Lab 2

Yuqi Shao 199507014208

December 2021

Problem 1

(b) Why do we use a replay buffer and target network in DQN?
The need for a reply buffer is due to that successive updates are strongly correlated (following a particular trajectory), which affects the convergence rate. The "dataset" for training DQN is the replay buffer.
The target network is used for stabilizing learning. Without a target network, the Q-learning targets would keep changing, introducing high instabilities. However, resent research has succeed in removing the need for a target network by introducing a mellowmax operator [1] .

(c) See code.

(d)

- Layout of network:
  2 hidden layers with 64 neurons on each. Each of the 2 layer has a ReLU as activation function.
- Optimizer: Adam
- Other parameters:
  $\gamma = 0.99$
  $L = 30000$
  $T_E = 500$
  $N = 128$
  $C = L/N$
  ϵ: linear decay with $\epsilon_{min} = 0.05, \epsilon_{max} = 1$, $Z = 90\%$ of the total number of episodes
  learning rate: 0.0008
- 2-norm clipping of the networks' gradients to 1
- Modifications: Combined experience replay

Due to the problem setting, time taken is not a too important issue for getting a high total reward, thus a discount factor of 0.99 (really close to 1) is chosen. Since there are many possible trajectories, comparably large L and N are chosen. Linear decay of ϵ is chosen, because the environment is complex and it would be good to explore for a bit more. Combined experience replay is used to prioritize the newest experience, since we have a large reply memory.

(e)

(1) During the training process, the episodic reward increases steadily. The total number of steps increases then start decreasing at around 300-400 episodes. It is reasonable since the agent first learns to control the flying and landing well, then it learns to lower irrelevant actions since there are costs associated with firing the engines.

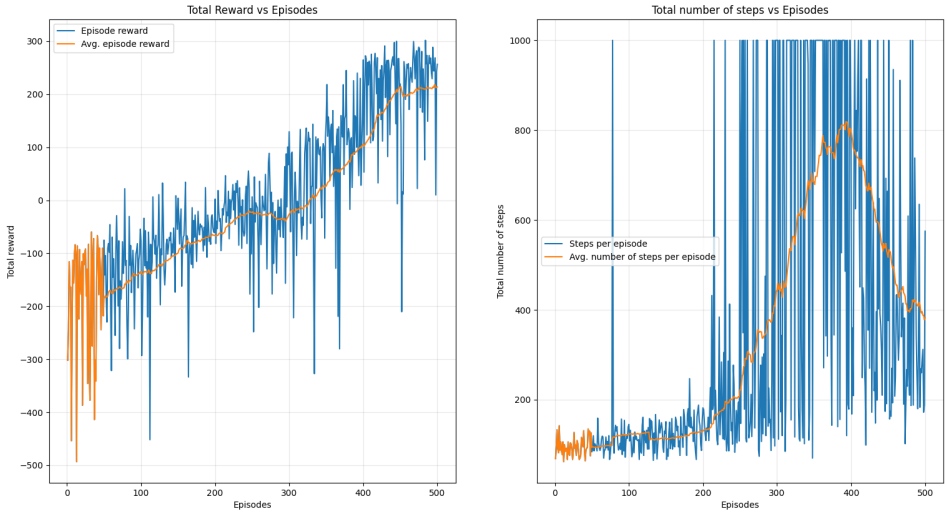

Figure 1: Total episodic reward and the total number of steps taken per episode during training

(2) Plots generated for discount factors $\gamma_1 = 1$ and $\gamma_2 = 0.5$.

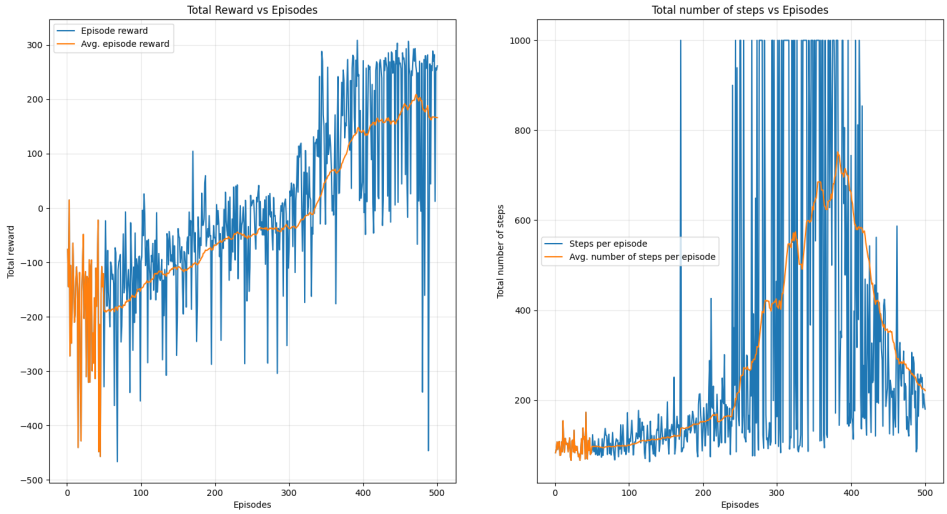

Figure 2: Total episodic reward and the total number of steps taken per episode during training $\gamma_1 = 1$

