Homework 3

Juanwu Lu (3037432593)

(M.Sc. Civil Engineering, UC Berkeley)

1 Part 1: Q-Learning

Question 1: basic Q-learning performance (DQN)

Results

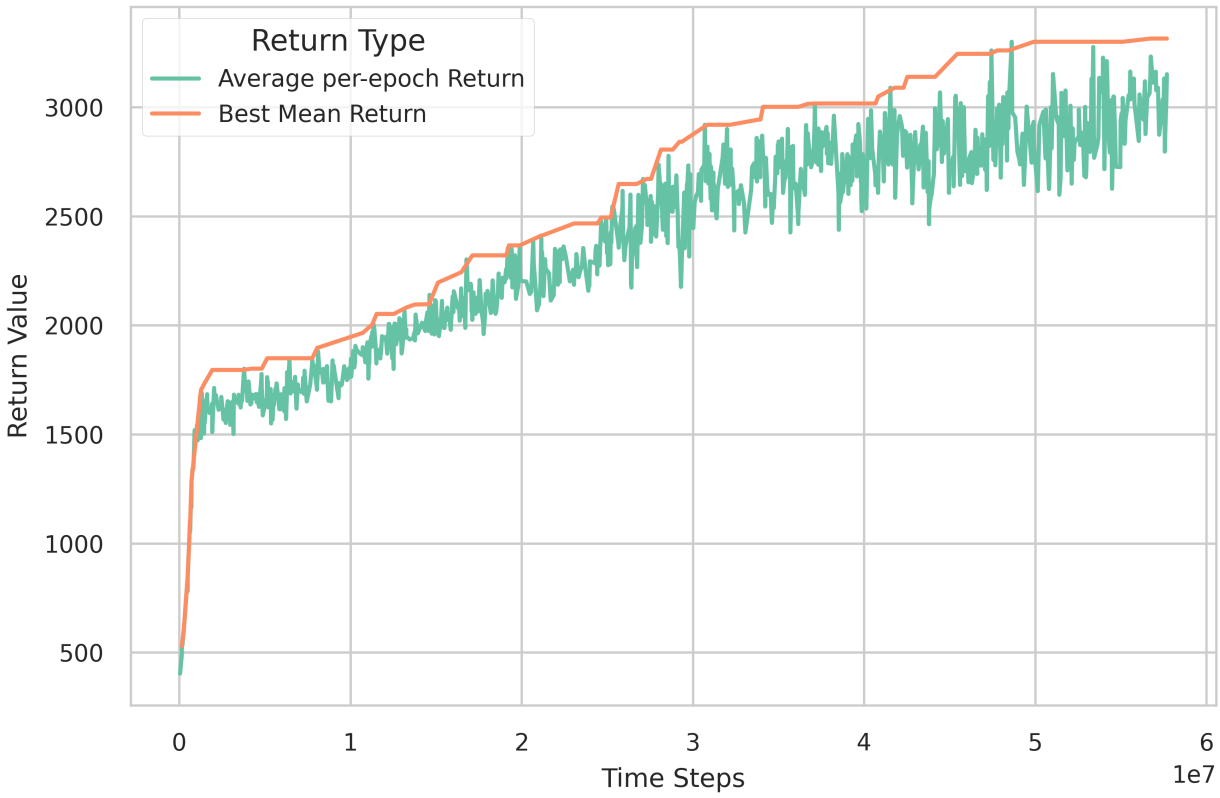

Figure 1. DQN performance on `Ms. Pac-Man` in average per-epoch return (cyan curve) and best mean return (red curve) versus number of time steps. Performance of the algorithm shows a step increasing trend. However, the average per-epoch return variances are also increasing.

