Deep Learning Practice Lab 8

0856056 Yu-Wun Tseng

June 2020

1 Introduction

The goal of this lab is to implement two deep reinforcement algorithms to solve LunarLander-v2. LunarLander is a simulation game where players control the lander so that the lander can land without crashing. The game has two versions, one with four actions and the other with continuous actions. The first algorithm is deep Q-network(DQN), which is suitable for discrete actions. The second algorithm is deep deterministic policy gradient(DDPG), which is suitable for continuous actions.

2 Experiment Setups

This lab is done by using Pytorch and the sample code provided by TA. The environment using the Gym toolkit published by OpenAI [1].

2.1 DQN

There are two network in DQN, one is behavior network ant the other is target network. The behavior network is used to generate the q-value with the current action in every episode. The target network is used to estimate the target q-value. The two network have the same architecture, and the target network uses the parameters of the behavior network to update the parameters once in a while. In my implementation, the network has three fully connected layers, except for the last layer, there is a relu layer behind each layer.

In DQN, the epsilon greedy is used. The model has a probability of epsilon to randomly choose actions, otherwise, choose the action with the highest q-value generated by the behavior network. The epsilon is 1 at the beginning, each episode will be multiplied by 0.995, and the minimum value is 0.1.

After selecting the action, the action will be executed in the current environment and the environment will return the next state, reward and whether the game is over. The state, action, reward, next state and whether the game is over will compose a transition set, every transition will be stored in reply memory. When training, the model will sample a mini-batch transitions from the reply memory.

To train the model and update the behavior network, we first sample a mini-batch transitions from the reply memory. Next, we need to calculate the highest q-value with the states sample from reply memory using the behavior network. After then, we calculate the highest next q-value with the next states using the target network. The target value is calculated using reward and the next q-value, as the equation below:

$$y_j = r_j + \gamma \times \max_{a'} \hat{Q}(s_{j+1}, a' | \theta^-) \tag{1}$$

We expect the q-value to be as similar to the next q-value as possible, in other words, the error between the q-value and the next q-value is as small as possible. Therefore, we use the mean squared error as the loss function to calculate the loss, and finally update the behavior network.

In addition, in order to update the target network, we copy the parameters from the behavior network to the target network every four episodes. While in my implement, I use the soft update to update the target network. The equation of the soft update is as below:

$$\theta^- = \tau\theta + (1 - \tau)\theta^- \tag{2}$$

Where θ^- is the parameter of the target network and the θ is the parameter of the behavior network. The soft update makes the target network more stable and the model performance is better.

The learning rate is 5e-4, the number of warm-up steps is 10000 and the capacity of reply memory is 10000. The mini-batch size is 128, the τ is 1e-3 and the γ is 0.99. The update frequency of the behavior network and the target network is 4. The episode is 1200, but early stop when the average reward of the nearly 100 episodes larger than 200.

2.2 DDPG

Similar to DQN, there is a behavior network, a target network and a reply memory in DDPG. Each network has a actor network and a critic network. The actor network generates the action distribution, the action is selected by adding Gaussian noise to the distribution. Given the action generated by the actor network and the current state, the critic network estimates the corresponding q-value. In this lab, the actor network has three fully connected layers. There is a relu layer behind the first and the second layer, and a tanh layer behind the third layer. There are also three layers fully connected layers in the critic network, but only the first and the second layer have a relu layer. Compared with the actor network, the input of the critic network is the concatenation of states and actions, not just states.

The behavior actor network is used to generate the action distribution given the current state, and to select the action by adding Gaussian noise to the distribution. After selecting the action, the action will be executed in the current environment as DQN, and return the transition information. The transitions will also be stored in reply memory.

To train the model and update the behavior network, we first calculate the critic loss. Similar to DQN, we use the behavior critic network is calculate the q-value with the states sample from reply memory. After then, we use target actor network to select the next action, and feed the action and next state to the target critic network to get the next q-value. The calculation method of the target value is the same as DQN. Finally, using the mean squared error as the loss function to calculate the loss of q-value and next q-value, and update the behavior critic network.