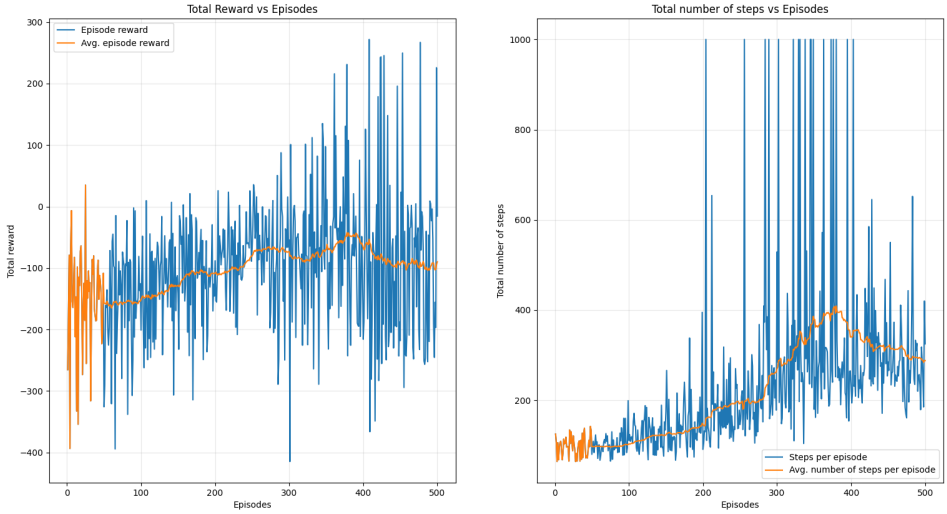

Figure 3: Total episodic reward and the total number of steps taken per episode during training $\gamma_2 = 0.5$

The performance of $\gamma_1 = 1$ is similar comparing to $\gamma_0 = 0.99$, whereas $\gamma_1 = 1$ converged to a

slightly lower value by chance. Around the end of training, the variance of total reward is higher for $\gamma_1 = 1$ comparing to $\gamma_0 = 0.99$, which indicates that it is more stable to have a slight discount on reward for better learning on reducing irrelevant actions. The training for $\gamma_2 = 0.5$ did not converge to a desired value. This is due to that the rewards are decreased significantly in time which is not a correct representation of the engineering problem. It changed the learning focus of the agent.

(3) Decreasing the number of episodes:

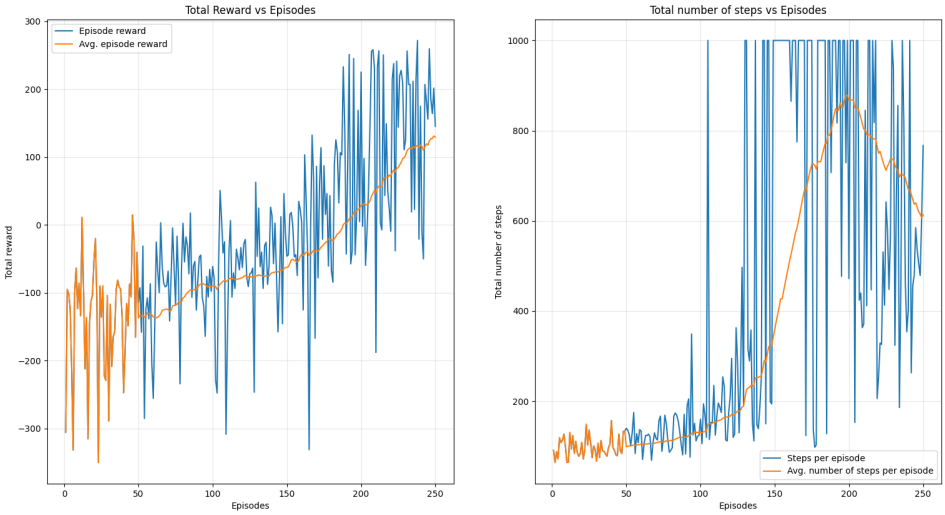

Figure 4: Total episodic reward and the total number of steps taken per episode during training $N_{episode} = 250$

The number of episode is decreased to 250 and we are able to reach slightly lower average episodic reward at the end.

Decreasing the memory size:

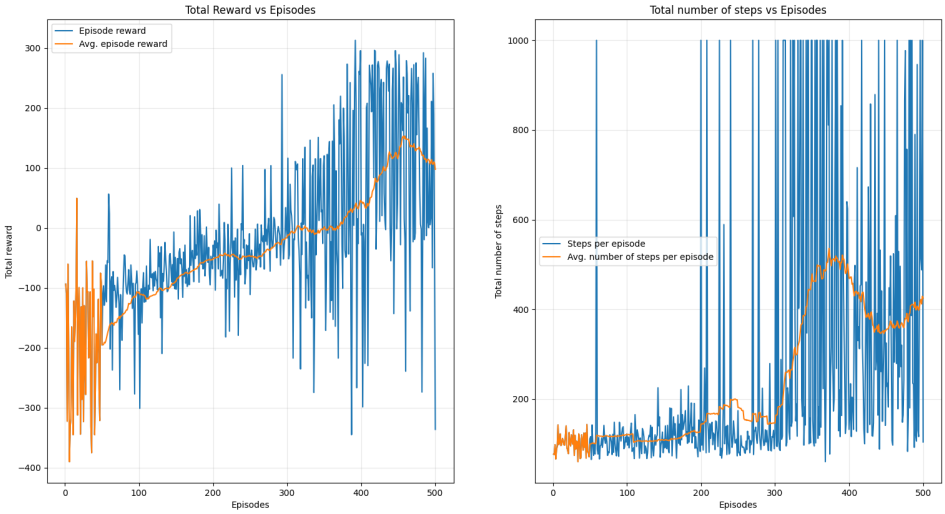

Figure 5: Total episodic reward and the total number of steps taken per episode during training $L = 5000$

When the memory size is decreased, lower average total reward and higher variance are observed at around the end of training.

