RL Final Project Report

> **312553024 江尚軒**

Methodology Introduction

In the RL Final Project, I employed **TD3** for both the Circle_cw and Austria maps.

The racecar_gym environment is a continuous action space, and my previous experience with TD3 in RL Homework 4 influenced my decision to utilize TD3 for this final project.

Experiment Design and Implementation

This section will outline my experiment design for the Circle_cw and Austria environments separately.

Circle_cw

Reward Function

1. Firstly, I use the original reward, which is the progress of the race car.

```
reward = (info['lap'] + info['progress'] - 1)
```

2. Secondly, I plus time as a reward function.

```
reward = (info['lap'] + info['progress'] - 1) + info['time
```

3. I subtracted the distance from the goal and added the distance to the obstacle.

```
reward = (info['lap'] - 1) + info['progress'] - info["dist
```

4. I subtracted collision penalties as a component of the reward function.

```
reward = (info['lap'] + info['progress'] - 1) - penalty * (
```

5. Finally, if the car collides with the wall, set the reward to 0 and terminate.

```
if info["wall_collision"]:
    terminates = True
    reward = 0
```

Action Space & Noise

1. After the first demo, I observed that the Circle_cw environment primarily requires the race car to turn right. Consequently, I opted to narrow down the action space range from 0 to 1. Furthermore, I modified the action noise to Ornstein-Uhlenbeck (OU) noise.

```
self.action_space = gym.spaces.box.Box(low=0., high=1., sh
self.noise = OUNoiseGenerator(noise_mean, noise_std)
```

2. Action space from 0.5 to 1.

```
self.action_space = gym.spaces.box.Box(low=0.5, high=1., s
self.noise = OUNoiseGenerator(noise_mean, noise_std)
```

3. Action space from 0.5 to 0.75.

```
self.action_space = gym.spaces.box.Box(low=0.5, high=0.75,
self.noise = OUNoiseGenerator(noise_mean, noise_std)
reward = original_reward
```

Resize Observation & Random Start Point

1. Following the tips from the TA, I resized the observation from 128*128 *to* 64*64 and randomized the starting point of the race car.

```
obs = cv2.resize(obs, (64, 64), interpolation=cv2.INTER_AR

if kwargs.get("options"):
    kwargs["options"]["mode"] = "random"
```

```
else:
    kwargs["options"] = {"mode": "random"}
```

2. Resized the observation from 128*128 *to* 64*64 and randomized the starting point of the race car.

```
obs = cv2.resize(obs, (32, 32), interpolation=cv2.INTER_AR

if kwargs.get("options"):
    kwargs["options"]["mode"] = "random"
else:
    kwargs["options"] = {"mode": "random"}
```

Rule-based

Due to the unsatisfactory performance of the aforementioned methods, I have decided to employ a rule-based approach to play the game.

I counted the pixels along the rightmost and leftmost columns representing the wall. The code is provided below:

```
left_wall = 0
right_wall = 0
for i in range(len(obs)):
    if obs[i][0] >= 140 and obs[i][0] <= 200:
        left_wall += 1

    if obs[i][-1] >= 140 and obs[i][-1] <= 200:
        right_wall += 1
```

1. If the left wall is greater than 50% of the observation length, then turn right, and vice versa.

```
action = [1.0, 0.15]

if left_wall >= len(obs)*0.5:
    action = [1.0, 0.3]
```

```
if right_wall >= len(obs)*0.5:
    action = [1.0, 0.0]
```

2. If there are 4 more pixels on the left than on the right, then turn right, and vice versa.

```
if left_wall - right_wall > 4:
    action = [1.0, 0.3]
elif right_wall - left_wall > 4:
    action = [1.0, 0.0]
else:
    action = [1.0, 0.15]
```

RL from Demonstration

1. I employed the rule-based method to determine the action, subsequently calculating the MSE loss between the rule-based action and the actor's action. I then combined this loss with the critic loss and performed model training.

```
action = self.actor_net(state)

rule_action = []
for i in range(state.shape[0]):
    rule_action.append(self.my_rule_2(state[i].unsqueeze(0
rule_action = torch.FloatTensor(np.array(rule_action))

critic_loss = self.critic_net1(state, action).mean()
rule_loss = criterion(action.to(self.device),
                      rule_action.to(self.device))

actor_loss = rule_loss - critic_loss
```

Austria

Rule-based

The left and right walls are the same as the Circle_cw environment.

