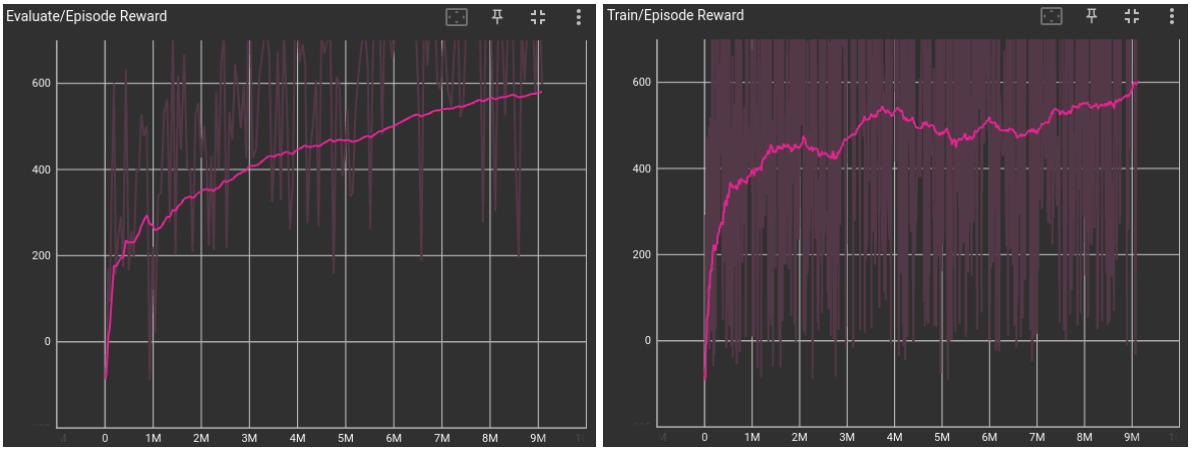
Lab 4 - TD3

312553024 江尚軒

■ Experimental Results (30%)

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

training curve



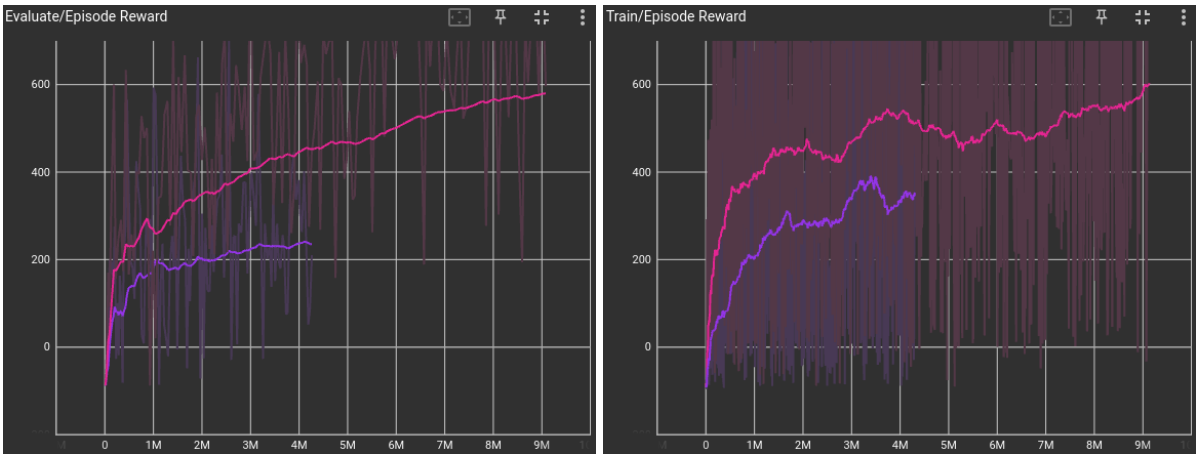
testing results

```
=================================================
Evaluating...
/home/adsl-1-2/anaconda3/envs/RL-Lab2/lib/python3.10/
bool`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
Episode: 1      Length: 999     Total reward: 866.22
Episode: 2      Length: 640     Total reward: 935.90
Episode: 3      Length: 560     Total reward: 629.26
Episode: 4      Length: 999     Total reward: 864.16
Episode: 5      Length: 668     Total reward: 703.01
Episode: 6      Length: 999     Total reward: 864.41
Episode: 7      Length: 606     Total reward: 722.63
Episode: 8      Length: 999     Total reward: 880.33
Episode: 9      Length: 999     Total reward: 843.06
Episode: 10     Length: 603     Total reward: 729.28
average score: 803.8260733662657
=================================================
```

