Lab4 Report

Twin Delayed DDPG

Student ID : 313554044

Student Name : 黃梓誠

NYCU Reinforcement Learning Fall 2024

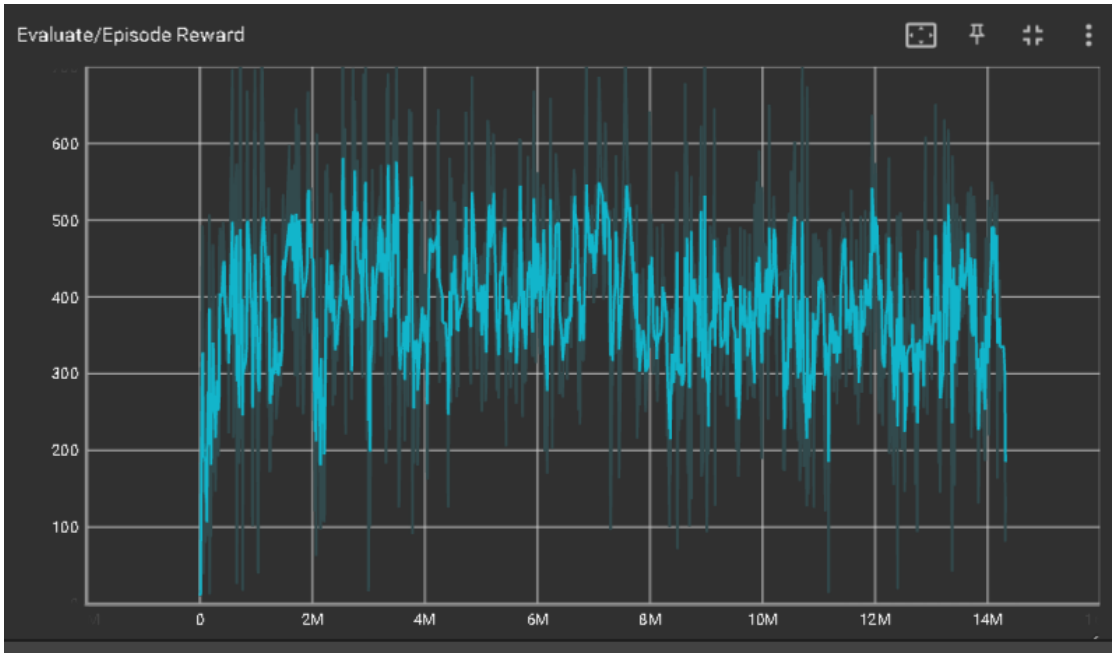
Date Submitted: November 24, 2024

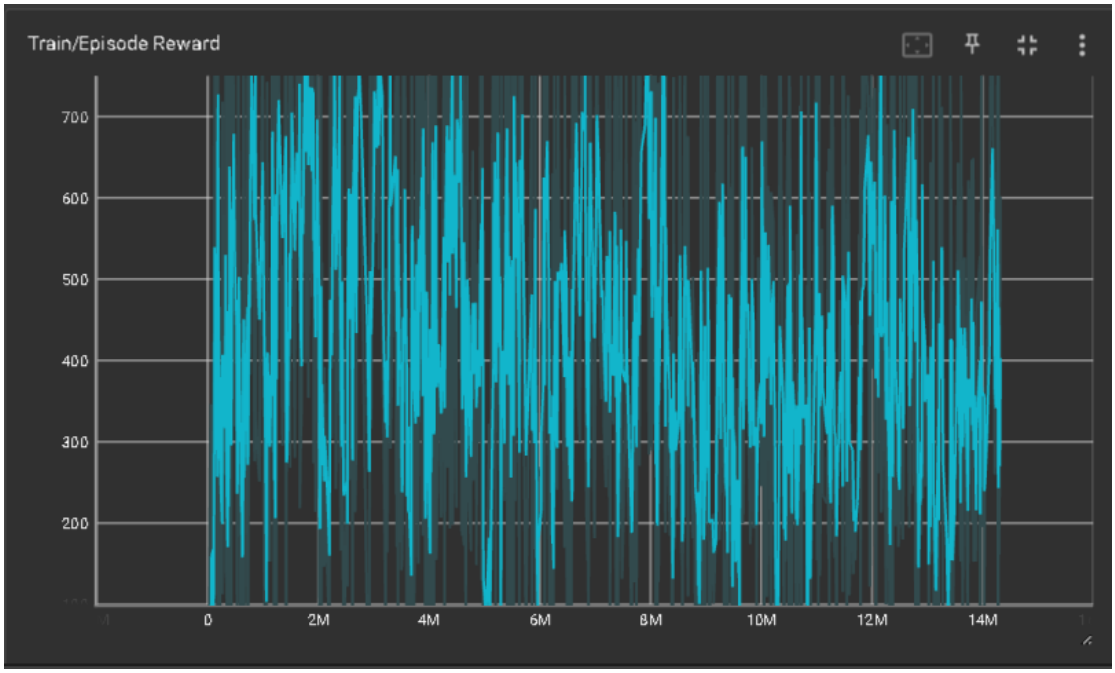
● **Report contains two parts:**

■ **Experimental Results (30%)**

(1) Screenshot of Tensorboard training curve and testing results on TD3.

■ Training curve





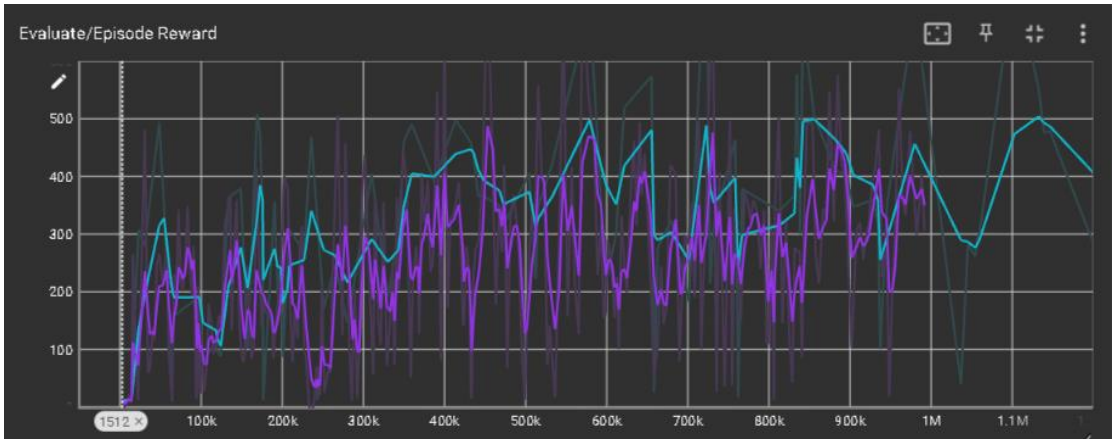
■ Testing results (10 games)

```
================================================
Evaluating...
Episode: 1      Length: 428     Total reward: 609.73
Episode: 2      Length: 330     Total reward: 486.76
Episode: 3      Length: 343     Total reward: 469.05
Episode: 4      Length: 687     Total reward: 879.30
Episode: 5      Length: 539     Total reward: 827.29
Episode: 6      Length: 276     Total reward: 433.84
Episode: 7      Length: 275     Total reward: 398.76
Episode: 8      Length: 373     Total reward: 434.92
Episode: 9      Length: 580     Total reward: 859.81
Episode: 10     Length: 178     Total reward: 147.51
average score: 554.6971055847955
================================================
```

(1) Screenshot of Tensorboard training curve and compare the performance of using twin Q-networks and single Q-networks in TD3, and explain (5%).

Purple: twin Q-networks

Blue: single Q-networks



Compare :

