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

Louis Tilloy

1 Q-learning

1.1 basic Q-learning performance

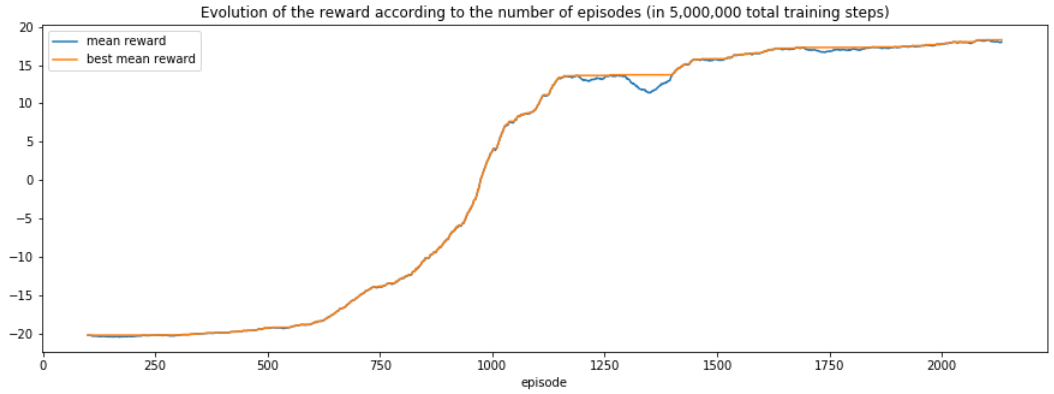

Figure 1 – (I forgot to add the timesteps in the pickle file when I trained the network, and realized too late that it was not possible to extract it from the default values dumped in the file. The training was still made with 5,000,000 steps. The default parameters were used.)

1.2 double Q-learning

Since my computer is too slow, I had to run double-q learning on the lunar lander task.

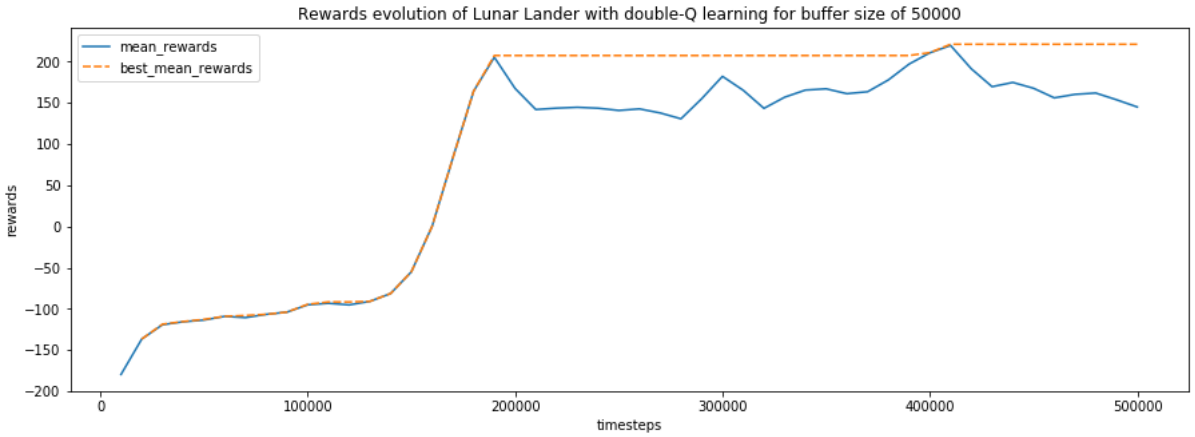

Figure 2 –

1.3 Experimenting with hyperparameters

I chose to experiment on the buffer size, since it is a non-trivial parameter to tune : If it is too short that the network focus too much on recent events while if it is too long, it still takes into account irrelevant examples, so there must be an optimal buffer size to train the network.

