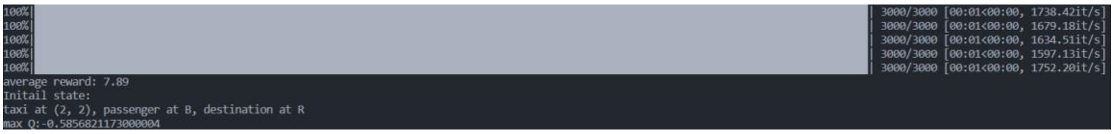
Homework 4:

Reinforcement Learning

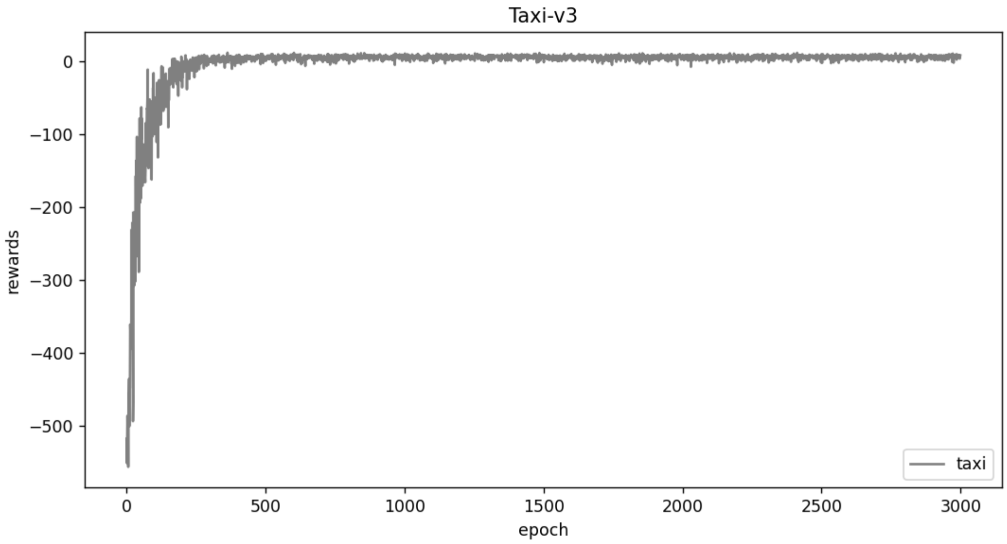
109550136 邱弘竣

Part I. Experiment Results (the score here is included in your implementation):

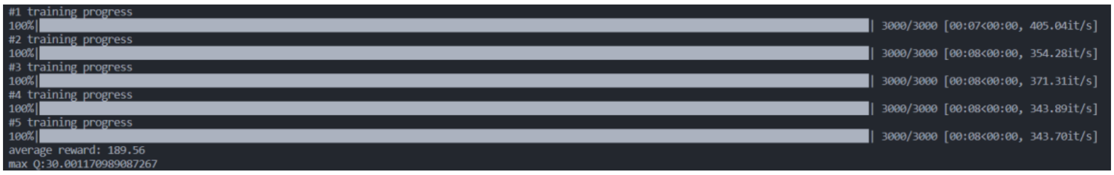
1. taxi.png:



