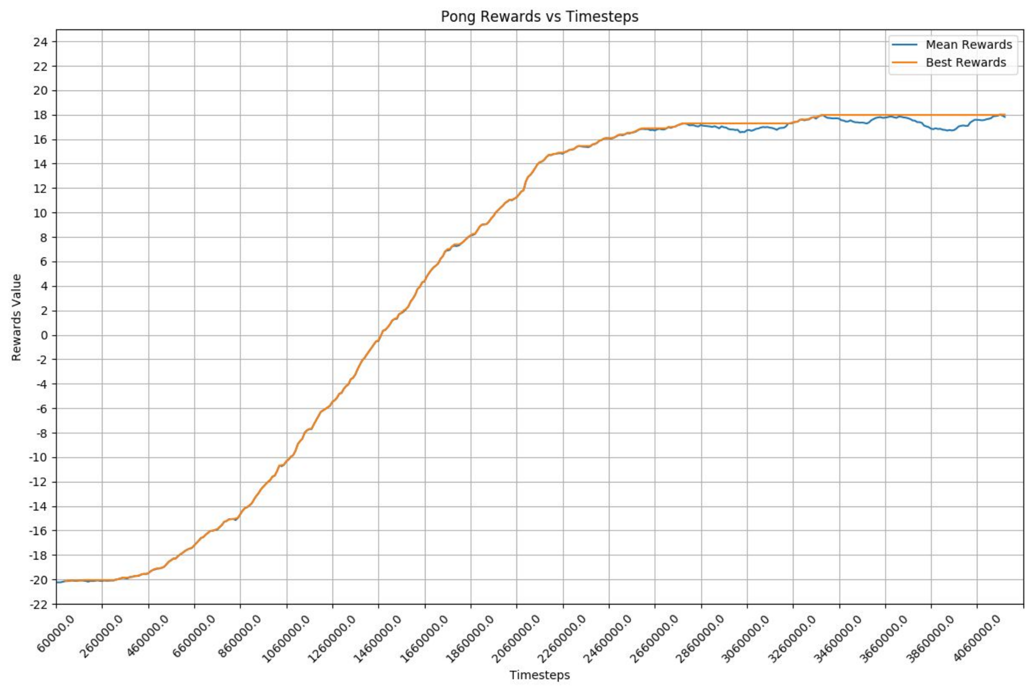
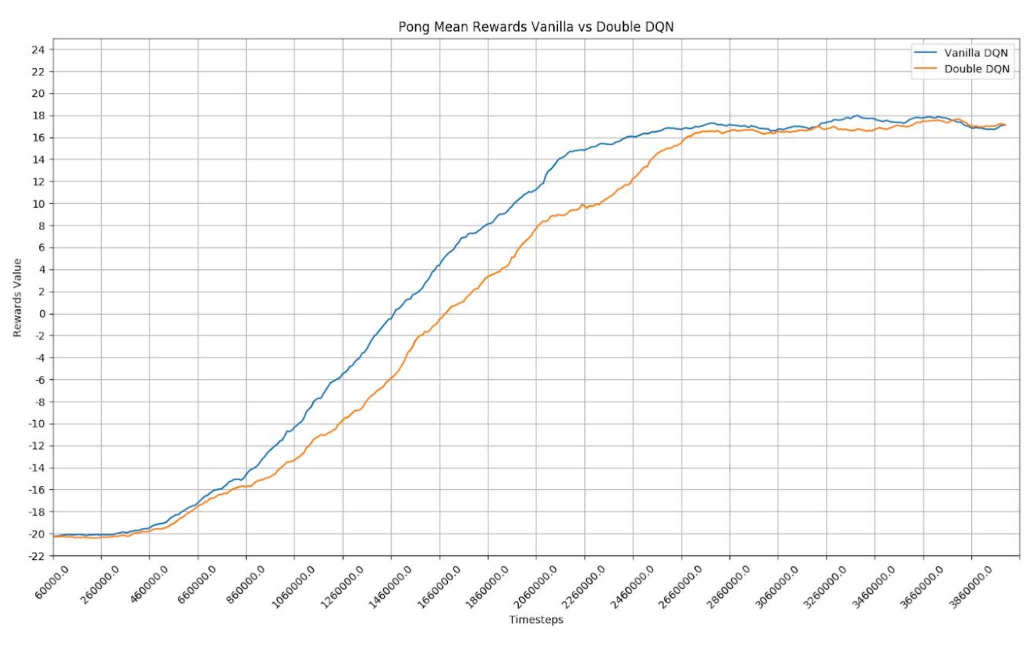
DQN : I used default settings for all questions. I hard coded writing and reading of the csv file for plotting. Question 2 and 3 were based on double Q-learning.
Usage: python q1_plot.py/q2_plot.py/q3_plot.py to plot graphs for dqn. Need to manually modify file paths.
Question 1: *default settings of hyperparameters, dqn_atari for ~4 million timesteps*