(f)
(1) 3D plot for $max_a Q$:

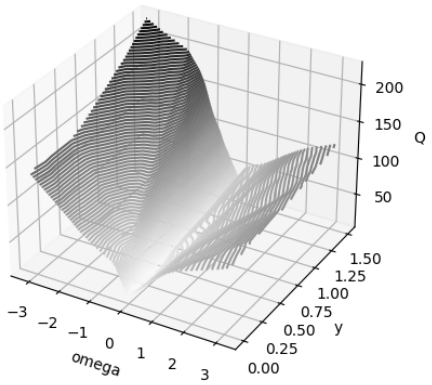

Figure 6: $max_a Q$

The plot shows a lower value at around $\omega = 0$ for any y. It means it is more desirable to be in the state that the spaceship is at an inclined angle and the corresponding action is firing the side engines. This coincided with the fact that firing side engines have lower cost comparing to firing main engines.

(2) 3D plot for $argmax_a Q$:

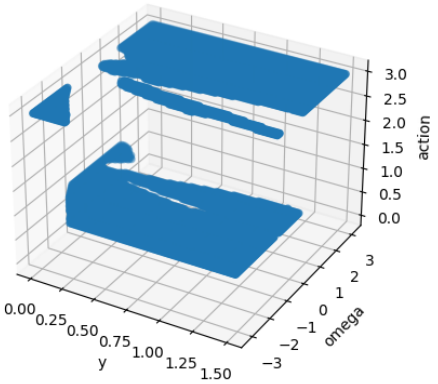

Figure 7: $argmax_a Q$

The plot is reasonable, since when the space ship is tilted, the best actions are firing corresponding left or right engines to adjust the spaceship towards horizontal.

(g)
Total episodic reward over 50 episodes of random agent vs Q-network (my agent):

Figure 8: Total episodic reward over 50 episodes

The random agent could rarely hit the positive rewards whereas the trained Q-network could achieve quite good rewards, which stays above 200 for most of the time. It means the training was successful.

(h) Find 'neural-network-1.pth' in the folder.

Problem 2

(b)
(1) Why don't we use the critic's target network when updating the actor network? Would it make a difference?
Because when updating the actor network, it should include the newest learned information. If critic's target network is used, θ is not in alignment with critic's network parameter but in alignment with critic's target network parameter, which leads to a mess.

(2) Is DDPG off-policy? Is sample complexity an issue for off-policy methods (compared to on-policy ones)?
DDPG is off-policy since the policy used to generate behaviour is unrelated to the policy that is evaluated and improved. Off-policy algorithms have low sample complexity, but are difficult to tune.

(c) See code.

(d)

(1) Explain the layout of the network that you used; the choice of the optimizer; the parameters that you used. Motivate why you made those choices.
Parameters implemented according to suggested.

- Layout of network:
  Critic: $state \rightarrow 400 \rightarrow ReLU \rightarrow 200 + actions \rightarrow ReLU \rightarrow output$
  Actor: $state \rightarrow 400 \rightarrow ReLU \rightarrow 200 \rightarrow ReLU \rightarrow output \rightarrow Tanh$
- Optimizer: Adam
- Other parameters:
  $\gamma = 0.99$
  $L = 30000$
  $T_E = 300$
  $\tau = 10^{-3}$
  $N = 64$
  $d = 2$
  $\mu = 0.15$
  $\sigma = 0.2$
  actor learning rate: $5 \cdot 10^{-5}$
  critic learning rate: $5 \cdot 10^{-4}$
- 2-norm clipping of the networks' gradients to 1

Large sizes of actor and critic networks provide more flexibility for accurate approximations. A close to 1 discount factor is the best for the same reason as problem 1. Large buffer size and medium batch size is chosen due to the high complexity of environment. The soft-update parameter τ is very small, improving the stability in updates of target networks and allowing a small number of d to be used for more frequent updates.

(2) In general, do you think it is better to have a larger learning rate for the critic or the actor? Explain why.
It is better to have a larger learning rate for the critic. Because the critic is evaluating actor, it is as if I fix policy π_θ for a long time to get a good critic.

(e)
(1)