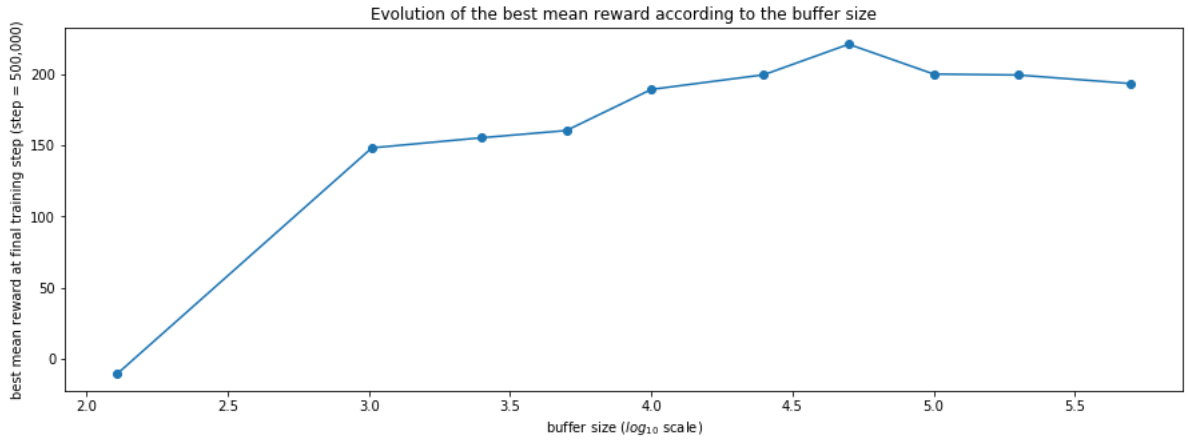

FIGURE 3 – (Data collected on lunar lander experiments) It seems like the default buffer size of 50,000 actually gives the best results. Small buffer size gives bad reward, while big buffer size tends to slightly reduce performance or even simply stagnate it.

The script ran to obtain this data was the following :

```
python run_dqn_lander.py -b buffer_size
```

2 Actor-Critic

2.1 Sanity check with Cartpole

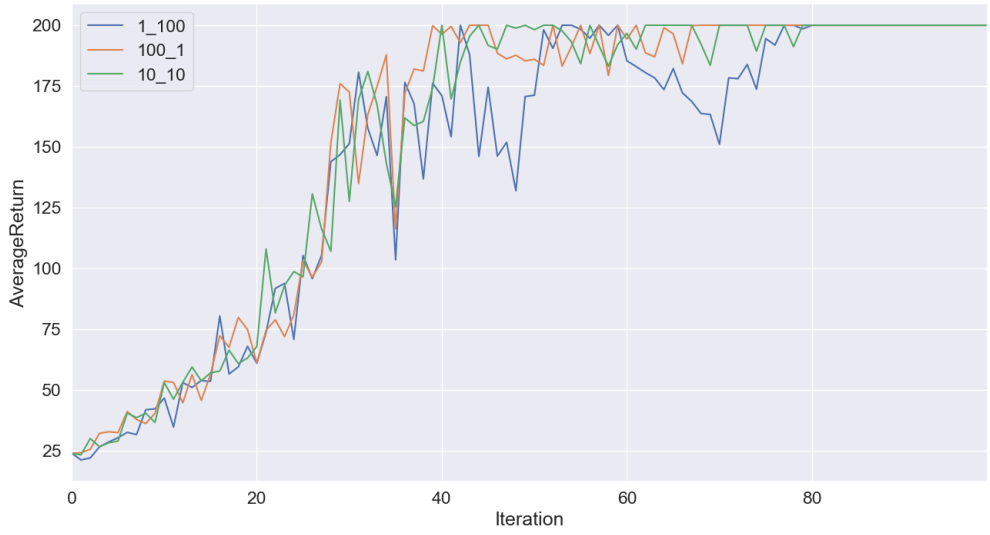

FIGURE 4 – Cartpole training with different hyper parameters for the update of the actor-critic (num_grad_steps_per_target_update _ num_target_updates). We can see that the average return has more variance with 100 gradient steps per target update, simply because after 100 gradient steps, the target error (the difference between the actual prediction the network would have given and the old target) is too high. The two others have similar performance, so it is better to take 10 gradient steps for each target update since it requires less computing than taking only 1 gradient step per target update.