1. If the count of pixels representing the left wall exceeds 40% but is less than 60% of the observation length, instruct the agent to turn right. If the count surpasses 60%, guide the agent to turn more sharply to the right, and vice versa.

```
action = [0.2, 0.0]

if left_wall >= len(obs)*0.4:
    action = [0.1, 0.75]
elif left_wall >= len(obs)*0.6:
    action = [0.05, 1.0]

if right_wall >= len(obs)*0.4:
    action = [0.1, -0.75]
elif right_wall >= len(obs)*0.6:
    action = [0.05, -1.0]
```

2. If there are 4 more pixels on the left than on the right, then turn right, and vice versa.

```
if left_wall - right_wall > 4:
    action = [1.0, 0.3]
elif right_wall - left_wall > 4:
    action = [1.0, 0.0]
else:
    action = [1.0, 0.15]
```

3. Momentum: To change the action, the same action must be taken multiple times consecutively.

```
if left_wall - right_wall > 4:      # right
    self.right_count += 1
    self.left_count = 0
    self.straight_count = 0
elif right_wall - left_wall > 4:    # left
    self.right_count = 0
    self.left_count += 1
    self.straight_count = 0
else:   # straight
```

```
    self.right_count = 0
    self.left_count = 0
    self.straight_count += 1

if self.right_count >= 1:
    self.action_state = "R"
if self.left_count >= 2:
    self.action_state = "L"
if self.straight_count >= 4:
    self.action_state = "S"

if self.action_state == "R":
    action = [0.01, 1.0]
if self.action_state == "L":
    action = [0.01, -1.0]
if self.action_state == "S":
    action = [0.1, 0.0]
```

RL from Demonstration

1. During the warm-up step, I incorporated the experiences generated by the rule-based method into the replay buffer. Subsequently, I utilized these experiences to train the TD3 model.

```
if self.total_time_step < self.warmup_steps:
    action = self.warmup_action(state)
else:
    # exploration degree
    sigma = max(0.1*(1-episode/self.total_episode), 0.01)
    action = self.decide_agent_actions(state, sigma=sigma)
```

2. I employed the rule-based method to determine the action, subsequently calculating the MSE loss between the rule-based action and the actor's action. I then combined this loss with the critic loss and performed model training.

```
action = self.actor_net(state)

rule_action = []
```

```
for i in range(state.shape[0]):
    rule_action.append(self.my_rule_2(state[i].unsqueeze(0
rule_action = torch.FloatTensor(np.array(rule_action))

critic_loss = self.critic_net1(state, action).mean()
rule_loss = criterion(action.to(self.device),
                      rule_action.to(self.device))

actor_loss = rule_loss - critic_loss
```

3. Exclusively training based on the rule-based loss.

```
action = self.actor_net(state)

rule_action = []
for i in range(state.shape[0]):
    rule_action.append(self.my_rule_2(state[i].unsqueeze(0
rule_action = torch.FloatTensor(np.array(rule_action))

rule_loss = criterion(action.to(self.device),
                      rule_action.to(self.device))

actor_loss = rule_loss
```

Method Comparison and Evaluation

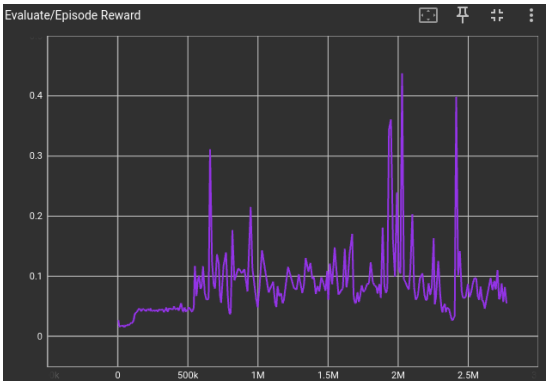
This section will outline my evaluation result for the Circle_cw and Austria environments separately.