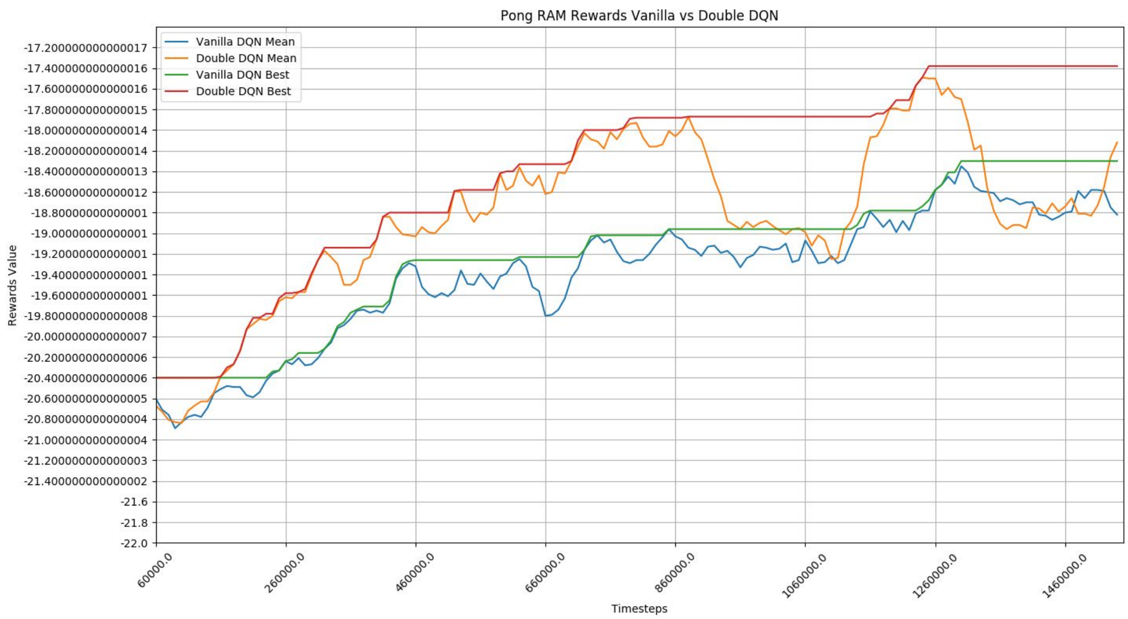


Question 2: *default settings of hyperparameters, dqn_atari for ~4 million timesteps*

