IFT6163 Assignment 4: Q-Learning Algorithms

Simon Chamorro

March 28, 2022

1 Q-Learning

For the experiments in this section, we implemented DQN as well as Double DQN and compared their performances on the gym environements LunarLander and MsPacman. All the experiments can be reproduced by running `scripts/01-dqn.sh`, while all the figures can be recreated by running `scripts/make_plots.py`.

1.1 Experiment 1 - DQN

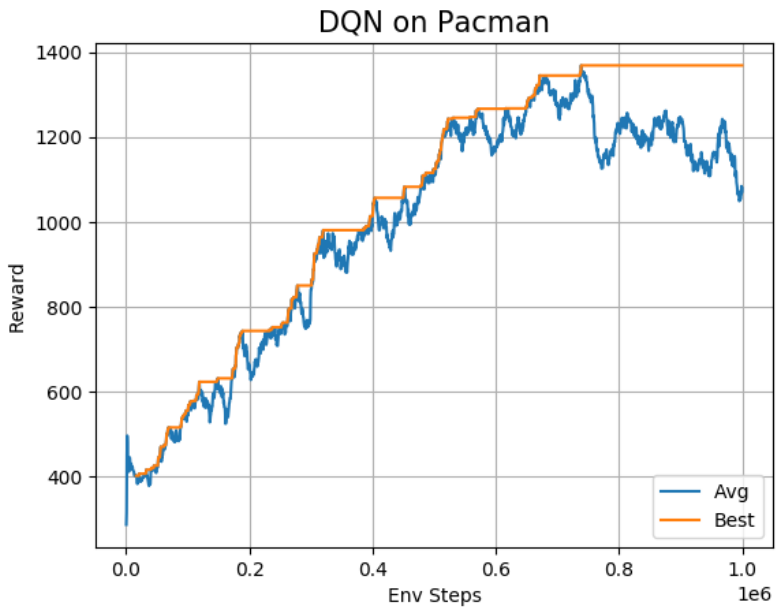
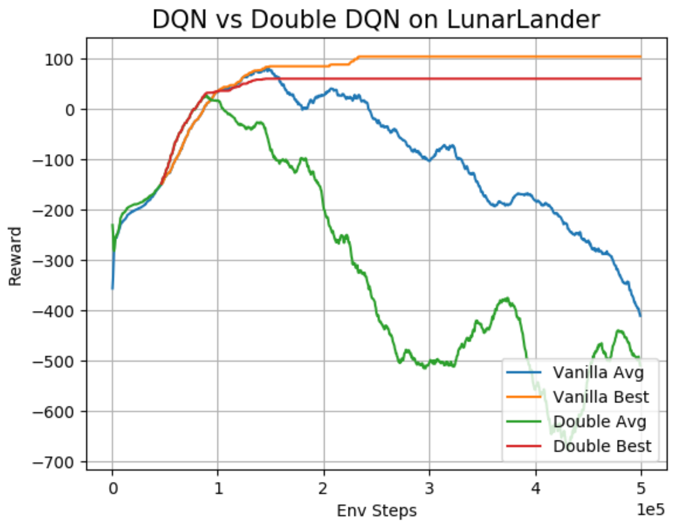
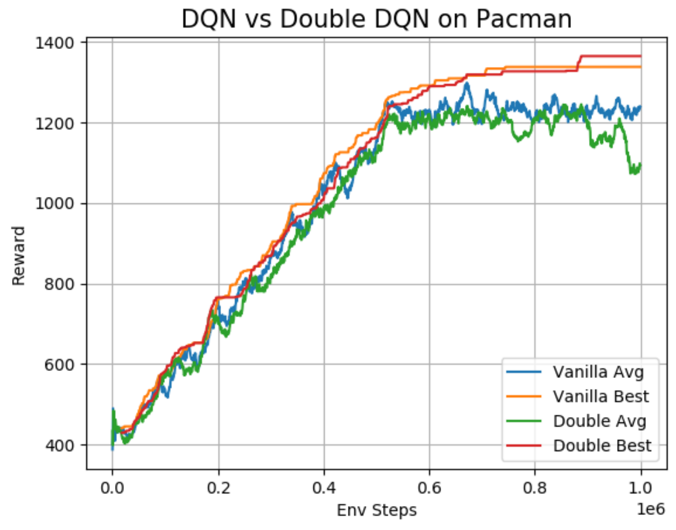


Figure 1: **Vanilla DQN Performance on Pacman.** This figure shows the performance of a Vanilla DQN agent on Pacman trained for 1M steps. The default parameters where used for this experiment.