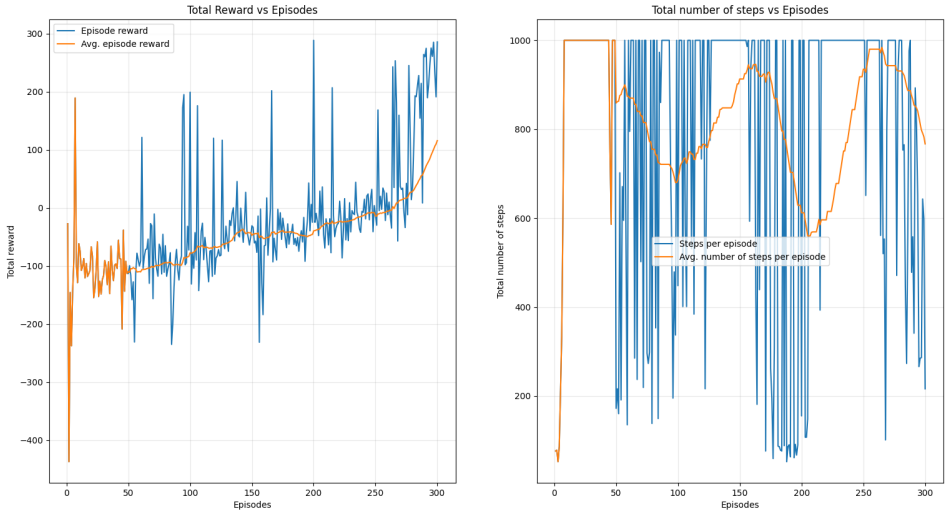

Figure 9: Total episodic reward and the total number of steps taken per episode during training

The training process shows a steady increase of total reward in the first 250 episodes, and a slightly steeper increase of total reward in the last 50 episodes. The last jump in the total reward corresponds to the decrease in steps taken in the figure on the right. The decrease in steps taken after the spaceship learned flying and landing well is important, which lowers the cost for firing engines.

(2) Plots generated for discount factors $\gamma_1 = 1$ and $\gamma_2 = 0.5$.

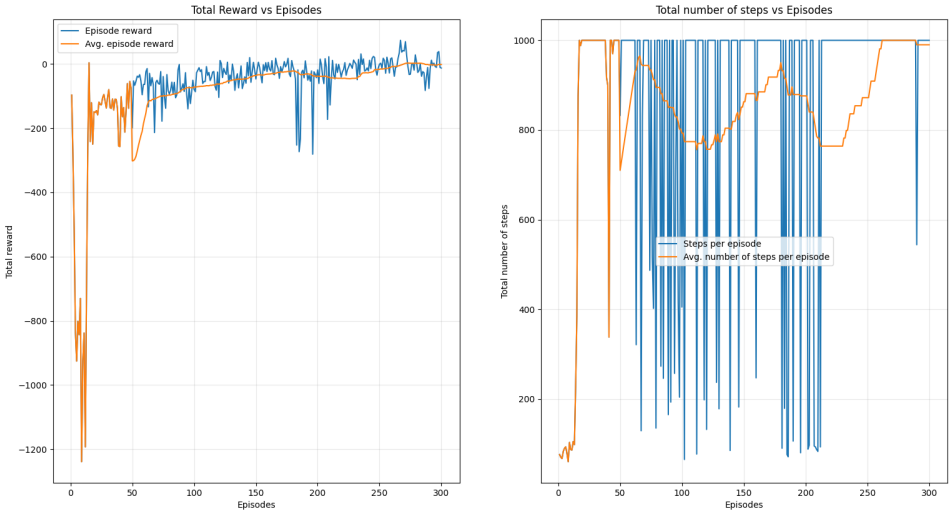

Figure 10: Total episodic reward and the total number of steps taken per episode during training $\gamma_1 = 1$

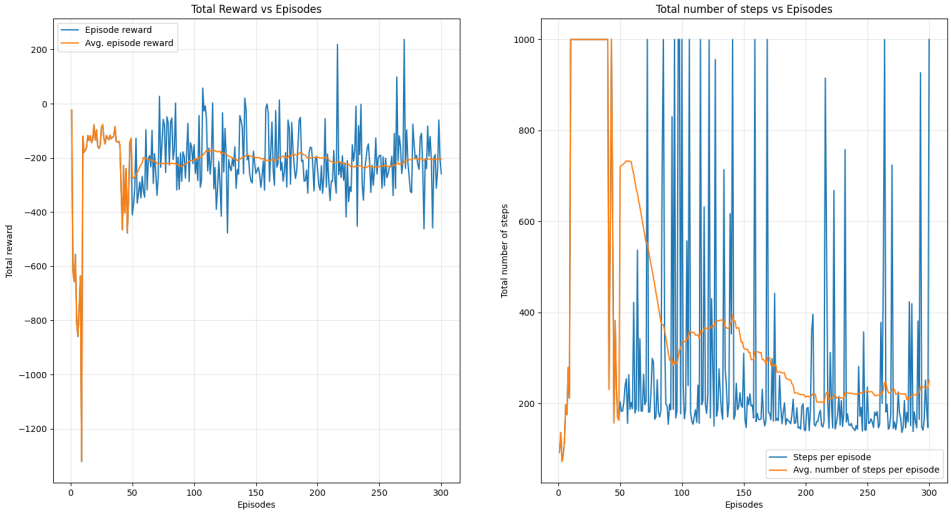

Figure 11: Total episodic reward and the total number of steps taken per episode during training $\gamma_2 = 0.5$

The reward of $\gamma_1 = 1$ converged to only around 0 at the end of training. From the plot on the

right you could see that it has trouble lowering number of steps with this setting. This is probably due to that no discount on the landing reward makes the effect of lowering firing actions not so significant. Again, the training for $\gamma_2 = 0.5$ did not converge to a desired value and the reason is same as what has been illustrated in problem 1. It is not a correct representation of system and changed the learning focus of agent.

