

Homework #4

Mahan Fathi

March 31, 2022

1 Question 1

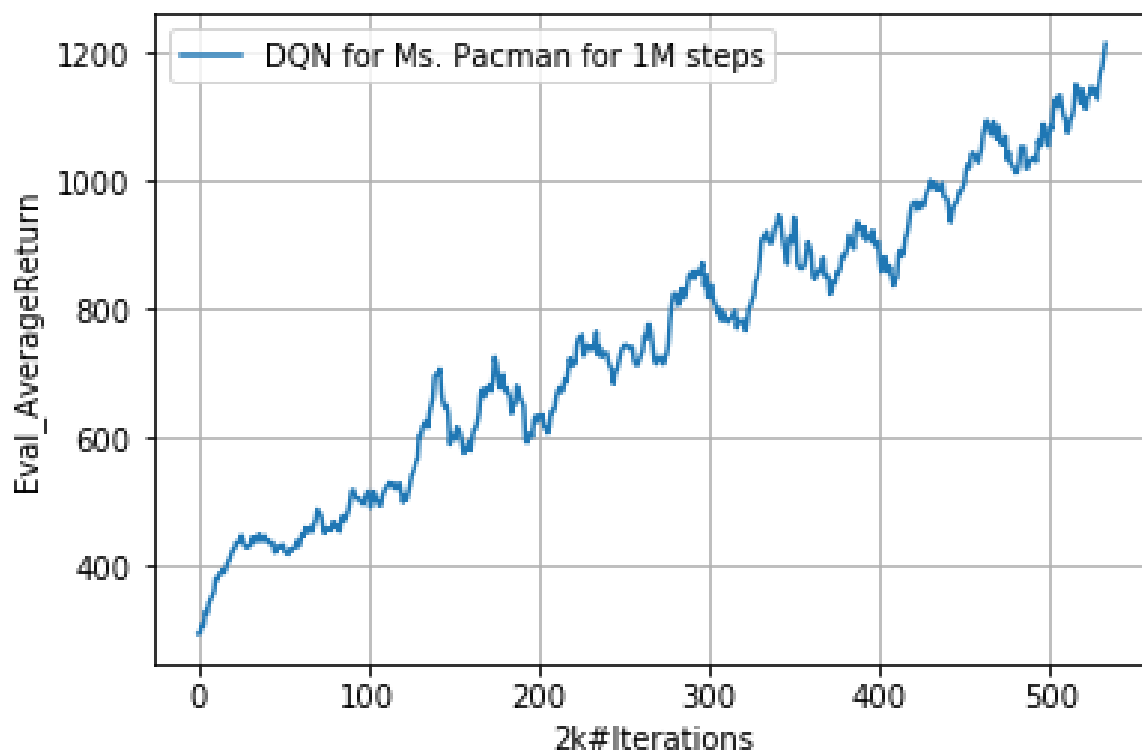


Figure 1: **Q1** MsPacman-v0 Single DQN

```
python run_hw4.py exp_name=q1 env_name=MsPacman-v0 n_iter=10000000
```

Listing 1: **Q1** Run command

2 Question 2

The plummet after reaching a total return of 150 has already been alluded to in the problem statement, and is the case also here.