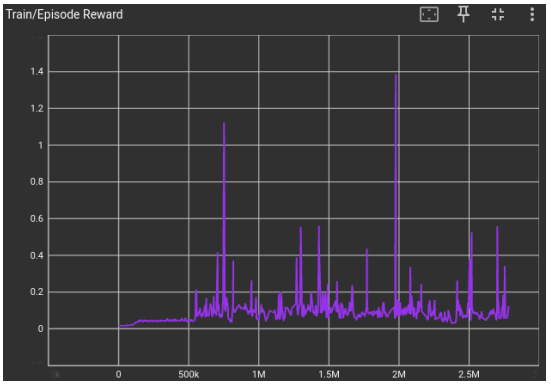
Each experimental result will correspond to one of the methods outlined above, and it can be matched through headers and indices for reference.

Circle_cw

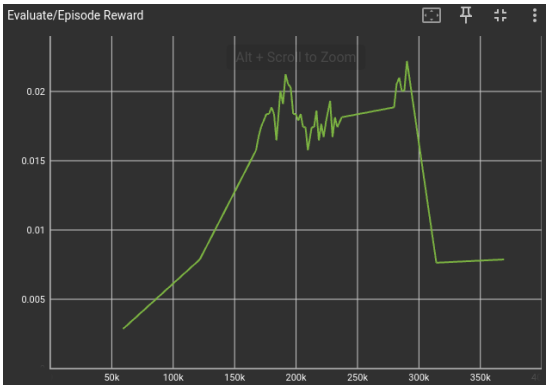
Reward Function

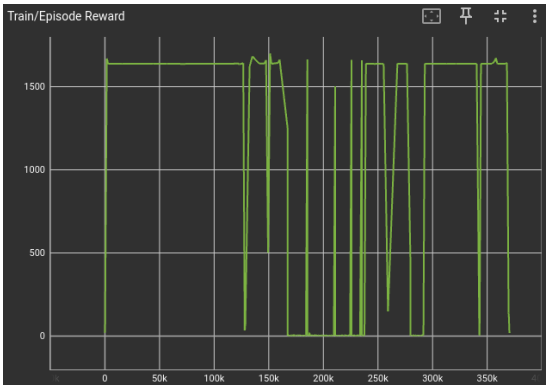
1. The reward experienced a slight increase, but the improvement was not substantial.



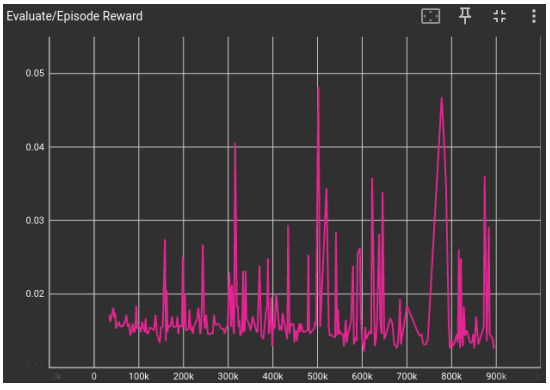


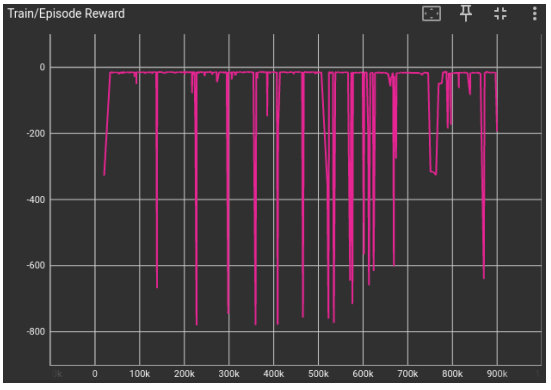
2. The reward is very small.



