RL final project report　　姓名:陳品翰　學號:313552042

Methodology Introduction:

1. Introduction of the methodology used for each map:

   這次的 final project 裡兩張地圖我都是採用 PPO 的訓練方式， PPO 是採用策略梯度法對 policy 直接更新，且更新的時候會有限制更新的幅度，用這個方式來避免和原本的策略相差太多，以保證整個訓練過程的穩定性和避免在學習的過程中有激烈的震盪，另外限制更新幅度還可以避免更新太多後偏離當前的方向，進而造成新的 policy 效能急遽降低的情況。

2. Explanation for the choice of methods for each map

   會採用 PPO 是因為我在訓練的過程中並沒有把動作改成離散的方式，而是直接採用連續動作，且在這個遊戲中有油門跟煞車兩個動作，PPO 採用直接用策略梯度對 policy 做更新，比較不會受到高維動作空間的影響，相較於 TD3 合適，且因為我有採用隨機起點的環境跟噪聲，PPO 可以更穩健的面對這種隨機的情況，不像 TD3 會對很多隨機的情況過於敏感。

Experiment Design and Implementation:

1. Description of the training processes:

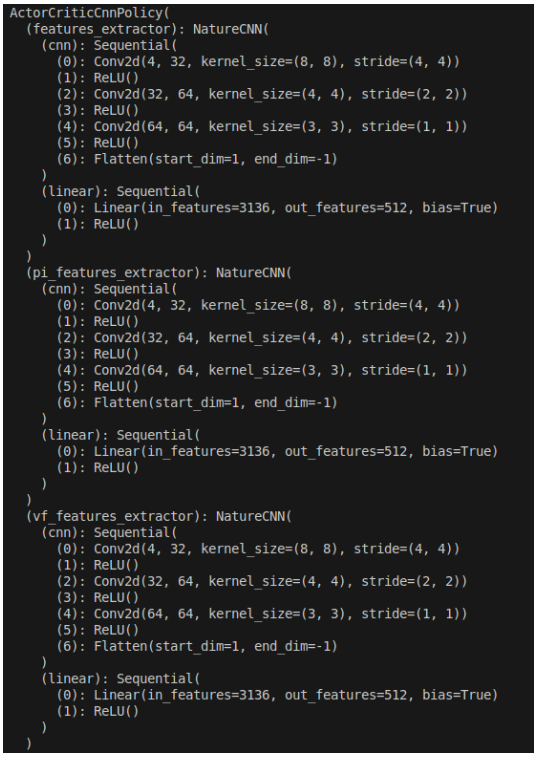
   在訓練個過程中，賽車一開始為隨機位子，且輸入到 model 的畫面為 4 * 84 * 84，也就是把 4 個畫面用 frame_stack 放在一起輸入到 model，且改為灰階的畫面，另外我有自己寫一個 reward function，以下附圖

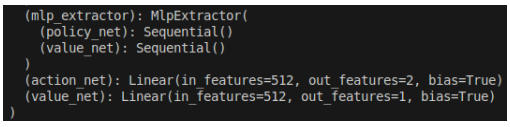
```
action_penalty_1 = -0.1 * np.abs(actions[self.motor_name])
action_penalty_2 = -0.1 * np.abs(actions[self.steering_name])
#print('action = ' , actions)
#print('motor_name = ' , actions[self.motor_name])
#print('steering_name = ' , actions[self.steering_name])
# Example reward based on progress (from `info` dictionary)
progress = info.get("progress")  # Assuming `info` contains a "progress" key

# Example collision penalty
collision_penalty = -0.8 if info.get("wall_collision", False) else 0
if(progress == self.pre_progress):
    progress_reward = -0.5
else:
    progress_reward = progress - self.pre_progress
# Combine rewards
reward = base_reward + progress_reward  + collision_penalty + action_penalty_1 + action_penalty_2
self.pre_progress = progress
return reward
```

   當有急遽的左右或煞車、油門時，會扣比較多分，用這個方式來避免做出劇烈的動作，另外當撞到牆時，會直接扣 0.8 分，停在原地扣 0.5 分，如果有成功向前，則加上他前進的距離。

   在訓練的過程中，我有採用多環境的方式，讓整個訓練可以加速。

2. Neural network architectures:





用的是 stable-baselines3 中的 Cnnpolicy 提供的 cnn model，可以看見一開始是用來接收輸入的圖片，接下來做特徵的提取，之後 Actor-Critic 分開，最後是輸出。

3. Details of the hyper-parameters

```
model = PPO("CnnPolicy" , vec_env ,
            learning_rate = 0.0005 , batch_size = 64 ,
            gamma = 0.95 , n_steps = 1024 , n_epochs = 10 ,
            verbose = 1 , device = "cuda" , tensorboard_log = "./log/modified_PPO")
```

可以看見採用的是 Cnn model，第二個為 environment，這邊放的是上面提到過的多環境，環境數為 4，接下來 learning rate 、 batch_size、 gamma 就如圖所示，n_steps 代表要更新前收集的步數，n_epochs 代表更新時的跌代次數，clip_range 採用預設的 0.2，entropy 則是預設的 0。