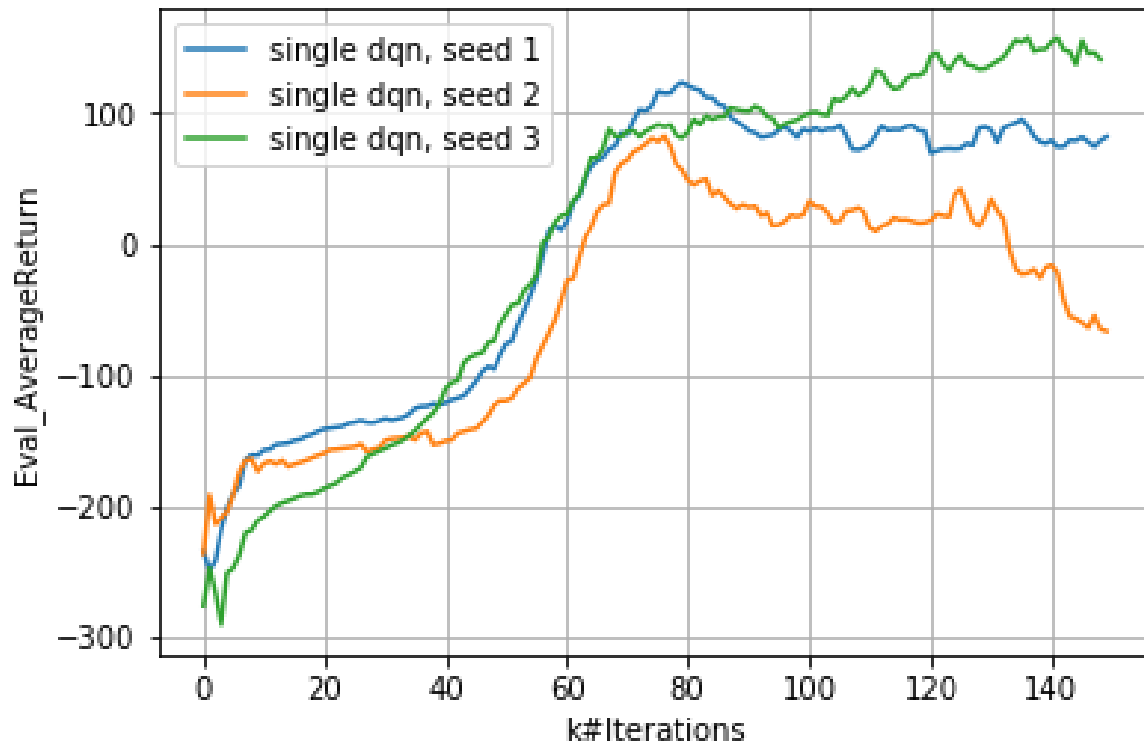


Figure 2: Q2 LunarLander-v3 Single DQN

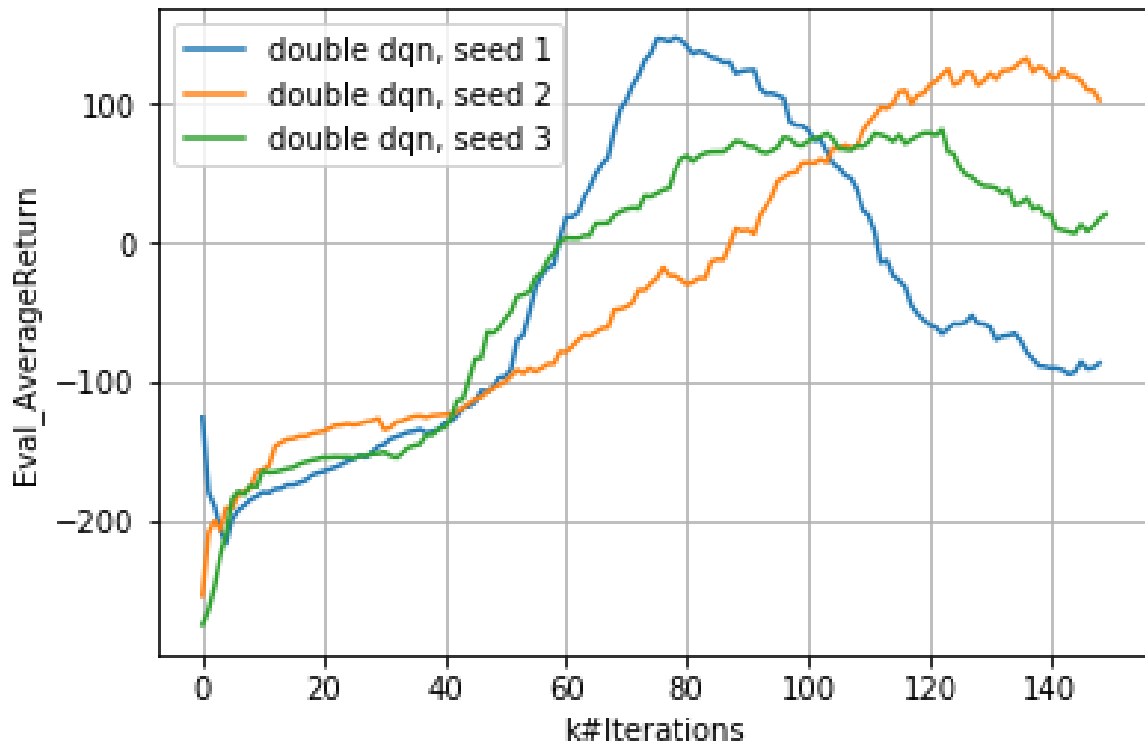


Figure 3: **Q2** LunarLander-v3 Double DQN

```
python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q2_dqn_1 seed=1 n_iter=1000000 double_q=null

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q2_dqn_2 seed=2 n_iter=1000000 double_q=null

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q2_dqn_3 seed=3 n_iter=1000000 double_q=null

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q2_doubledqn_1 seed=1 n_iter=200000

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q2_doubledqn_2 seed=2 n_iter=200000

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q2_doubledqn_3 seed=3 n_iter=200000
```