After update the behavior critic network, we calculate the actor loss. We want to maximize the expected value of the q-values, thus, the loss is the negative sum of q-values.

$$\nabla_\theta J(\mu_\theta) \approx \mathbb{E}_\mu[\nabla_\theta Q(s_t, \mu(s_t|\theta)|\omega)] \tag{3}$$

We use the behavior actor network to select the action, and use the behavior critic network to calculate the q-value. Finally, use the negative value of the sum of q-values as loss to update the network.

To update the target network, the soft update is implemented. The detail of the soft update is mentioned in the previous session. Both the actor network and the critic network using the same method. However, unlike DQN, the target networks are updated every episode.

The learning rate of actor network is 1e-3, and the learning rate of critic network is 1e-3. The number of warm-up steps is 10000 and the capacity of reply memory is 500000. The mini-batch size is 64, the τ is 5e-3 and the γ is 0.99.

2.3 Explanation

The discount factor gamma controls the portion of the target value affected by the next q-value. If the gamma is 1 means that the next q-value is taken into account in the target value. If the gamma is 0 means that the model only use the reward to update the parameters.

Greedy action selection is to choose the actions randomly, but there is no guarantee that the action will get the highest reward. The epsilon-greedy algorithm uses epsilon to control action selection strategies, i.e., exploitation or exploration. The model has a probability of epsilon to explore actions with higher reward, and has a probability of (1-epsilon) to exploit the best action to obtain the highest reward.

In order to maintain stability, the target network does not frequently change the parameters significantly. For general DQN, the update frequency of the target network is less than the behavior network. For DDPG, due to the large gamma, the proportion of updated parameters is small. Therefore, the model can use a more stable network to calculate the target value.

The larger buffer size can store more transitions and contain a wide range of experiences, thus, the model will have more stable behavior. If the buffer size id too large, the model will learn slowly. The smaller buffer size stores less transitions, the model can learn faster. However, if the buffer size is too small, the model will overfit the small amount transitions.

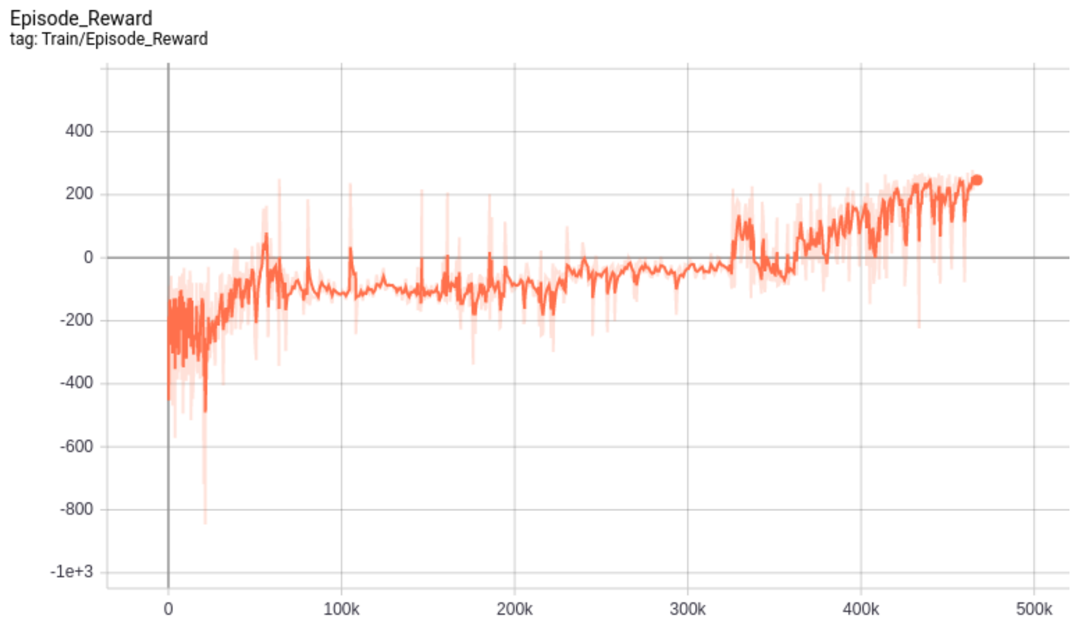


Figure 1: Episode Reward Curve of DQN

3 Results

3.1 DQN

The training episode reward curve is shown in Figure 1, training ewma reward curve is shown in Figure 2. The number of training episodes is 1800, but early stopping at 820 as Figure 3.