Codes

```
python cs285/scripts/run_hw3_dqn.py --env_name MsPacman-v0 --exp_name q1 -gpu_id $1
```

Question 2: double Q-learning (DDQN)

Results

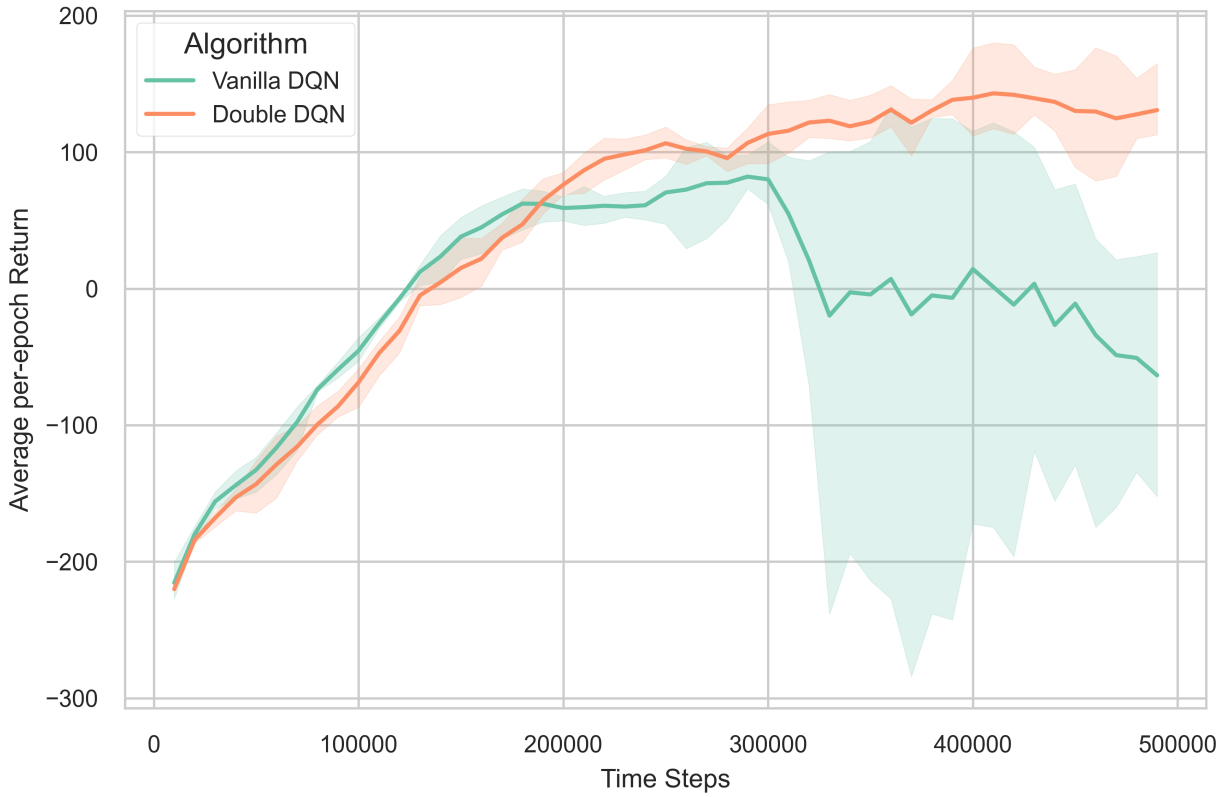

Figure 2. Average training per-epoch return with respect to vanilla (green curve) and double (orange curve) DQN. The double DQN successfully prevents rewards from decreasing after around 30,000 training steps and achieves a higher score.