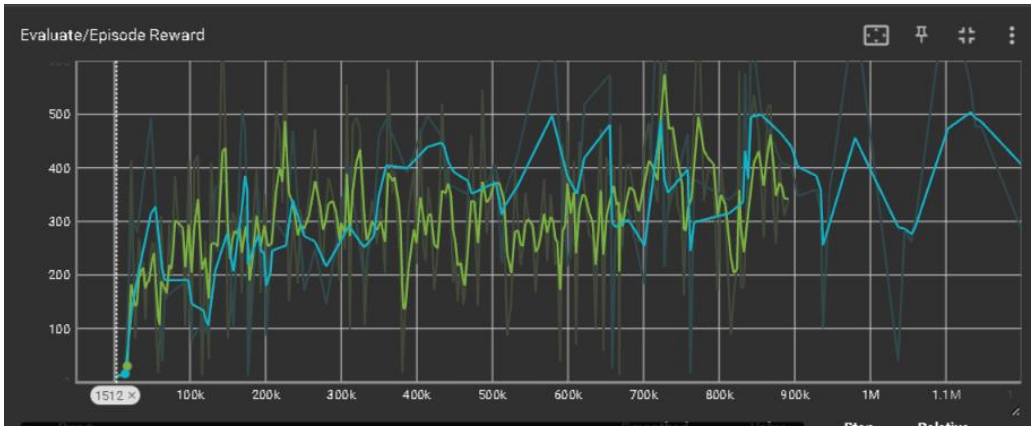
在比較 TD3 和不使用 twin Q-networks 的 TD3 時，我認為使用 twin Q-networks 的方法和 Double DQN 的核心概念非常相似，兩者的目標都是為了避免 Q 值的高估問題。

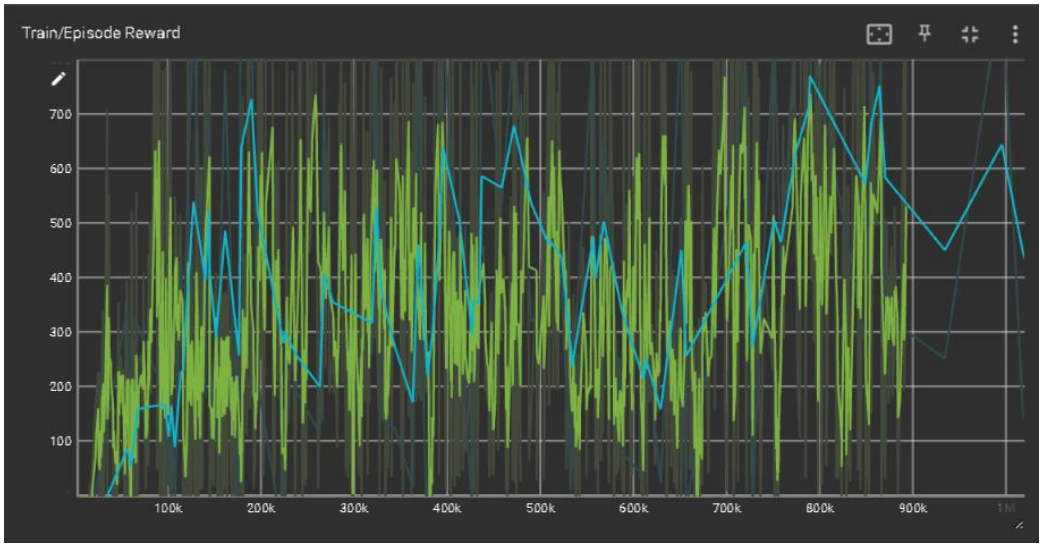
從實作的角度來看，TD3 的 twin Q-networks 是通過兩個 critic 網路分別預測 Q 值，並在其中選取最小值作為目標值。相比之下，這種方法的實作要比 Double DQN 的方法更加簡單和直接，因為後者需要額外設計目標 Q 網路更新的機制。

在實驗結果中，我觀察到不使用 twin Q-networks 的 TD3 在前期可能比較穩定，但後期性能明顯低於完整的 TD3 ，這表明 twin Q-networks 是 TD3 中非常重要的一部分。它有效地解決了高估問題，提升了演算法的穩定性和準確性。因此，從我的角度來看，twin Q-networks 的引入是 TD3 成功的關鍵之一。

---

(2) Screenshot of Tensorboard training curve and compare the impact of enabling and disabling target policy smoothing in TD3, and explain (5%).
Enable:blue / Disable : Green