(3) See figures below for experiments with memory size $L = 5000$, $L = 100000$ and $L = 200000$. Decreasing the memory size:

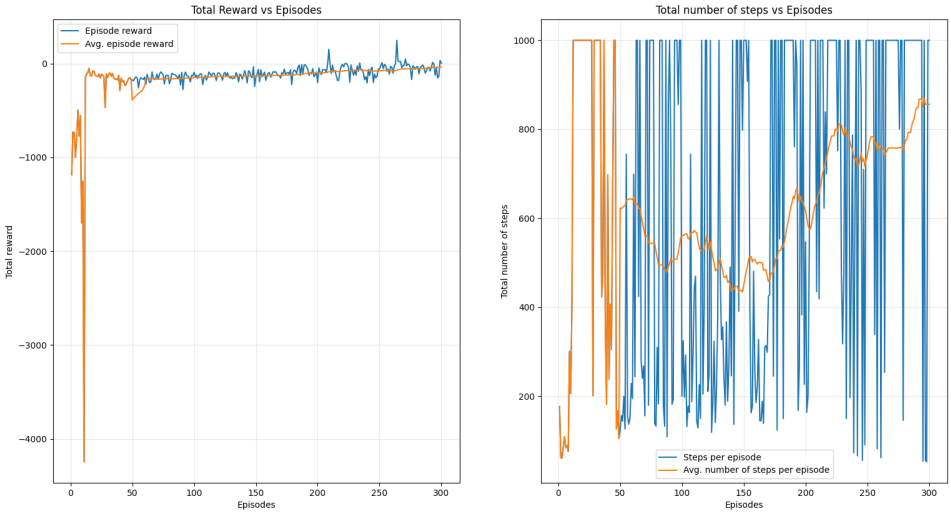

Figure 12: Total episodic reward and the total number of steps taken per episode during training $L = 5000$

Increasing the memory size:

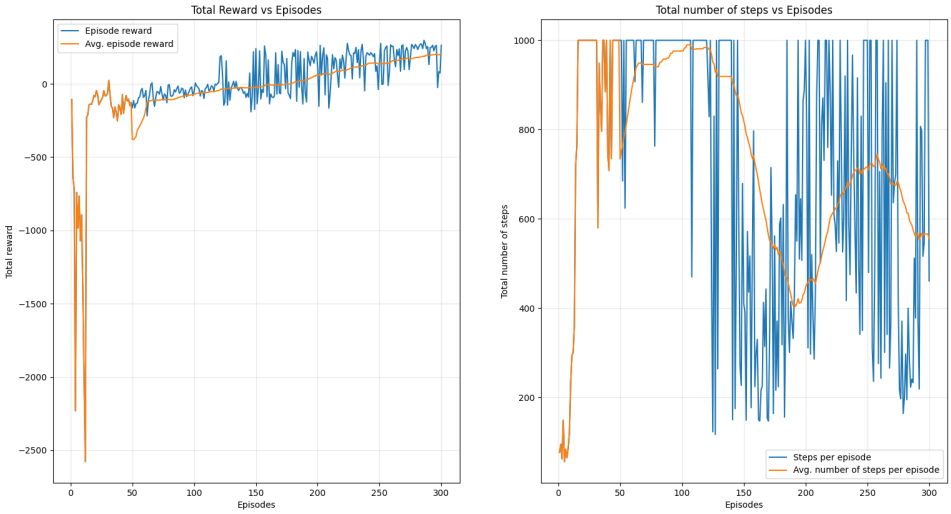

Figure 13: Total episodic reward and the total number of steps taken per episode during training $L = 100000$

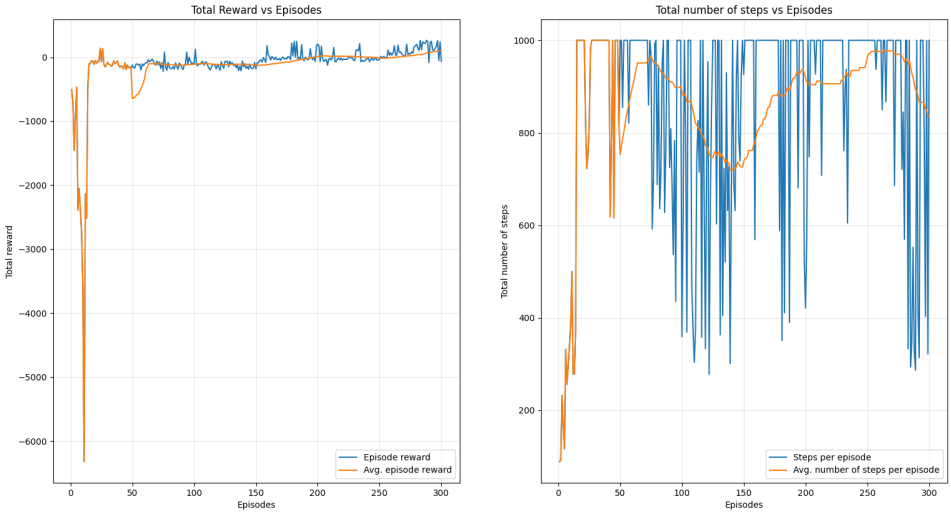

Figure 14: Total episodic reward and the total number of steps taken per episode during training $L = 200000$

When a memory size of 5000 is used, it is not converging to the desired value. When a memory size of 100000 is used, the average episode reward reached around 200 at the end and the training

process looks more stable. A memory size of 200000 does not show convergence to the desired value. In theory, the memory size should not be too large or too small. In this large state-space system we need an extremely large buffer size (200000) to encounter the problem of the buffer size to be too large such that the sampled experiences are too old for training. Larger buffer also takes larger computation time, so it is preferable to have a medium buffer size which is just enough for accurate training.