Codes

```
echo "Running Homework 3 Question 2";
python $1 --env_name LunarLander-v3 --exp_name q2_dqn_1 --seed 1 -gpu_id $2;
python $1 --env_name LunarLander-v3 --exp_name q2_dqn_2 --seed 2 -gpu_id $2;
python $1 --env_name LunarLander-v3 --exp_name q2_dqn_3 --seed 3 -gpu_id $2;

python $1 --env_name LunarLander-v3 --exp_name q2_doubledqn_1 --double_q --seed 1 -gpu_id $2;
python $1 --env_name LunarLander-v3 --exp_name q2_doubledqn_2 --double_q --seed 2 -gpu_id $2;
python $1 --env_name LunarLander-v3 --exp_name q2_doubledqn_3 --double_q --seed 3 -gpu_id $2;
echo "Question 2 Done!"
```

Question 3: experimenting with hyperparameters

Results

For this question I explore the influence of different `hidden_size` settings of the Q-function network on the training process. Intuitively, higher number of neurons yields better ability to represent non-linear relationships, which explains why performance under `hparam3` are lower compared to the others at the earlier stage of the training. But with sufficient training, all network structures converge to a similar performance level.

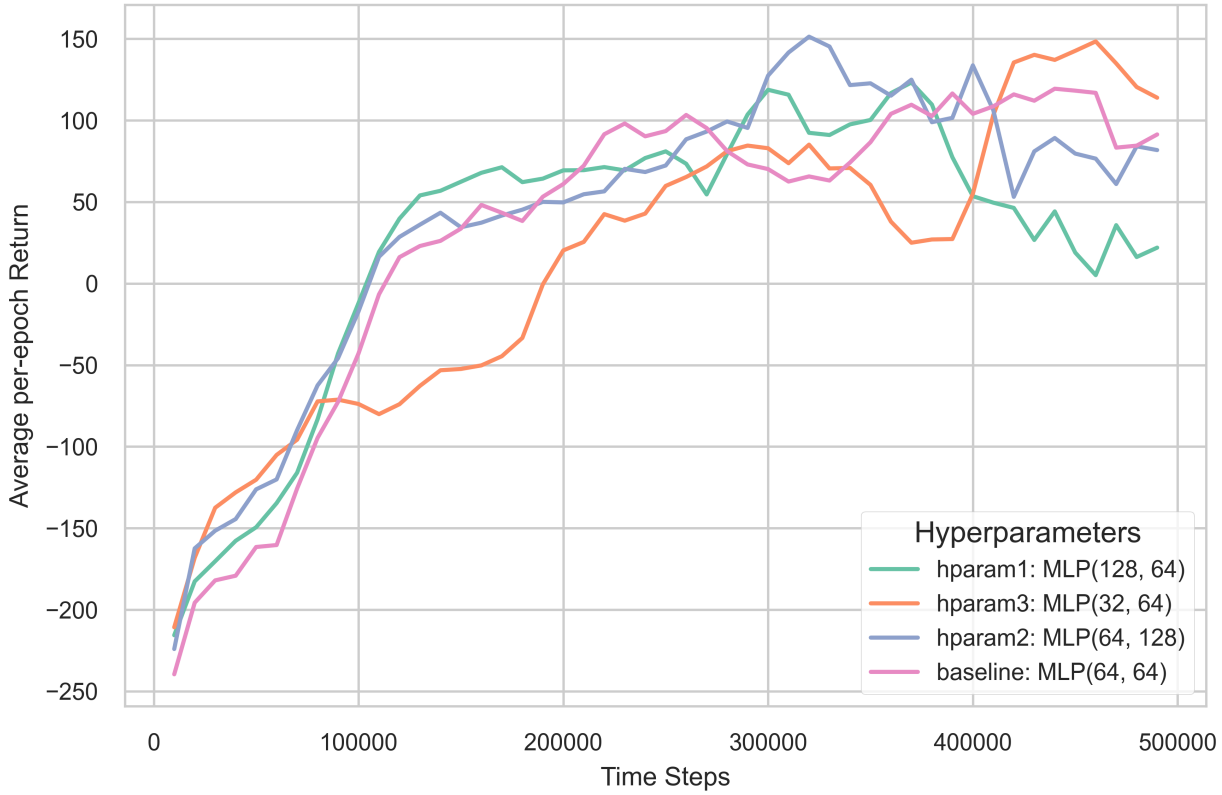

Figure 3. Average per-epoch training returns with respect to training time steps given different Q-function neural network structures.

