EL2805 Reinforcement Learning
Computer Lab 2

Jingxuan Mao	Yuqi Zheng
001214-9068	990122-1243
jmao@kth.se	yuqizh@kth.se

December 23, 2022

1 Problem 1: Deep Q-Networks (DQN)

(b)

Reason for using a replay buffer and target network in DQN: The replay buffer stores experience collected in different training episodes. When we randomly sample experience from the memory, we could remove the correlation between the samples of the experience since they can come from various episodes, which exploits the data and improves the learning efficiency of the model.
We use a fixed target for certain successive steps in order to avoid too frequent update of θ which could make the network get tracked.

(d)

- Layout of the network: We use a 3-layer network with input dimension of 8 as that of state, 64 neurons in each layer, and output dimension of 4 as the number of actions. Since there are not too many data, simple neural network is enough to achieve a good learning result.
- Choice of the optimizer: We use the Adam optimizer because it can avoid the network getting stuck in the local minima. In addition, it can adjust the learning rate automatically.
- Parameters $(\gamma, L, T_E, C, N, \epsilon)$: By trail and error, we found that with $(\gamma = 0.99, L = 100000, T_E = 400, C = 5, N = 64)$, the network could converge successfully. ϵ is changing in different episodes, decreasing exponentially from 0.9 to 0.5, since the agent should more tend to choose the action based on trained policy network instead of acting randomly.

(e)

(1) The total episodic reward and the total number of steps taken per episode during training are shown in Fig. 1.

We can see that as the training proceeds, the total reward is rising in a fluctuating manner. And from a certain time instant, the total reward would have a strong improvement. Also, the number of steps in each episode is gradually decreasing during the training, implying the agent learns the right action.

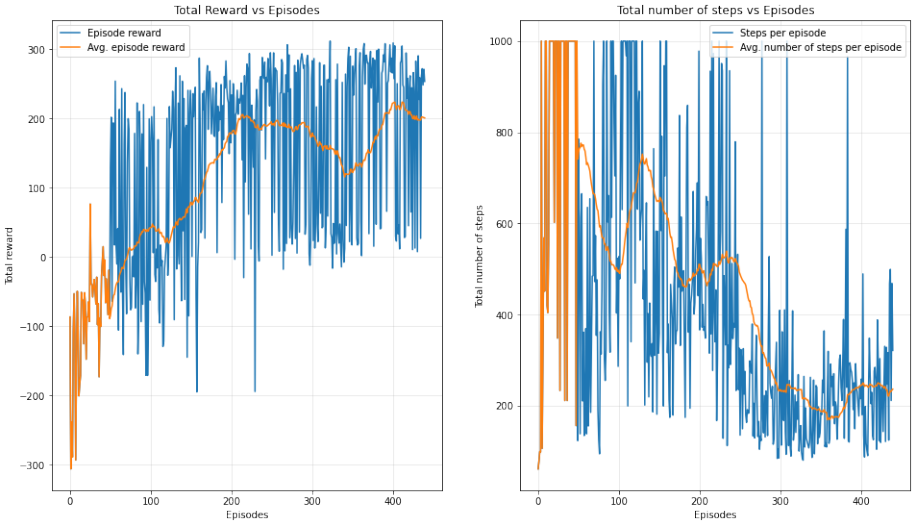

Figure 1: Total reward and total number of steps vs. episode, $\gamma_0 = 0.99$.

(2) The plots for $\gamma_1 = 1$ and $\gamma_2 = 0.5$ are shown in Fig. 2 an Fig. 3.

We can see that for neither $\gamma_1 = 1$ nor $\gamma_2 = 0.5$, the agent can learn the right policy within 400 episodes, where for $\gamma_0 = 0.99$, the agent could reach the average score of 200 successfully. Therefore, the choice of γ is of vital importance.

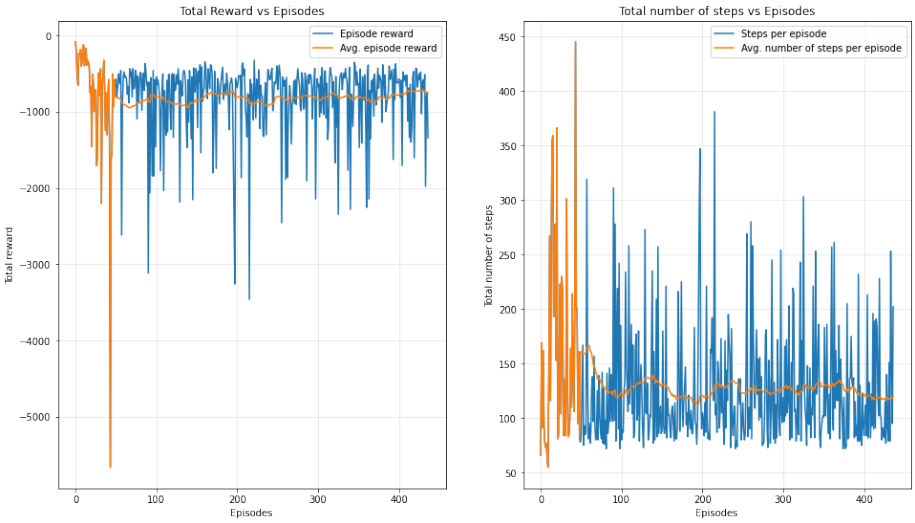

Figure 2: Total reward and total number of steps vs. episode, $\gamma_1 = 1$.