2.2 Run actor-critic with more difficult tasks

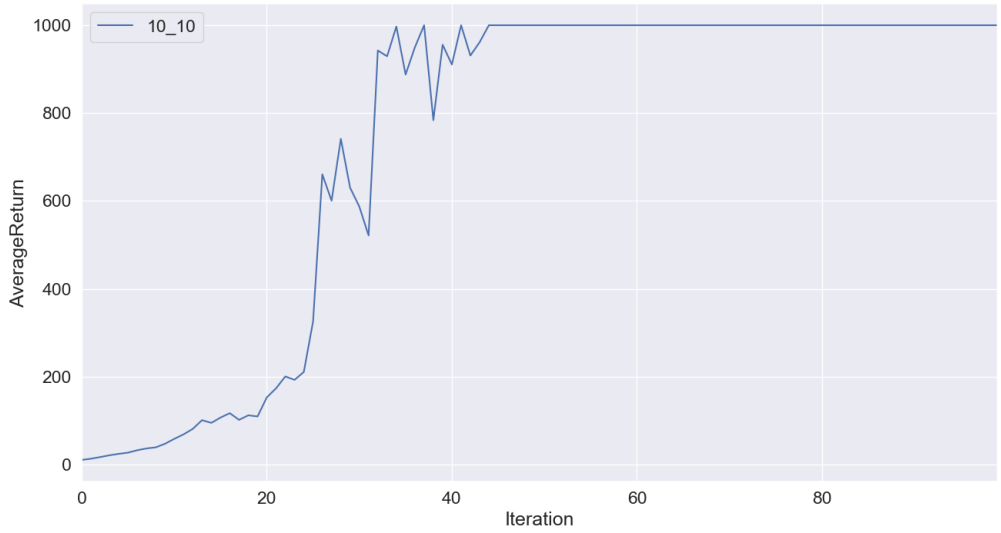

FIGURE 5 – InvertedPendulum training with default hyperparameters, 10 gradient step per target update and 10 target update per policy gradient step.

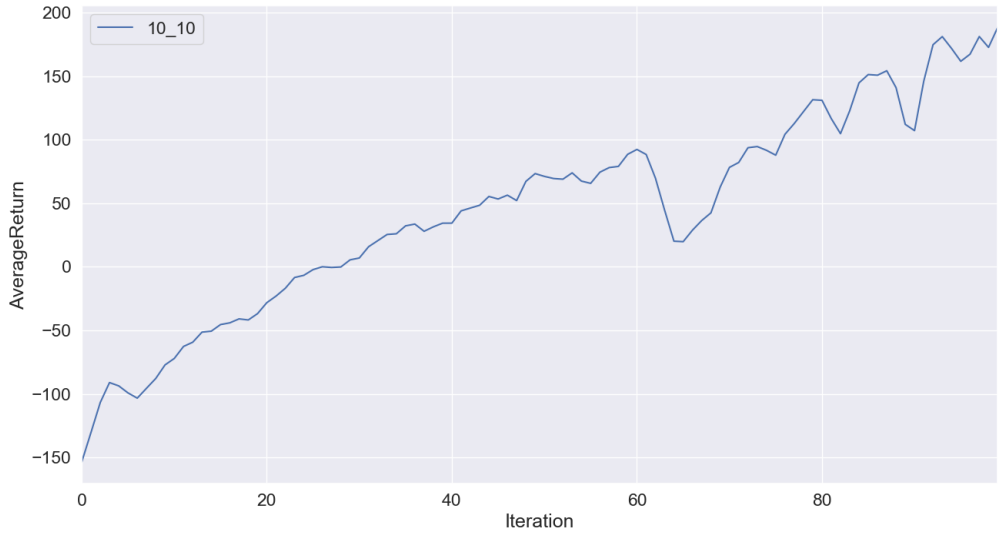

FIGURE 6 – HalfCheetah training with default hyperparameters except for the discount set at 0.95 10 gradient step per target update and 10 target update per policy gradient step.