Listing 2: **Q2** Run commands

3 Question 3

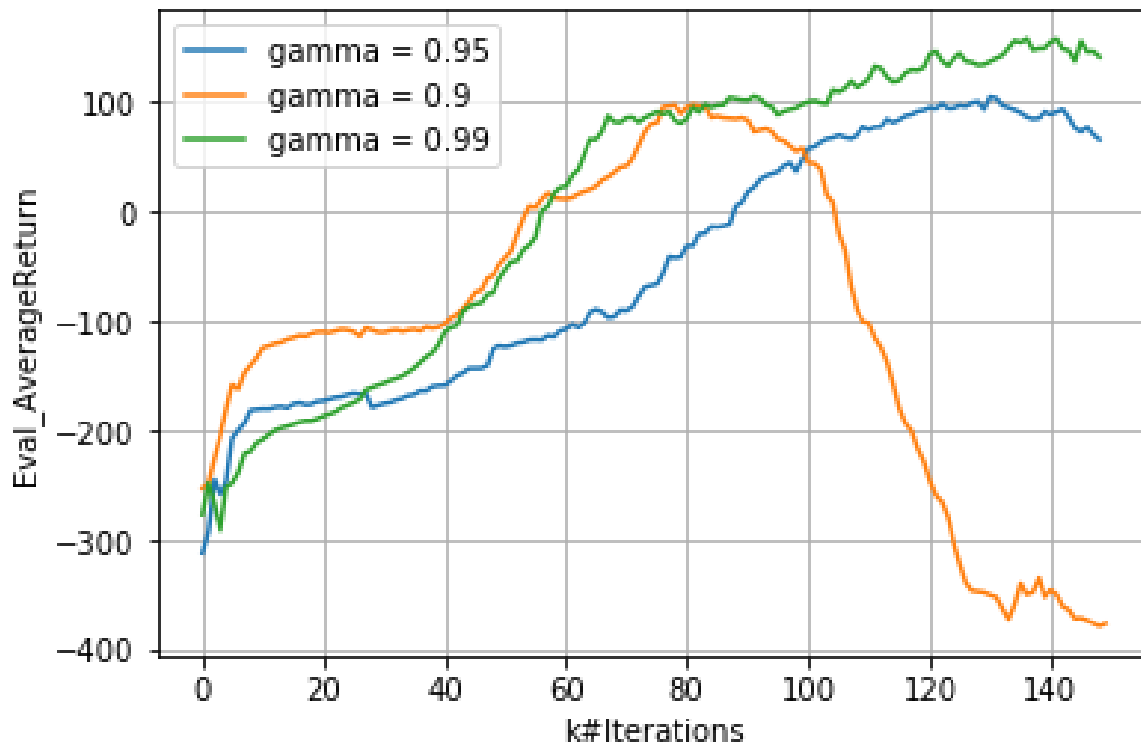


Figure 4: Q3 LunarLander-v3 with ablations for gamma

```
python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q3_gamma0.9 n_iter=1000000 gamma=0.9

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q3_gamma0.95 n_iter=1000000 gamma=0.95

python run_hw4.py env_name=LunarLander-v3 \
    exp_name=q3_gamma0.99 n_iter=1000000 gamma=0.99
```

Listing 3: Q3 Run commands

Here we have chosen `params['gamma']` as our ablation variable. The reason behind this is that one convenient handle we have over the bias and variance trade-off in a Q-Learning algorithm, is the discount factor. My speculation was that the fall in the return after 150 could be addressed by making the Q values more or less sensitive to the changes in the rewarding pattern of the environment. We can see from plots that indeed $\gamma = 0.99$ performs best, as it corresponds to a scenario in which the algorithm is least fixated on the current reward and takes into account the future rewards more, as opposed to other γ values.

