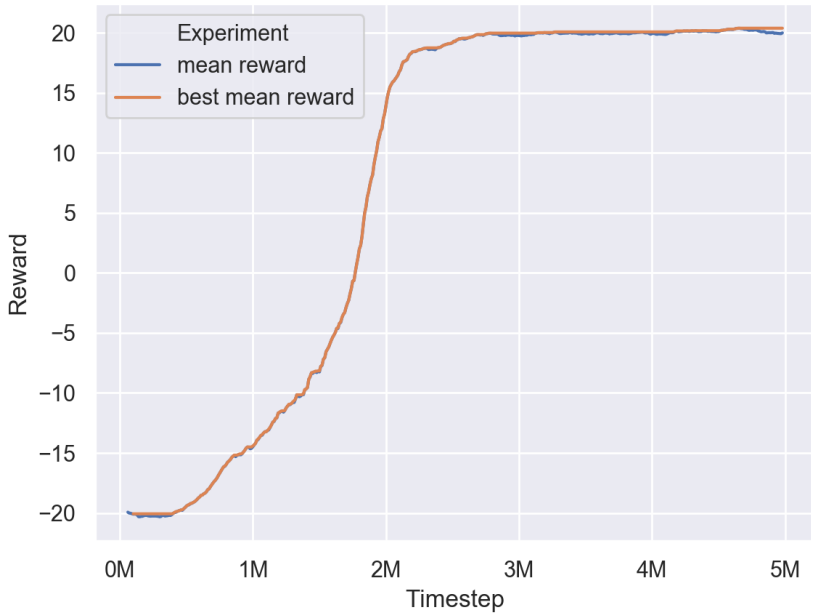
Berkeley DRL HW3

Anton Makiievskyi

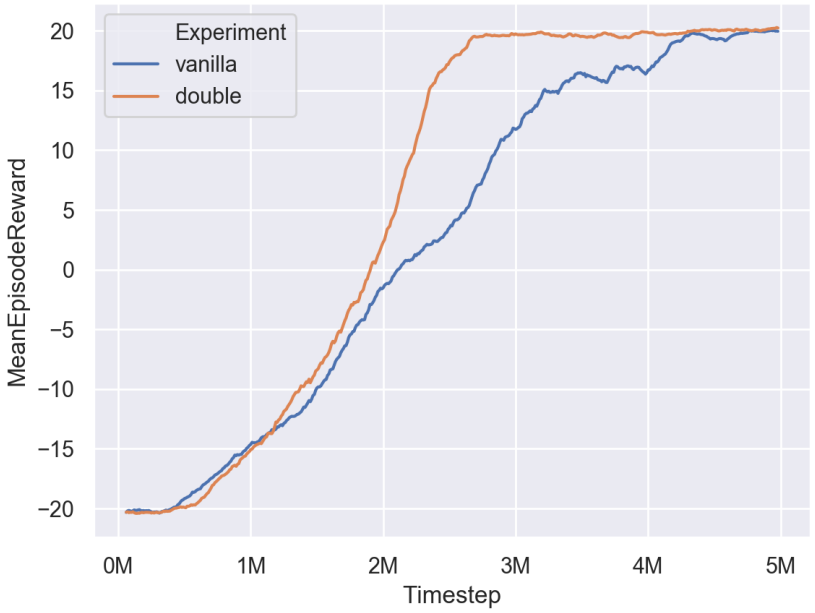
PART I

Question 1: Basic Q-Learning Performance

I ran the algorithm with the default parameters. The mean 100 episode reward is almost indistinguishable from the best mean reward



Question 2:
Below is the comparison of the double DQN to vanilla DQN using the modified network architecture which performed a little better then the default one