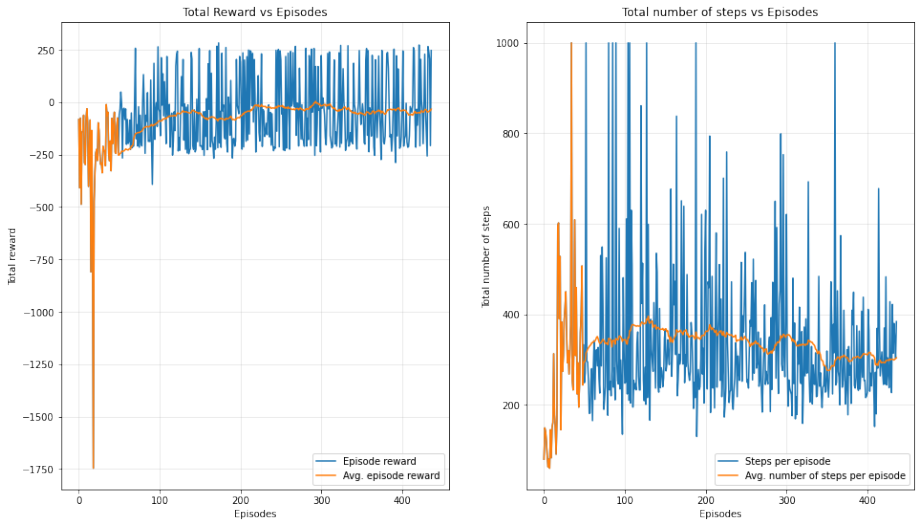

Figure 3: Total reward and total number of steps vs. episode, $\gamma_2 = 0.5$.

(3) For $\gamma_0 = 0.99$, the plots for episodes of 200 and 600 are shown in Fig. 4 and Fig. 5.

We can see that the agent can reach an average score of 200 after 400 episodes. The total reward is not stable enough in 200 episodes, although it is close to 200. After 600 episodes, the total reward is stable, but it is not necessary to train so many episodes. Once the average reward achieves 200, the training can be terminated.

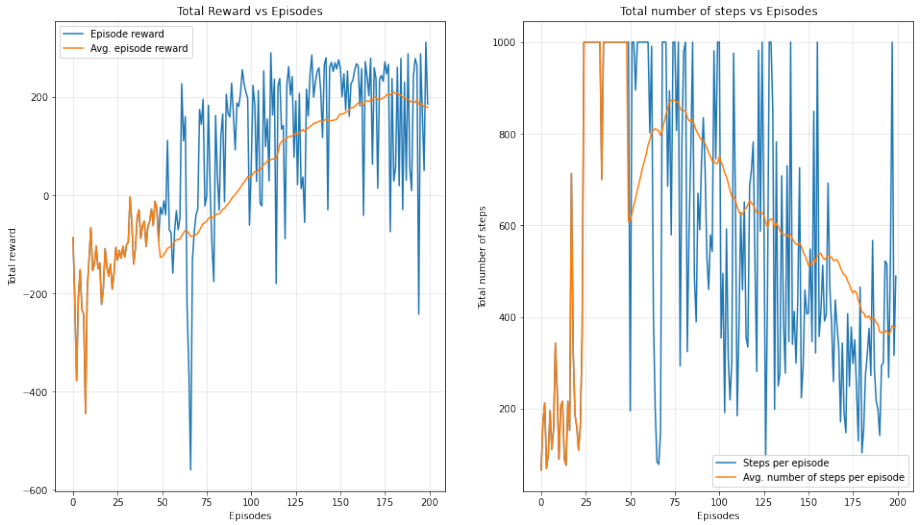

Figure 4: Total reward and total number of steps vs. episode, 200 episodes.

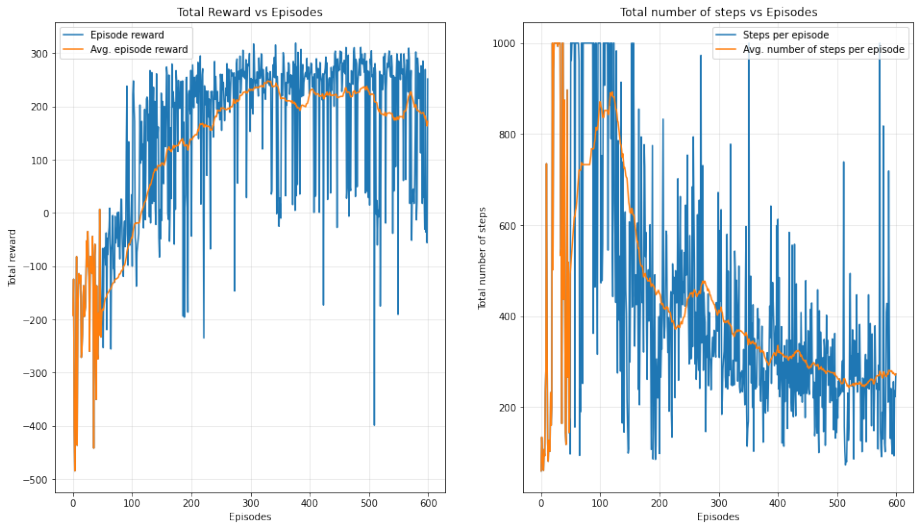

Figure 5: Total reward and total number of steps vs. episode, 600 episodes.

(f)

(1) For $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$, $Q_\omega(s(y, \omega), \pi_\theta(s(y, \omega)))$ is shown in Fig. 6.

We can see that the Q values are symmetric about the angle and the minima occur when ω is 0, which means that the the left side has the same influence as the right side. However, we can also find that the values are not strictly symmetric, so there's space for improvements in the algorithm.

