CS294-112 Deep Reinforcement Learning HW3: Q Learning and Actor Critic

Nathan Lambert, nol@berkeley.edu,

October 5, 2018

1 Q-Learning

Question 1: basic Q-Learning Performance:

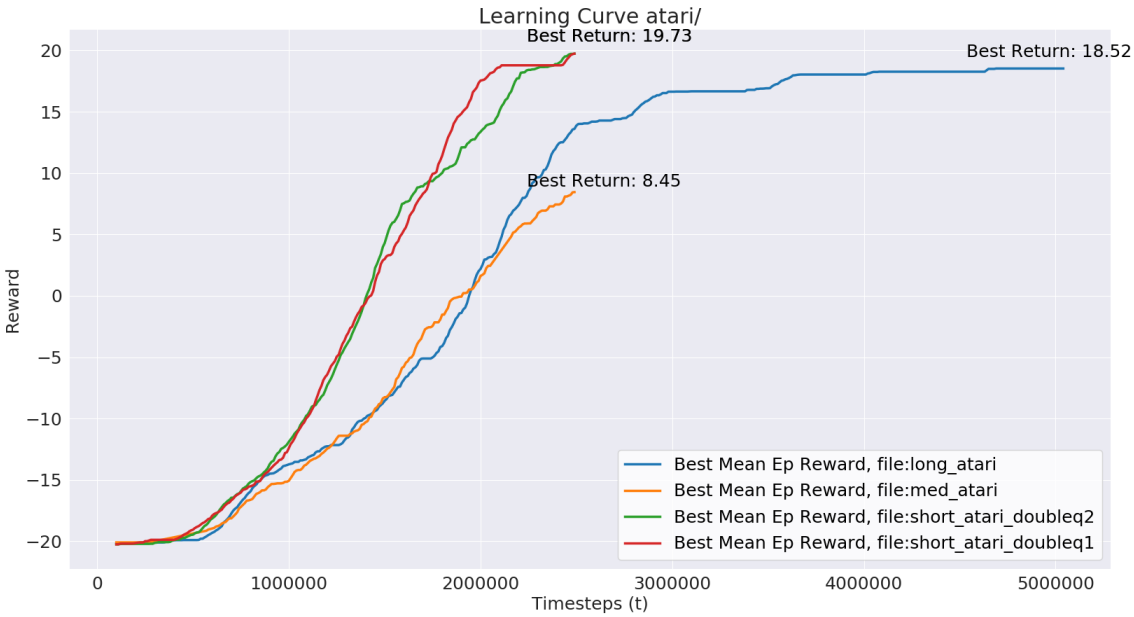

Figure 1: Q learning on images. Across a couple different seeds there is substantially different performance, but it definitely works!

Question 2: double Q-Learning:

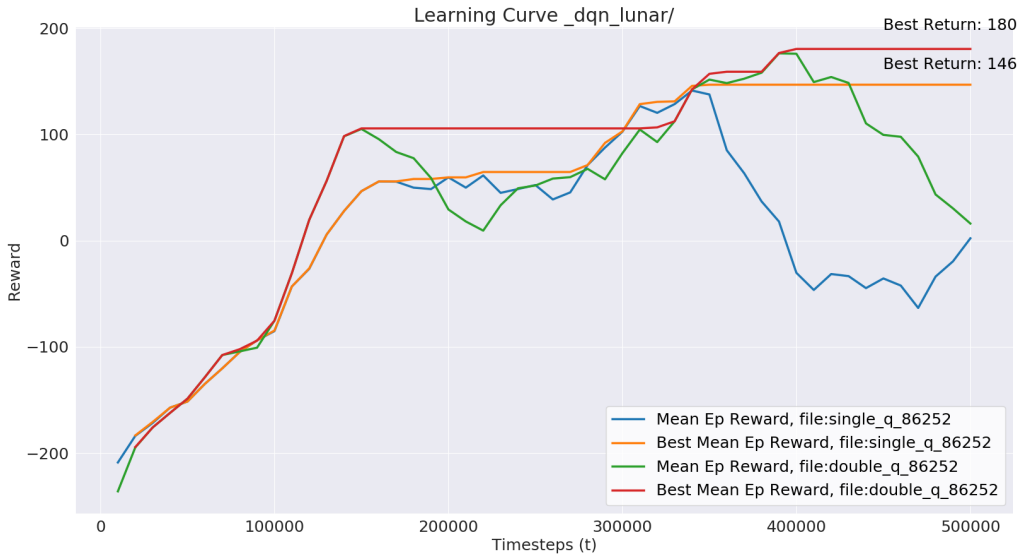

Figure 2: DQN on lunar lander. Double Q learning learned slightly faster and reached a higher ceiling on this task as expected.

