LAB 4

TA 陳昱丞

2023/11/7

yucheng.cs11@nycu.edu.tw

Deadline: 2023/12/3 (Sun) 23:59

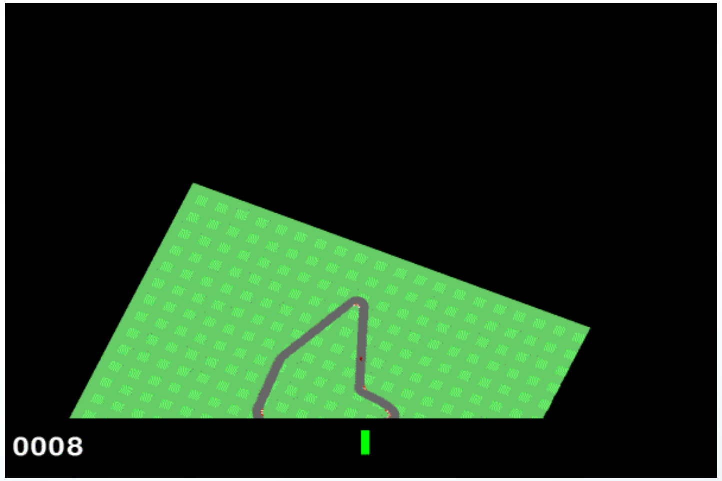
Demo: 2023/12/4 (Mon) 18:00

In this lab,

Must use sample code, otherwise no credit.

CarRacing-v2

- Introduction:
  - The easiest control task to learn from pixels - a top-down racing environment. The generated track is random every episode. Some indicators are shown at the bottom of the window along with the state RGB buffer. From left to right: true speed, four ABS sensors, steering wheel position, and gyroscope.
- Observation space:
  - The whole image
- Action space:
  - Steering (-1 is full left, +1 is full right)
  - Gas (0~1)
  - Breaking (0~1)



https://www.gymlibrary.dev/environments/box2d/car_racing/

Twin Delayed DDPG (TD3)

- Deep Deterministic Policy Gradient (DDPG)

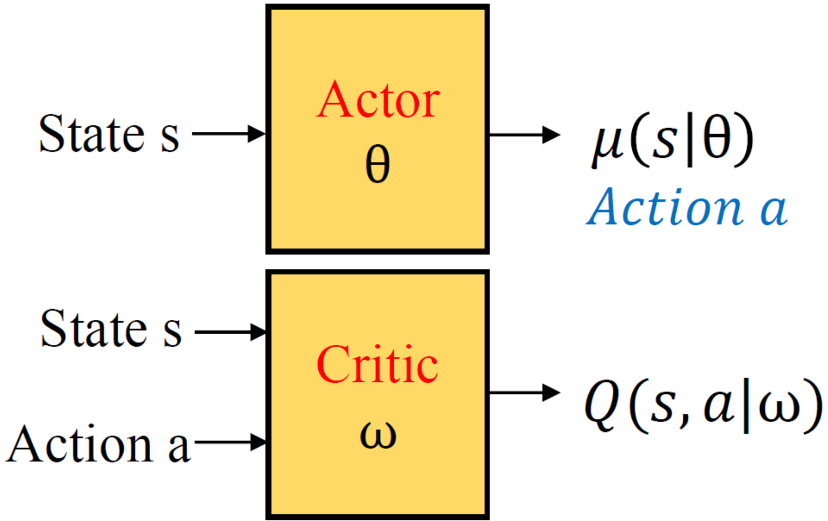
Critic estimates value of current action by Q-learning

TD error

$$\nabla_\omega L_Q(s_t, a_t|\omega)$$
$$= \left(\left(r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1}|\theta)|\omega)\right) - Q(s_t, a_t|\omega)\right) \nabla_\omega Q(s_t, a_t|\omega)$$

Actor updates policy in direction suggested by critic (DDPG):

$$\nabla_\theta J(\mu_\theta) \approx \mathbb{E}_\mu[\nabla_\theta Q(s_t, \mu(s_t|\theta)|\omega)]$$
$$= \mathbb{E}_\mu\left[\nabla_a Q(s_t, a|\omega)\Big|_{a=\mu(s_t|\theta)} \nabla_\theta \mu(s_t|\theta)\right]$$