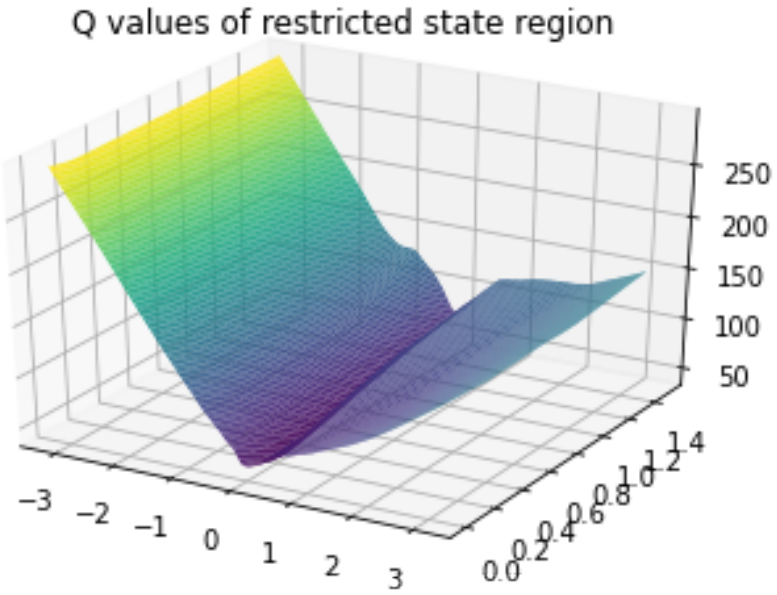

Figure 6: Q values of restricted state region.

(2) For $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$, $\pi_\theta(s(y, \omega))$ is shown in Fig. 7.

We can see that there are four kinds of action values and a clear procedure of transition in the figure. When the angle is negative, the lander takes

action 0. As the angle increases, the lander will successively take action 1, 2 and 3. The transition between actions are gradual and fluctuating, so there's planes existing in the middle of different values.

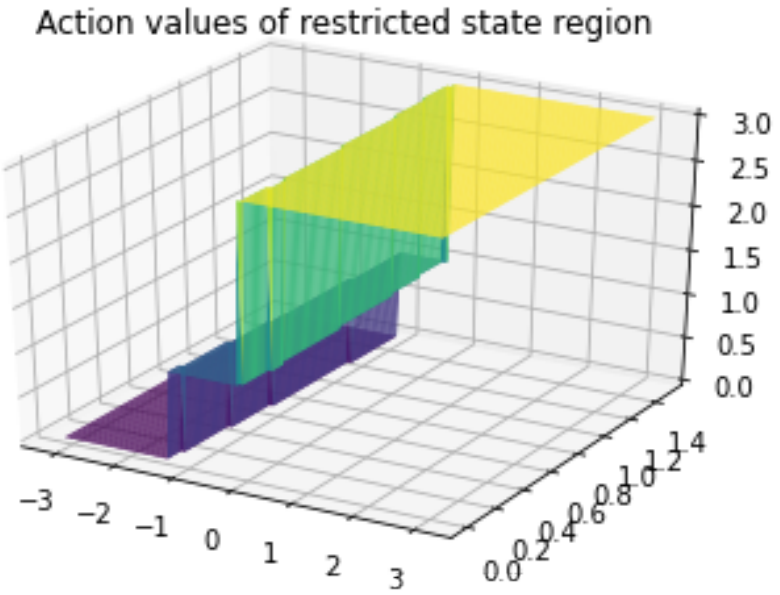

Figure 7: Action values of restricted state region.

(g)

The episode rewards of trained agent and random agent in 50 episodes are shown in Fig. 8

Figure 8: Episode reward of trained agent and random agent in 50 episodes.

(h)

The policy can pass the test successfully as shown in Fig. 9.

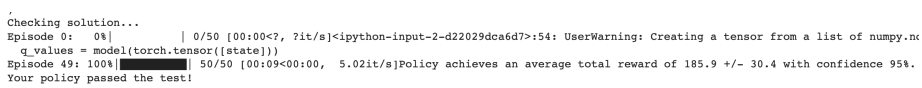

Figure 9: Verification result of the policy.

2 Problem 2: Deep Deterministic Policy Gradient (DDPG)

(b)

(1) We do not use the critic's target network when updating the actor network. Because we replicate the actor-critic network as target network to separate the exploration and the approximation of target. Therefore, in training procedure, the actor network and the target network should be independent, which means the output of target network ought to not affect the updating of actor network. Also, there's lag in the update of parameters of target network, so this could make the training more stable.

(2) DDPG is an off-policy learning algorithm. Sample complexity is an issue for off-policy methods. For DDPG, the update of the behavior and the target policies are separate: the behavior policy is used for exploration, while the target policy is used for function approximation. Also, we sample experience from the memory buffer so the transitions for update can be from previous policy not the current one. Therefore, it has a higher sample complexity. Off-policy learning tries to find a local optimal solution without policy behavior, thus it requires more time.

(d)

- Layout of the network: We use a 3-layer network with 400 neurons in the first layer, and 200 neurons in the second layer. The differences between the actor and critic networks are the dimension of their input and tanh activation is implemented in actor network to clip the action value between -1 and 1.
- Choice of the optimizer: We use the Adam optimizer because it can avoid the network getting stuck in the local minima. In addition, it can adjust the learning rate automatically.
- Parameters: By trail and error, we found that with ($\gamma = 0.99, L = 30000, T_E = 300, N = 64, d = 2, \tau = 0.001, lr_{actor} = 5e-5, lr_{critic} = 5e-4$), the network could converge successfully in this condition.
- The critic should have a larger learning rate than the actor. Otherwise, the Q value obtained from the past policies cannot represent the true value.

(e)

(1) The total episodic reward and the total number of steps taken per episode during training are shown in Fig. 10.

We can see that the training process of DDPG is more stable than that of DQN.