4 Question 4

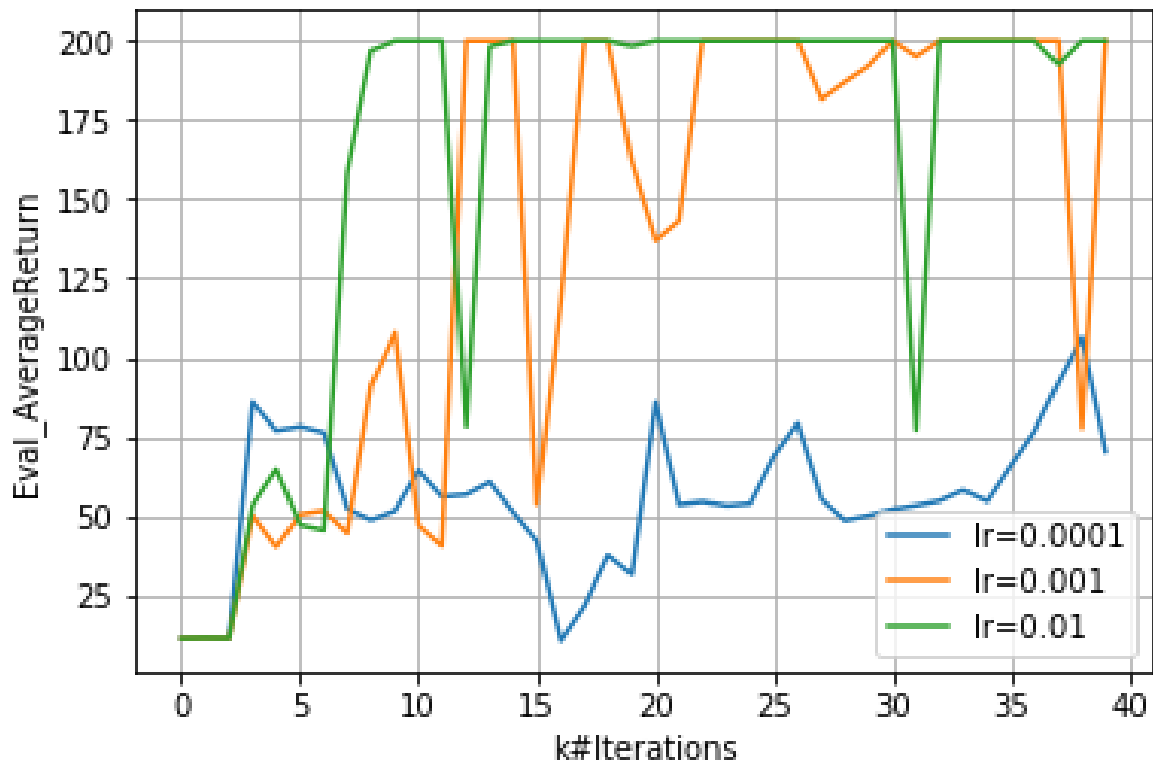


Figure 5: Q4 DDPG for InvertedPendulum-v2, learning_rate ablations

```
python run_hw4.py exp_name=q4_ddpg_lr0.001 rl_alg=ddpg \
    env_name=InvertedPendulum-v2 atari=false n_iter=40000 learning_rate=0.001

python run_hw4.py exp_name=q4_ddpg_lr0.01 rl_alg=ddpg \
    env_name=InvertedPendulum-v2 atari=false n_iter=40000 learning_rate=0.01

python run_hw4.py exp_name=q4_ddpg_lr0.0001 rl_alg=ddpg \
    env_name=InvertedPendulum-v2 atari=false n_iter=40000 learning_rate=0.0001

python run_hw4.py exp_name=q4_ddpg_uf1 rl_alg=ddpg \
    env_name=InvertedPendulum-v2 atari=false n_iter=40000 policy_update_frequency=1

python run_hw4.py exp_name=q4_ddpg_uf10 rl_alg=ddpg \
    env_name=InvertedPendulum-v2 atari=false n_iter=40000 policy_update_frequency=10

python run_hw4.py exp_name=q4_ddpg_uf100 rl_alg=ddpg \
    env_name=InvertedPendulum-v2 atari=false n_iter=40000 policy_update_frequency=100
```

