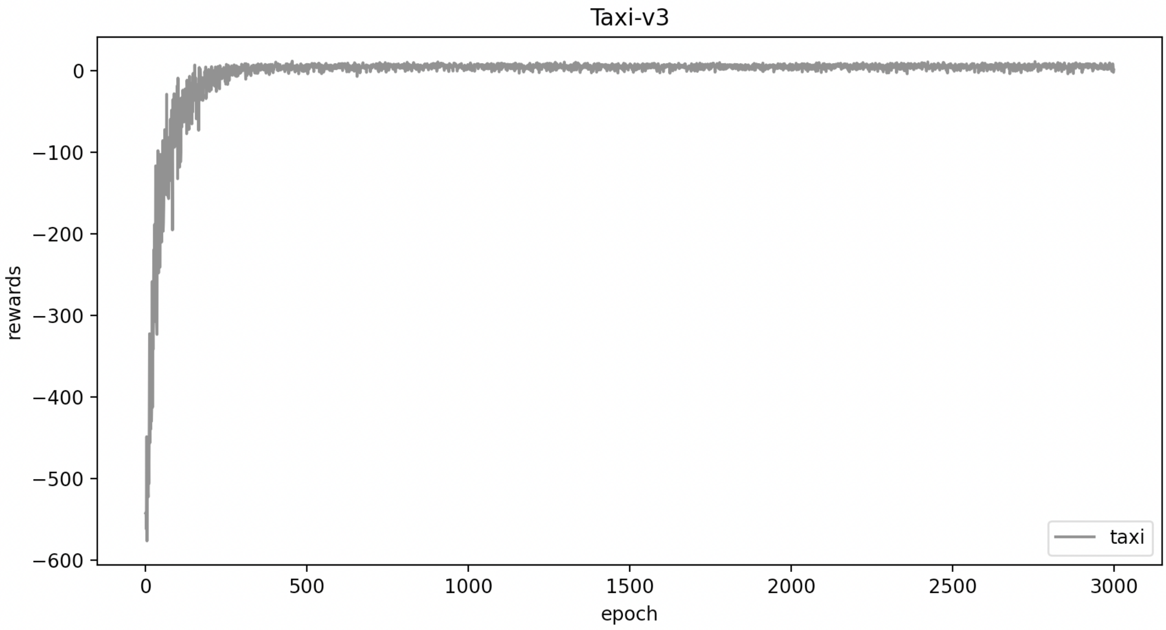
109550182 莊婕妤

AI Homework4 Report

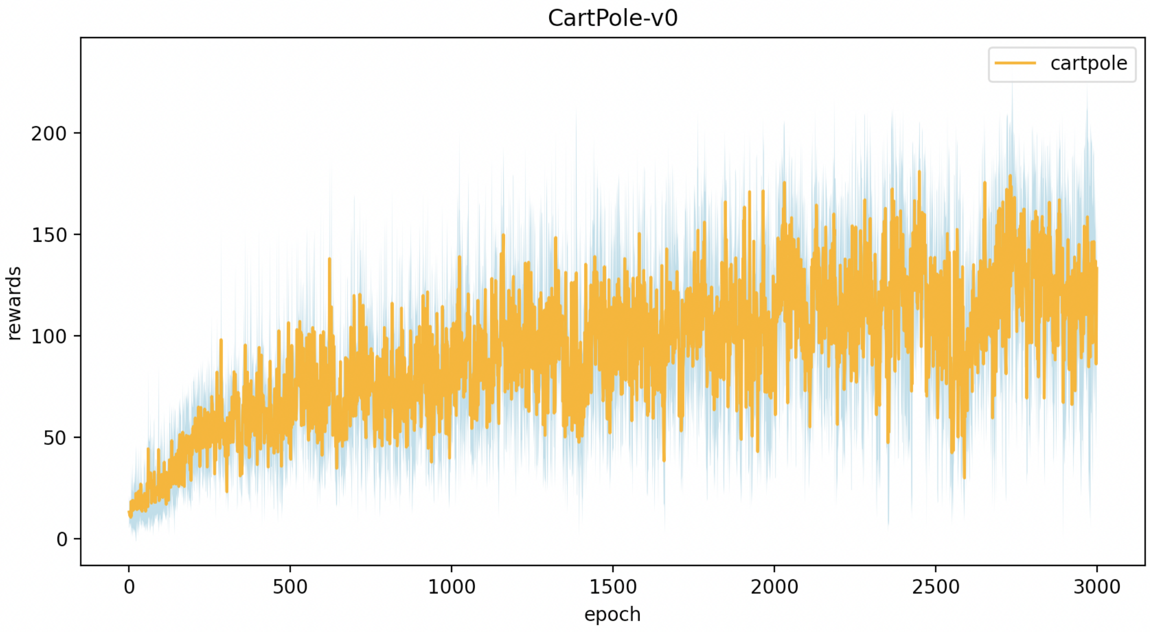
Part I. Experiment Results:

