Using Reinforcement Learning to play Breakout

NOTE: Environment.yml file is attached so the results can be exactly replicated. If there are any problems during your run, please create a conda environment using this. I used PyTorch version 1.5.0 for this assignment. I don't have a working GPU on my laptop so the model evaluation was done on CPU. The model is capable of being tested on the GPU as well.

References

- https://jonathan-hui.medium.com/rl-dqn-deep-q-network-e207751f7ae4
- https://medium.com/@markelsanz14/introduction-to-reinforcement-learning-part-4-double-dqn-and-dueling-dqn-b349c9a61ea1
- https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
- https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
- https://arxiv.org/abs/1511.06581

Training Process

To try to solve this challenge, I tried using Dueling DQN. Based on the paper I read, the training process for Dueling DQN and Vanilla DQN was the same. The only difference was in the model architecture. The Dueling DQN was shown to have performed significantly better than vanilla DQN in the published paper so I decided to jump to that implementation. The model architecture was the same as the one described in the official paper.

```
(conv1): Conv2d(4, 32, kernel_size=(8, 8), stride=(4, 4))
(conv2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
(conv3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
(fc1): Linear(in_features=3136, out_features=512, bias=True)
(V): Linear(in_features=512, out_features=1, bias=True)
(A): Linear(in_features=512, out_features=4, bias=True)
```

I started training on the Dueling DQN with the following parameters:

- Batch size: 32
- Buffer length: 20,000
- Start Learning: 5,000 steps
- Episodes: 25,000
- Epsilon decay steps: 100,000
- Minimum epsilon: 0.025
- Learning rate: 1.5 x e-4

- Loss function: Huber loss

I got the following graph

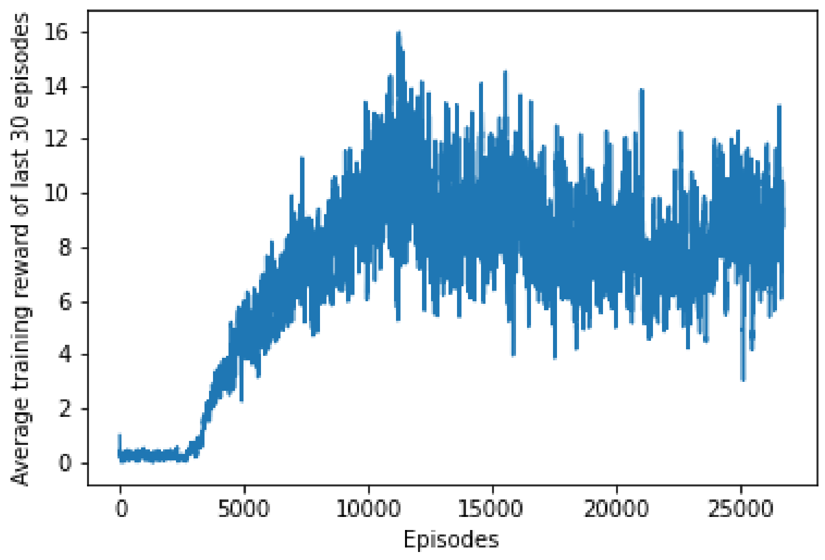


Figure 1: First attempt with the Dueling DQN

The training resulted in the following graph with the best results around the 11,000th episode. To verify if the results were not a one-off fluke, I conducted the experiment again with 20,000 episodes

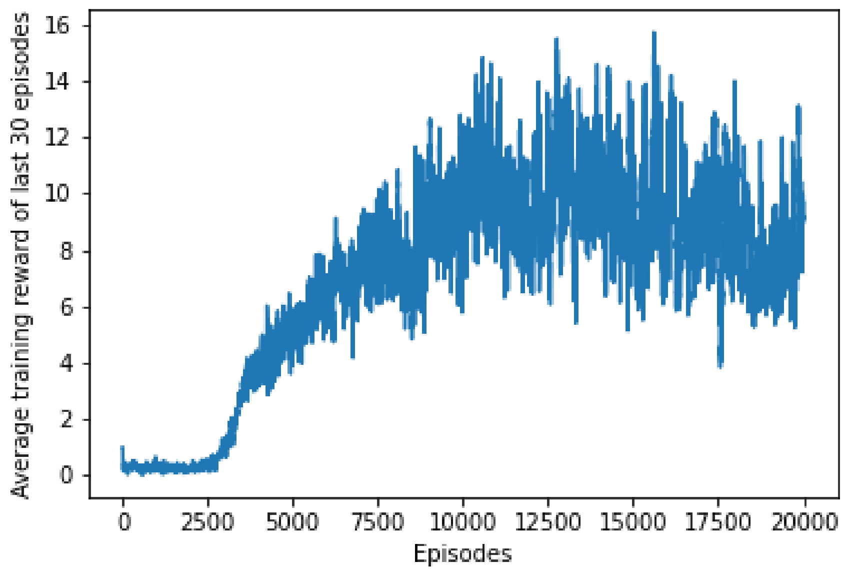


Figure 2: Second attempt with the Dueling DQN and the submission training graph

The best results were achieved between 13,000 and 13,500 episodes.