The testing average reward is 244 as shown in Figure 4, and the testing reward curve is shown in Figure 5.

3.2 DDPG

The training episode reward curve is shown in Figure 6, training ewma reward curve is shown in Figure 7. The number of training episodes is 1200 as Figure 8.

The testing average reward is 285 as shown in Figure 9, and the testing reward curve is shown in Figure 10.

4 Discussion

I found that there are two methods to randomly select actions in DQN. One is to sample the actions from action space, the other is to use a random number between 1 to 4. I use the latter, and my classmate use the former. The performance of the models are not much different.

At the beginning, I didn't use the soft update in DQN. The model still couldn't learn after 1800 episodes. After I discussed with my classmate and watched the DDPG videos, I found that the soft update not only used in DDPG, but also commonly used on DQN. Thus, I added it into my model. After I used the soft update, the model can learn well.

References

[1] *Gym*. URL: `https://gym.openai.com/`.

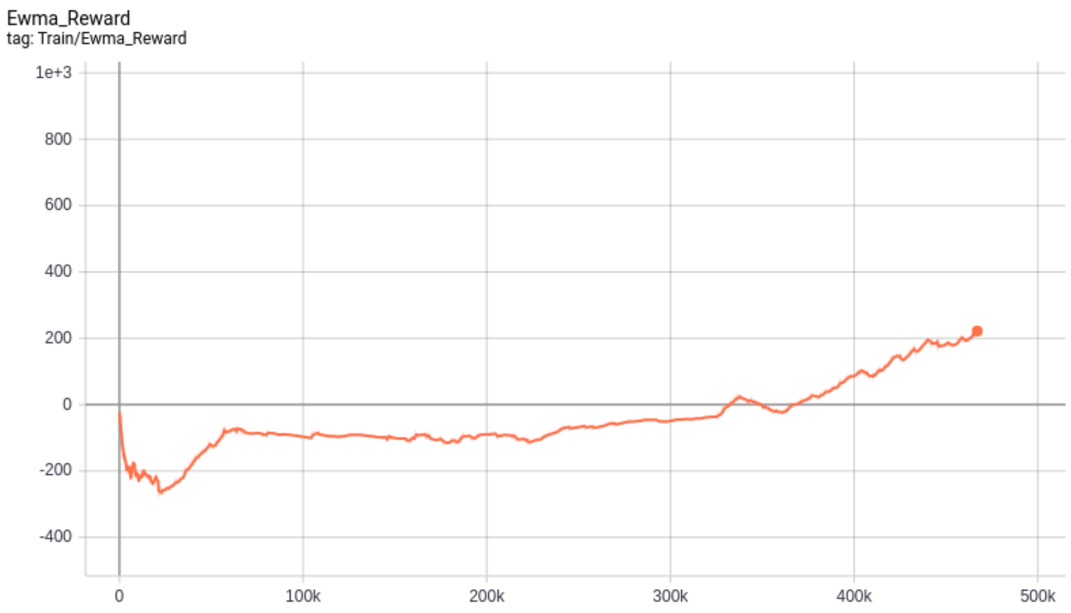
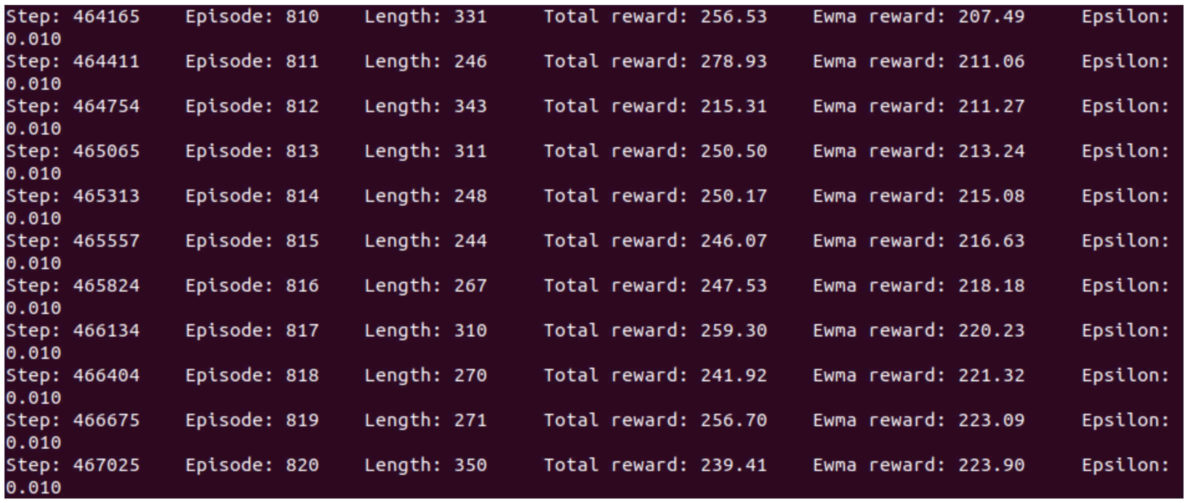
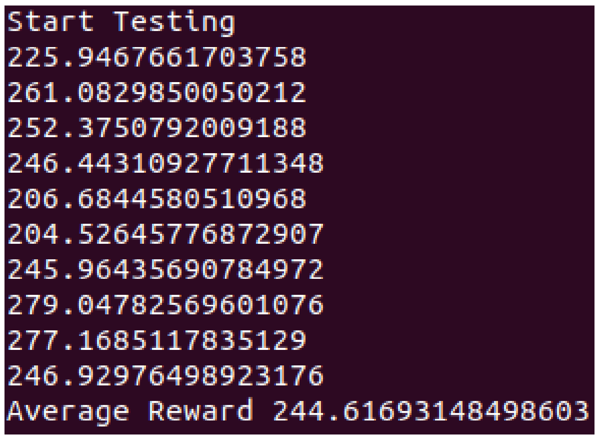