Please paste taxi.png, cartpole.png, DQN.png and compare.png here.

1. taxi.png

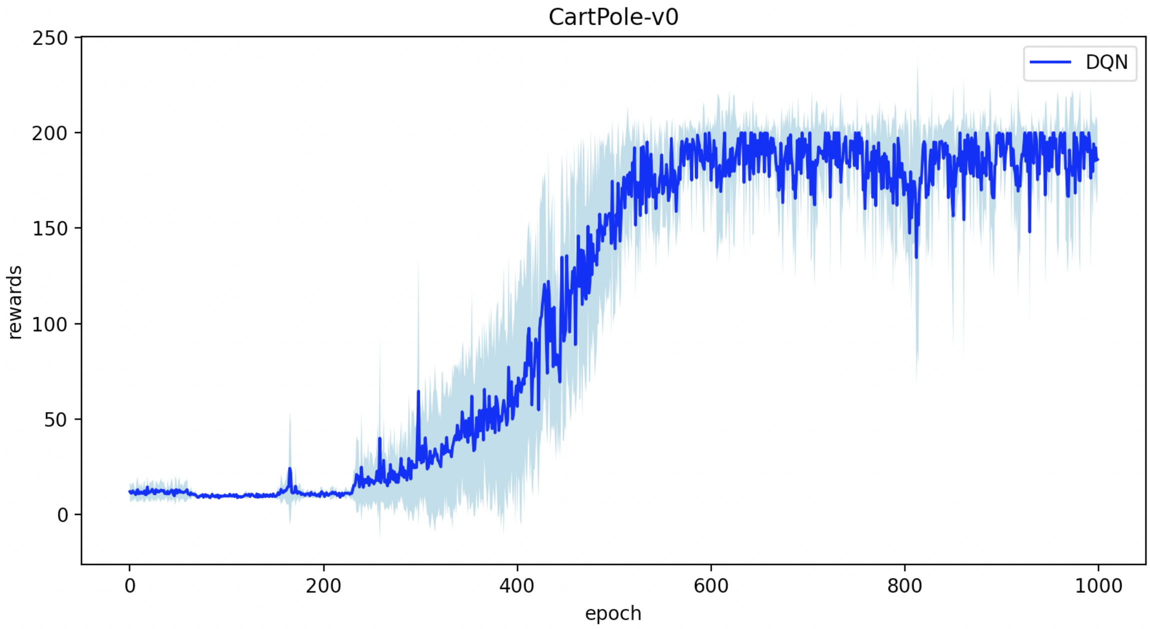


2. cartpole.png

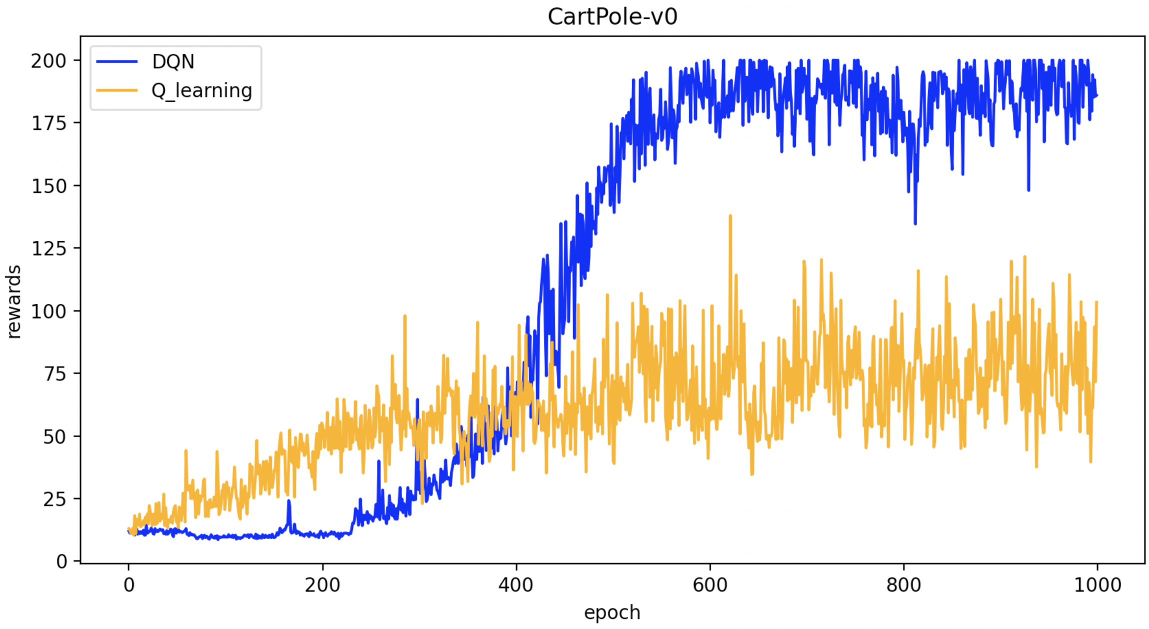


109550182 莊婕妤

3. DQN.png



4. compare.png



109550182 莊婕妤

Part II. Question Answering (50%):

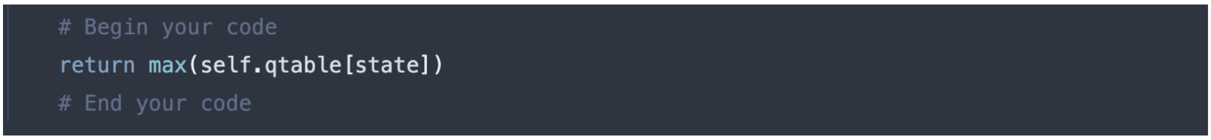
1. Calculate the optimal Q-value of a given state in Taxi-v3 (the state is assigned in google sheet), and compare with the Q-value you learned (Please screenshot the result of the "check_max_Q" function to show the Q-value you learned). **(4%)**

   The Optimal Q-Value I Calculated :

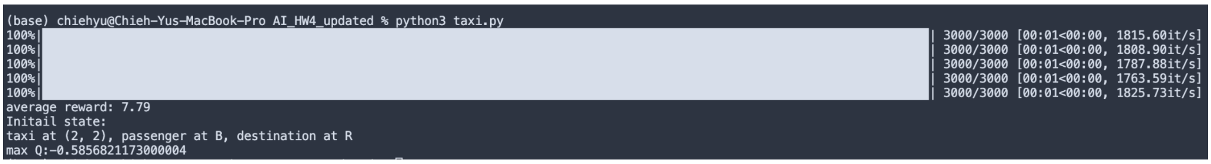
   Since the initial state is at (2, 2) and we have to send the passenger from B to R, the optimal solution needs 11 steps. Thus, the equation I used is below :

$$
\begin{aligned}
&(-1) \times (1-\gamma^{num\_steps})/(1-\gamma) \ + \ 20 \times \gamma^{num\_steps} \\
&= (-1) \times (1-0.9^{11})/(1-0.9) \ + \ 20 \times 0.9^{11} \\
&= -0.58568211...
\end{aligned}
$$

