# Deep reinforcement Learning with VizDoom

**Arpit Agarwal**
arpita1
Robotics Institute
arpita1@andrew.cmu.edu

**Yilun Chen**
yilunc1
Robotics Institute
chenyilun16@cmu.edu

**Chaojing Duan**
chaojind
Department of ECE
chaojind@andrew.cmu.edu

## 1 Introduction

We plan to tackle the issues of playing a First-Person-Shooting (FPS) game in a 3D environment, which involves a wide range of skills, such as navigating through a map, collecting items, recognizing and fighting enemies, etc. These characteristics cause it much more challenging than playing most Atari games. Furthermore, states are partially observable and the agent navigates a 3D environment in a first-person perspective, making the task more suitable for real-world robotics applications as well as providing the practical and the important value for our project.

## 2 Related Work

We briefly overview two recent published architectures trained successfully on Doom that may serve as baseline for our work.

[5] has proposed an AI-agent with the ability to remember previous states in order to select optimal actions and applied it to play the death matches in FPS games using only the pixels on the screen in partially observable 3D environments. Exploring the map to collect items and find enemies has been achieved in the first phase with simple DQN method and the second phase focuses on fighting enemies when they are observed based on DRQN method with game feature augmentation. This architecture is more suitable for real-world robotics application because of a wide variety of skills it could develop and purposes it could achieve.

[12] presented a framework based on A3C with convolutional neural networks (CNN) and trained an AI agent in Doom, wining the champion of Track1 in ViZDoom AI Competition 2016 by a large margin. This work used actor-critic models [9] to jointly estimate a value function $V(s)$ and a policy function $\pi(a|s)$. The former gives the expected reward of the current states and the latter provides a probability distribution on the candidate actions $a$ for the current state $s$.

## 3 Methods

To address the difficult task of Doom in deathmatch mode, we propose two key points: incorporate domain knowledge into the existing network; learn the complex goal by dividing it into subtasks and learning each subtask separately under Hierarchical DQN framework.

More specifically, we plan to respectively train the agent to learn to navigate through *wayhome* environment, pick up medicine through *health* environment, pick up ammo through *ammo* environment, turn and shoot enemy through *defend* environment. After the agents are trained with these basic skills, we train a high-level planning network to integrate these behaviors for deciding which behavior mode the agent should follow at each time step. We also use a pre-trained ImageNet classification model to promote training results.

### 3.1 Deep Q-Networks and Variants

We applied these methods in Doom to improve the performance of the basic DQN algorithm.

1. *DQN*: By applying the Bellman equation as an iterative update, the optimal Q-function can be represented
$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a')|s,a] \tag{1}$$
A Q-network can be trained by minimizing a sequence of loss functions at the current time step $t$,
$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'}(y_t - Q_{\theta_t}(s,a))^2 \tag{2}$$
2. *Double DQN*: Double DQN[10] removes upward bias caused by $\max_a Q(s,a,w)$. The current Q-network $w$ is used to select actions and the older Q-network $w^-$ is used to evaluate actions
$$I = (r + \gamma Q(s', \arg\max_{a'} Q(s',a',w), w^-) - Q(s,a,w))^2 \tag{3}$$
3. *Prioritized replay*: Prioritized replay[8] weights experience according to surprise. It store experience in priority queue according to Temporal Difference error. The prioritizing framework saves important transitions more frequently, and therefore learns more efficiently.
$$TD\ error = |r + \gamma \max_{a'} Q(s',a',w^-) - Q(s,a,w)| \tag{4}$$
4. *Dueling network*: Dueling network[11] split Q-network into two channels: action-independent value function V(s,v) and action-dependent advantage function A(s,a,w). The idea is to generalize learning across actions without imposing any change to the underlying DQN.
$$Q(s,a) = V(s,v) + A(s,a,w) \tag{5}$$

### 3.2 Exploration Policies

We tried three difference exploration policies to explore the environment in different ways.

1. $\epsilon$-*greedy*: Choose an epsilon and choose a random number. If the number is greater than epsilon, choose the max value action. Otherwise, choose a random action.
2. *softmax*: Choose a random number and select the action by a multinomial probability ordered by $P(a) = e^{Q(a)}/sum(e^{Q(a)})$.
3. *Shifted Multinomial*: Similar to softmax but chooses the action by the order.
$$Q_{shifted}(a) = Q(a) - \min(avg(\min(Q(a)), \min(Q(a))) \tag{6}$$
$$P(a) = \frac{Q_{shifted}(a)}{\sum(Q_{shifted}(a))} \tag{7}$$

### 3.3 Augmented Network

As reported in [5], typical DQN methods does not perform well on 3D-world Doom environment and the learned agents cannot detect enemies. So we propose to apply a learning methodology with pre-knowledge in 3D environment, which is more similar to the human intuition comparing with the related works mentioned above. Typically, we use a pre-trained ImageNet model to extract substantial features of the environment. We believe this method is plausible because we have observed high accuracy in classification rates of objects like enemy in Doom environment. After we subtract a good representation of the visual input, we augment it with other substantial game information like health and ammo to learn the proposed policy. This pre-trained network serves as the domain knowledge in Doom game and greatly relieves the difficulty of training.