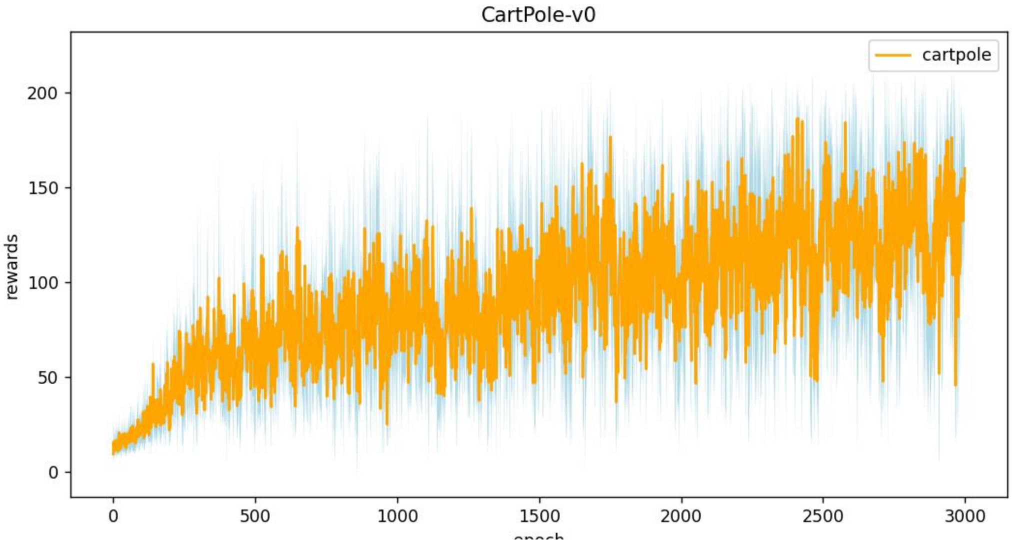
```
100%|████████████████████████████████████████| 3000/3000 [00:01<00:00, 1738.42it/s]
100%|████████████████████████████████████████| 3000/3000 [00:01<00:00, 1679.18it/s]
100%|████████████████████████████████████████| 3000/3000 [00:01<00:00, 1634.51it/s]
100%|████████████████████████████████████████| 3000/3000 [00:01<00:00, 1597.13it/s]
100%|████████████████████████████████████████| 3000/3000 [00:01<00:00, 1752.20it/s]
average reward: 7.89
Initail state:
taxi at (2, 2), passenger at B, destination at R
max Q:-0.5856821173000004
```