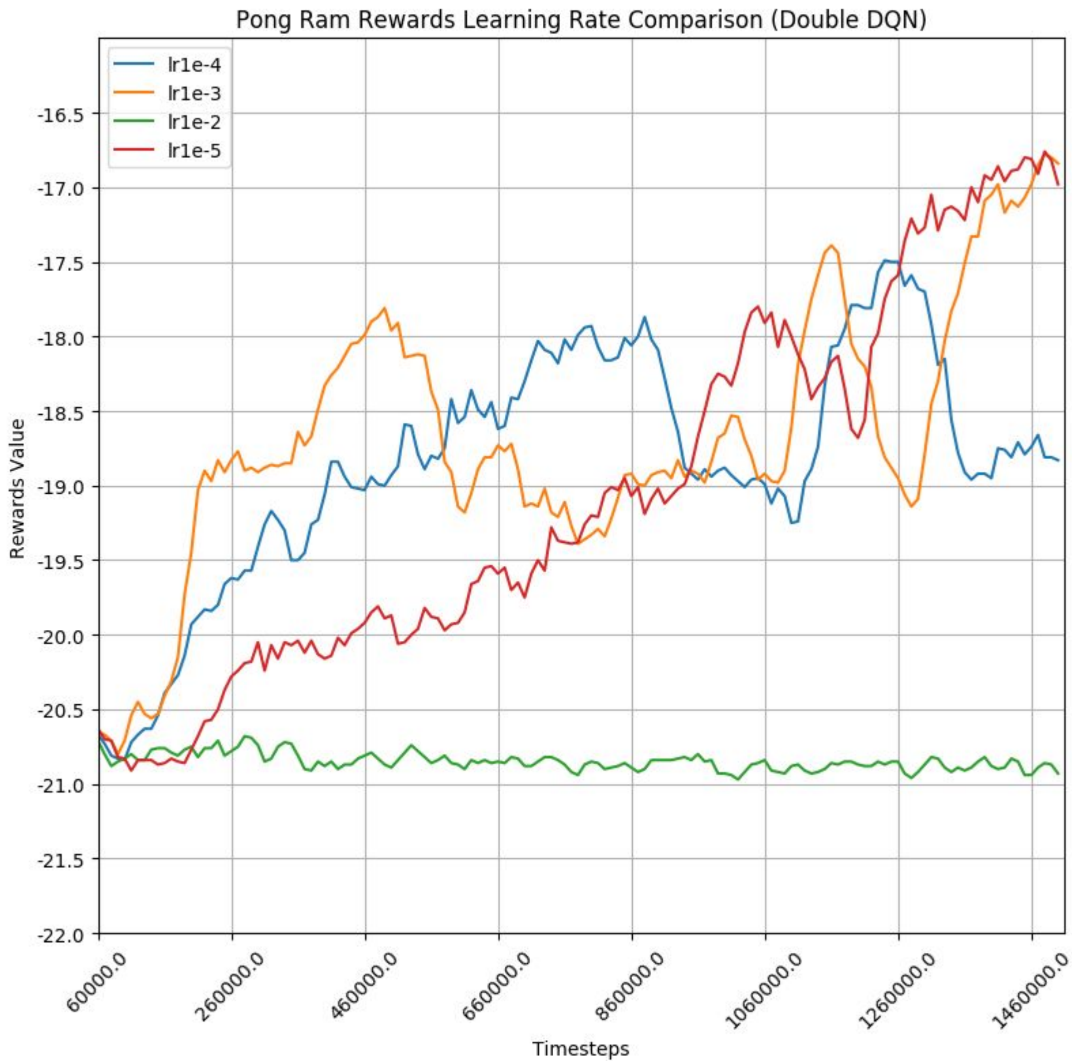


Actually I was not sure whether my algorithm was wrong or environment’s variance caused double Q-learning to be worse than vanilla DQN in early stages of training (but it did surpassed vanilla’s mean best rewards near the end of 4 million timesteps). Due to time constraints, I ran the comparison in dqn_ram again and double Q-learning was obviously outperforming vanilla DQN. So I think my implementation of double Q-learning should be correct.