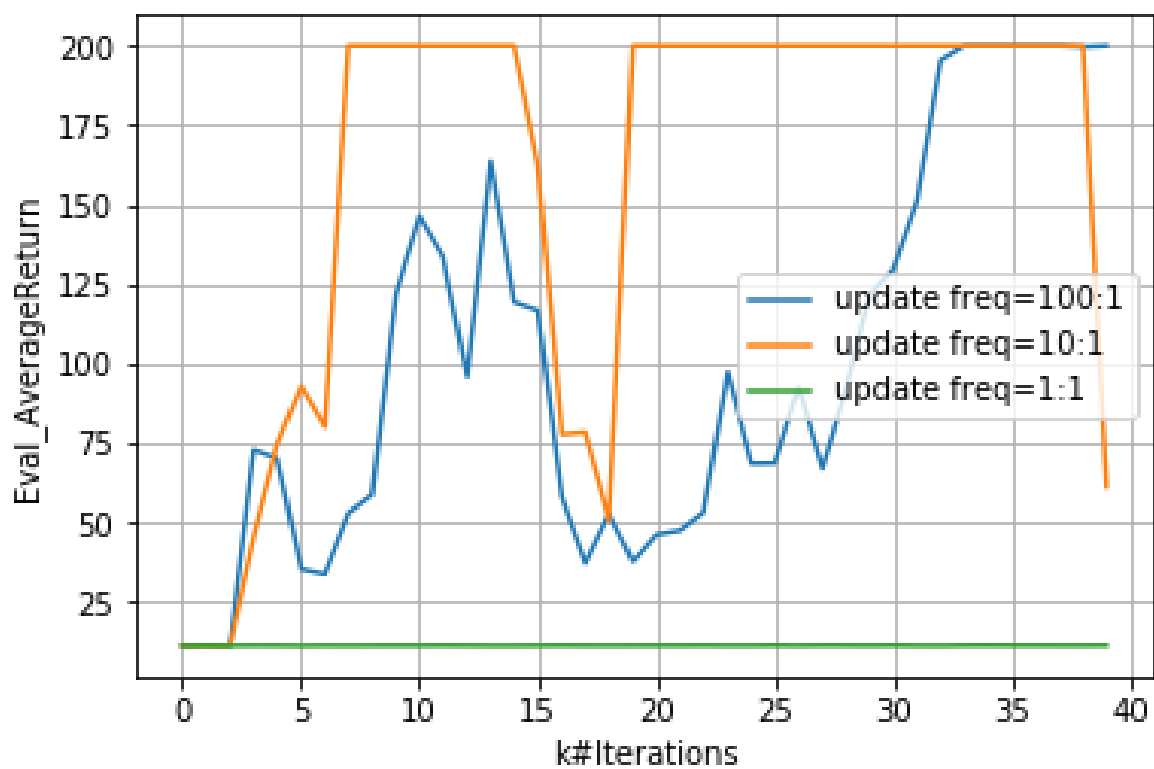


Figure 6: **Q4** DDPG for InvertedPendulum-v2, policy_update_frequency ablations

5 Question 5

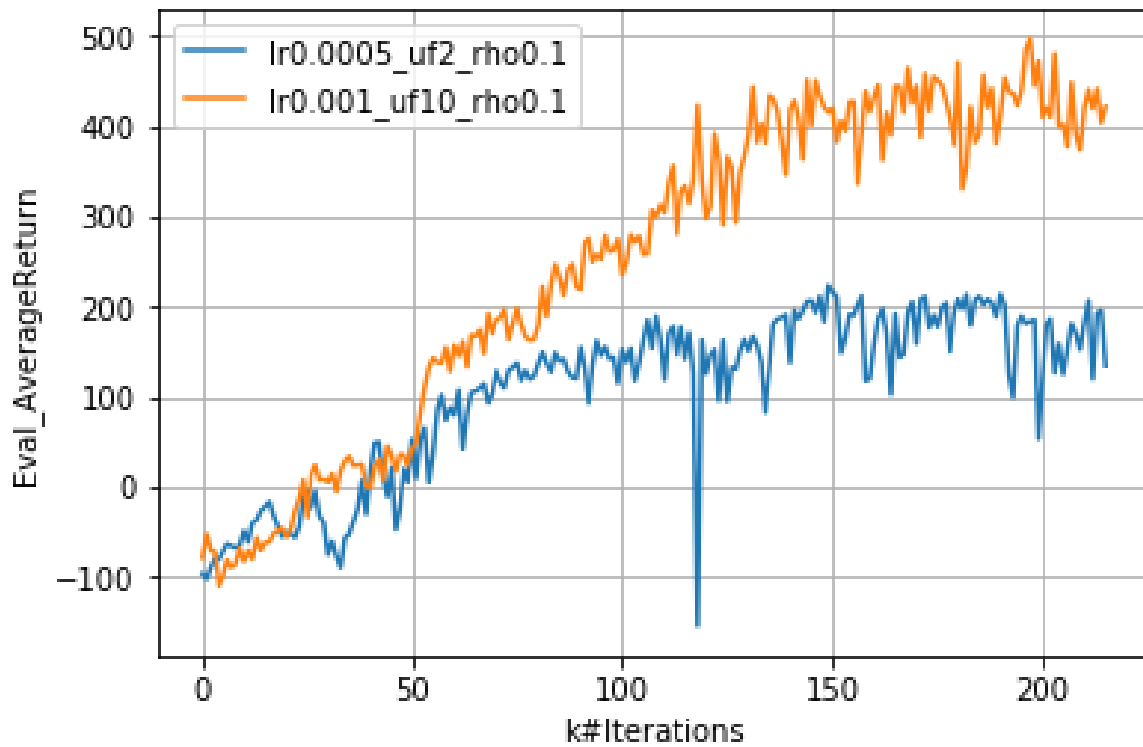


Figure 7: **Q5** DDPG for HalfCheetah-v2, with two different learning schedule

DDPG manages to solve this task and as you can see, given that

```
python run_hw4.py exp_name=q5_ddpg_hard_lr0.001_uf10 rl_alg=ddpg \
  env_name=HalfCheetah-v2 atari=false n_iter=100000 \
  learning_rate=0.001 policy_update_frequency=10

python run_hw4.py exp_name=q5_ddpg_hard_lr0.001_uf100 rl_alg=ddpg \
  env_name=HalfCheetah-v2 atari=false n_iter=100000 \
  learning_rate=0.001 policy_update_frequency=100
```