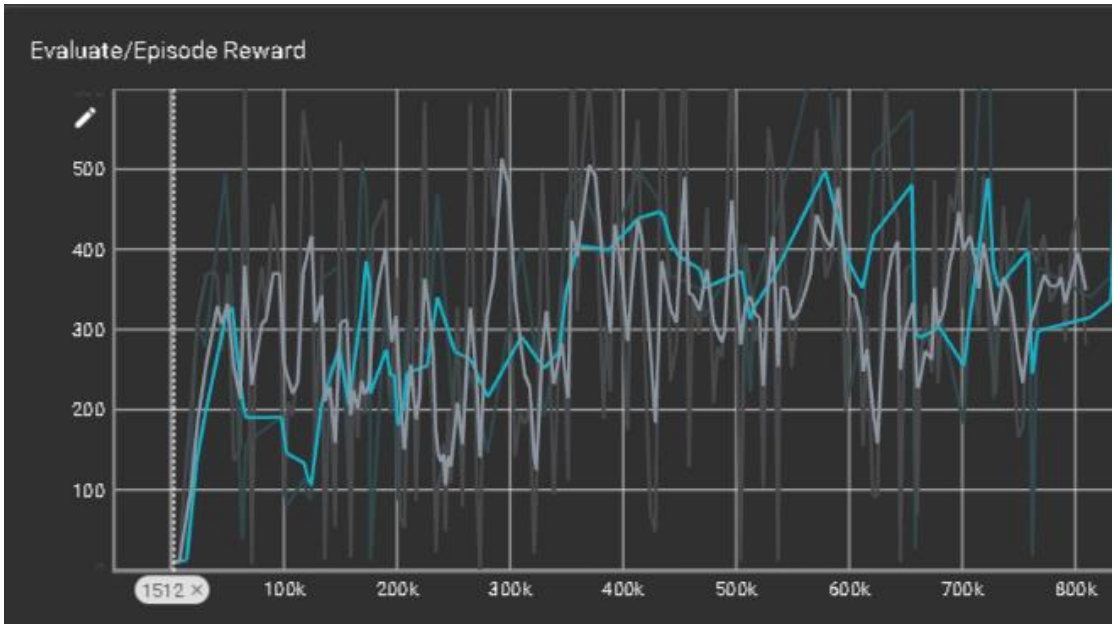
Compare :

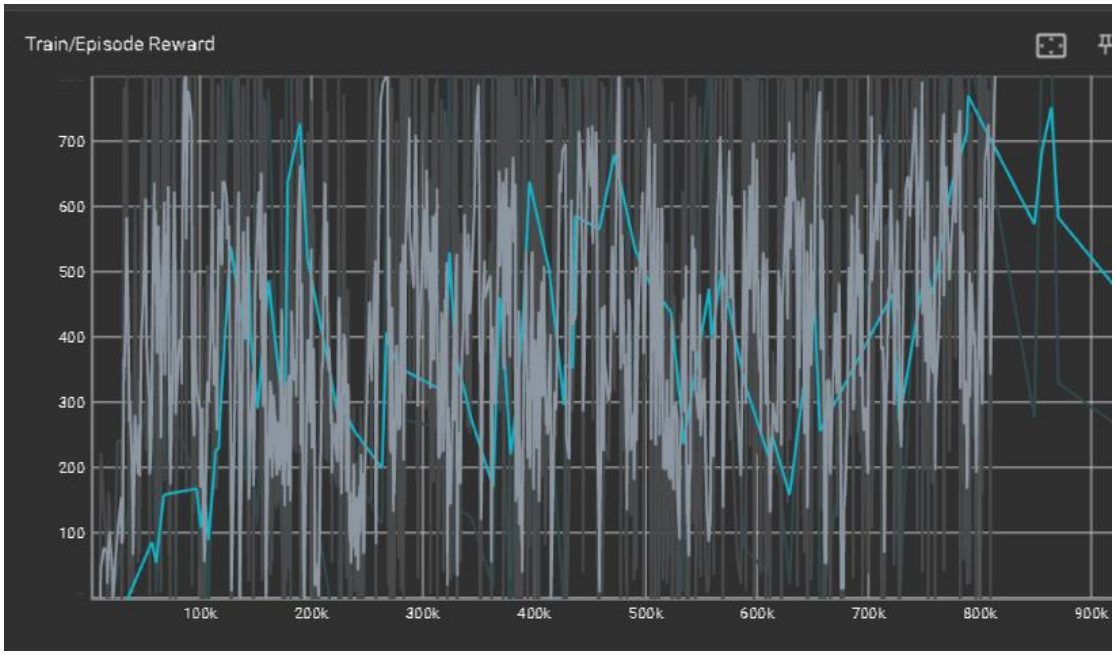
target policy smoothing 的核心是引入 noise 來平滑 target action，避免 policy network 過度專注於單一的 Q 值預測，從而降低過度擬合的風險：