default settings of hyperparameters, dqn_ram for ~1.5 million timesteps

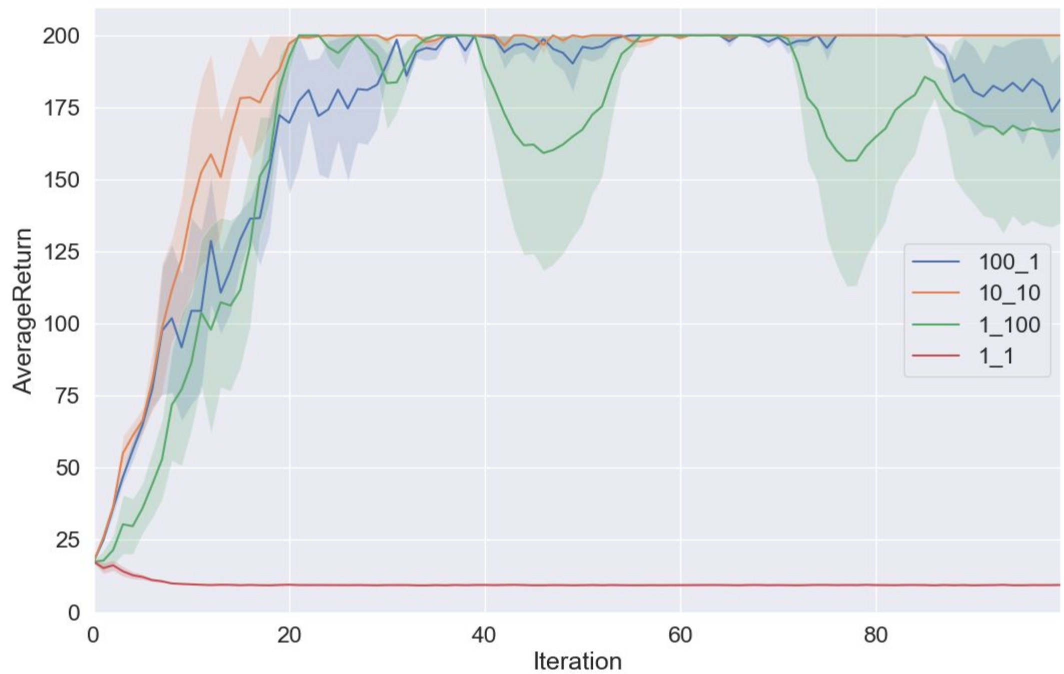


Question 3: *default settings of hyperparameters, dqn_ram for ~1.5 million timesteps*