4. List of packages, tools, or resources used

這個作業我是使用 stable-baselines3 提供的 PPO 函式跟 Cnnpolicy 的 model 來做訓練的。

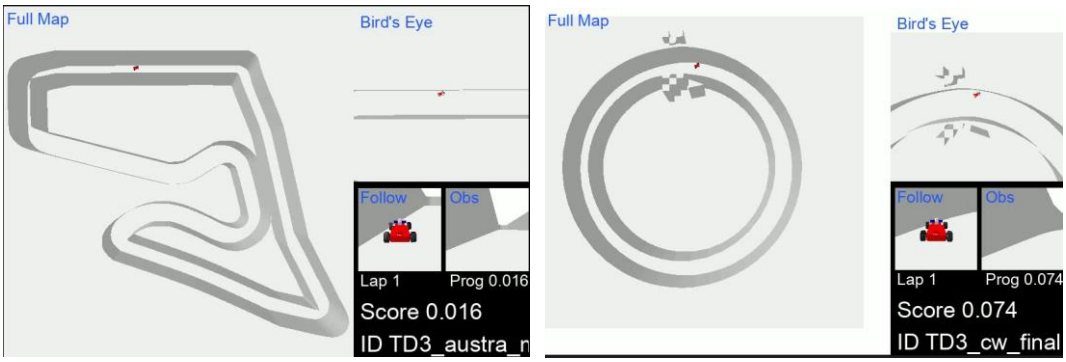
Method Comparison and Evaluation

1. Try different methods and show their results

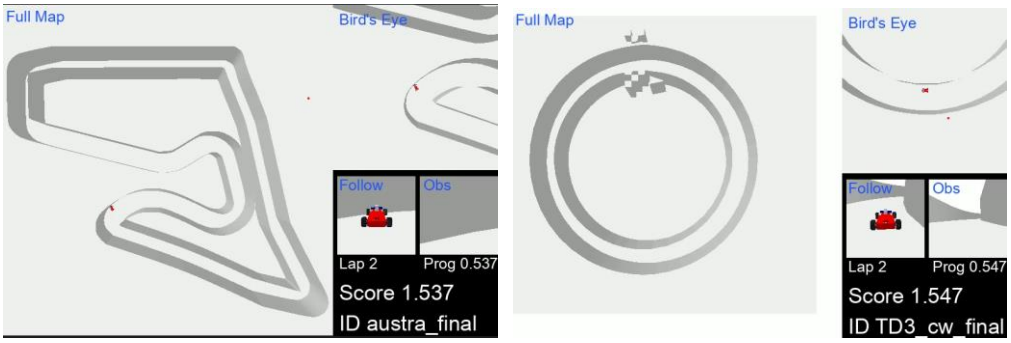
這個作業我有嘗試用 TD3 跟 PPO，下面直接展示遊戲最後的結果，沒附上 tensorboard 是因為 reward 我有自己改過，所以圖表的結果跟實際的結果會

有很大的落差。

下圖為 TD3 的遊戲結果，可以看到一進遊戲馬上就撞牆了。



下圖為 PPO 的遊戲結果，可以看到相較於 TD3 好非常多。



2. Comparing the effectiveness of different approaches

   TD3 採用函數逼近方法，在適當的 hyper-parameter 下可能可以有很好的結果，但因為環境有噪聲的關係，所以需要花很長的時間去調整。

   PPO 則可以更好的適配這個環境，也可以有很好的穩定度跟高維連續動作空間的適應性。

3. Analyzing the successful and unsuccessful cases

   就像我前面說的，TD3 需要花上很多的時間調整參數才可能有好的結果，如果參數設定不好，則因為對於隨機的環境敏感，所以造成結果非常的差，PPO 則可以適配這個環境，適當的更新 policy 以達到更好的成績。

4. Discussing the key observations and insights

   在我設定的 reward function 中，noise 是重要的因素，因為劇烈的動作會扣分，且停在原地扣很多分也盡量不可撞牆，所以如果 noise 剛好提供了一個動作適應當下的情況，則可以加很多分並且讓模型學到，如此一來便可以漸漸地跑完整個賽道。

Challenges and Learning Points

1. Encountered challenges during the implementation process

最大的挑戰就是我採用 TD3 的時候，無論我怎麼調整參數跟 reward function，最後的結果都很差，當我改用 PPO 後，可以得到一個相對於 TD3 好非常多的結果，之後我漸漸地更改 reward function 的計算方式，最後得到上面附圖的結果。

2. The learnings from these challenges
   在寫這個 project 以前我以為 TD3 很適配賽車的環境，但是在實行的過程中我發現 TD3 要 train 的好需要有很多的先備知識，來讓調整參數跟 reward function 時可以更有依據，我也認知到了 PPO 適合的環境是哪種，也切身體會到了選擇適合的 method 對於訓練效率的重要性。

Future Work

1. The proposal of ideas for potential improvements
   如果 PPO 調整 entropy 可能可以讓整個 model 更有探索的可能性，以此或許可以縮短訓練的時間，也可能會有更多不一樣的、更好的賽車路線。
2. The suggestion of additional research directions for the future
   希望可以研究 entropy 設定為多少時，對於整體訓練的提升最大，也希望可以研究出 learning rate 在什麼時候是最好的調整時機，以保證訓練的過程可以時刻都有效。