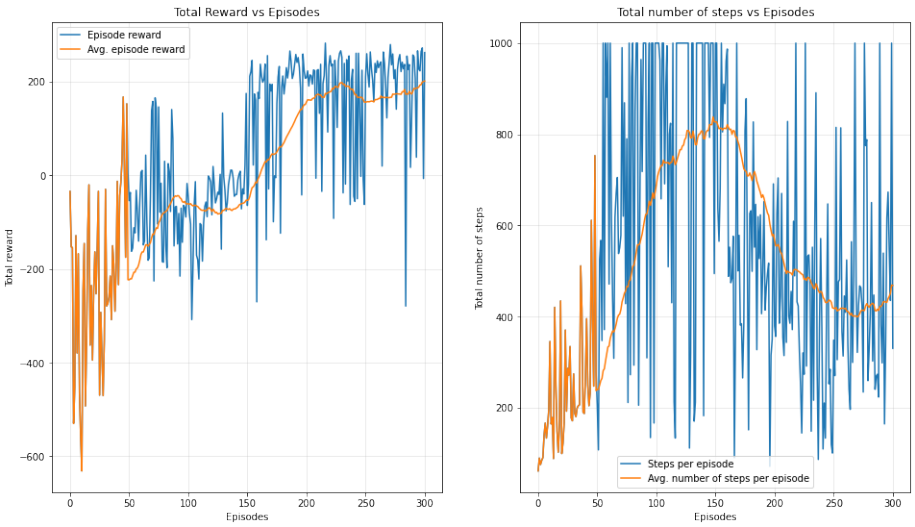

Figure 10: Total reward and total number of steps vs. episode, $\gamma_0 = 0.99$.

(2) The plots for $\gamma_1 = 1$ and $\gamma_2 = 0.5$ are shown in Fig. 11 an Fig. 12.

We can see that for neither $\gamma_1 = 1$ nor $\gamma_2 = 0.5$, the agent can learn the right policy within 300 episodes, where for $\gamma_0 = 0.99$, the agent could reach the average score of 200 successfully. It may take a lot more episodes for them to learn the correct policy. Therefore, the choice of γ is of vital importance. Only with appropriate γ, the agent could successfully learn the policy fast.

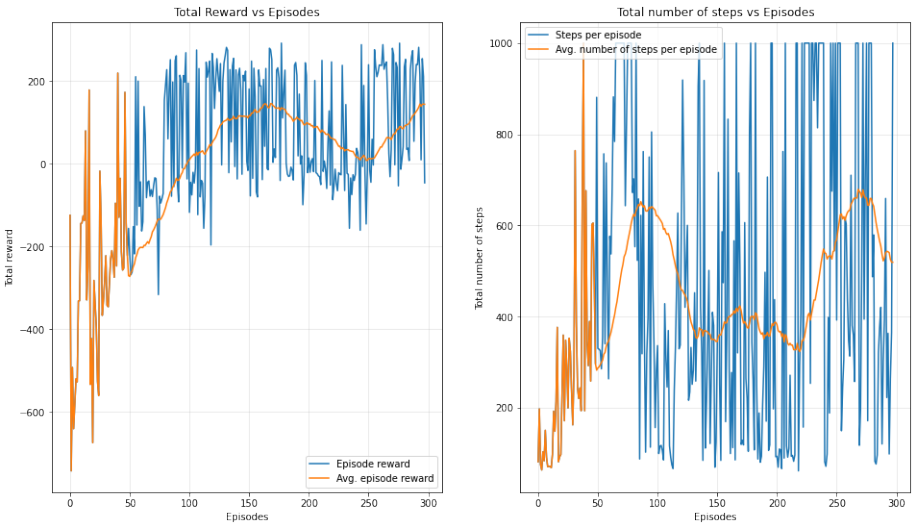

Figure 11: Total reward and total number of steps vs. episode, $\gamma_1 = 1$.

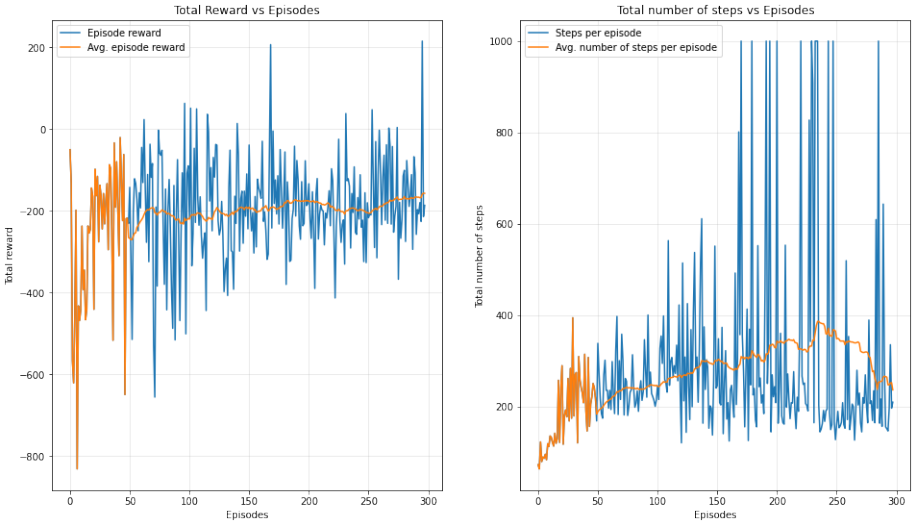

Figure 12: Total reward and total number of steps vs. episode, $\gamma_2 = 0.5$.

(3) For $\gamma_0 = 0.99$, the plots for memory sizes of 10000 and 50000 are shown in Fig. 13 and Fig. 14.

We can see that with a smaller memory size of 10000, the agent could achieve the average score of 200, i.e., learn the correct policy successfully. While for the larger memory size of 50000, the agent cannot reach the average score of 200 within 300 episodes. It may require more episodes to learn the right policy. For smaller memory size, the training is more stable.