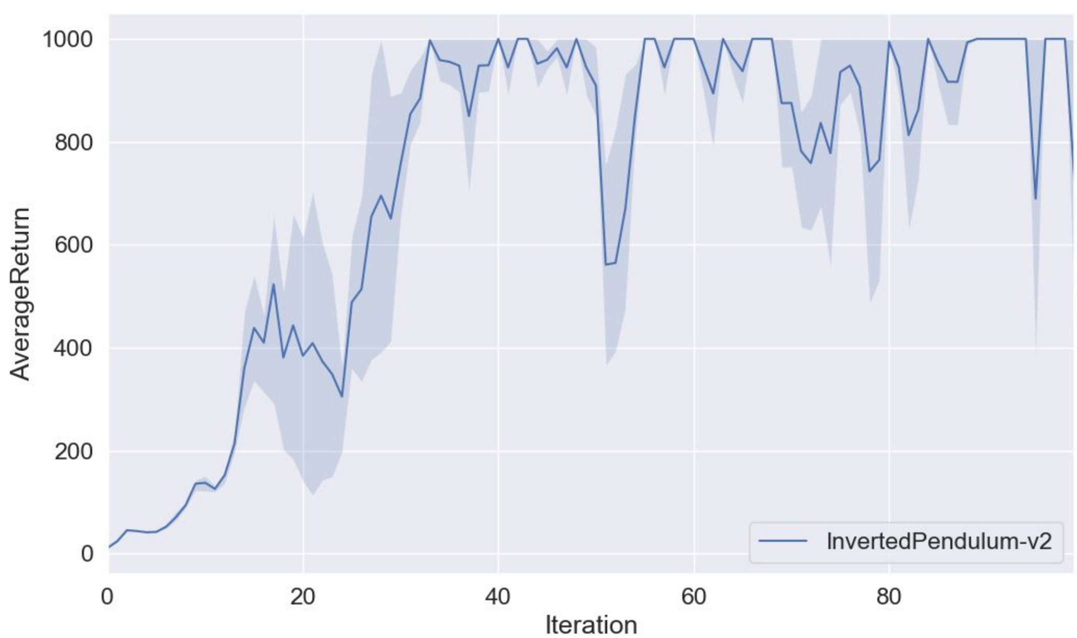


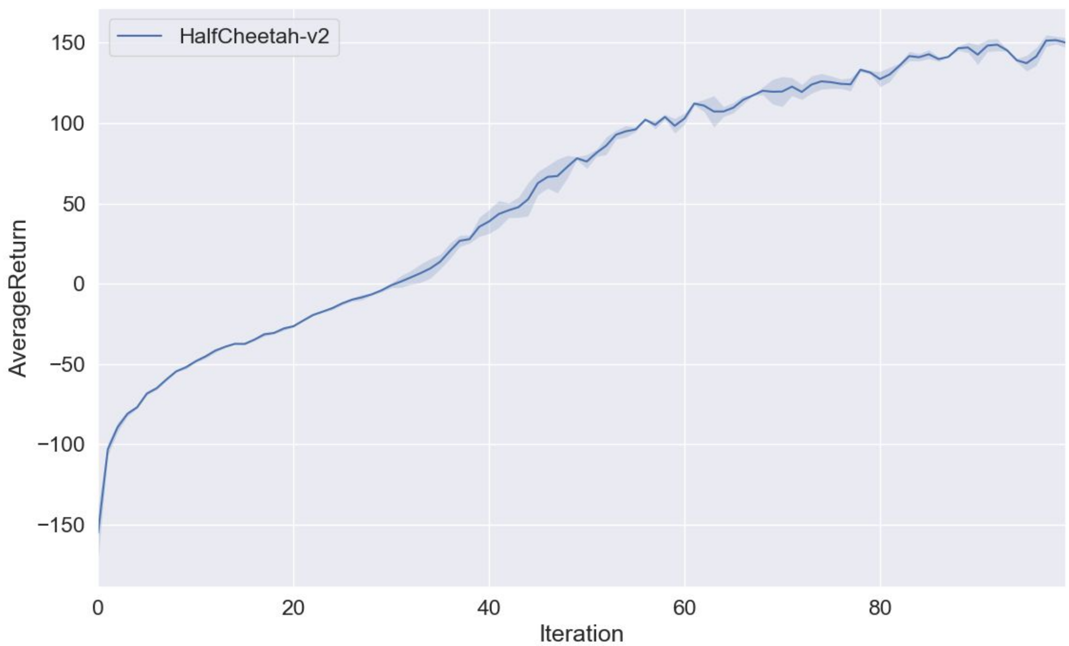
I chose to compare different learning rates, 1e-3 and 1e-5 all have better performances than default 1e-4, at least for the 1.5 million timesteps I ran. 1e-2 did not seem to work.

Actor-critic:
q1:
Obviously, -ntu 10 -ngsptu 10 had the best performance.



q2: -ntu 10 -ngsptu 10




HalfCheetah-v2
AverageReturn
Iteration
150
100
50
0
−50
−100
−150
0
20
40
60
80

Bonus question:
I experimented with a more complex and expressive critic network--two more layers than actor, and each layer with 32 more neurons, learning rate is 1.25 times actor's learning rate (0.025).

I chose HalfCheetah as InvertedPendulum had high variance in hw2. As shown here, a more complex critic networks led to higher returns in later stages of training and the variance was much smaller.