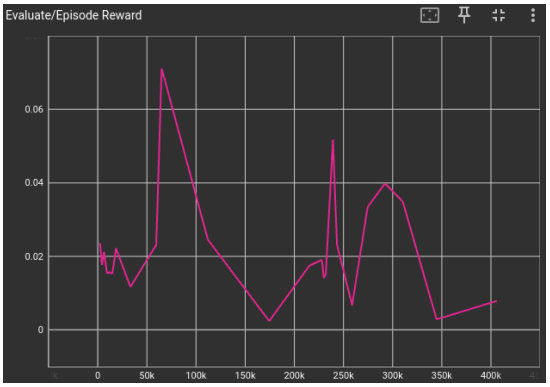


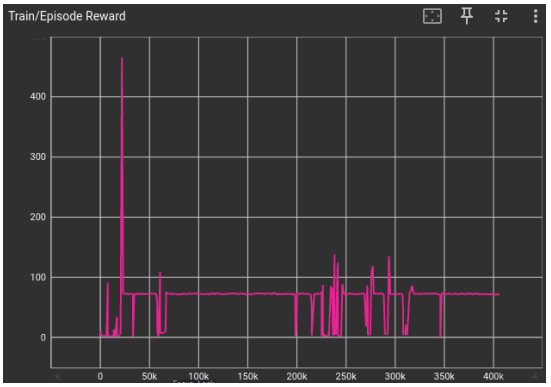
3. The curve undergoes significant fluctuations.



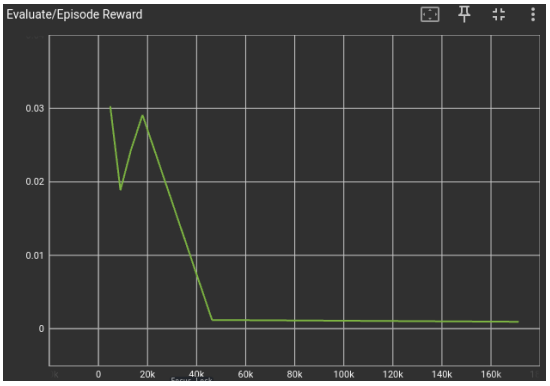


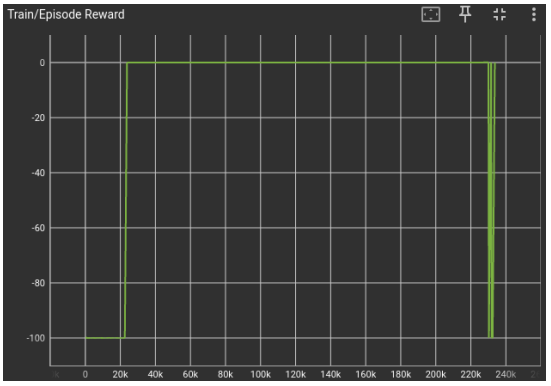
4. There is no clear upward trend in the reward.





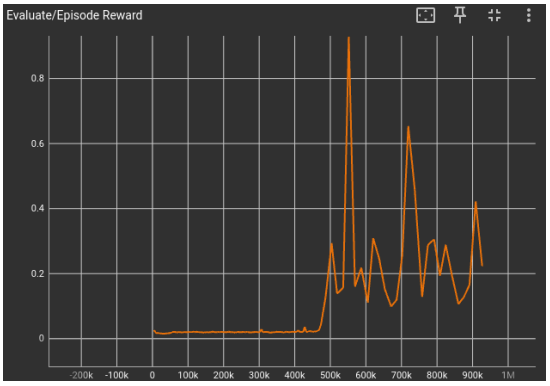
5. The reward is very poor because the racecar is afraid of hitting the wall, prompting it to choose to stay in place and not move.

