as average humans in several scenarios.

References

[Bengio *et al.*, 2009a] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[Bengio *et al.*, 2009b] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[Dosovitskiy and Koltun, 2016] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[Florensa *et al.*, 2017] Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *CoRR*, abs/1707.05300, 2017.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Hausknecht and Stone, 2015] Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015.

[Held *et al.*, 2017a] David Held, Xinyang Geng, Carlos Florensa, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. *arXiv preprint arXiv:1705.06366*, 2017.

[Held *et al.*, 2017b] David Held, Xinyang Geng, Carlos Florensa, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. *CoRR*, abs/1705.06366, 2017.

[Karpathy and Van De Panne, 2012] Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence*, pages 325–330. Springer, 2012.

[Kempka *et al.*, 2016] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. The best paper award.

[Khamassi *et al.*, 2005] Mehdi Khamassi, Loïc Lachèze, Benoît Girard, Alain Berthoz, and Agnès Guillot. Actor–critic models of reinforcement learning in the basal ganglia: From natural to artificial rats. *Adaptive Behavior*, 13(2):131–148, 2005.

[Lample and Chaplot, 2016a] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. *CoRR*, abs/1609.05521, 2016.

[Lample and Chaplot, 2016b] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. *CoRR*, abs/1609.05521, 2016.

[Matiisen *et al.*, 2017] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *CoRR*, abs/1707.00183, 2017.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[Ng *et al.*, 1999] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[Pentina *et al.*, 2015] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5492–5500, 2015.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016.

[Wu and Tian, 2016] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. 2016.

Appendix

Map editor We design our curriculum map using the tools **SLADE** [3] this is an opensource map editor for doom-like game.

Our code Our code[4] is opensourced on github, we adapt to some basic implementation of DQN and DRQN. and implement our own training, testing, curriculum training interface.

[3]http://slade.mancubus.net/index.php?page=news

[4]https://github.com/jeffchy/Artificial-Idiot