   It seems that the value is quite the same as the Q-value I learned.



```
        # Begin your code
        return max(self.qtable[state])
        # End your code
```



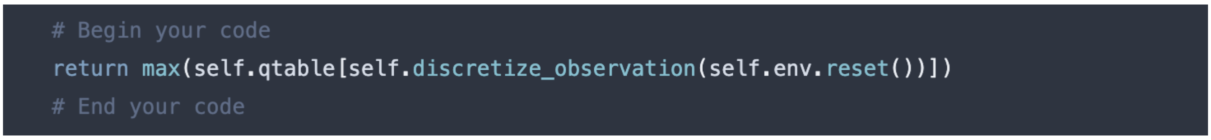
```
(base) chiehyu@Chieh-Yus-MacBook-Pro AI_HW4_updated % python3 taxi.py
100%|██████████████████████████████████████████████████| 3000/3000 [00:01<00:00, 1815.60it/s]
100%|██████████████████████████████████████████████████| 3000/3000 [00:01<00:00, 1808.90it/s]
100%|██████████████████████████████████████████████████| 3000/3000 [00:01<00:00, 1787.88it/s]
100%|██████████████████████████████████████████████████| 3000/3000 [00:01<00:00, 1763.59it/s]
100%|██████████████████████████████████████████████████| 3000/3000 [00:01<00:00, 1825.73it/s]
average reward: 7.79
Initail state:
taxi at (2, 2), passenger at B, destination at R
max Q:-0.5856821173000004
```

2. Calculate the max Q-value of the initial state in CartPole-v0, and compare with the Q-value you learned. (Please screenshot the result of the "check_max_Q" function to show the Q-value you learned) **(4%)**

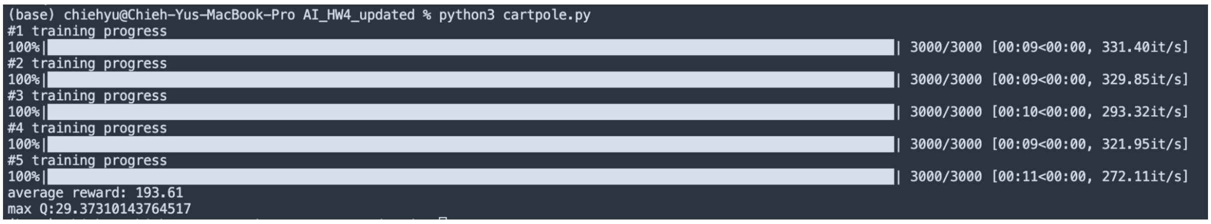
   The Optimal Q-Value I Calculated :

$$
\begin{aligned}
&(1-\gamma^{average\_reward})/(1-\gamma) \\
&(1-0.97^{193.61})/(1-0.97) \\
&= 33.2417...
\end{aligned}
$$

   In comparison to the Q-value I learned. The one I calculated is larger and I think the reason why is that there may be low estimated Q-value for early episodes.



```
        # Begin your code
        return max(self.qtable[self.discretize_observation(self.env.reset())])
        # End your code
```



```
(base) chiehyu@Chieh-Yus-MacBook-Pro AI_HW4_updated % python3 cartpole.py
#1 training progress
100%|██████████████████████████████████████████████████| 3000/3000 [00:09<00:00, 331.40it/s]
#2 training progress
100%|██████████████████████████████████████████████████| 3000/3000 [00:09<00:00, 329.85it/s]
#3 training progress
100%|██████████████████████████████████████████████████| 3000/3000 [00:10<00:00, 293.32it/s]
#4 training progress
100%|██████████████████████████████████████████████████| 3000/3000 [00:09<00:00, 321.95it/s]
#5 training progress
100%|██████████████████████████████████████████████████| 3000/3000 [00:11<00:00, 272.11it/s]
average reward: 193.61
max Q:29.37310143764517
```

3.

   a. Why do we need to discretize the observation in Part 2? **(2%)**

      The original states are continuous, which leads to infinite Q-values. Thus, we discretize the observation to make the states discrete so that we can build up the look-up table.

   b. How do you expect the performance will be if we increase “num_bins”? **(2%)**

      By increasing num_bins, the discretized states will be more precise, so I think that the performance will be better.

   c. Is there any concern if we increase “num_bins”? **(2%)**

      The training may take much more time since the number of states will increase. Also, it may take more space for the Q-table.