■ Experimental Results and Discussion of bonus parts (Impact of Twin Q-Networks, Target Policy Smoothing, Delayed Policy Update Mechanism, Action Noise Injection) (bonus) (30%)

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

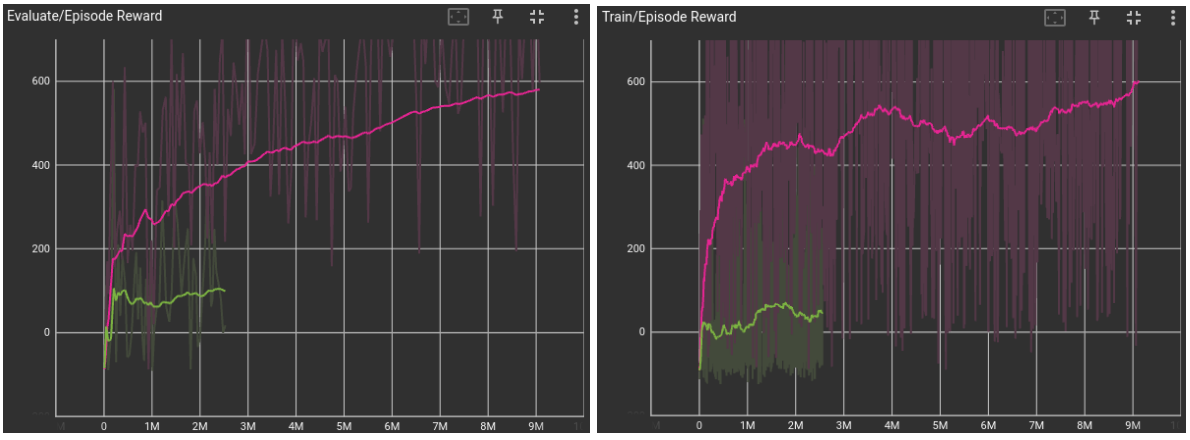
pink: TD3 with twin Q-networks
purple: TD3 with single Q-network



Using twin Q-networks in TD3 mitigates overestimation bias, improving Q-value estimation, robustness to function approximation errors, exploration stability, and generalization. Twin networks provide a conservative policy update, preventing aggressive changes. Overall, they enhance stability, accuracy, and robustness in the learning process.

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

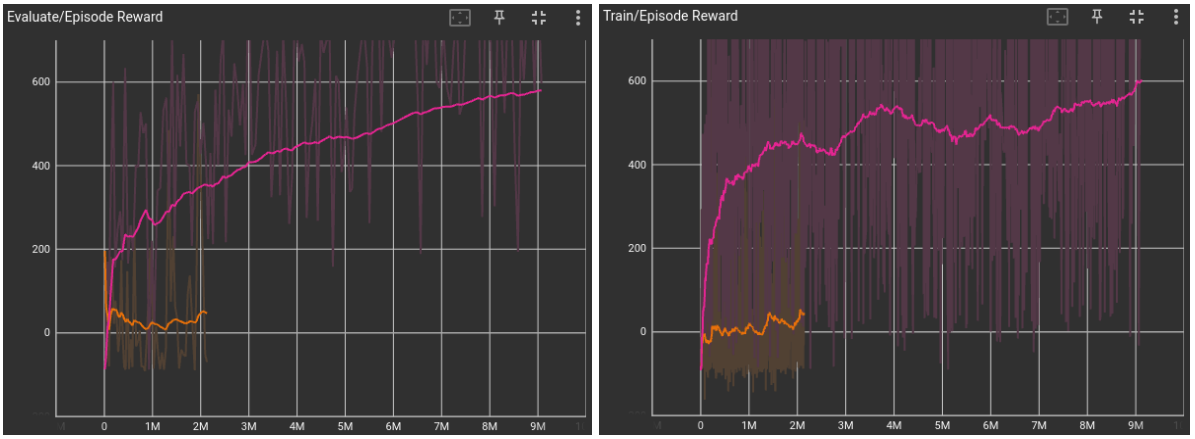
pink: TD3 with target policy smoothing
green: TD3 without target policy smoothing



Enabling target policy smoothing in TD3 improves stability and robustness during training by adding a noise term to the target actions. This regularization mitigates overfitting to specific actions and prevents the policy from getting stuck in local optima. It encourages a smoother policy update, leading to better exploration and convergence, ultimately enhancing the overall performance and training effectiveness of the TD3 algorithm.