1.2 Experiment 2 - Double DQN



(a) LunarLander



(b) Pacman

Figure 2: **DQN vs Double DQN on LunarLander and Pacman.** This figure shows the performance of DQN and Double DQN. For each configuration, 3 seeds were run and averaged. Figure a) shows the results on LunarLander and figure b) shows the results on Pacman. All hyperparameter values are the default ones. We notice that double DQN performs worse than regular DQN in LunarLander for the seeds we tested, but performs better on Pacman, reaching a higher best performance at the end.

1.3 Experiment 3 - DQN Exploration

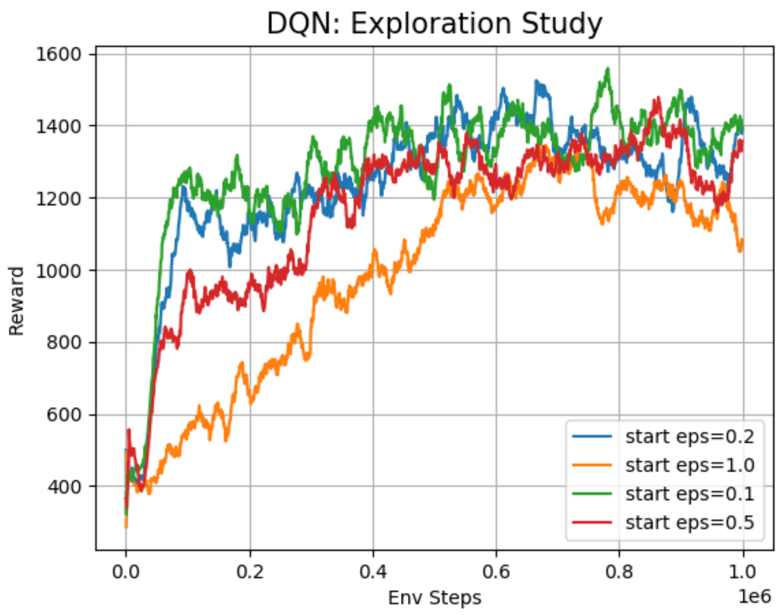
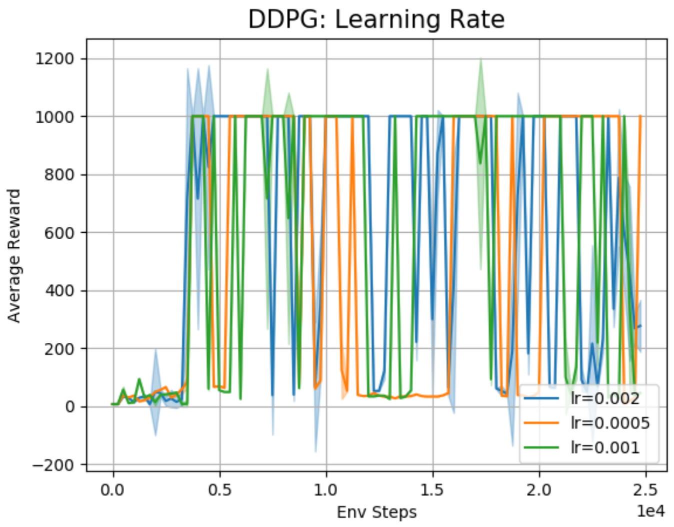


Figure 3: **DQN with different exploration schedules.** This figure shows the performance of DQN agents with different exploration schedules on Pacman trained for 1M steps. The default parameters where used for this experiment except for exploration. All agents have a schedule going from <start_eps> to 0.1 over 1M steps. We notice that the agent with a constant epsilon of 0.1 is the one that performs best. This could be explained by the fact that the agent can already start doing some exploitation early on in the training process and visit better states.