Codes

```
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v3 --exp_name q3_hparam1 -gpu_id $1;
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v3 --exp_name q3_hparam2 -gpu_id $1;
python cs285/scripts/run_hw3_dqn.py --env_name LunarLander-v3 --exp_name q3_hparam3 -gpu_id $1;
```

2 Part 2: Actor-Critic

Question 4: sanity check with `Cartpole-v0`

Results

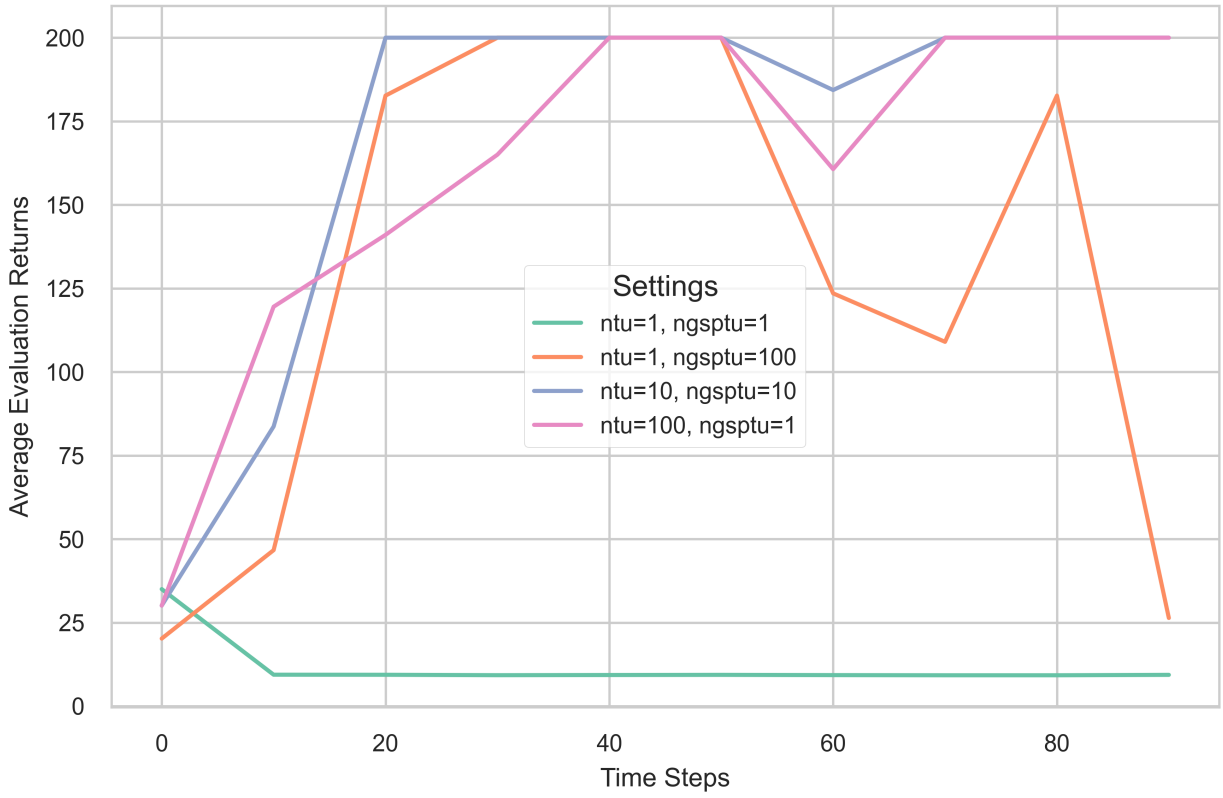

Figure 4. Average evaluation returns with respect to training time steps given different `ntu` and `ngsptu` settings. From the results, 10 target updates along with 10 gradient steps per target update yields the best results among the four different settings.