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

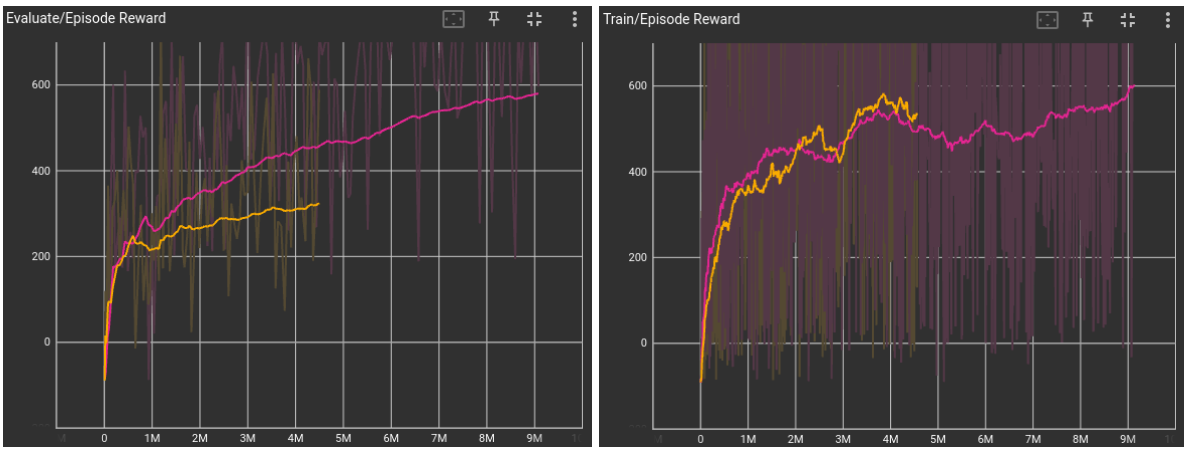
pink: TD3 with delayed update steps = 4
orange: TD3 without delayed update steps = 1



Delayed policy updates in TD3 address issues of stability, exploration, and overestimation bias, leading to more robust and efficient learning. Empirical results and theoretical considerations support the use of delayed updates as a crucial component for achieving superior performance in reinforcement learning tasks.

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

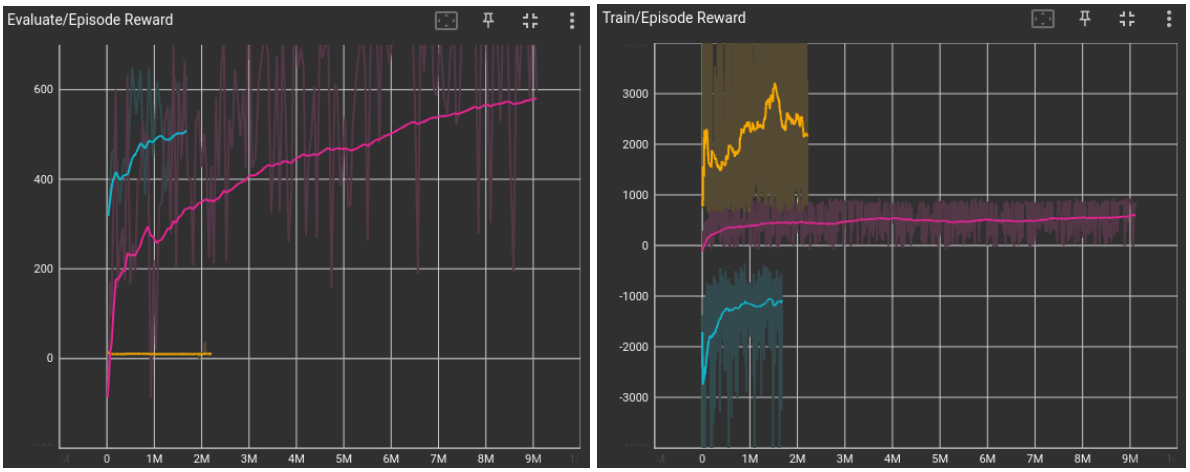
pink: TD3 with OU Noise
yellow: TD3 with Gaussian Noise



OU Noise is preferred over Gaussian Noise in TD3 because it introduces correlated noise, providing temporal stability, smoother exploration in action space, and reducing overestimation bias in Q-value estimates. Its mean-reverting property ensures consistent exploration, making it beneficial for learning in continuous action spaces.

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

pink: TD3 with a default reward function
blue: TD3 with my first reward function (reward -= 0.1 * grass_pixel_count)
yellow: TD3 with my second reward function (reward += 0.1 * road_pixel_count)



I modified the reward function to **"reward -= 0.1 * grass_pixel_count"** and **"reward += 0.1 * road_pixel_count."** In my opinion, the initial reward function is more effective as it imposes a negative penalty for grass contact, promoting the car's stability in the center of the road. Conversely, the second reward function is less optimal as it provides a positive reward for staying in the middle of the road, which may not be suitable for this specific environment.