### 3.4 Solve by Hierarchical Division

Learning goal-directed behavior in environments with sparse feedback is a major challenge for reinforcement learning algorithms. [5, 12] has reported the difficulty when the agent has to jointly learn to navigate and shoot simultaneously. The primary difficulty arises due to insufficient exploration and causes an agent to be unable to learn robust value functions. That's why we need to incorporate high-level planning into the reinforcement learning mechanism.

By following the ideas in [4], we divide the task of gaining high score into several subtasks: navigation, picking up medicine, picking up ammo and shooting enemy. We use hierarchical-DQN framework to integrate hierarchical value functions and operate at different temporal scales. The top-level value function learns a policy over intrinsic goals, and a lower-level function learns a policy over atomic actions to satisfy the given goals. We use functions over the detection entities of the ImageNet classification network as goal specifications. In this way, intrinsically motivated agents can explore new behavior for its own sake rather than to directly solve problems, which provides an efficient space for exploration in complicated environments.

# 4 Experiments

## 4.1 Dataset and Prepocessing

For our project, we develop upon Doom on OpenAI gym, which is a testbed for RL algorithms and supports AI bots to learn to play games through raw pixels. We obtain and create the dataset through interactions with the gym environment. Typically we construct tuples $e_t = (s_t, a_t, r_t, s_{t+1})$ in the replay memory, where $s_t$ refers to the states(4 recent image frames), $a_t$ refers to legitimate actions, $r_t$ refers to reward functions. The input of the network is the states and the output is our learned choices of actions, aka, the policy.

Our basic preprocessing procedure includes down scaling to $80 \times 60$, gray scaling, and frame skipping with parameter 4. These preprocessing steps are necessary for both computation efficiency and training stability.

## 4.2 Evaluation Method

The expected reward for 100 episode is used as the evaluation of our approach for subgoal learning.

For final evaluation purpose, we plan to use the commonly used Frags per episode, the number of killed minus the number of suicides for the agent in one round of game. An AI is stronger if its frags is ranked higher against others

## 4.3 Preliminary Result

Currently, We have trained our agent using DDQN algorithm in several simple scenarios, including *basic, defend, health* and *wayhome* environments. Figure 1 shows the reward curve during training in *basic, defend* and *health* environments. Figure 2 gives an empirical impression of the scenery in *basic, defend* and *health* environments, respectively.

We can see that in *basic* environment, the reward function converges fast to a high value and remains stable. In practice, the agent performs well in turning and shooting the enemy when reward is higher than 10. Though the learned reward function seems unstable in *defend*, the actual policy of the agent is acceptable and seems intelligent enough to find the enemy and shoot [1]. However in the *health* environment, we can tell from the graph that the agent actually learns nothing. It just follows the simple policy of going straight all the time. The reason may be that it can't detect the medicine on the ground, so it just follows a random strategy, which implies our idea of incorporating object detection algorithm in the network.

In *deathmatch* environment, it shows the poor performance with low reward on 100 episodes in Figure 3 when you train the typical DDQN algorithm without any advance exploration strategy. The learned policy just shoots all the time no matter there is an enemy in the front or not, as Figure 4 shows.

We also analyze the performance of using different exploration methods like $\epsilon$-greedy, shifted multinomial and softmax. We can tell from Figure 5 that there is hardly any difference of exploration efficiency in the Basic scenario. We argue this may be due to the simplicity of the environment.

[1] The visualization of the learned policy in *defend* environment after 10k episodes can be found here: `https://www.youtube.com/watch?v=nHHsWRd3qKI&feature`

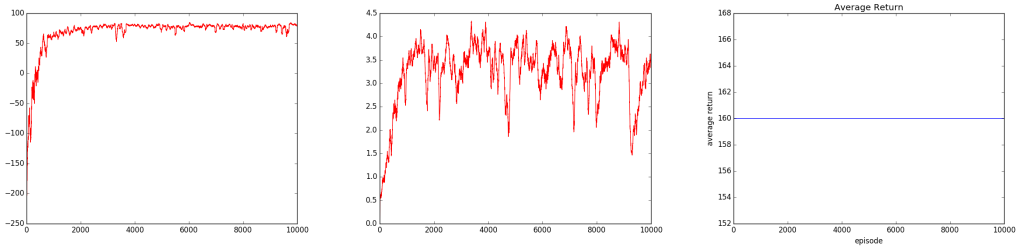


Figure 1: These three figures show the average training reward using DDQN algorithm with 10k episodes.From left to right, the agent is trained in basic, defend and health environment respectively.