(f) (1) 3D plot for Q_ω:

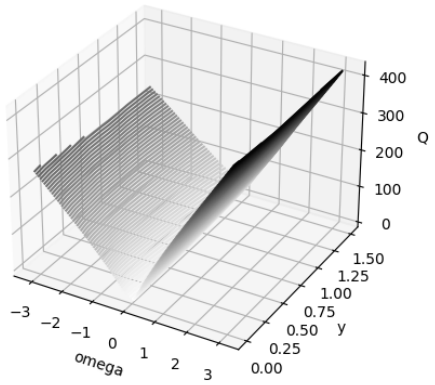

Figure 15: Q_ω

Same as problem 1, the plot shows a lower value at around $\omega = 0$ for any y. It means it is more desirable to be in the state that the spaceship is at an inclined angle and the corresponding action is firing the side engines. This coincided with the fact that firing side engines have lower cost comparing to firing main engines.

(2) 3D plot for Engine direction:

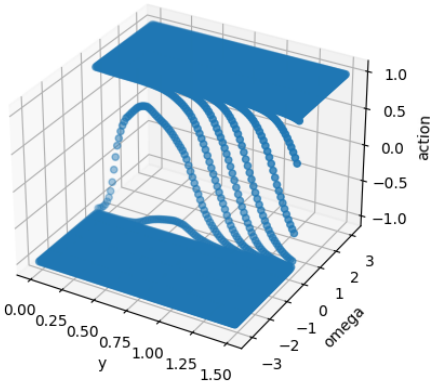

Figure 16: Engine direction

Same as problem 1, the plot is reasonable, since when the space ship is tilted, the best actions are firing corresponding left or right engines to adjust the spaceship towards horizontal.

(g)
Total episodic reward over 50 episodes of random agent vs actor-network (my agent):

Figure 17: Total episodic reward over 50 episodes

The random agent could rarely hit the positive rewards whereas the trained actor-network could achieve quite good rewards, which stays above 200 for most of the time. It means the training was successful.

(h) Find 'neural-network-2-actor.pth' and 'neural-network-2-critic.pth' in the folder.

Problem 3

(b)
(1) Why don't we use target networks in PPO?
Because the aim for target networks is to stablize training, and in PPO the stablization is already achieved by training everything after an episode finished and constraining the policy update. This allows to perform several epochs of training in each episode.

(2) Is PPO an on-policy method? If yes, explain why. Furthermore, is sample complexity an issue for on-policy methods?
PPO is an on-policy method. The reason is that $\pi_{\theta_{old}}$ is the old policy of the same actor network, which is considered in the update of θ parameters. The sample complexity of on-policy methods is higher than off-policy methods.

(c) See code.

(d)
(1) Explain the layout of the network that you used; the choice of the optimizer; the parameters that you used. Motivate why you made those choices.
Parameters implemented according to suggested.

- Layout of network:
  Critic: $state \rightarrow 400 \rightarrow ReLU \rightarrow 200 \rightarrow ReLU \rightarrow output$
  Actor:

$$state \rightarrow 400 \rightarrow ReLU \rightarrow \begin{cases} \text{Head 1:} & \rightarrow 200 \rightarrow ReLU \rightarrow output \rightarrow Tanh \\ \text{Head 2:} & \rightarrow 200 \rightarrow ReLU \rightarrow output \rightarrow Sigmoid \end{cases}$$

- Optimizer: Adam
- Other parameters:
  $\gamma = 0.99$
  $T_E = 1600$
  $\epsilon = 0.2$
  $M = 10$
  actor learning rate: $1 \cdot 10^{-5}$
  critic learning rate: $1 \cdot 10^{-3}$

(2) Do you think that updating the actor less frequently (compared to the critic) would result in a better training process (i.e., more stable)?
For PPO algorithm it is not very meaningful to make the actor update less frequently. The stability is already controlled by clipping the policy so that the policy will not change much after one single update. If both clipping and lowering update frequency are applied it might lead the networks not trained enough.

(e)
(1)

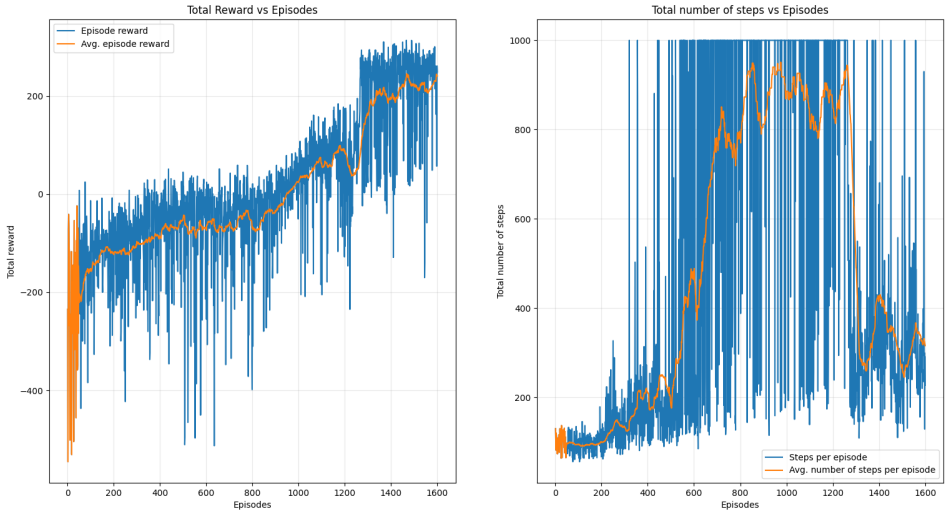

Figure 18: Total episodic reward and the total number of steps taken per episode during training

There is a steep jump of the total reward during episode 1200-1400, which corresponds to the dramatic decrease of steps taken during episode 1200-1400, shown by the figures. The decrease in steps taken after the spaceship learned flying and landing well is important, which lowers the cost for firing engines.