Question 3: hyper-parameters:

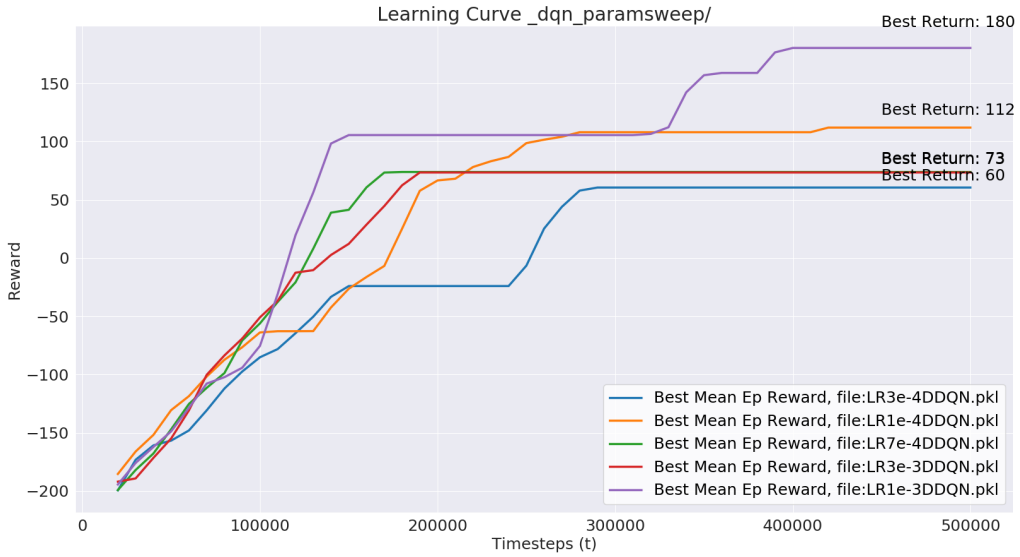

Figure 3: Learning rate comparison. Used lunar lander. This plot shows the sensitivity of learning rate. The best parameter by a substantial amount was the included parameter, and there is a noticeable dropoff when deviating from that.

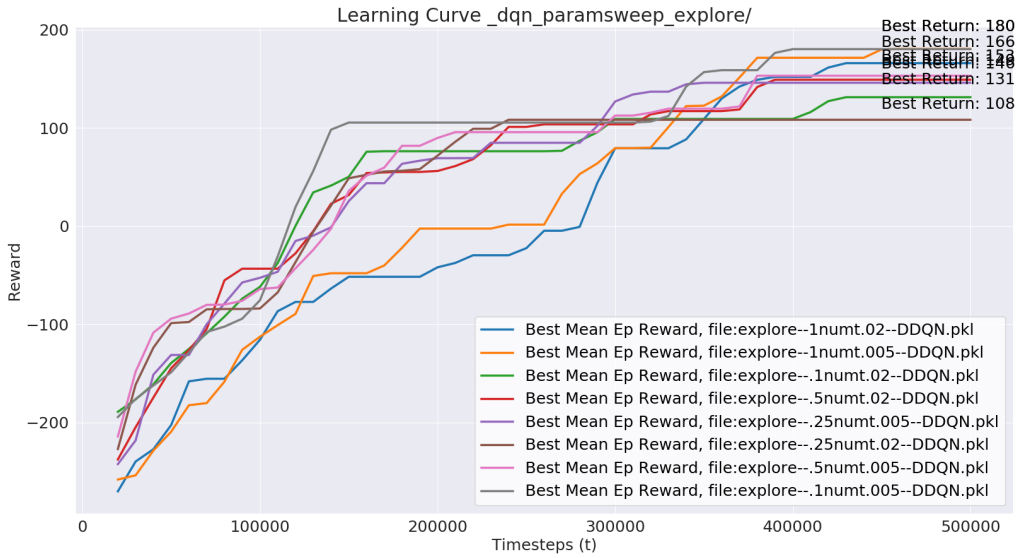

Figure 4: Explore rate comparison. Used lunar lander. Here the biggest takeaway is the difference between the upper bunch of lines and the blue/orange pair. The Blue/Orange pair is slower to learn as expected because the explore rate decreases substantially lower. Additionally upon closer inspection, you can see that the policies with lower converged exploration rate has a higher steady state reward, as expected. My naming convention is explore– AnumbB. Here, A is the percentage of the data the explore rate decreases over and B is the final explore rate.