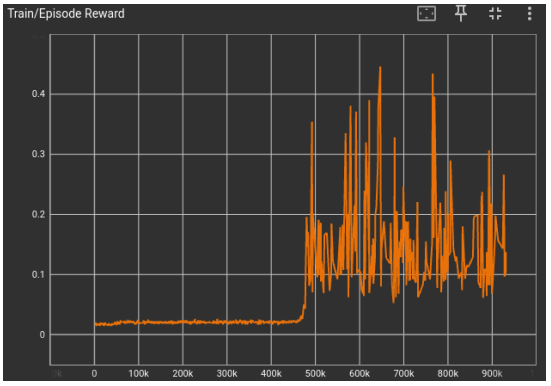




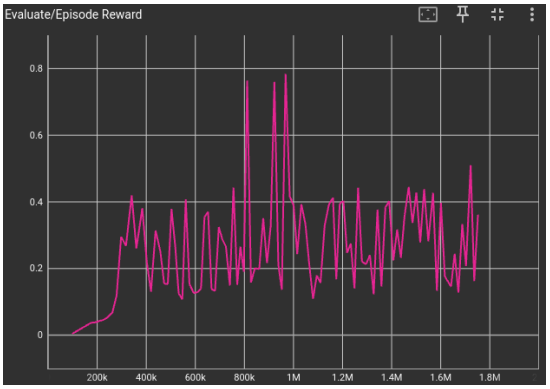
Action Space & Noise

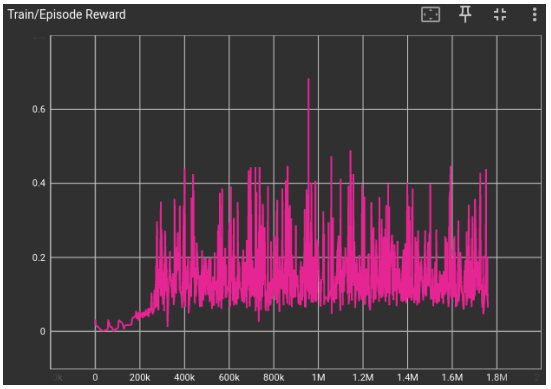
1. From the graph, it can be observed that the reward experiences a sudden increase but then remains around 0.1 to 0.2.



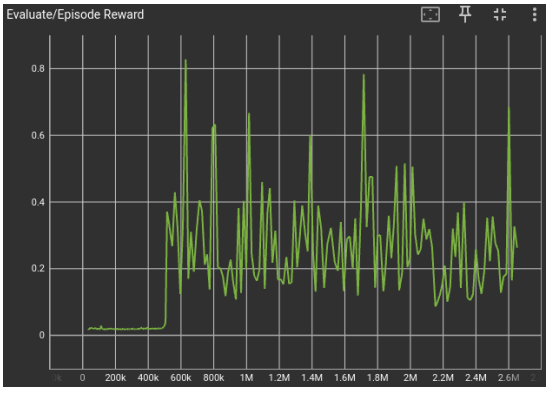


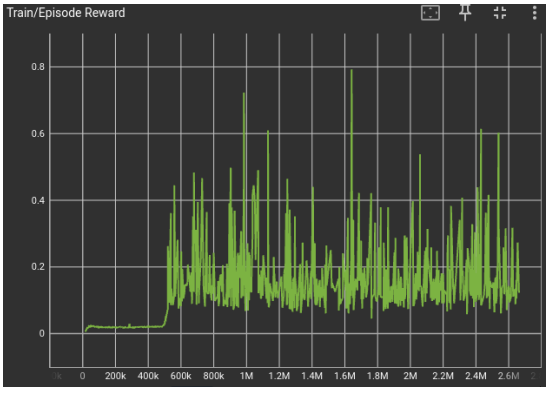
2. I observed that the reward increased earlier and has been oscillating between 0.2 and 0.4, but there hasn't been any further upward trend.





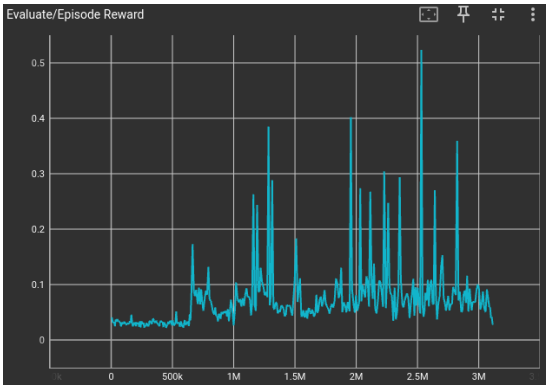
3. This graph looks similar to the previous one, and there is no apparent trend of continued increase, too.