(2) Plots generated for discount factors $\gamma_1 = 1$ and $\gamma_2 = 0.5$.

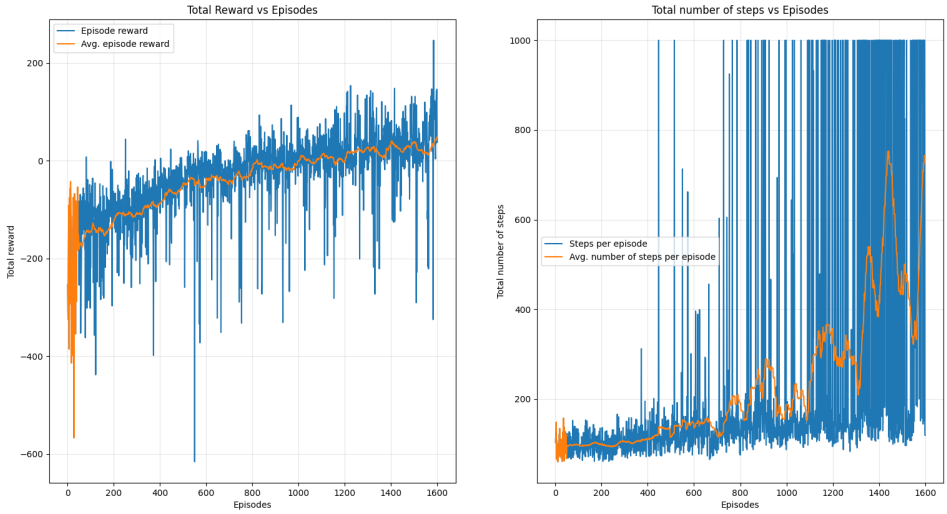

Figure 19: Total episodic reward and the total number of steps taken per episode during training $\gamma_1 = 1$

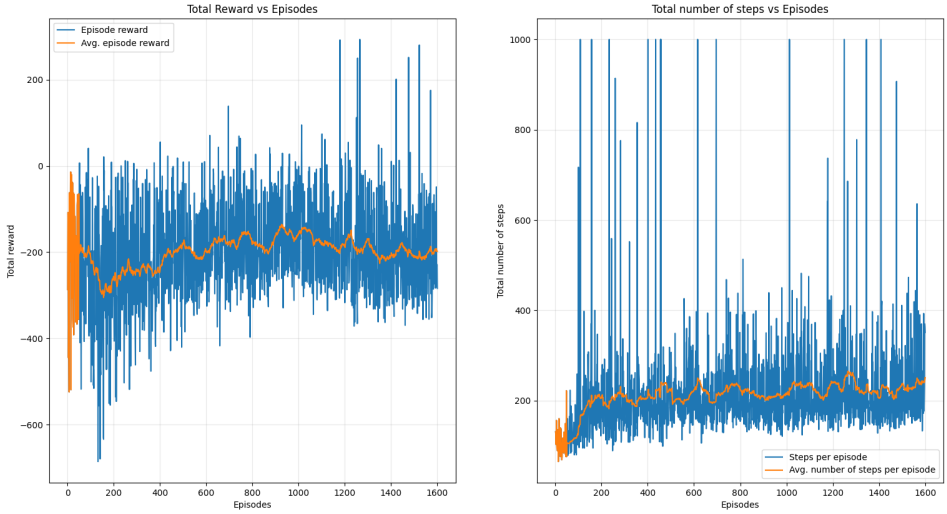

Figure 20: Total episodic reward and the total number of steps taken per episode during training $\gamma_2 = 0.5$

Both the reward of $\gamma_1 = 1$ and $\gamma_2 = 0.5$ did not converge to a desired value. Comparing to

problems 1 and 2, it is found that all algorithms performances are quite bad with $\gamma_2 = 0.5$, but DQN is less sensitive to $\gamma_1 = 1$ than DDPG and PPO. Whether this is due to algorithm property or differences between discrete and continuous actions space needs further exploring. Overall, $\gamma_0 = 0.99$ is a good choice for such systems.

(3) Plots generated for $\epsilon = 0.1$ and $\epsilon = 0.3$.