Codes

```
echo "Running Homework 3 Question 4..."
python $1 --env_name CartPole-v0 -n 100 -b 1000 --exp_name q4_ac_1_1 -ntu 1 -ngsptu 1 -gpu_id $2
python $1 --env_name CartPole-v0 -n 100 -b 1000 --exp_name q4_ac_100_1 -ntu 100 -ngsptu 1
    -gpu_id $2
python $1 --env_name CartPole-v0 -n 100 -b 1000 --exp_name q4_ac_1_100 -ntu 1 -ngsptu 100
    -gpu_id $2
python $1 --env_name CartPole-v0 -n 100 -b 1000 --exp_name q4_ac_10_10 -ntu 10 -ngsptu 10
    -gpu_id $2
echo "Done!"
```

Question 5: Run Actor-Critic with more difficult tasks

Results

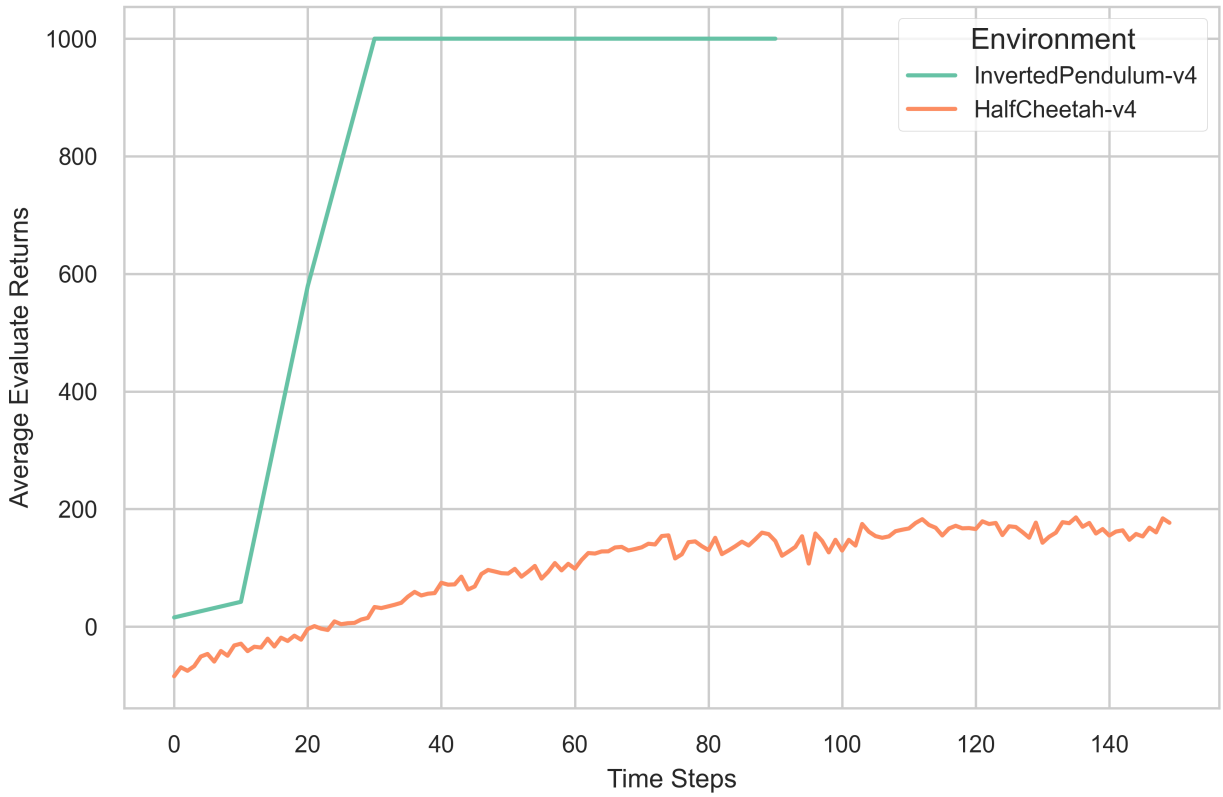

Figure 5. Average evaluation return with respect to training steps of running Actor-Crtic algorithm with `InvertedPendulum-v4` (green) and `HalfCheetah-v4` (orange).