6 Question 6

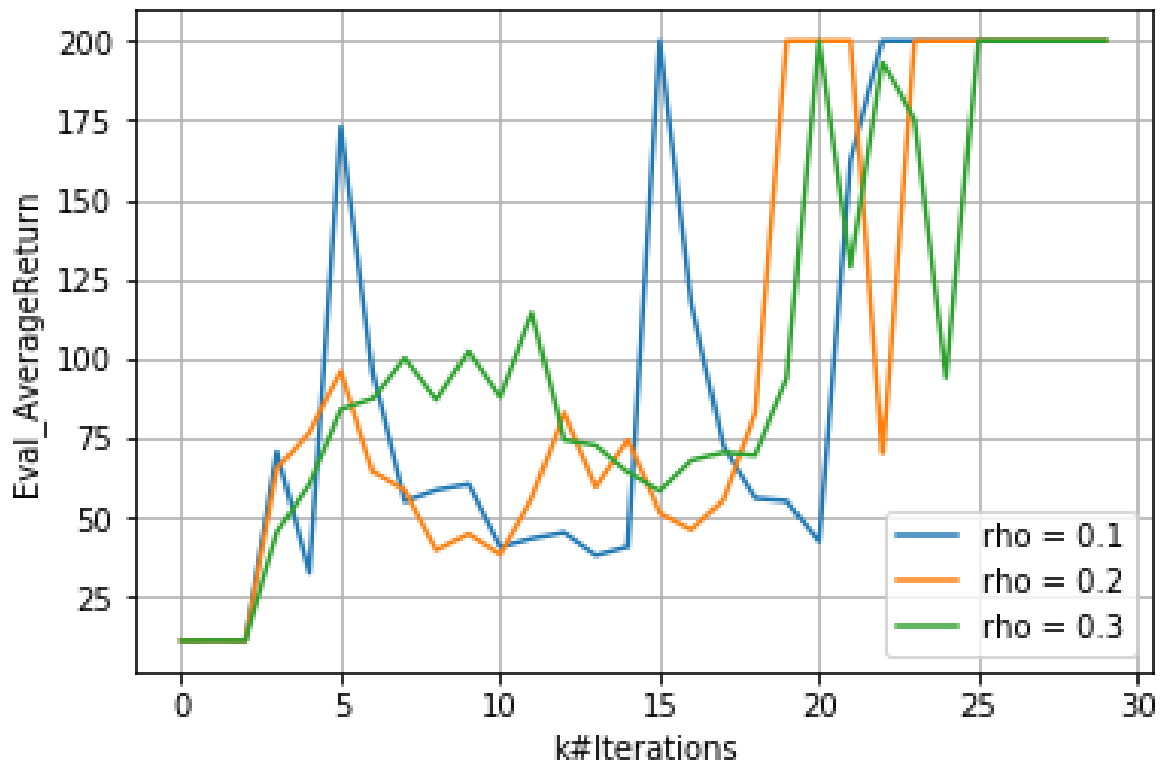


Figure 8: Q6 TD3 for InvertedPendulum-v2, td3_target_policy_noise ablations

```
python run_hw4.py exp_name=q6_td3_rho0.1 rl_alg=td3 \
  env_name=InvertedPendulum-v2 atari=false n_iter=30000 \
  learning_rate=0.001 td3_target_policy_noise=0.1

python run_hw4.py exp_name=q6_td3_rho0.2 rl_alg=td3 \
  env_name=InvertedPendulum-v2 atari=false n_iter=30000 \
  learning_rate=0.001 td3_target_policy_noise=0.2

python run_hw4.py exp_name=q6_td3_rho0.3 rl_alg=td3 \
  env_name=InvertedPendulum-v2 atari=false n_iter=30000 \
  learning_rate=0.001 td3_target_policy_noise=0.3

python run_hw4.py exp_name=q6_td3_rho0.1_shape2 rl_alg=td3 \
  env_name=InvertedPendulum-v2 atari=false n_iter=100000 \
  learning_rate=0.001 td3_target_policy_noise=0.1 n_layers_critic=2
```