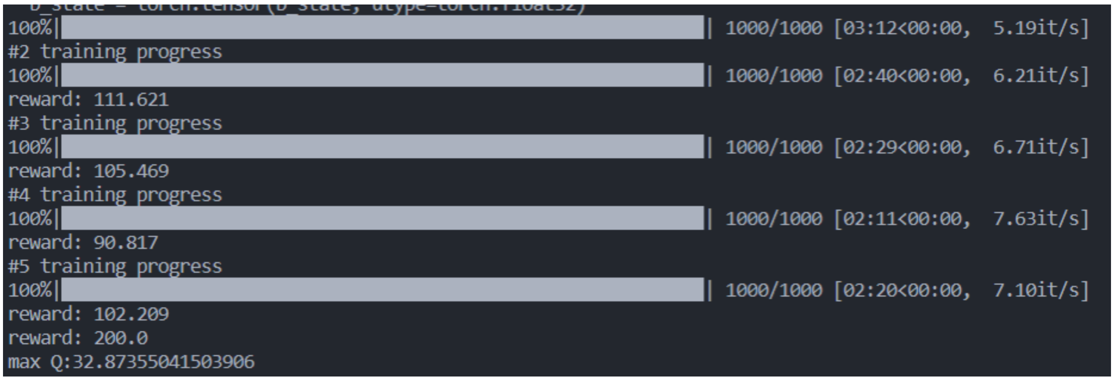
2. cartpole.png



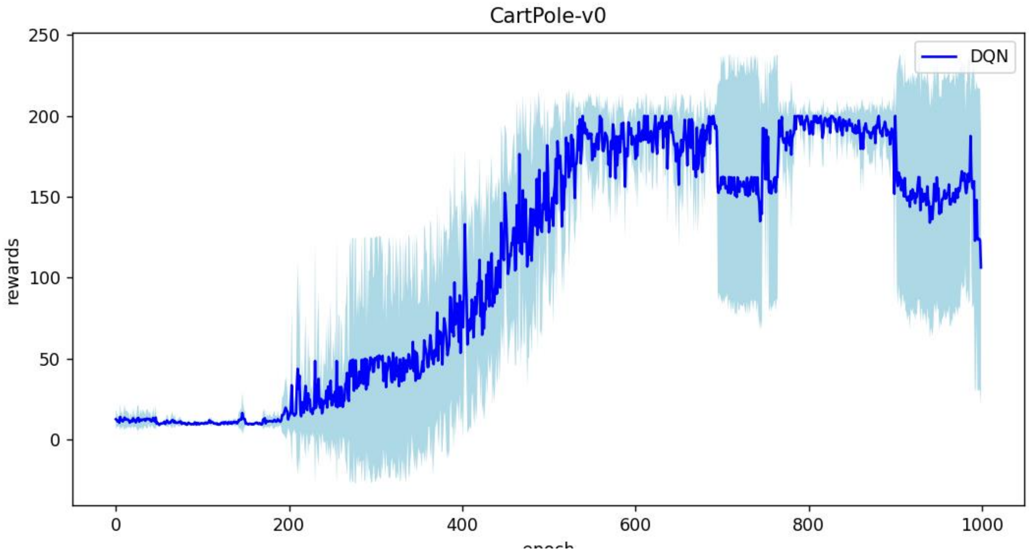
```
#1 training progress
100%|████████████████████████████████████████| 3000/3000 [00:07<00:00, 405.04it/s]
#2 training progress
100%|████████████████████████████████████████| 3000/3000 [00:08<00:00, 354.28it/s]
#3 training progress
100%|████████████████████████████████████████| 3000/3000 [00:08<00:00, 371.31it/s]
#4 training progress
100%|████████████████████████████████████████| 3000/3000 [00:08<00:00, 343.89it/s]
#5 training progress
100%|████████████████████████████████████████| 3000/3000 [00:08<00:00, 343.70it/s]
average reward: 189.56
max Q:30.001170989087267
```



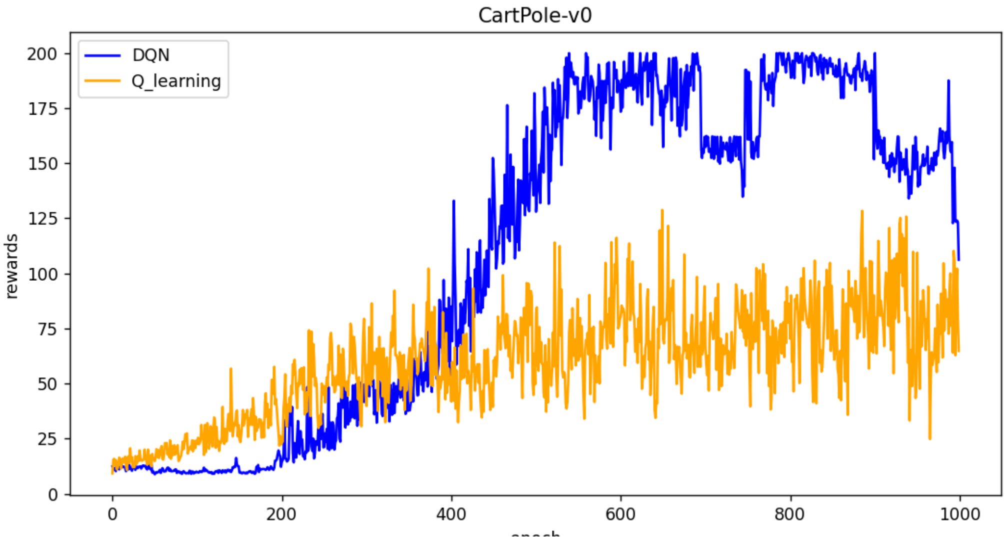
3. DQN.png



```
b_state = torch.tensor(b_state, dtype=torch.float32)
100%|██████████████████████████████████████████| 1000/1000 [03:12<00:00,  5.19it/s]
#2 training progress
100%|██████████████████████████████████████████| 1000/1000 [02:40<00:00,  6.21it/s]
reward: 111.621
#3 training progress
100%|██████████████████████████████████████████| 1000/1000 [02:29<00:00,  6.71it/s]
reward: 105.469
#4 training progress
100%|██████████████████████████████████████████| 1000/1000 [02:11<00:00,  7.63it/s]
reward: 90.817
#5 training progress
100%|██████████████████████████████████████████| 1000/1000 [02:20<00:00,  7.10it/s]
reward: 102.209
reward: 200.0
max Q:32.87355041503906
```