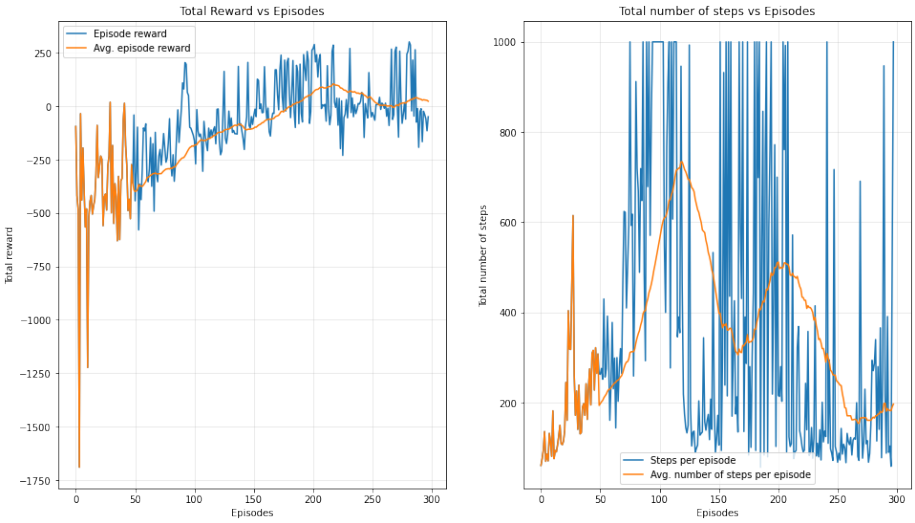

Figure 13: Total reward and total number of steps vs. episode, $L = 10000$.

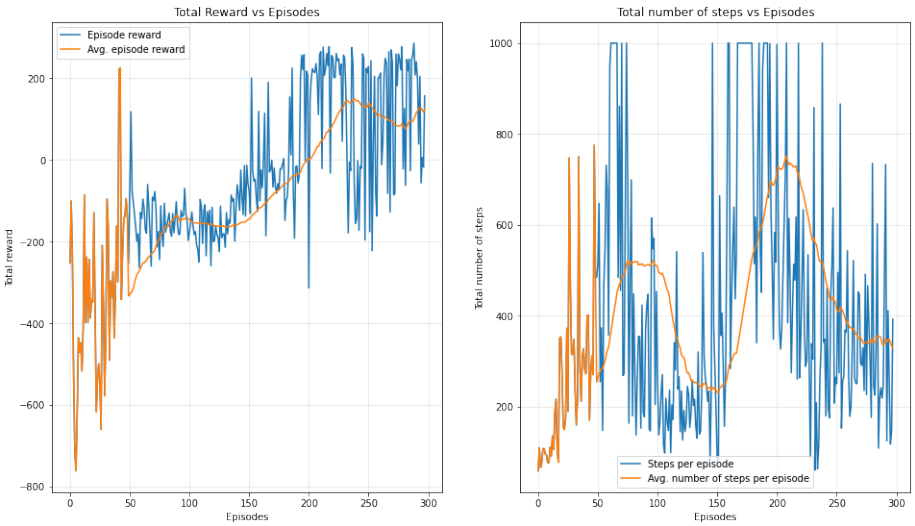

Figure 14: Total reward and total number of steps vs. episode, $L = 50000$.

(f)

(1) For $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$, $Q_\omega(s(y,\omega), \pi_\theta(s(y,\omega)))$ is shown in Fig. 15.

We can see that the Q values are still symmetric about the angle and the minima occur when ω is 0, which means that the the left side has the same influence as the right side. The symmetricity of DDPG is better than that of DQN.

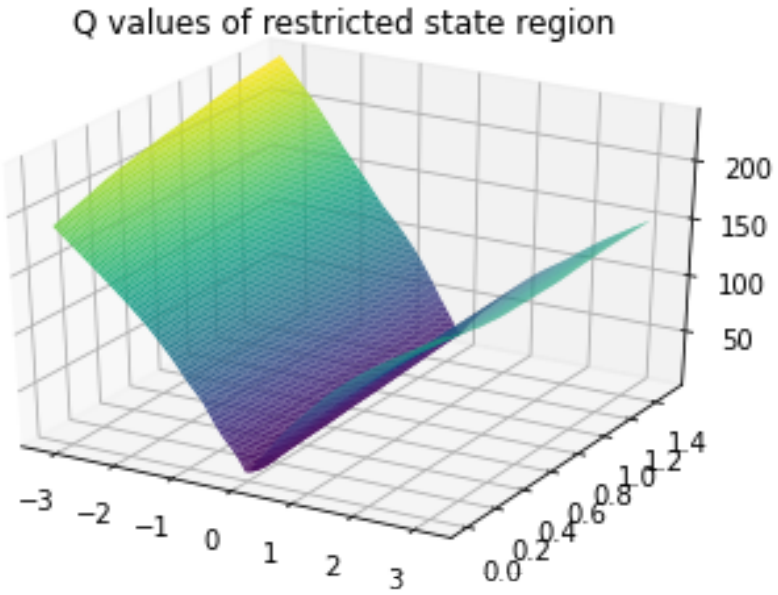

Figure 15: Q values of restricted state region.

(2) For $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$, $\pi_\theta(s(y,\omega))$ is shown in Fig. 16.

We can see that the engine values are separated into two sub-regions by $\omega = 0$. In each region, the engine values are constant. There is a transition

plane between where the engine values fluctuate within -1 to 1.

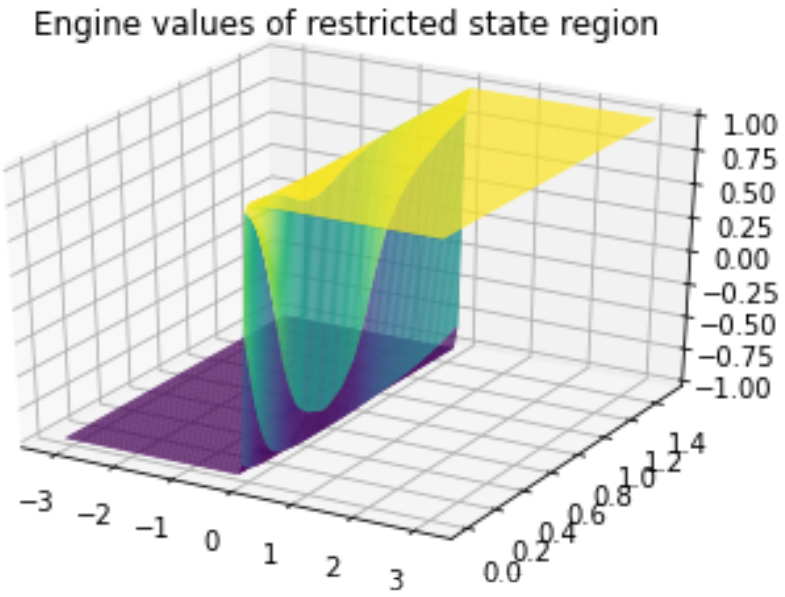

Figure 16: Action values of restricted state region.