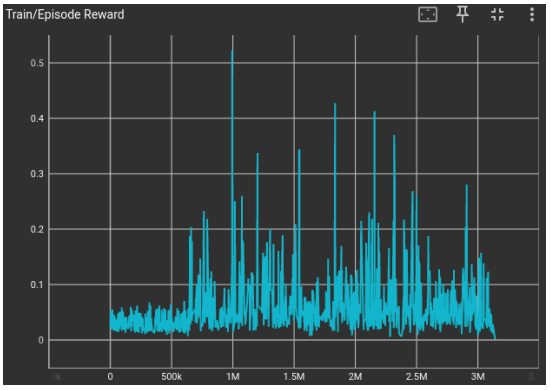




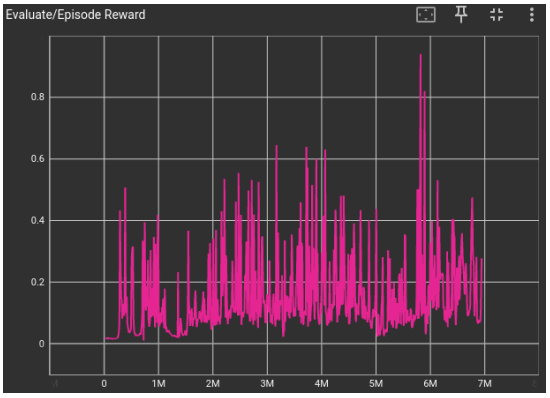
Resize Observation & Random Start Point

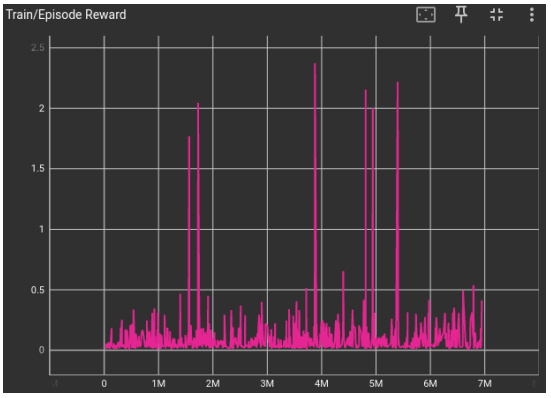
1. There is a slight upward trend, but it remains quite unstable.





2. Similar to the previous graph, although there is a slight upward trend, it still remains quite unstable.





Rule-based

1. If the left wall is greater than 50% of the observation length, then turn right, and vice versa.

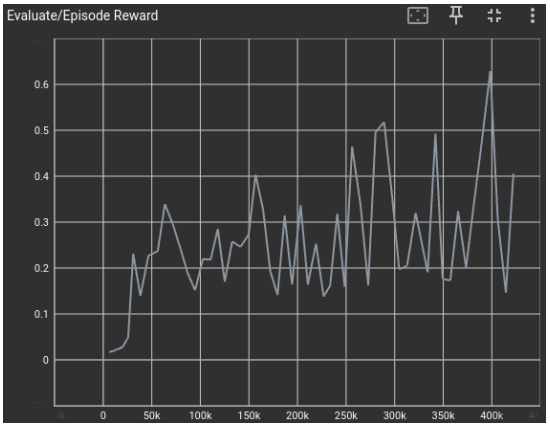
   Best score: 1.6551

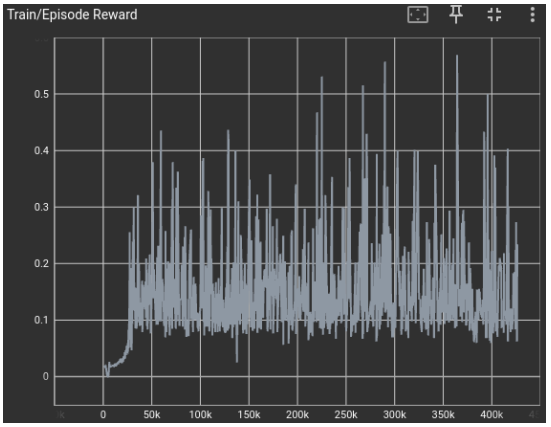
2. If there are 4 more pixels on the left than on the right, then turn right, and vice versa.

   Best score: 1.6695

RL from Demonstration

1. There is an overall upward trend, but the oscillations are still substantial.





Austria

Rule-based

1. If the count of pixels representing the left wall exceeds 40% but is less than 60% of the observation length, instruct the agent to turn right. If the count surpasses 60%, guide the agent to turn more sharply to the right, and vice versa.