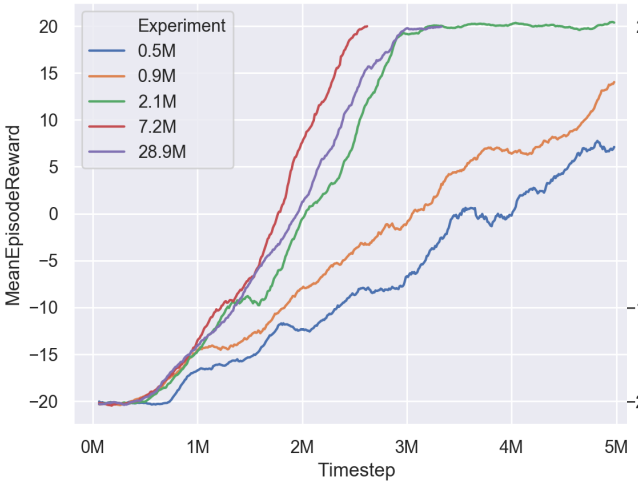


Question 3:
I chose to experiment with the network architecture in an attempt to decrease the time for the network to train . The default architecture took ~6hours to reach the reward of 20 in the best case and it has ~8M parameters. Reducing the network size to 0.5M parameters did not speed up the training significantly. I concluded that most of the time must be consumed by calls to the game emulator.

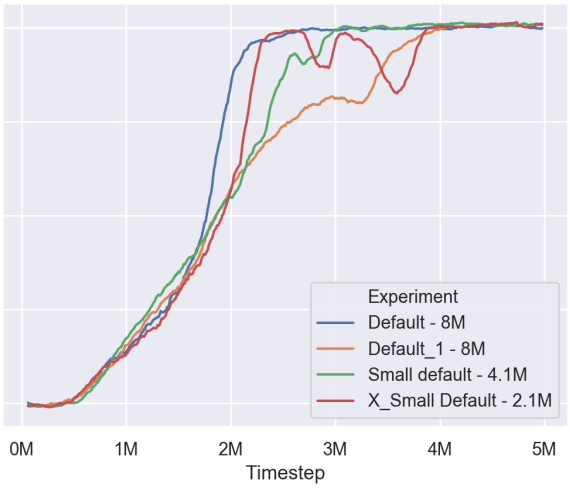
I used 6 more architectures with 0.5M, 0.9M, 2.1M, 4.1M, 7.2M and 28.9M parameters. There was a quite clear trend between the number of parameters and the speed of convergence. However after doing the additional run of the default architecture it showed a very different and much improved performance (best overall). It means that any conclusions should be made with caution, and each network should probably be tested on several different random seeds to make a meaningful conclusion.

Below are the learning curves for the different architectures with the number of trainable parameters specified in the legend. The actual architectures can be seen in the file run_dqn_atari.py. I also include the table of time it took for each network to reach the reward of 20.

2 convolutional layers models:



3 Convolutional layers models



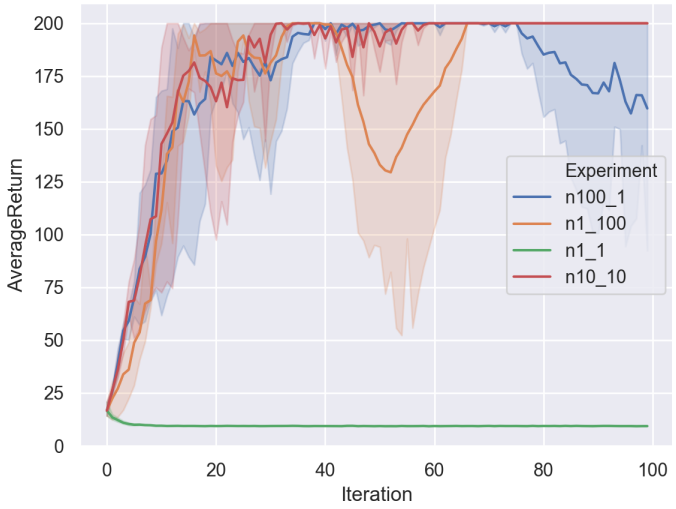
Number of parameters in the network in Millions	Time to reach mean reward of 20 in minutes
0.5M	Never
0.9M	Never
2.1M	318
7.2M	329
28.9M	345

Number of parameters in the network in Millions	Time to reach mean reward of 20 in minutes
2.1M	387
4.1M	297
8M	307
8M	411

PART II

Question 1:

The run with 10 target updates and 10 gradient steps per update reached the maximum reward the fastest and also was more stable. Evidently other two setting were unbalanced and didn't provide either enough gradient steps to reach a target or enough target updates to provide good targets.



Question 2:

Below are the learning curves for the actor-critic algorithms on HalfChettah and InvertedPendulum environments