(g)

The episode rewards of trained agent and random agent in 50 episodes are shown in Fig. 17

Figure 17: Episode reward of trained agent and random agent in 50 episodes.

(h)

The policy can pass the test successfully as shown in Fig. 18.

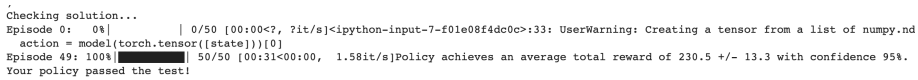

Figure 18: Verification result of the policy.

3 Problem 3: Proximal Policy Optimization (PPO)

(b)

(1) We need target network to learn the value when we use Bellman optimality equation. But in PPO we implement the A2C scheme and the target and action are directly yielded by critic and actor networks. Therefore there's no longer the problem of moving target thus no need of target networks.

(2) Yes PPO is an on-policy method. It updates the networks immediately and only based on the experience in the current episode and clear the buffer at the end of each episode. No replay buffer and no sampling so all transitions gathered for update can only be obtained in current policy.

(d)

- Layout of the network: For the critic network, we use a 3-layer network with 400 neurons in the first layer followed by ReLU activation, and 200 neurons in the second layer which is also followed by ReLU activation. For the actor network, we use an input layer with 400 neurons followed by ReLU activation, whose output will be then fed into two heads, one for μ and one for σ^2. For both heads, we use a 2-layer network with 200 neurons in the first layer followed by ReLU activation. Especially, we use tanh activation for μ output, and sigmoid activation for σ^2 output.

- Choice of the optimizer: We use the Adam optimizer because it can avoid the network getting stuck in the local minima. In addition, it can adjust the learning rate automatically.

- Parameters: By trail and error, we found that with $(\gamma = 0.99, T_E = 1600, lr_{actor} = 1e-5, lr_{critic} = 1e-3, M = 10, \epsilon = 0.2)$, the network could converge successfully in this condition.

- If the actor is updated less frequently, the learning process would be more stable. Because the less frequent update of the actor network could reduce the errors in the critic network, which means that the value estimate used for the update of the actor network would have a smaller variance. In this way, both the actor network and the critic network would be more stable. Thus, the errors accumulated would be smaller, which would lead to a more stable training process.

(e)

(1) The total episodic reward and the total number of steps taken per episode during training are shown in Fig. 19.

We can see that the training process of PPO is much more stable than that of DDPG and DQN. What's more, the number of steps taken in each episode is increasing and then suddenly drop to about 400 when the agent learn to solve the problem with more then 200 points, which is different from DDPG and DQN.

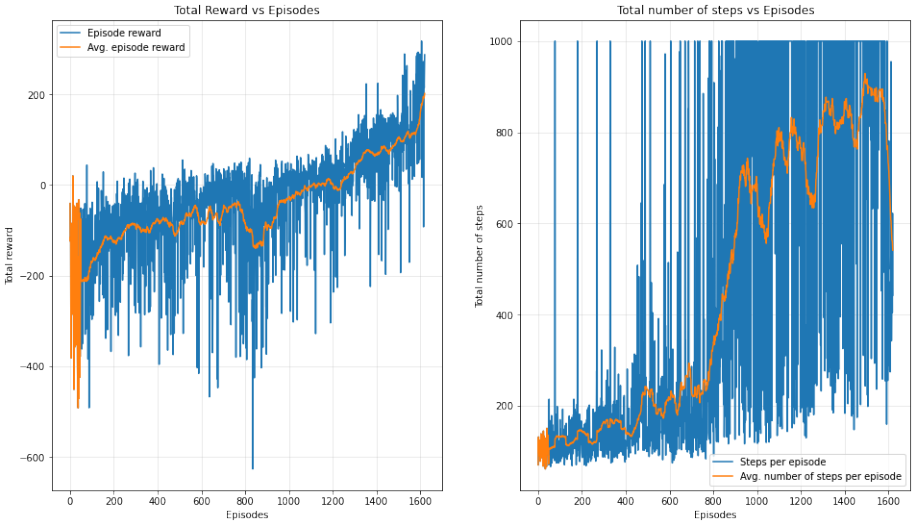

Figure 19: Total reward and total number of steps vs. episode, $\gamma_0 = 0.99$.

(2) The plots for $\gamma_1 = 1$ and $\gamma_2 = 0.5$ are shown in Fig. 20 an Fig. 21.

We can see that for neither $\gamma_1 = 1$ nor $\gamma_2 = 0.5$, the agent can learn the right policy within 1600 episodes, where for $\gamma_0 = 0.99$, the agent could reach the average score of 200 successfully. After a few more episodes, it's likely for them to learn the correct policy given the increasing episode reward curve. Therefore, the choice of γ is of vital importance. Only with appropriate γ, the agent could successfully learn the policy with fewer training time. For $\gamma_2 = 0.5$, the reward only increase a bit within 1600 episodes and the agent could hardly learn the correct policy even though learning for a much longer time.