Codes

```
NTU=$2
NGSPTU=$3
GPUID=$4

echo "Running Homework 3 Question 5";
python $1 --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.95 -n 100 -l 2 -s 64 -b 5000
    -lr 0.01 --exp_name q5_${NTU}_${NGSPTU} -ntu $NTU -ngsptu $NGSPTU -gpu_id $GPUID;
python $1 --env_name HalfCheetah-v4 --ep_len 150 --discount 0.90 --scalar_log_freq 1 -n 150 -l 2
    -s 32 -b 30000 -eb 1500 -lr 0.02 --exp_name q5_${NTU}_${NGSPTU} -ntu $NTU -ngsptu $NGSPTU
    -gpu_id $GPUID
echo "Done!"
```

3 Part 3: Soft Actor-Critic

Question 6: Run Soft Actor-Critic with more difficult tasks

Results

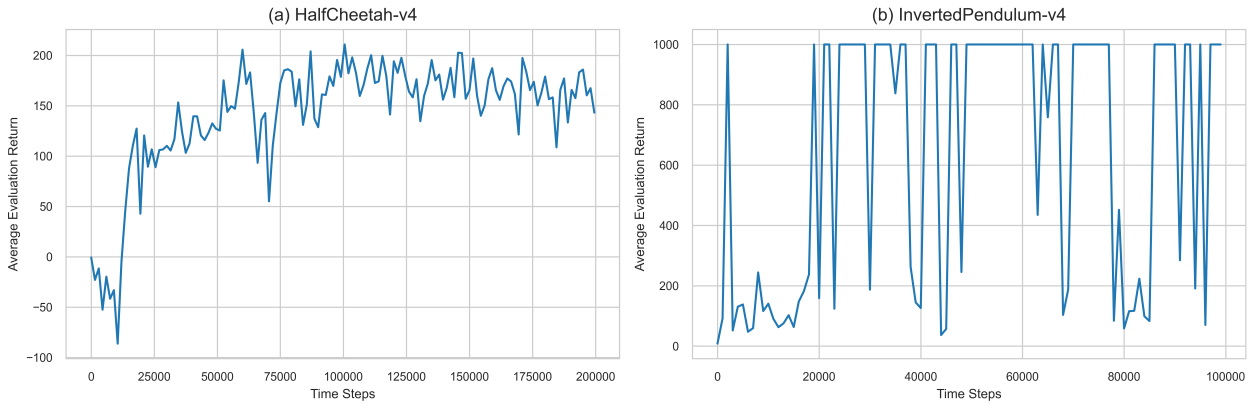

Figure 6. Average Evaluation Returns running Soft Actor-Critic on `HalfCheetah-v4` (left) and `InvertedPendulum-v4` (right). Average rewards on `HalfCheetah-v4` reaches around 200 after 50,000 steps, and rewards on `InvertedPendulum-v4` reaches $1,000$ after 20,000 steps. However, rewards on the second environment oscillate drastically indicating a potential high variance estimation of the state-value function.

Codes

```
echo "Running Homework 3 Question 6";
python cs285/scripts/run_hw3_sac.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.99
    --scalar_log_freq 1000 -n 100000 -l 2 -s 256 -b 1000 -eb 2000 -lr 0.0003 --init_temperature
    0.1 ----exp_name q6a_sac_InvertedPendulum_<parameters> --seed 1 -gpu_id $1;
python cs285/scripts/run_hw3_sac.py --env_name HalfCheetah-v4 --ep_len 150 --discount 0.99
    --scalar_log_freq 1500 -n 2000000 -l 2 -s 256 -b 1500 -eb 1500 -lr 0.0003 --init_temperature
    0.1 --exp_name q6b_sac_HalfCheetah_<parameters> --seed 1 -gpu_id $1
echo "Done!"
```