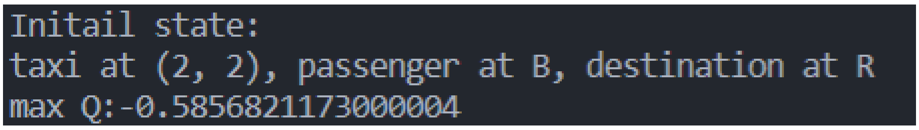
4. compare.png



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%)



```
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%)

3.

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

Continuous data costs the memory space too much, and it causes too many states to visit during training.

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

Increasing “num_bins” will get better performance.

Because more bins can let the model account for more specific state space.

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

it might cost too much storage space, time, and computational power.

4. Which model (DQN, discretized Q learning) performs better in Cartpole-v0,

and what are the reasons? (3%)

DQN performs better. Because neural network can match different environment and automatically extract feature from environment. However, Q table is fixed.

5.

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

Because we have to take into account exploration and exploitation and get balance.

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

We will only consider the things we have already learned, and don’t have to explore the new action we have never checked before.

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, we can use other algorithm such as the softmax action selection strategy which decides the relative levels of exploitation and exploration by mapping values into action probabilities.

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

We have finished the training.

In testing section, we don’t need to explore new actions. Instead, we use the actions which are explored when training.

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

The only goal is to pick one using the current weight instead of storing all the tensors involved in computing the output into a graph, then it’s better to use torch.no_grad(). Without using it, the code may be a bit slower.

7.

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

No.

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

If we overestimate on one Q network, then we have the second will hopefully control this bias when we would take the max.

It not only improves accuracy in the action-values but also improves the policy learned.

c. What are the disadvantages? (2%)

It costs more time and space to maintain two network.

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%)

replay buffer can “remember” the things it already learned. It stores the transitions that the agent observes, allowing us to reuse this data later. It is not necessary to implement a replay buffer, but if we have a huge replay buffer, we can make training a little volatile.

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

It’s hard for us to sent all the data to train.

A batch size means that samples from the training dataset will be used to estimate the error gradient before the model weights are updated

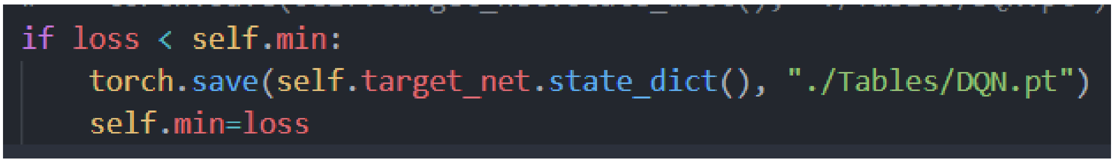
c. Is there any effect if we adjust the size of the replay buffer(memory)

or batch size? Please list some advantages and disadvantages. (2%)

Bigger batch size needs more memory and each step is time consuming. However, bigger the batch size, lesser is the noise in the gradients and so better is the gradient estimate. This allows the model to take a better step towards a minima.

9.

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



```
if loss < self.min:
    torch.save(self.target_net.state_dict(), "./Tables/DQN.pt")
    self.min=loss
```

Loss is the value computed after loss function.

b. What are the reasons? (2%)

I expect loss to be as small as possible. If the value after learning and the expected value is almost the same, it means that this module is nearly the best module.

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

When I was in senior high school, I took the courses about RL (the simplest one). Teacher taught in a very easy way to let us know how the neural network works. As a result, mentioned to RL, I thought I had to write a lot of code about gradient descent, forward propagation and backward propagation. After this homework, I knew that python is very convenient because we can import modules like PyTorch to help us implement what we want to do easily.