Figure 2: Ewma Reward Curve of DQN



```
Step: 464165    Episode: 810    Length: 331    Total reward: 256.53    Ewma reward: 207.49    Epsilon:
0.010
Step: 464411    Episode: 811    Length: 246    Total reward: 278.93    Ewma reward: 211.06    Epsilon:
0.010
Step: 464754    Episode: 812    Length: 343    Total reward: 215.31    Ewma reward: 211.27    Epsilon:
0.010
Step: 465065    Episode: 813    Length: 311    Total reward: 250.50    Ewma reward: 213.24    Epsilon:
0.010
Step: 465313    Episode: 814    Length: 248    Total reward: 250.17    Ewma reward: 215.08    Epsilon:
0.010
Step: 465557    Episode: 815    Length: 244    Total reward: 246.07    Ewma reward: 216.63    Epsilon:
0.010
Step: 465824    Episode: 816    Length: 267    Total reward: 247.53    Ewma reward: 218.18    Epsilon:
0.010
Step: 466134    Episode: 817    Length: 310    Total reward: 259.30    Ewma reward: 220.23    Epsilon:
0.010
Step: 466404    Episode: 818    Length: 270    Total reward: 241.92    Ewma reward: 221.32    Epsilon:
0.010
Step: 466675    Episode: 819    Length: 271    Total reward: 256.70    Ewma reward: 223.09    Epsilon:
0.010
Step: 467025    Episode: 820    Length: 350    Total reward: 239.41    Ewma reward: 223.90    Epsilon:
0.010
```

Figure 3: Training Result of DQN



```
Start Testing
225.9467661703758
261.0829850050212
252.3750792009188
246.44310927711348
206.6844580510968
204.52645776872907
245.96435690784972
279.04782569601076
277.1685117835129
246.92976498923176
Average Reward 244.61693148498603
```