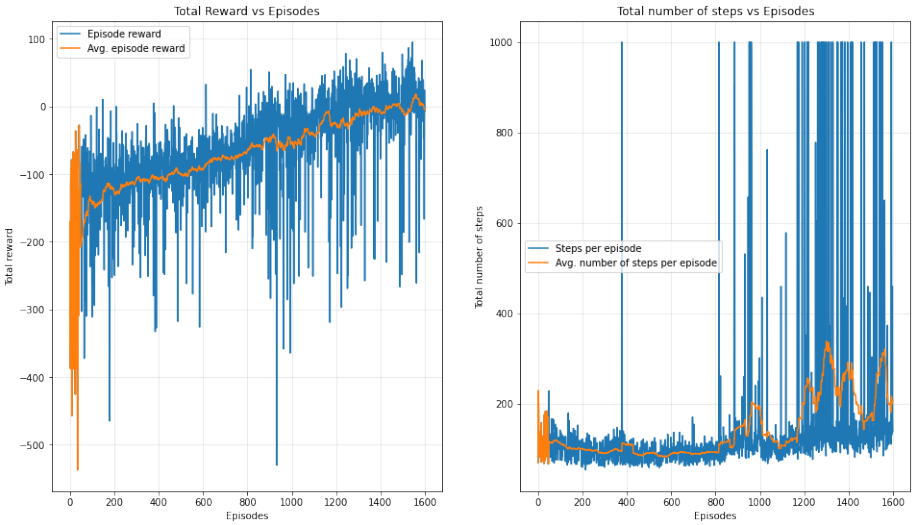

Figure 20: Total reward and total number of steps vs. episode, $\gamma_1 = 1$.

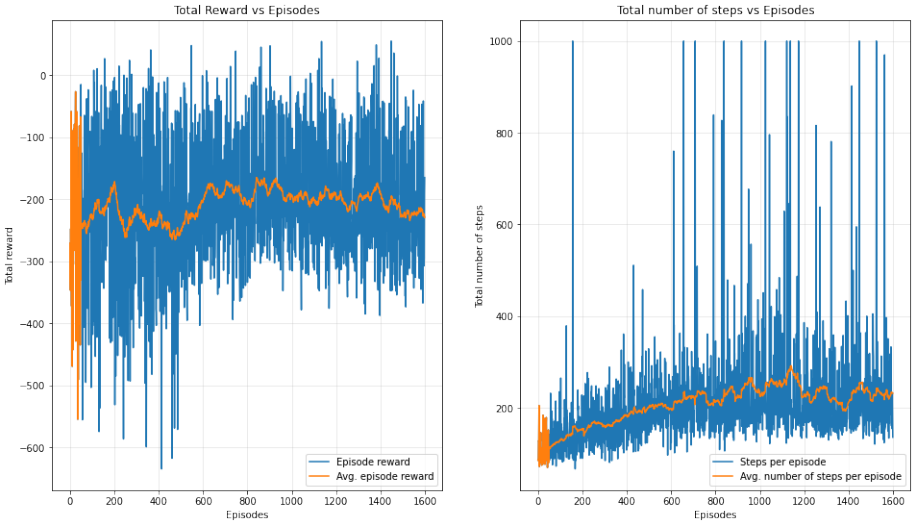

Figure 21: Total reward and total number of steps vs. episode, $\gamma_2 = 0.5$.

(3) For $\gamma_0 = 0.99$, the plots for ϵ of 0.1 and 0.5 are shown in Fig. 22 and Fig. 23.

We can see that for neither $\epsilon = 0.1$ nor $\epsilon = 0.5$, the training process can achieve convergence within 1600 episodes as it does for $\epsilon = 0.2$. However, the rewards for $\epsilon = 0.1$ and $\epsilon = 0.5$ are rising steadily as the episode increases, which means that longer time is required for the agent to learn the correct policy. For $\epsilon = 0.2$, the training is the most stable with the least variations.

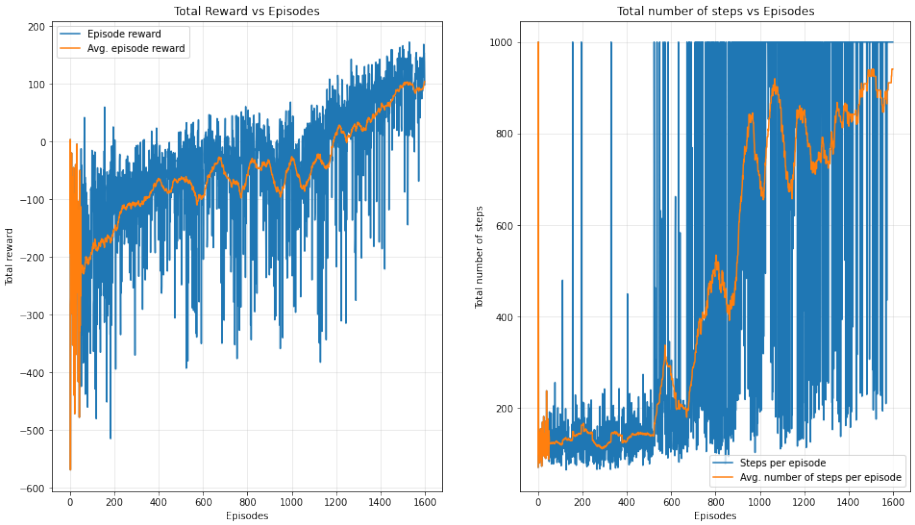

Figure 22: Total reward and total number of steps vs. episode, $\epsilon = 0.1$.