1. **Enable** target policy smoothing：
    - 通過添加噪聲，TD3 在訓練過程中更加穩健，有效緩解了 Q 函數的高估偏差，從而提升了穩定性。
    - 回合獎勵的成長趨勢更平滑，模型能更有效學習。
2. **Disable** target policy smoothing：
    - 沒有 noise 的干預可能導致策略在某些狀態下過度樂觀優化，從而增加 Q 值估計的不穩定性。
    - 模型仍能學習，但表現的波動性較大，並且對隨機性的敏感性更高。

(3) Screenshot of Tensorboard training curve and compare the impact of delayed update steps and compare the results, and explain (5%).

Origin : Blue / Delayed update steps : Gray





Compare :

delayed policy updates 在 TD3 中的作用是降低 Q 函數更新頻率，使其在更新時有更準確的目標，從而減少高估偏差：

1. 使用 delayed policy updates（灰色曲線）：
    - Policy network 和 Q network 的更新頻率不同步，讓 Q network 有更多時間學習穩定的目標。降低了過度更新引起的不穩定性，提升了訓練與評估的穩定性。
2. 不使用 delayed policy updates（藍色曲線）：
    - Policy network 和 Q network 同步更新可能導致 Q 函數目標估計的過度偏差。更頻繁的更新導致模型容易出現震盪現象，從而降低了穩定性。

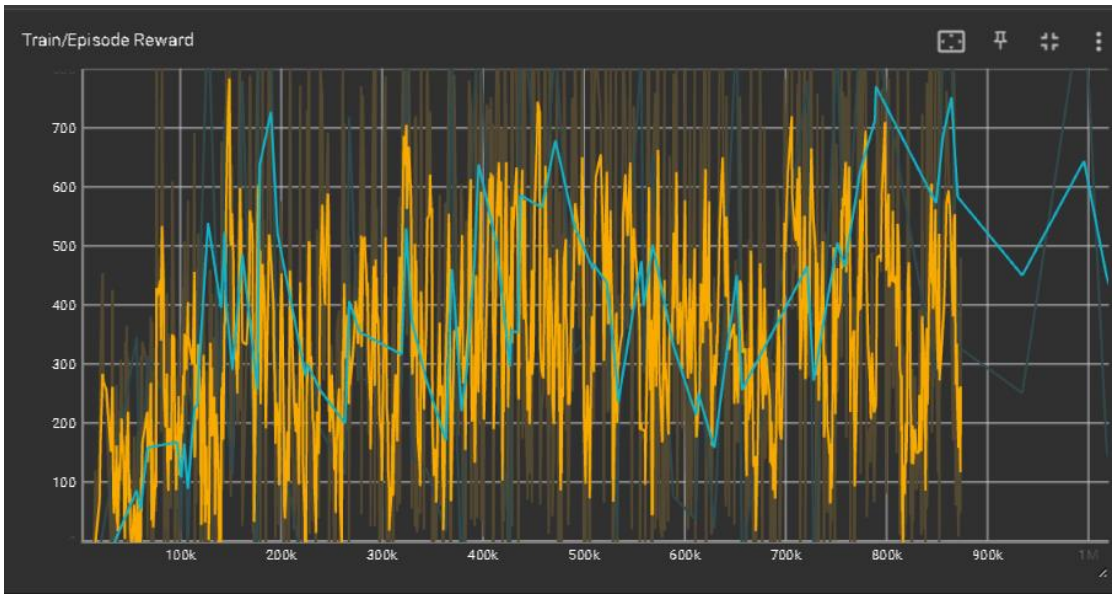
結論

delayed policy updates 理論上能有效提升 TD3 的訓練穩定性，避免頻繁更新帶來的不穩定性。但因為訊連時間不長 所以我只能得出使用 delayed policy updates 能夠讓震盪限制在一個範圍內，雖然長久下來最終的性能可能接近，但使用延遲更新的模型在應對隨機性和訓練波動時具有更好的表現。

---

(4) Screenshot of Tensorboard training curve and compare the effects of adding different levels of action noise (exploration noise) in TD3, and explain (5%).