Figure 2: These three figures show the Doom scenarios 1)Basic environment: Skill development are aiming, moving left and moving right 2)Defend the center: Skill developed are aiming, turning left and turning right 3)Health Gathering environment: Skill developed are collecting things and navigation

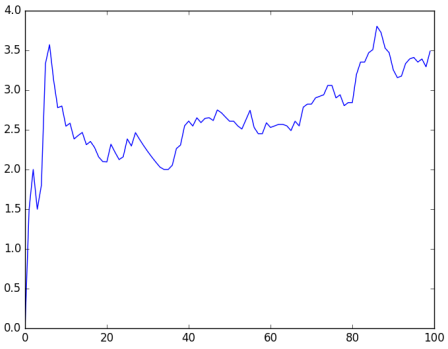

Figure 3: Average reward for Death match scenario for 100 episodes. The agent receives reward +1 for killing the monster

Figure 4: Preprocessed input images from Death Match scenario

## 5 Future work

### 5.1 Timeline and Basic Analysis

1. *3 Oct. - 7 Nov.* Train DQN, Double DQN and other existing DQN algorithms in OpenAI gym Doom environment. Compare the results obtained by different methods. Design game specific methods like Augmented Network and Hierarchical DQN.

   We have met the goal and actually gone farther.

2. *8 Nov. - 15 Nov.* Try Bootstrapped DQN[6], the core idea of which is randomizing value functions, leading to a promising approach to efficient exploration with good generalization.

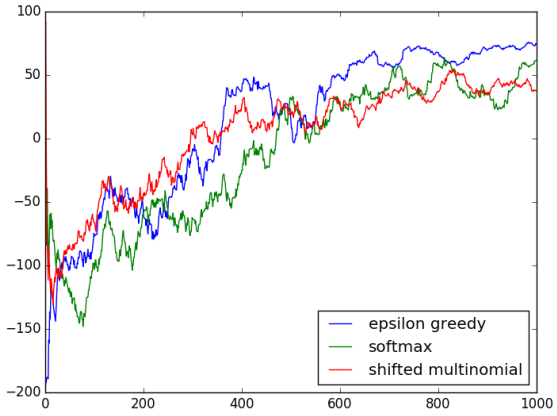


Figure 5: Reward curve over different exploration strategies on Basic scenario in Doom.

Compared with Bootstrapped DQN, Variational Information Maximizing Exploration (VIME) [7] requires a Bayesian neural network which is incompatible with our current architecture.

3. *16 Nov. - 23 Nov.* Investigate the topic of memory-based Deep Reinforcement Learning with Doom.

   The state of art DQN with temporal architectural like DRQN[3] and its variations simply model the DQN with one LSTM/RNN extra layer, leading their low performance when long term memory is needed.

4. *24 Nov. - 30 Nov.* Try to apply the ideas of Neural Turing Machine[1] and its recent advance[2] in RL to our architecture.

   By introducing an extra space for memory storage, the agent may possibly be able to rely on the previously obtained information.

5. *1 Dec. - 2 Dec.* Wrap up our work, write the paper and prepare for the poster session.

6. *3 Dec. - 8 Dec.* Discuss with TAs and modify our work according to the advice. Complete the final report and the poster.

### 5.2 Division of Work

- Arpit: Programming and in charge of the architecture of the codebase, realization of the algorithms and training. Investigates new ideas.
- Chaojing: Programming and contributing to codebase in visualization. Paper writing. Lead discussion and communication.
- Yilun: Programming and contributing to the realization of the algorithms. Paper writing. Investigating new ideas.

## References


[1] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[2] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[3] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[4] Tejas D Kulkarni, Karthik R Narasimhan, Ardavan Saeedi, and Joshua B Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *arXiv preprint arXiv:1604.06057*, 2016.

[5] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. *arXiv preprint arXiv:1609.05521*, 2016.

[6] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

[7] Yan Duan John Schulman Filip De Turck Rein Houthooft, Xi Chen and Pieter Abbeel. Variational information maximizing exploration. *arXiv preprint arXiv:1605.09674v2*, 2016.

[8] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[9] Sutton Richard Stuart. Temporal credit assignment in reinforcement learning. *Available from ProQuest Dissertations and Theses Global*, 1984.

[10] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR, abs/1509.06461*, 2015.

[11] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[12] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. *conference paper at ICLR under review*, 2017.