An approach to make AI Agent play DOOM

Kartik Thakral
PhD19004

Dimpy Varshni
MT19022

1. Introduction

Reinforcement learning algorithms have allowed AI agents to perform well on Atari games with a human-level better performance. Training AI agents on games are considered to be good because this allows agents to explore real-world like environment. In recent years, first-person shooter games particularly Doom have been explored much in the research area because of its resemblance to realistic 3D world environment. To help researchers explore more in the domain, a platform for developing deep reinforcement learning techniques was proposed in 2016 called Vizdoom which allows developing AI bots in somewhat limited 2D like and 3D environments. In this project, we explored deep Q-networks for these two environments of Doom.

2. Methodology

Two different versions of deep reinforcement learning algorithms are used for training agents : (i) Deep Q- networks for training agent in 2-D like environment [1] and (ii) Dueling Double Deep Q-Networks for training agent in 3-D like environment [2]. Overview of implementations of both of these networks are given in section below.

2.1. Deep Q-Networks

Main components of DQN implemented to train agent are as follows:

2.1.1 Preprocessing of frames

This stage deals with the preprocessing of frames before feeding them into DQN, which involves first converting rgb frames into grayscale frames followed by cropping the screen to remove the roof portion of the frames as it doesn't contribute anything to agent learning. The image is further normalized and resized. The resulting output frame from this stage has size of 84x84.

2.1.2 Stacking the frames

Single frame is not the only input to DQN but the stack of frames of size 4, this is done to overcome the problem of temporal limitation.

2.1.3 Deep Q-network using Convolutional layers

The Deep Q-network used for the purpose consists of three convolutional layers, details of which can be seen in Figure 1. Batch Normalization and Elu is used as an activation function. The convolutional layers are followed by one fully connected layer consisting of 512 hidden units and output layer consisting of 3 hidden units (as there are 3 actions possible - left, right and shoot).

2.1.4 Experience Replay

Experience replay is used in DQNs is used in the implementation to preserve previous experiences played by agent in memory and also random sampling is done to address the problem of correlation between experiences.

Figure 1. DQN is used for training agent on this environment

2.2. Dueling Double Deep Q-Networks

The implementation of this network also utilizes subsection 2.1.1 and section 2.1.2 of section 2.(Figure 2). As the environment is different from the 2-D like, frames are resized here to 100x120 in pre-processing stage.

Deep Q-Network and Target Network used in the DDDQN is same as used in 2-D like environment.

Along with this, prioritized experience replay [3] was used with DDQN network that is used to improve the performance of these networks.

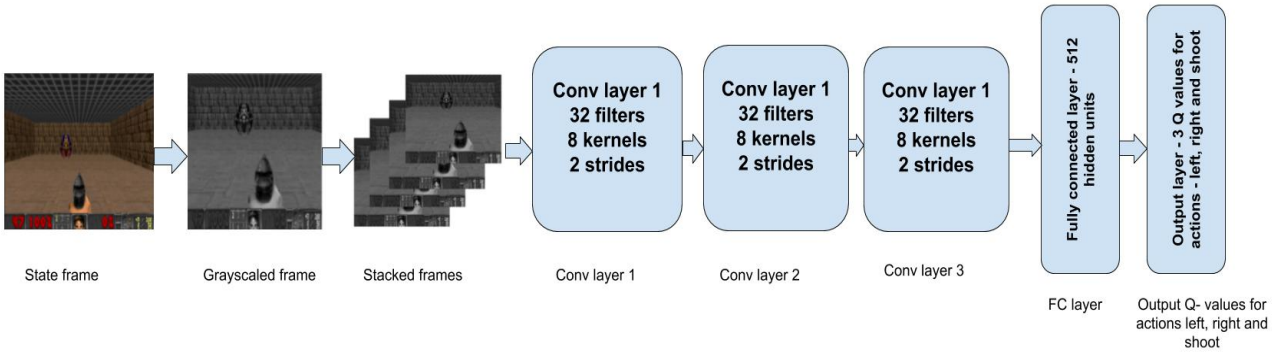

Figure 2. Deep Q - Network average rewards per 50 training episodes of total 950 finished episodes in total 1000 training episodes

3. Results and Analysis

Average Results obtained by both trained agents is shown in Table 1. It was observed that for 2-D like environment agent was able to learn well enough to score good and choose right action for a given state. Performance of the network and agent while training on 1000 episodes for finished episodes can be seen in figures 3 and 4.