   Best score: 0.4003

2. If there are 4 more pixels on the left than on the right, then turn right, and vice versa.

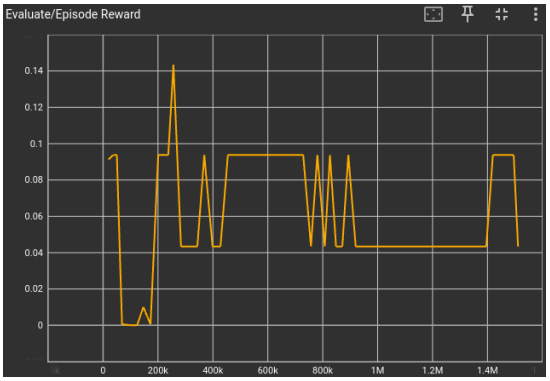
   Best score: 1.9515

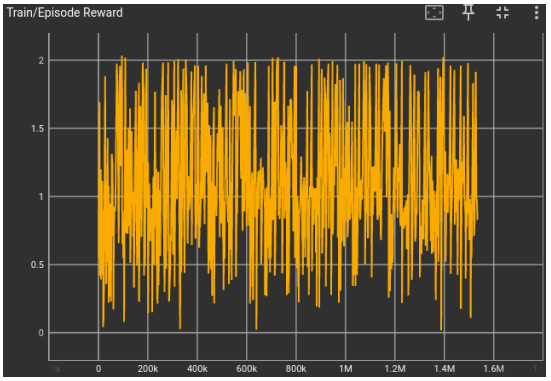
3. Momentum: To change the action, the same action must be taken multiple times consecutively.

   Best score: 1.5500

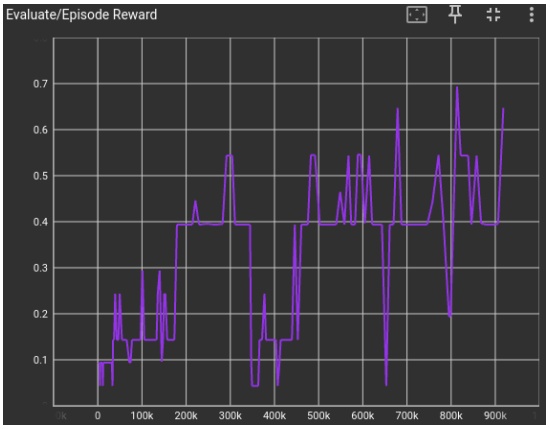
RL from Demonstration

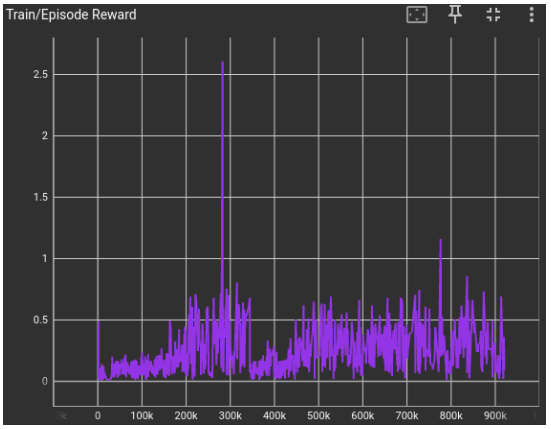
1. Even though the training reward is high, the evaluating reward is low. I guess this is because the actions are not generated by the actor.



