# Experiment

For this project, I tried to implement and train an agent to successfully solve “LunarLander-v2” in OpenAI Gym[[1]](#footnote-2). The objective of Lunar Lander is maximizing your score by landing on the landing pad and saving the fuels. To train an agent to solve this game, I followed the DQN algorithm from the paper “Human-level control through deem reinforcement learning”[[2]](#footnote-3). To approximate the Q-function, I used PyTorch with two hidden linear layers with 100 dimensions. I also used following parameters for the training and will compare these with other values later.

1. Target update: 10
2. Gamma: 0.99
3. Batch size: 20

# Graph: reward at each traing episode while training my agent

Chart

Description automatically generated

At the beginning, the agent couldn’t get high scores as much as 100. As we can see the score increased around episode No. 100. However, it decreased temporally after episode No. 200. And it increased again after episode 350 and stabilized. Since it returned goo result, I stop the training the agent.

# Chart, line chart, histogram Description automatically generatedGraph: reward using trained agent

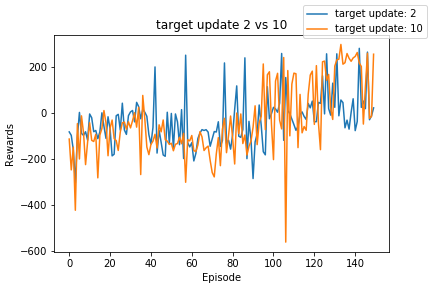
# Since we are using the agent trained from the previous example, the score is much higher and stabilized compared from the beginning compared to the previous one.

# Tuning parameters

Regarding the parameter tunings for the algorithm, I chose three parameters. 1) Dimensions of each layer 2) Gamma value 3) Batch size. Followings are the range of each parameter:

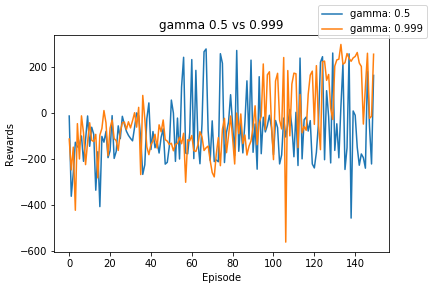
1. Target update: [5, 10]
2. Gamma: [0.7, 0.99]
3. Batch size: [1, 20]

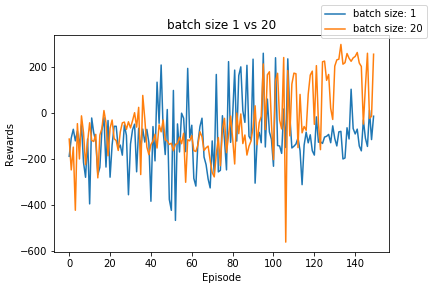
Result is the following parameters showed the best performance for 150 episodes.

1. Target update

Target update didn’t show much difference at the beginning. However, the target update of 10 showed better performance as the training progress.

1. Gamma

  
Gamma value didn’t show much difference at the beginning, but gamma value with higher value showed better performance as the training proceeds.

1. Batch size

Batch size of 20 showed better performance. However, since as the batch size increase, it took more time to train the agent.

# why i chose the DQN algorithm

First of all, it is one of the most famous algorithm to train agents to play video game without specific model. This means, the agent can learn without knowing about the environment. Also, status in video game is pretty huge, and to solve this with table algorithm, it takes a lot of storge. The DQN algorithm help us to approximate the function without saving all information about the status and rewards of the actions.

1. https://gym.openai.com/envs/LunarLander-v2/ [↑](#footnote-ref-2)
2. https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf [↑](#footnote-ref-3)