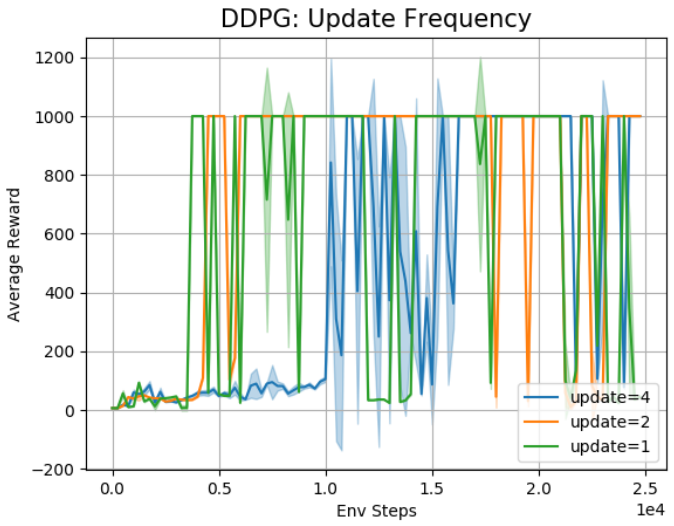
2 DDPG

For the experiments in this section, we implemented DDPG and evaluated its performance on the gym environements InvertedPendulum and HalfCheetah. All the experiments can be reproduced by running `scripts/02-ddpg.sh`, while all the figures can be recreated by running `scripts/make_plots.py`.

2.1 Experiment 4 - Experiments on InvertedPendulum



(a) Learning Rate



(b) Update Frequency

Figure 4: **DDPG Hyperparameters on InvertedPendulum.** This figure shows the performance of DDPG with different hyperparameters. Figure a) shows the results for different learning rates and figure b) shows the results for different update rates for the Q-network. All the other parameter values are the default ones. The best learning rate is 0.002, but all agents perform very similarly. Perhaps a more difficult environment would help differentiate results better. Regarding update frequencies, a value of 1 performs best.

2.2 Experiment 5 - DDPG on HalfCHeetah

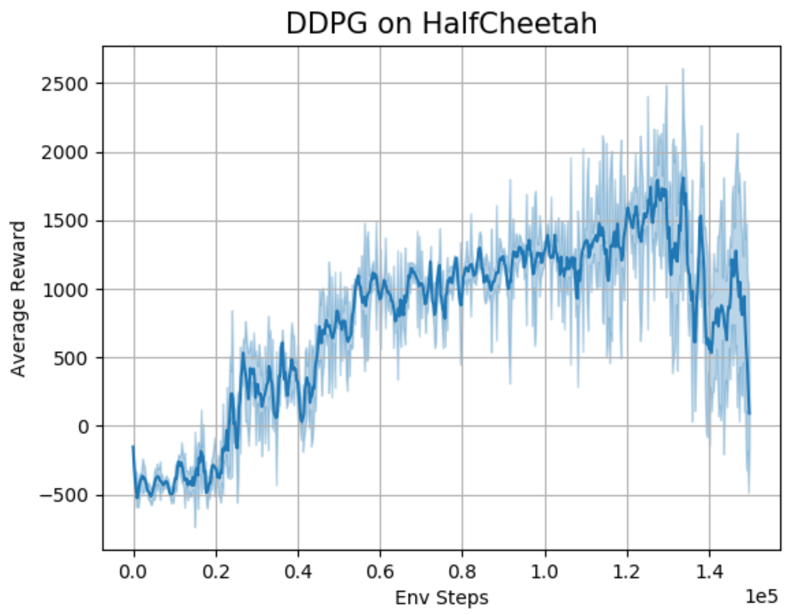
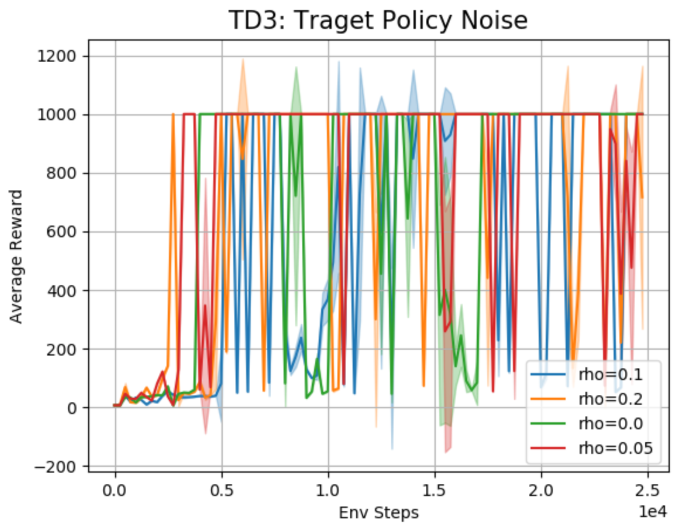


Figure 5: **DDPG on HalfCheetah.** This figure shows the performance of DDPG on HalfCHeetah. The default parameters where used for this experiment except for learning rate. A learning rate of 0.002 was used since this was the most successful configuration on the previous experiment. The episode lenght is 1000 and we can see that the agent reaches a performance of more than 1500 after 130k steps.