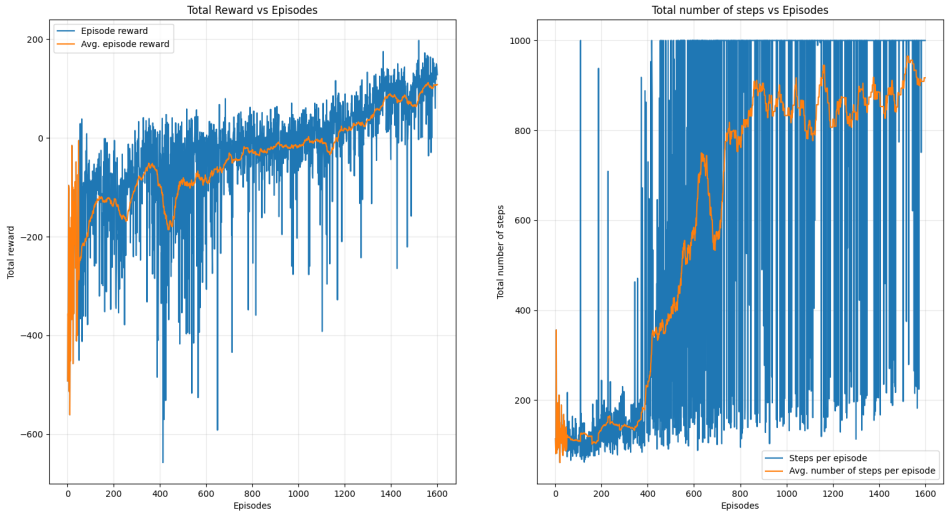

Figure 21: Total episodic reward and the total number of steps taken per episode during training $\epsilon = 0.1$

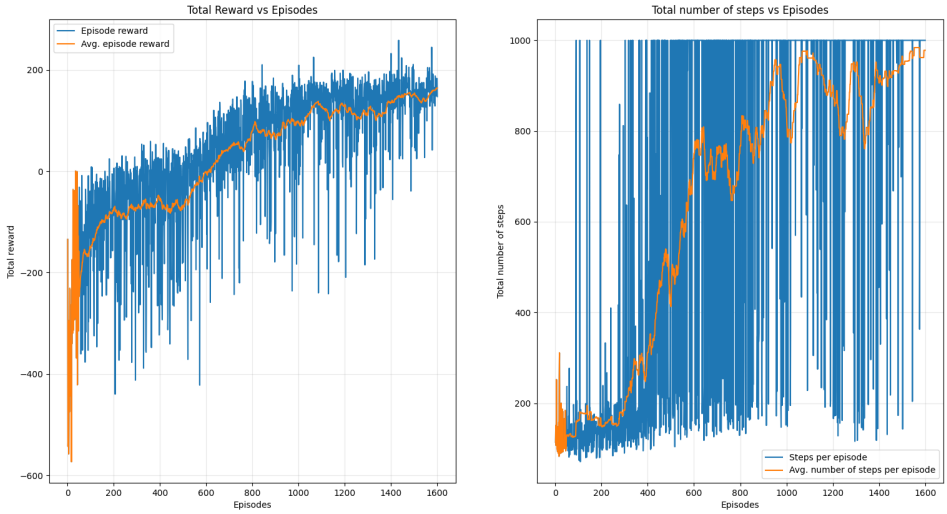

Figure 22: Total episodic reward and the total number of steps taken per episode during training $\epsilon = 0.3$

It is found that both $\epsilon = 0.1$ and $\epsilon = 0.3$ did not reach a total reward as high as $\epsilon = 0.2$ at the end. ϵ can neither be too big nor too small, and $\epsilon = 0.2$ seems to be an optimal value.

(f)
(1) 3D plot for V_ω:

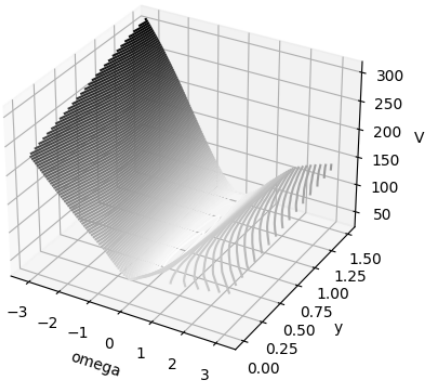

Figure 23: V_ω

Same as problem 1, the plot shows a lower value at around $\omega = 0$ for any y. It means it is more desirable to be in the state that the spaceship is at an inclined angle and the corresponding action is firing the side engines. This coincided with the fact that firing side engines have lower cost comparing to firing main engines.

(2) 3D plot for $\mu_{\theta 2}$:

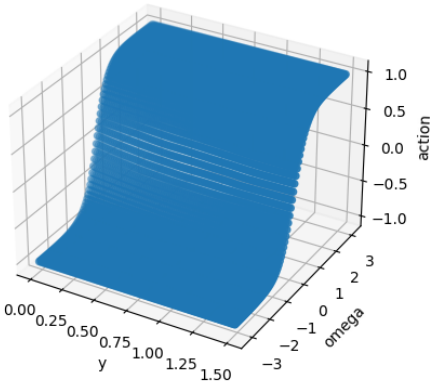

Figure 24: Engine direction Mean

Same as problem 1 & 2, the plot is reasonable, since when the space ship is tilted, the best actions are firing corresponding left or right engines to adjust the spaceship towards horizontal. Since the action is modeled as a distribution, the transitions are smooth.

(g)
Total episodic reward over 50 episodes of random agent vs actor-network (my agent):

Figure 25: Total episodic reward over 50 episodes

The random agent could rarely hit the positive rewards whereas the trained actor-network could achieve quite good rewards, which stays above 200 for most of the time. It means the training was successful.

(h) Find 'neural-network-3-actor.pth' and 'neural-network-3-critic.pth' in the folder.

References

[1] Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: Removing the need for a target network in deep q-learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2733–2739. International Joint Conferences on Artificial Intelligence Organization, 7 2019.