2 Implement Actor Critic on Top of Policy Gradient

Question 1: Cart Pole Sanity Check:

Figure 5: Comparison of the number of time steps to gradient steps learning curves. It makes a lot of sense that you need multiple gradient steps per time update, and using 1 time step per update is wasting compute power. The best values given to us are 10 & 10, but I'm sure better parameters could be found

Question 2: More Difficult Tasks:

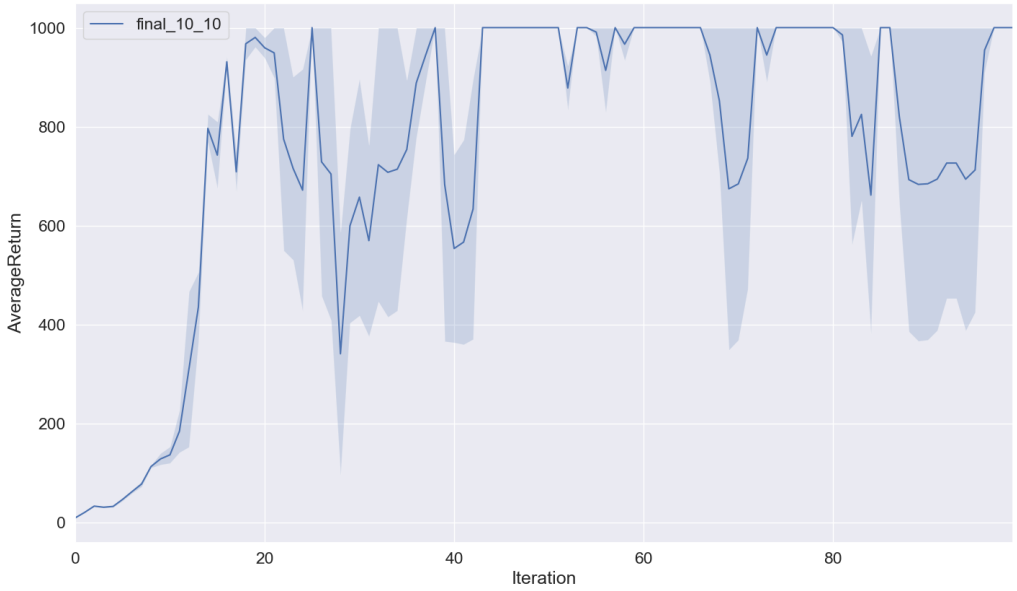

Figure 6: The **Inverted Pendulum** learning curve is similar to hw2

Figure 7: The **half cheetah** curve is similar to hw2.

3 Bonus

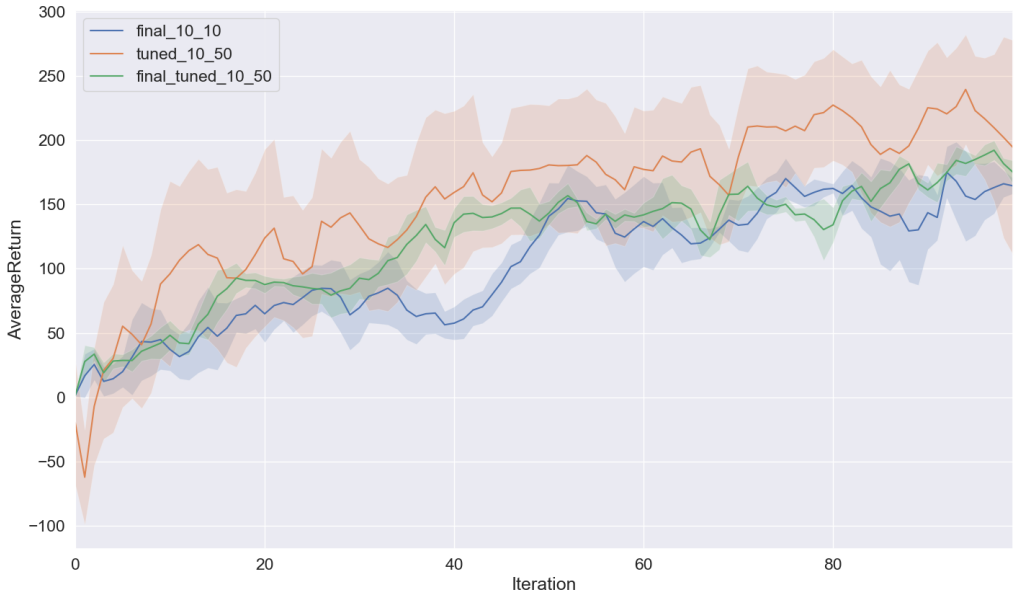

Figure 8: I did a very limited amount of parameter tuning to try and improve the half cheetah performance. I simply added two more hidden layers, doubled the hidden width, and added more gradient steps per target update and the performance got noticably better, but not by a crazy margin.