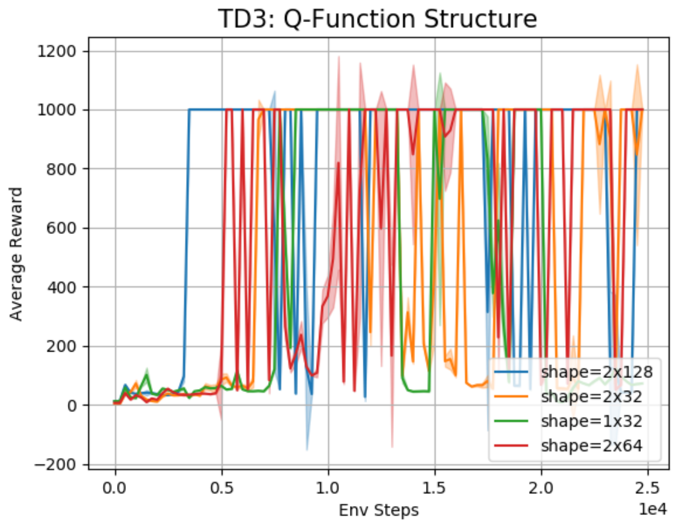
3 TD3

For the experiments in this section, we implemented TD3 and evaluated its performance on the gym environements InvertedPendulum and HalfCheetah. We also compare the performance of TD3 vs DDPG on HalfCheetah. All the components of TD3 were implemented: action noise, a delayed policy update and Q-value clipping with an additional network. All the experiments can be reproduced by running `scripts/03-td3.sh`, while all the figures can be recreated by running `scripts/make_plots.py`.

3.1 Experiment 6 - Experiments on InvertedPendulum



(a) Target Policy Noise



(b) Q-Network Architecture

Figure 6: **TD3 Hyperparameters on InvertedPendulum.** This figure shows the performance of TD3 with different hyperparameters. Figure a) shows the results for different target policy noise and figure b) shows the results for different Q-network architectures. All the other parameter values are the default ones. It us hard to determine which noise value is better. A value of 0.01 achieves maximum performance faster, but 0.005 seems more stable. Perhaps a more difficult environment would help differentiate results better. For further experiments, a value of 0.01 will be used. Regarding Q-network shape, two layers of size 128 performs best. This is probably due to the larger model capacity.

3.2 Experiment 7 - TD3 on HalfCHeetah

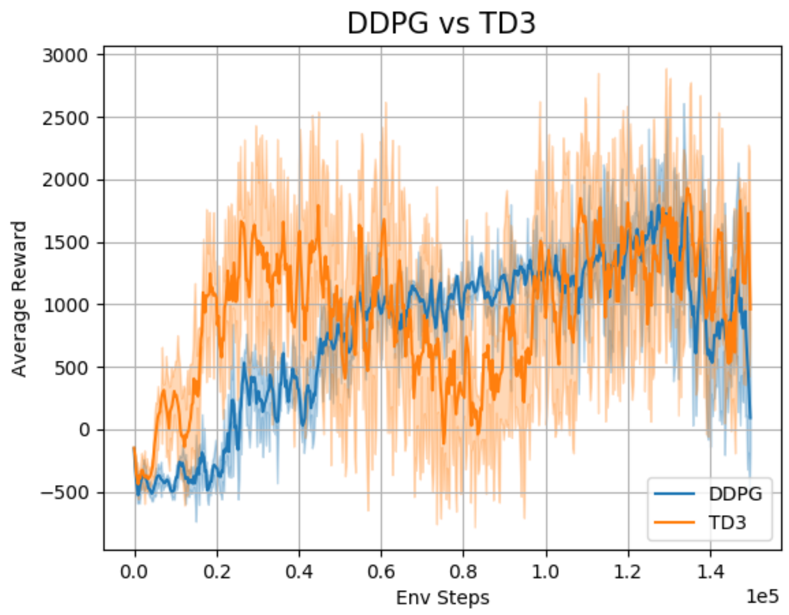


Figure 7: **TD3 and DDPG on HalfCheetah.** This figure shows the performance of TD3 on HalfCHeetah. The default parameters where used for this experiment except for the Q-network size. A network with two layers of size 128 was used since this was the most successful configuration on the previous experiment. The episode lenght is 1000 and we can see that the agent reaches a performance of more than 1500 after only around 40k steps. The results from figure 5 are also included here for comparison. We notice that TD3 outperforms DDPG.