Figure 4: Testing Result of DQN

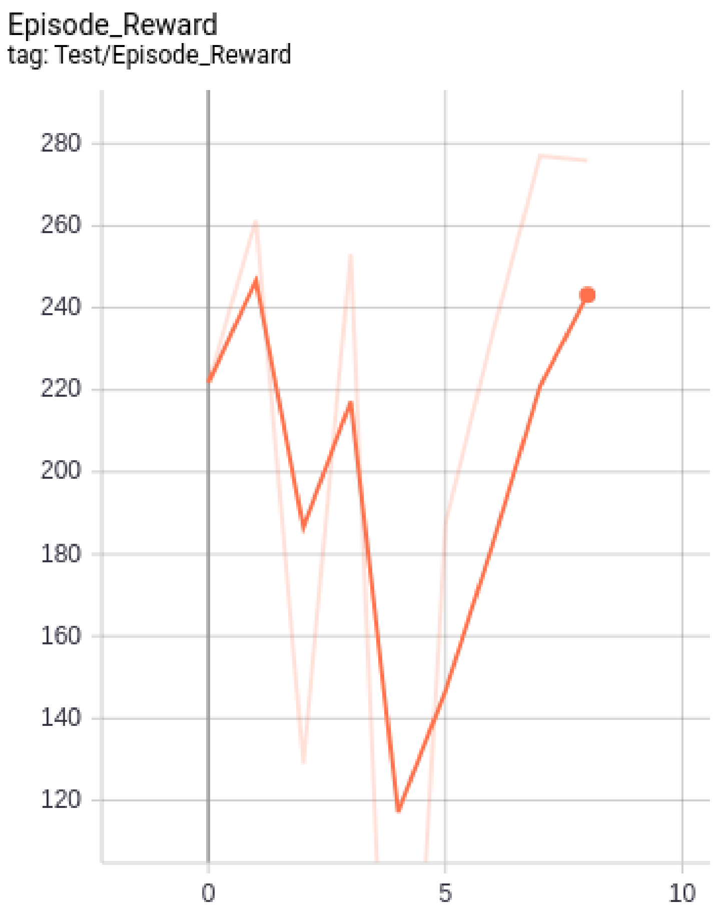


Figure 5: Testing Reward Curve of DQN

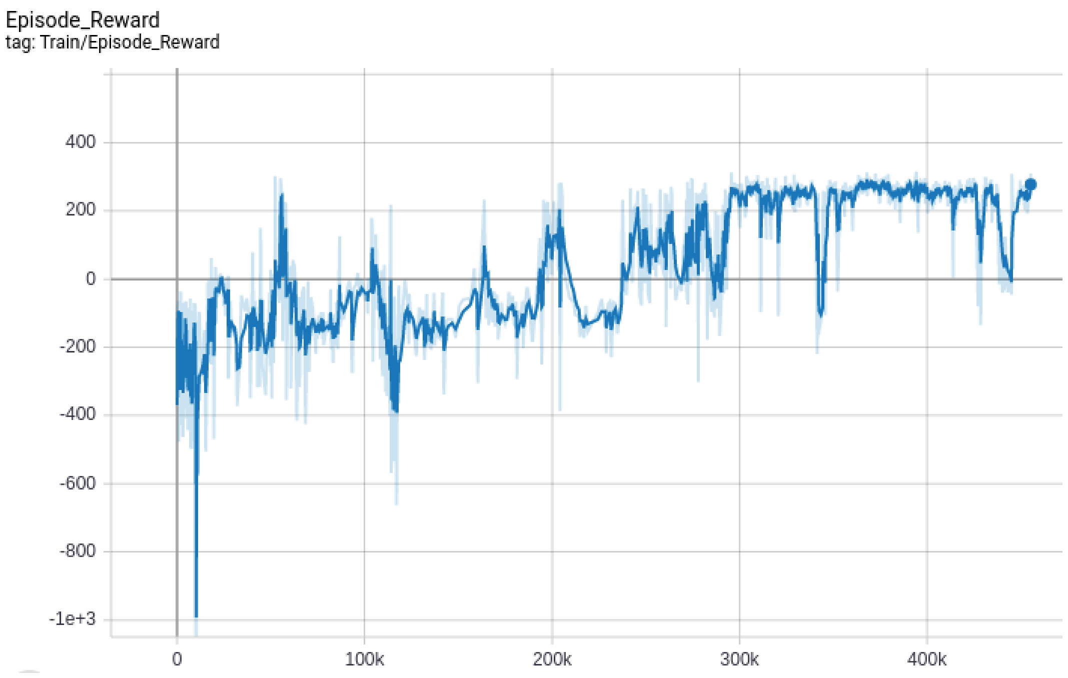