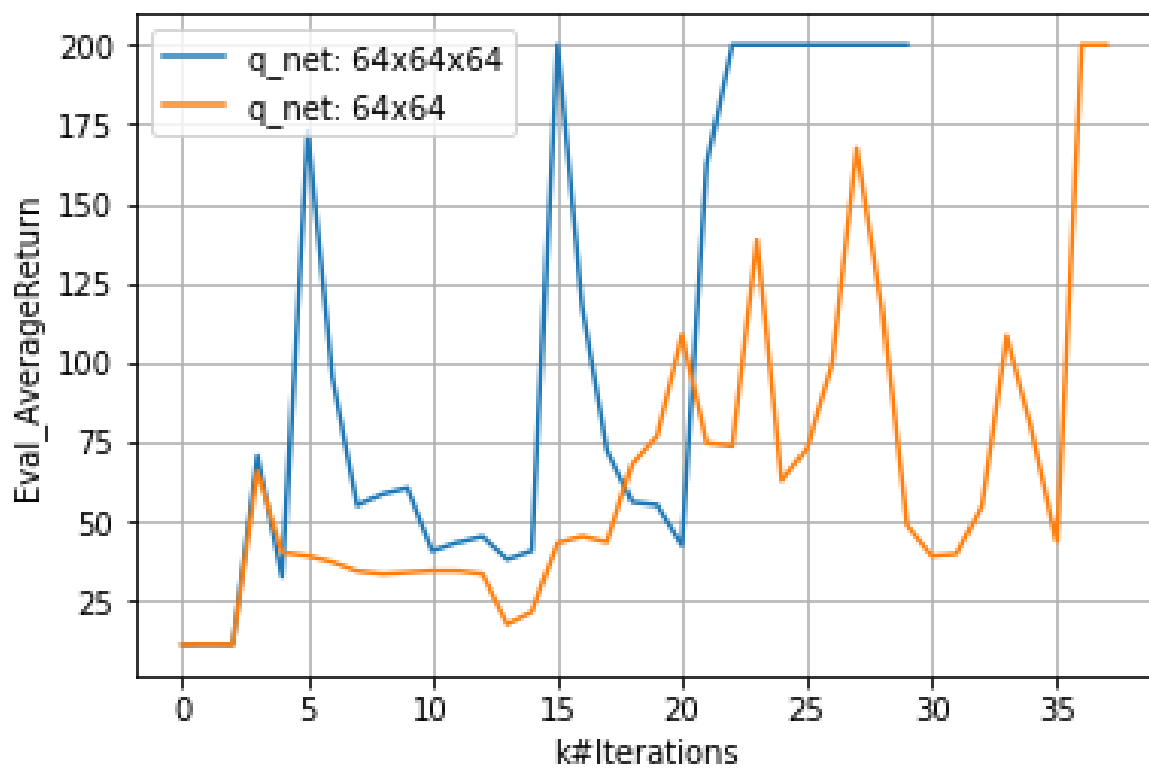


Figure 9: **Q6** TD3 for InvertedPendulum-v2, Q-network size ablations

7 Question 7

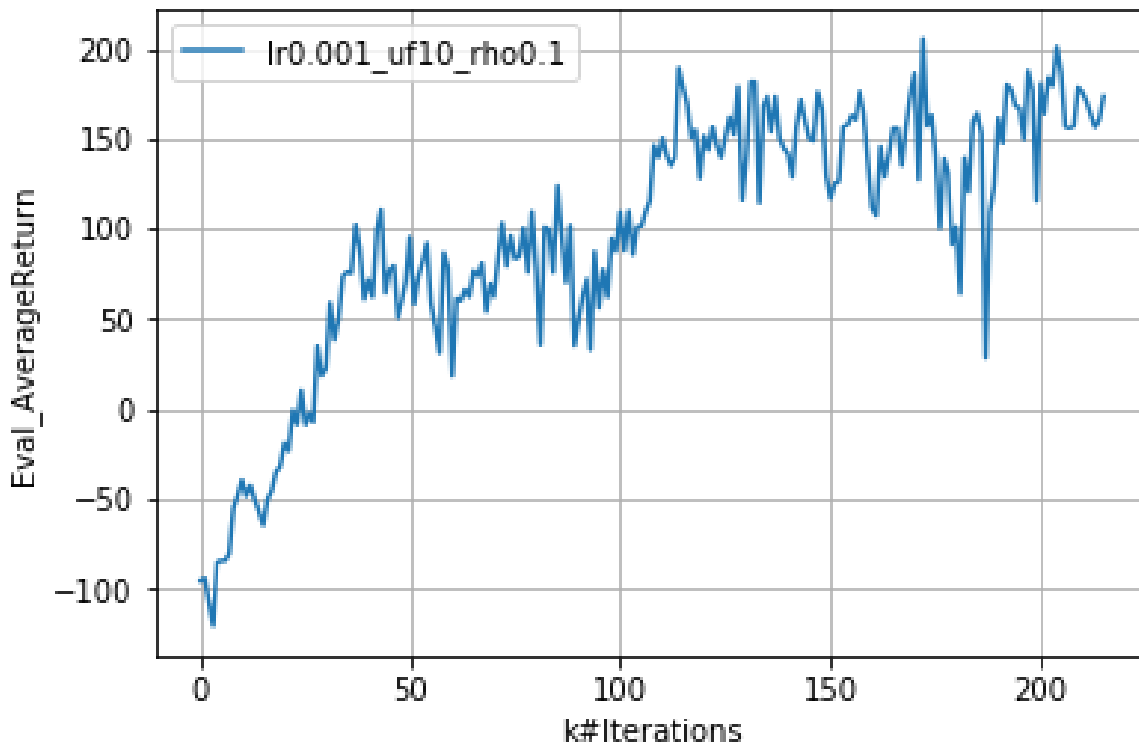


Figure 10: **Q7** TD3 for HalfCheetah-v2

```
python run_hw4.py exp_name=q7_td3_hard_lr0.001_uf10_rho0.1_shape3 \
    rl_alg=td3 n_layers_critic=3 \
    env_name=HalfCheetah-v2 atari=false n_iter=1000000 \
    learning_rate=0.001 policy_update_frequency=10 td3_target_policy_noise=0.1
```

```
python run_hw4.py exp_name=q7_td3_hard_lr0.0005_uf2_rho0.1_shape3 \
    rl_alg=td3 n_layers_critic=3 \
    env_name=HalfCheetah-v2 atari=false n_iter=1000000 \
    learning_rate=0.0005 policy_update_frequency=2 td3_target_policy_noise=0.1
```

There is a huge difference between the performance of DDPG and TD3 in the task of `HalfCheetah-v2`, both in terms of the sample efficiency and in the overall pattern of learning (learning curves). This difference stems from the smoothing property of the noise added on top of the actions sampled from the policy, when forming the Q-network target update. This trick was introduced by TD3 to prevent the policy network to come up with hacky ways to exploit the structure of the imperfect Q-network, however, here this trick is somewhat unnecessary, as it introduces too much bias and apparently the Q-network is already in good shape in the context of this specific task. DDPG seems to be more sample efficient, at least in comparison with TD3 with $\rho = 0.1$, i.e. the noise on top of target actions. Admittedly, TD3 is not tuned to the best of its abilities here, and would be jumping to conclusions to dismiss it as inferior to DDPG, solely based on this experiment.