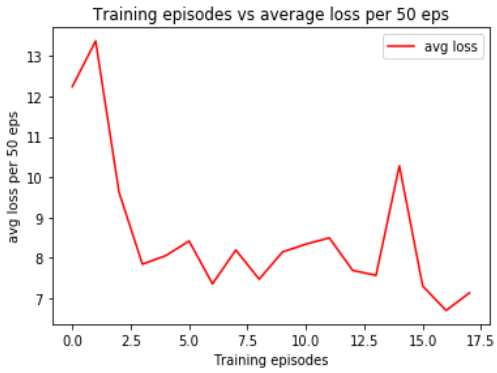

Figure 3. Deep Q - Network average loss per 50 training episodes of total 950 finished episodes in total 1000 training episodes

Although, the target for 2-D like environment was achieved with good performance of agent but it was not the case for 3-D like environment. Because of the limitation in the available hardware resources, it became difficult to train Dueling Double Deep Q-Networks with desirable settings, although when hyperparameters were adjusted then we were able to train the agent but it didn't give satisfactory

Figure 4. Deep Q - Network average rewards per 50 training episodes of total 950 finished episodes in total 1000 training episodes

results for the purpose. Through the visualization it was observed that in one such adjustable setting agent learned to kill one of the two enemies before moving forward resulting in negative reward while in the other setting agent learnt to move forward but without killing any of the two enemies resulting in positive reward as the reward for the Deadly Corridor environment is the distance between the agent and the vest. The performance of the trained agent on Deadly Corridor environment for 200 training episodes is shown in Figure 5 and 6. It is visible that agent's learning direction was not wrong in the adjustable settings.

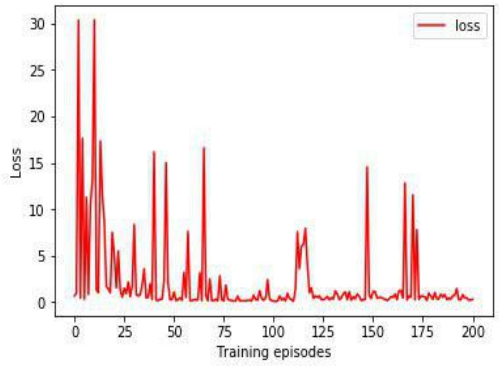

Figure 5. DDDQN Network loss while training agent on 200 episodes of total 5000 training episodes

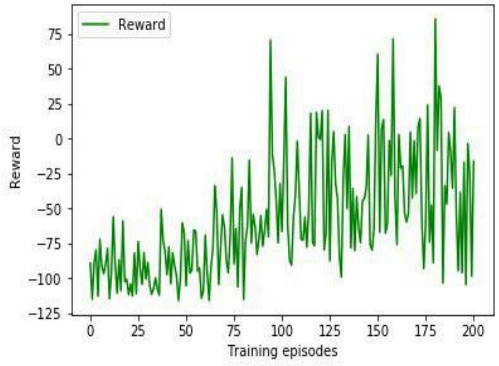

Figure 6. DDDQN Network reward while training agent on 200 episodes of total 5000 training episodes.

4. Relevant details about DQN algorithm implemented

- Learning rate for DQN : 0.0002
- Discount factor : 0.95
- Total training episodes : 500
- Maximum Steps per episode while training : 100
- Batch size for neural network input : 64
- Memory size for Replay buffer : 100000

5. Relevant details about DDDQN algorithm implemented

- Learning rate for DQN : 0.00025
- Discount factor : 0.95
- Total training episodes : 5000
- Maximum Steps per episode while training : 100
- Batch size for neural network input : 30000
- Memory size for Replay buffer : 30000

6. References

1. https://www.cs.toronto.edu/ vmnih/docs/dqn.pdf
2. https://arxiv.org/pdf/1509.06461.pdf
3. https://arxiv.org/pdf/1511.05952.pdf