4. Which model (DQN, discretized Q learning) performs better in Cartpole-v0, and what are the reasons? **(3%)**

   From my observation, DQN performs better than discretized Q learning. I think that this is because DQN uses the neural network to learn to predict Q-values, which won’t require a lot of space as the other method.

5.

   a. What is the purpose of using the epsilon greedy algorithm while choosing an action? **(2%)**

      To balance exploration and exploitation, which leads to better reward.

   b. What will happen if we don’t use the epsilon greedy algorithm in the CartPole-v0 environment? **(3%)**

      Exploration and exploitation will not be balanced. Thus there might be situations where there is no exploitation, which makes the agent not exploiting what it learns , ending up with a very low utility. On the other hand, if there is no exploration, the agent will not try other actions once it finds an optimal policy.

c. Is it possible to achieve the same performance without the epsilon greedy algorithm in the CartPole-v0 environment? Why or Why not? **(3%)**

Yes, since we can also balance exploration and exploitation by the randomized probability matching algorithm. The algorithm works by sampling a probability distribution that describes the probable mean of the payoff.

d. Why don't we need the epsilon greedy algorithm during the testing section? **(2%)**

It is because we do not need to randomly explore states in the testing section, we only need to find the optimal result.

6. Why is there "with torch.no_grad():" in the "choose_action" function in DQN? **(3%)**

It reduces the training time by disabling gradient descent and backpropagation.

7.

a. Is it necessary to have two networks when implementing DQN? **(1%)**

No, it is not necessary.

b. What are the advantages of having two networks? **(3%)**

By having two networks, the Q-learning algorithm will be more stable and it solves the problem of overestimation, preventing the Q-values to diverge.

c. What are the disadvantages? **(2%)**

I think the disadvantage may be that we have to maintain two different networks, which takes more space and slightly more time.

8.

a. What is a replay buffer(memory)? Is it necessary to implement a replay buffer? What are the advantages of implementing a replay buffer? **(5%)**

The replay buffer is used to store experience tuples, resulting from past experiences. They are gradually added while interacting with the environment. To implement a replay buffer is not necessary. However, with this technique, we can make better use of our past experience rather than using the most recent experience.

Advantages of Experience Replay :

1. It enables the data to be reused, which means that we can learn from the same experience multiple times. This brings to more efficient use of past experiences.
2. We get a better convergence behavior by making the problem more like a supervised learning problem. (tackles the problem of unstable training)

b. Why do we need batch size? **(3%)**

Batch size determines the number of samples trained. The reason why we need batch size is that it is nearly impossible to handle a huge amount of data at a time due to hardware / memory constraints and other factors.

c. Is there any effect if we adjust the size of the replay buffer(memory) or batch size? Please list some advantages and disadvantages. **(2%)**

Larger replay buffer or batch size means more data are stored or sampled. This leads to more accurate approximation, while increasing the time of each iteration.

Smaller replay buffer or batch size means that less data is stored or sampled. This results in a shorter training time while having a lower accuracy.

9.

a. What is the condition that you save your neural network? **(1%)**

I chose to save my neural network when the mean of total_rewards is above 30.

b. What are the reasons? **(2%)**

This is because I only want to save my neural network when the rewards are high enough. Since the max_Q I calculated is about 32, I think a number of 30 would be appropriate.

109550182 莊婕妤

10. What have you learned in the homework? **(2%)**

I learned what OpenAI Gym and Pytorch are and some functions related to them. I also got more familiar with how Q-learning and DQN are implemented under different environments. Although this homework is quite challenging to me, I still think this is a great practice for me to really implement those RL algorithms learnt in class.