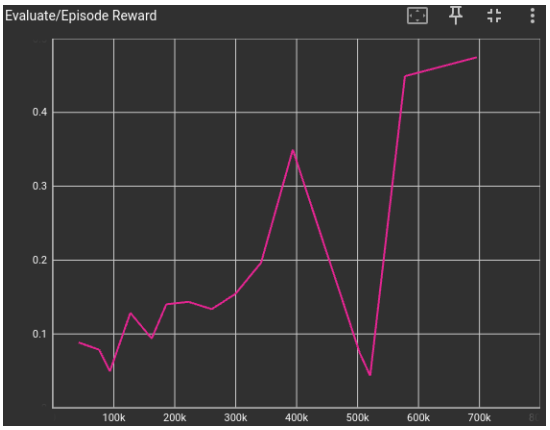


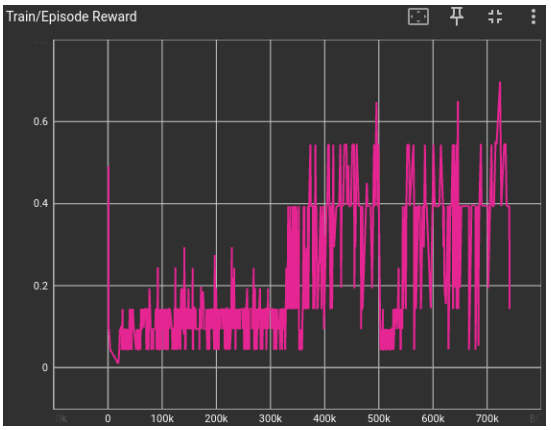
2. The training curve does not show a clear upward trend, while the evaluation curve demonstrates an ascending pattern. Although it can reach a maximum of 0.7, it still lags significantly behind the performance achieved by the rule-based approach.





3. The performance did not meet expectations, and there is a noticeable decline of around 500K, suggesting the possibility of overfitting.





Demo Result

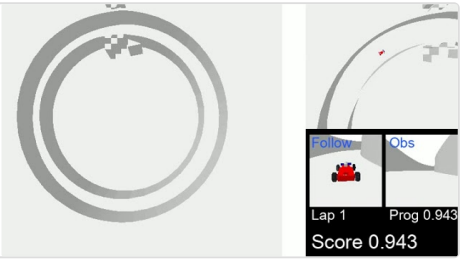
As the rule-based method outperformed other approaches significantly, I ultimately chose to use the rule-based method for the final demo.

Circle_cw

RL Final Project Circle

Best score: 1.7005

https://youtu.be/0Fsr1BjZeB0



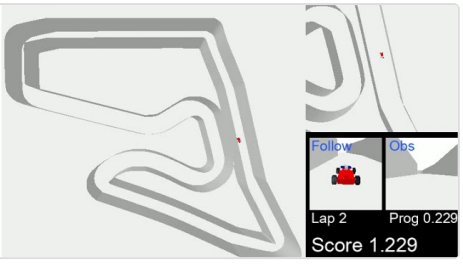
Rank	SID	Score	Env Time (sec)	Acc Time (sec)	Video
1	312553051	1.7649	25.020	409	Link
2	312551034	1.7136	25.020	234	Link
3	411551039	1.7029	25.020	227	Link
4	311551142	1.7017	25.020	226	Link
5	312553024	1.7005	25.020	218	Link
6	111101018	1.6969	25.020	186	Link

Austria

RL Final Project Austria

312553024 江尚軒

https://youtu.be/sMvP-rLaIlA



Rank	SID	Score	Env Time (sec)	Acc Time (sec)	Video
1	411551039	3.4122	100.020	1199	Link
2	312581020	3.3499	100.020	1051	Link
-	--- Aquila ---	3.3040	100.020	165	Link
3	310551002	3.1812	100.020	915	Link
4	311553043	3.0881	100.020	1067	Link
-	--- Libra ---	3.0403	100.020	791	Link
5	312551034	2.9327	100.020	707	Link
6	110705013	2.7137	100.020	615	Link
7	311551142	2.6507	100.020	1086	Link
8	312551113	2.5500	96.920	559	Link
9	109705001	2.3726	100.020	900	Link
10	312553024	2.3121	100.020	744	Link

Challenges and Learning Points

Challenges

1. In the beginning, I was not very familiar with the environment, and it took a considerable amount of time to comprehend it.
2. No matter how I modified the reward function, it consistently performed poorly. In the end, I realized that not making any changes to the reward function was the most effective approach.
3. Because the observation is in 3D, it is challenging to determine whether to turn left or right, especially in some sharp turns that are difficult to navigate. I believe this is the reason why TD3 struggles to train effectively.
4. During the first demo, I struggled to surpass even the simplest baseline, which was a bit disheartening for me.

Learning Points

1. Understanding a new environment.
2. Modifying reward function and action space.
3. Methods of RL from demonstration, including combining the critic loss and rule loss.

Future Work

1. The current observation is too complex; perhaps there are methods to simplify the observation.
2. Even though the reward functions I have tried did not yield satisfactory results, I believe there must be a reward function that can outperform the original one.
3. Because TD3 involves a continuous action space, it is challenging to train. In my next attempt, I will consider switching to a discrete action space, for example, using PPO.