Figure 6: Episode Reward Curve of DDPG

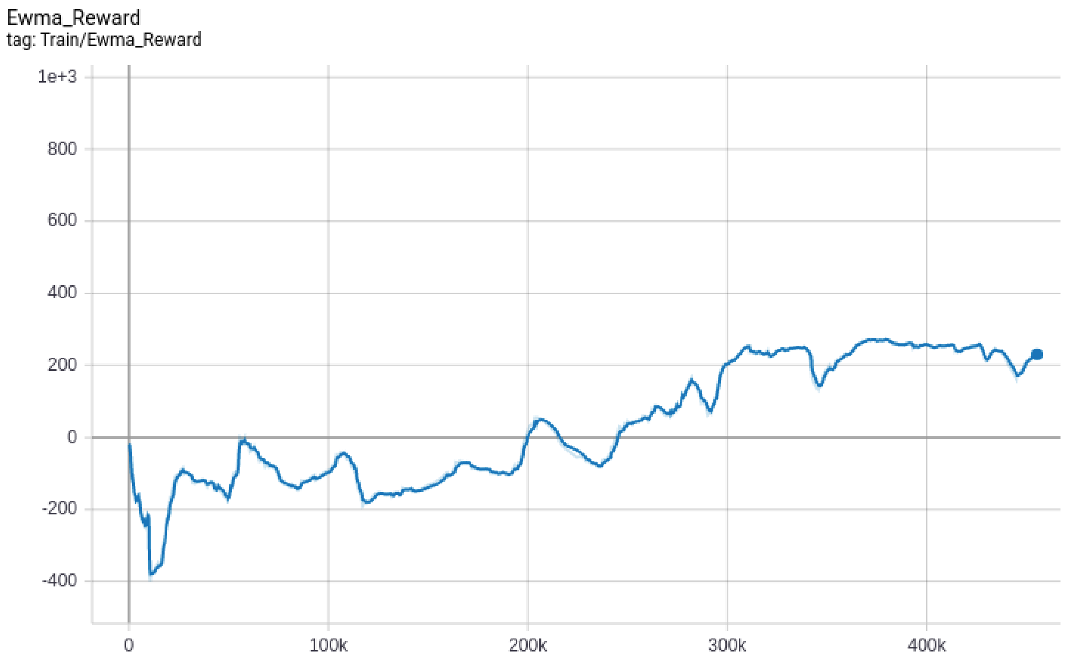
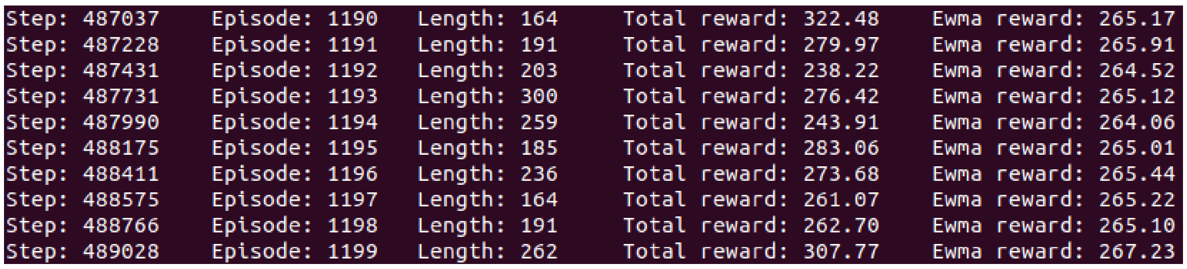
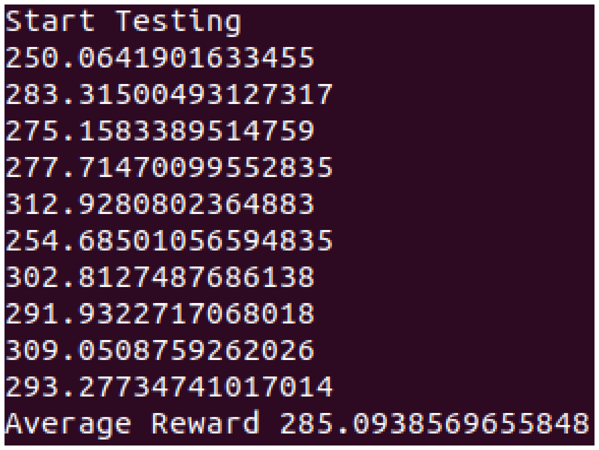