Looking at both figures 1 and 2, it can be seen that the model starts plateauing between 10,000 to 15,000 episodes.

I tried improving the results by changing the following parameters from the previous run

- Buffer length: 100,000
- Start Learning: 10,000 steps
- Epsilon decay steps: 300,000

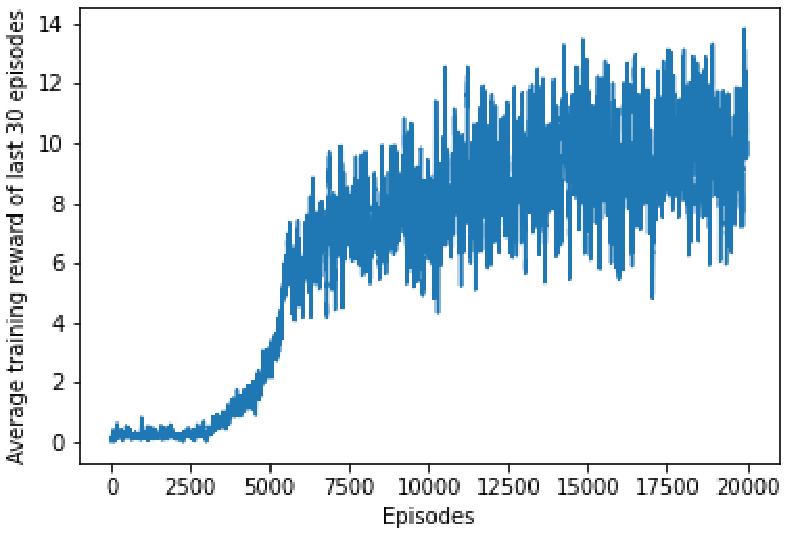
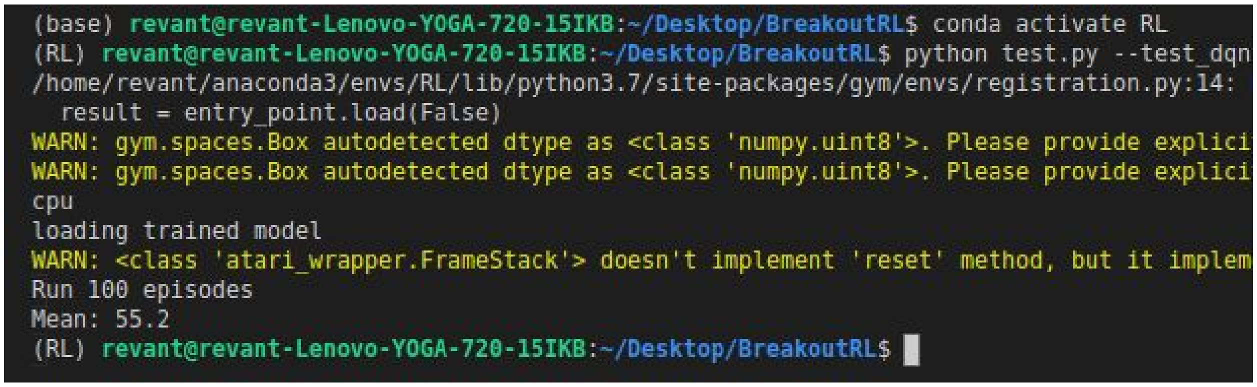


Figure 3: Third attempt with the Dueling DQN

This model did not offer any significant improvements over the previous models. Granted it was still training and did not plateau at 20k episodes, the testing results were not that significant. In the interest of time and resources, I used the model saved at the 13,000th episode in the **second** attempt (figure 2). It is saved as **test_model.pth**. The model gave a reward value of **55.2.** The screenshot is attached below:-



```
(base) revant@revant-Lenovo-YOGA-720-15IKB:~/Desktop/BreakoutRL$ conda activate RL
(RL) revant@revant-Lenovo-YOGA-720-15IKB:~/Desktop/BreakoutRL$ python test.py --test_dqn
/home/revant/anaconda3/envs/RL/lib/python3.7/site-packages/gym/envs/registration.py:14:
  result = entry_point.load(False)
WARN: gym.spaces.Box autodetected dtype as <class 'numpy.uint8'>. Please provide explici
WARN: gym.spaces.Box autodetected dtype as <class 'numpy.uint8'>. Please provide explici
cpu
loading trained model
WARN: <class 'atari_wrapper.FrameStack'> doesn't implement 'reset' method, but it implem
Run 100 episodes
Mean: 55.2
(RL) revant@revant-Lenovo-YOGA-720-15IKB:~/Desktop/BreakoutRL$
```