Origin(高斯) : Blue / Perlin　noise : Yellow



在程式中我使用了三種不同於高斯的 **noise** 並比較高斯和 **Perlin** 的區別，其中我使用到的 **noise** 資料如下 ：

1. Uniform Noise: 雜訊值從一個固定範圍內均勻取樣。

2. Perlin Noise: 使用連續且平滑變化的雜訊。

3. Sinusoidal Noise: 基於正弦波的噪聲，可以用於週期性幹擾。

Compare :

1. 高斯 noise（Blue）：高斯噪聲具有隨機且 no bias 的特性，能夠有效平衡探索和利用，適合大多數環境。

2. **Perlin noise（Yellow）：**雖然 Perlin 噪聲的連續性可能更符合某些環境的動態需求，但在 TD3 中，其過於平滑的特性可能導致探索行為過於偏向特定區域，從而降低探索效率。在探索過程中，模型可能需要更多的嘗試才能擺脫次優解。

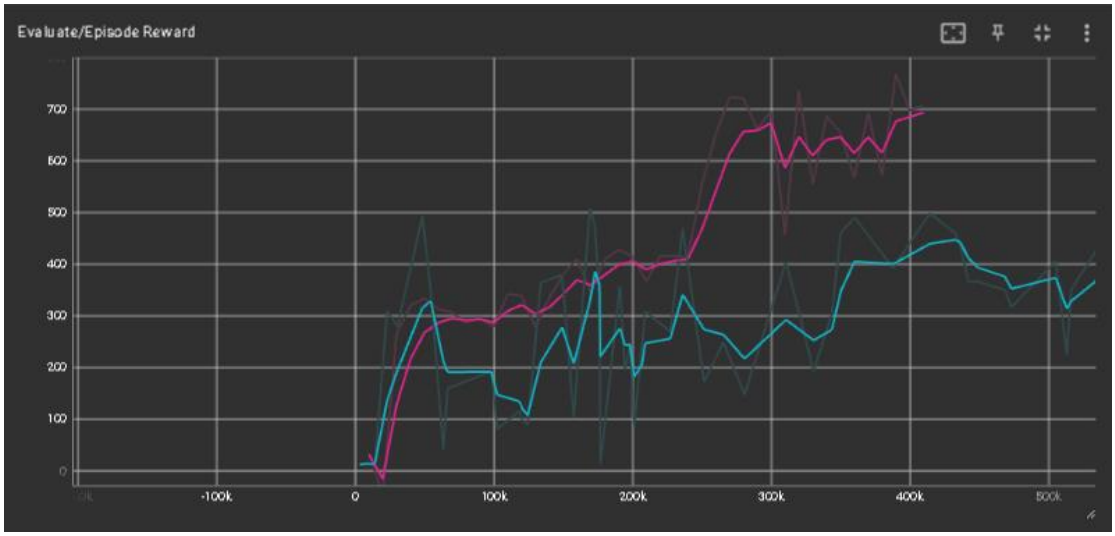
從這次比較中可以看出:

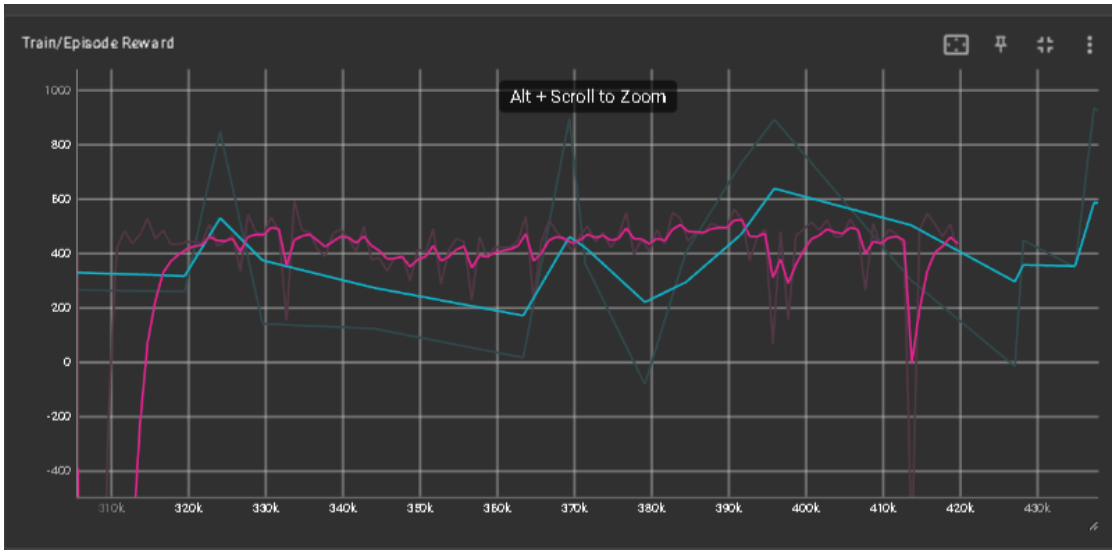
1. 標準高斯噪聲能提供穩定且有效的探索方式，使得 TD3 模型能夠在訓練過程中逐步提升表現，這對於許多穩定性要求較高的應用場景十分有利。
2. Perlin 噪聲雖然在某些情況下可能帶來探索的多樣性，但其過於平滑的特性也導致探索行為有偏，最終回報表現出明顯的不穩定，甚至容易陷入局部次優解。

這表明了在應用像 TD3 這樣的方法時，應充分考慮 noise 類型的影響，尤其是在探索與穩定性之間的平衡上，可能需要結合不同的 noise 設計更為適應環境特性的混合策略。

(5) Screenshot of Tensorboard training curve and compare your reward function with the original one and explain why your reward function works better(10%)

My reward function（pink） / origin reward function（blue）





Compare :

```
mpclab-gl@mpclabgl-Z690-AORUS-ELITE-DDR4:/mnt/md0/chen-
_reward_v5.py
================================================
Evaluating...
Episode: 1      Length: 999     Total reward: 854.41
Episode: 2      Length: 935     Total reward: 881.75
Episode: 3      Length: 756     Total reward: 664.11
Episode: 4      Length: 812     Total reward: 870.89
Episode: 5      Length: 823     Total reward: 917.60
Episode: 6      Length: 999     Total reward: 893.36
Episode: 7      Length: 806     Total reward: 849.80
Episode: 8      Length: 999     Total reward: 873.60
Episode: 9      Length: 833     Total reward: 916.60
Episode: 10     Length: 999     Total reward: 876.27
average score: 859.8388513200398
================================================
```

原始的 reward 函數根據車輛是否偏離賽道進行懲罰，並直接終止回合，這種方式過於簡單，無法提供足夠的反饋來幫助模型學習多樣化的駕駛策略。

首先，我的 reward 函數增加了對賽道佔用比例的考量，當車輛偏離賽道時，會根據偏離程度進行懲罰，而非立即結束回合，從而給模型更多的學習機會。

其次，我的設計鼓勵車輛保持高速行駛與直線駕駛，並對過度剎車進行適當懲罰，這些措施能夠促進模型學習穩定駕駛行為。

此外，我的 reward 函數還引入了「訪問新賽道區塊」的獎勵機制，鼓勵模型不斷探索未經過的區域，這對於提高整體車速有顯著幫助。

有趣的是，我的函數還加入了甩尾機制，希望他能學會甩尾。在彎道中成功甩尾給予獎勵，並對失控的甩尾行為進行懲罰，這不僅提高了模型的駕駛技巧，還使其能夠更靈活地應對不同的路況。

最後，完成賽道也會給予獎勵，這樣的終點獎勵能激勵模型專注於完成整圈賽道，從而提升模型的目標導向能力。

從訓練結果來看，我的 reward 函數在訓練的穩定性與表現的上限上均優於原始設計。由於我的函數能夠根據多維度的行為提供精細反饋，使得模型**更快學會優化駕駛策略**，從而達到更高的回報分數。