Figure 7: Ewma Reward Curve of DDPG



```
Step: 487037    Episode: 1190   Length: 164     Total reward: 322.48    Ewma reward: 265.17
Step: 487228    Episode: 1191   Length: 191     Total reward: 279.97    Ewma reward: 265.91
Step: 487431    Episode: 1192   Length: 203     Total reward: 238.22    Ewma reward: 264.52
Step: 487731    Episode: 1193   Length: 300     Total reward: 276.42    Ewma reward: 265.12
Step: 487990    Episode: 1194   Length: 259     Total reward: 243.91    Ewma reward: 264.06
Step: 488175    Episode: 1195   Length: 185     Total reward: 283.06    Ewma reward: 265.01
Step: 488411    Episode: 1196   Length: 236     Total reward: 273.68    Ewma reward: 265.44
Step: 488575    Episode: 1197   Length: 164     Total reward: 261.07    Ewma reward: 265.22
Step: 488766    Episode: 1198   Length: 191     Total reward: 262.70    Ewma reward: 265.10
Step: 489028    Episode: 1199   Length: 262     Total reward: 307.77    Ewma reward: 267.23
```

Figure 8: Training Result of DDPG



```
Start Testing
250.0641901633455
283.31500493127317
275.1583389514759
277.71470099552835
312.9280802364883
254.68501056594835
302.8127487686138
291.9322717068018
309.0508759262026
293.27734741017014
Average Reward 285.0938569655848
```

Figure 9: Testing Result of DDPG

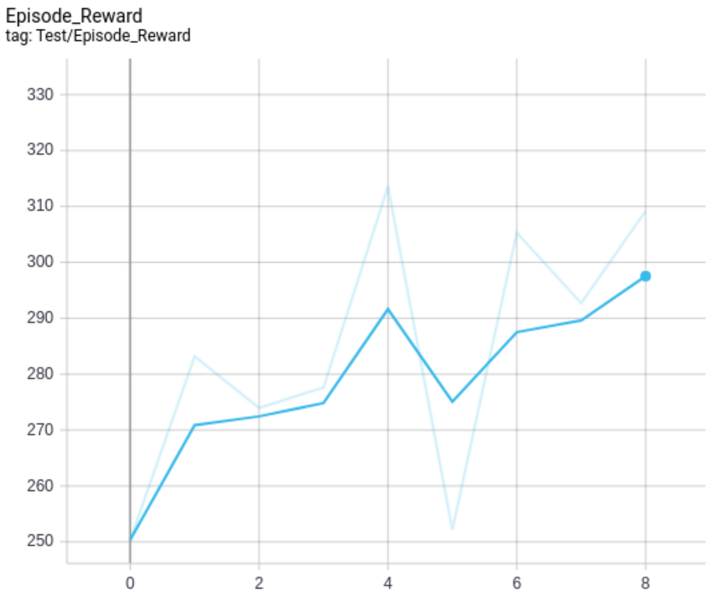


Figure 10: Testing Reward Curve of DDPG