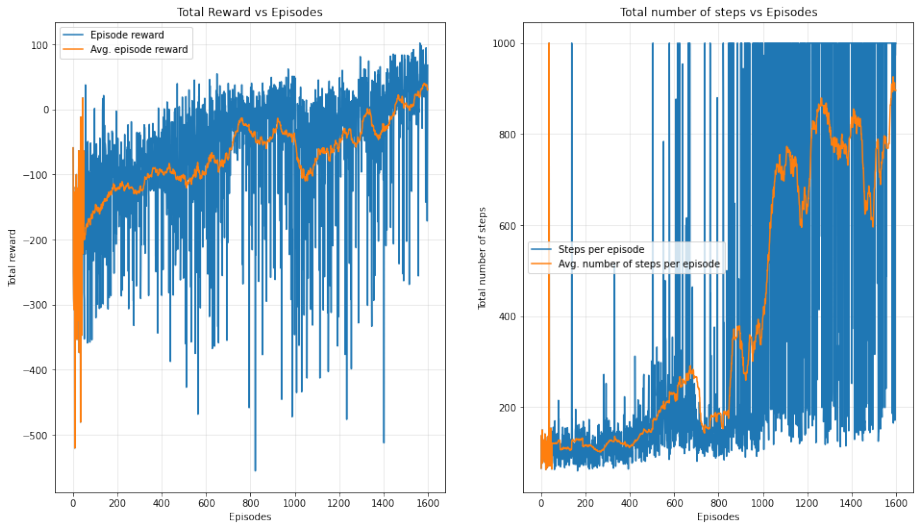

Figure 23: Total reward and total number of steps vs. episode, $\epsilon = 0.5$.

(f)

(1) For $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$, $Q_\omega(s(y, \omega), \pi_\theta(s(y, \omega)))$ is shown in Fig. 24.

We can see that the Q values are perfectly symmetric about the angle and the minima occur when ω is 0, which means that the the left side has the same influence as the right side. The symmetricity of PPO is the best among the three algorithms. Also, the transition of Q values under PPO is the smoothest.

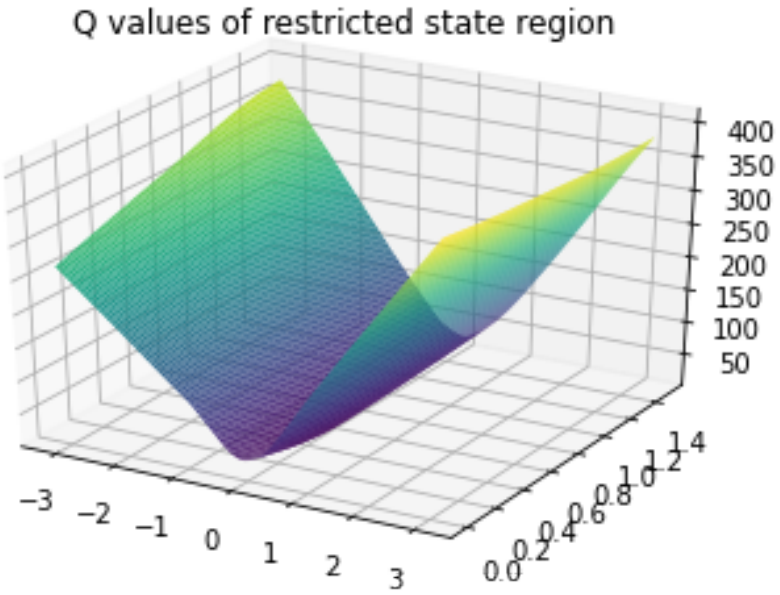

Figure 24: Q values of restricted state region.

(2) For $y \in [0, 1.5]$ and $\omega \in [-\pi, \pi]$, $\pi_\theta(s(y, \omega))$ is shown in Fig. 25.

We can see that the engine values are rising quite smoothly as the angle

increases. So the control of lunar lander is steady and deft, without abrupt alteration.

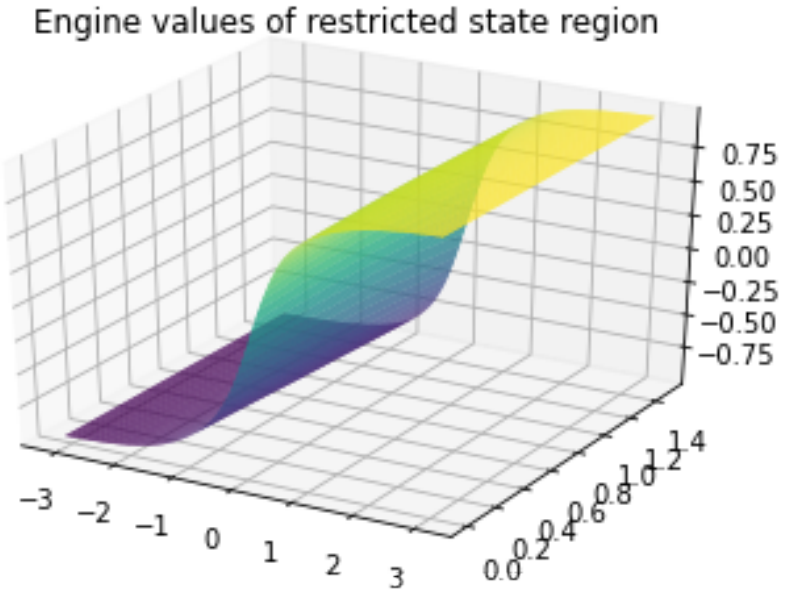

Figure 25: Action values of restricted state region.

(g)

The episode rewards of trained agent and random agent in 50 episodes are shown in Fig. 26

Figure 26: Episode reward of trained agent and random agent in 50 episodes.

(h)

The policy can pass the test successfully as shown in Fig. 27.

```
,
Checking solution...
Episode 0:   0%|          | 0/50 [00:00<?, ?it/s]<ipython-input-3-ad47f2b6d26c>:33: UserWarning: Creating a tensor from a list of numpy.nda
  mu, var = model(torch.tensor([state]))
Episode 49: 100%|██████████| 50/50 [00:33<00:00,  1.49it/s]Policy achieves an average total reward of 190.4 +/- 31.7 with confidence 95%.
Your policy passed the test!
```

Figure 27: Verification result of the policy.