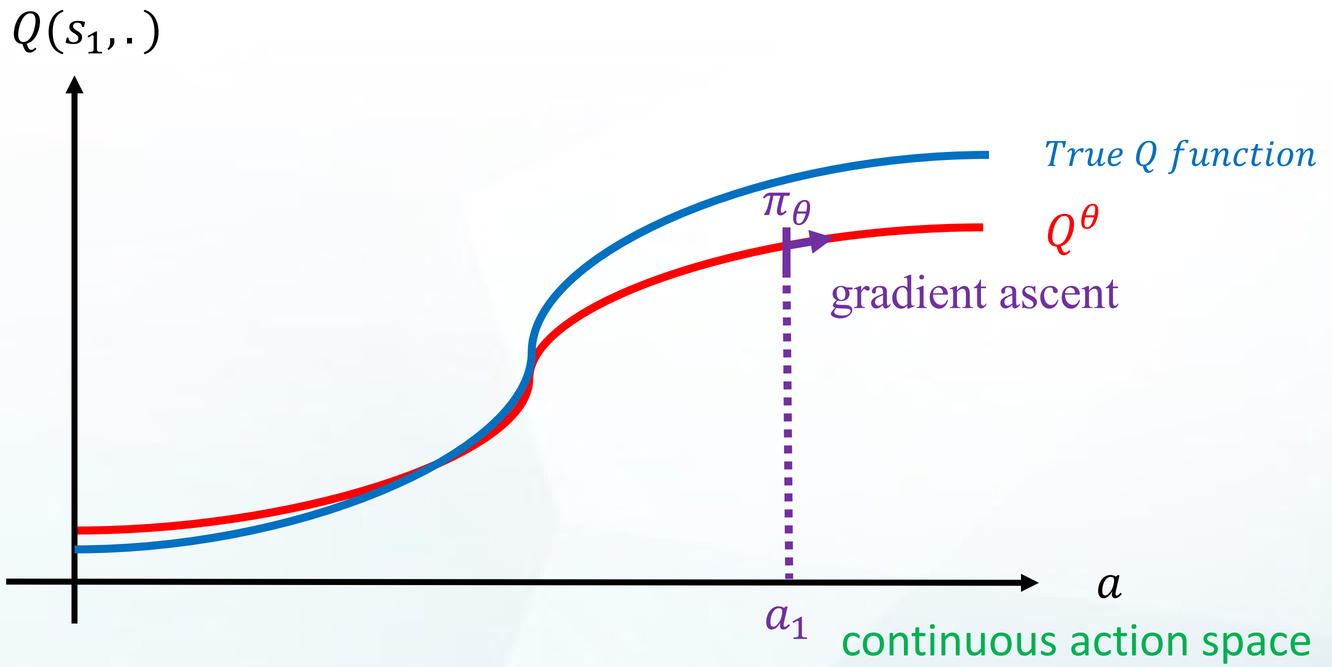


Twin Delayed DDPG (TD3)

Actor update:

$$\nabla_\theta J(\mu_\theta) \approx \mathbb{E}_\mu[\nabla_\theta Q(s_t, \mu(s_t|\theta)|\omega)]$$

$$= \mathbb{E}_\mu\left[\nabla_a Q(s_t, a|\omega)\Big|_{a=\mu(s_t|\theta)} \nabla_\theta \mu(s_t|\theta)\right]$$



Twin Delayed DDPG (TD3)

- TD3: Add 3 tricks in DDPG
  1. Clipped Double Q-Learning for Actor-Critic
  2. Delayed Policy Updates
  3. Target Policy Smoothing Regularization

Twin Delayed DDPG (TD3)

Algorithm – TD3 algorithm:

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau \theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau \phi + (1 - \tau)\phi'$
  end if
end for

1. Clipped Double Q-Learning for Actor-Critic

2. Delayed Policy Updates

3. Target Policy Smoothing Regularization

TODO

- Solve CarRacing-v2 using TD3.
- (Bonus) Ablation study:

3 tricks

1. Screenshot of Tensorboard training curve and compare the performance of using twin Q-networks and single Q-networks in TD3, and explain (5%).
2. Screenshot of Tensorboard training curve and compare the impact of enabling and disabling target policy smoothing in TD3, and explain (5%).
3. Screenshot of Tensorboard training curve and compare the impact of delayed update steps and compare the results, and explain (5%).

exploration noise

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

reward function

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

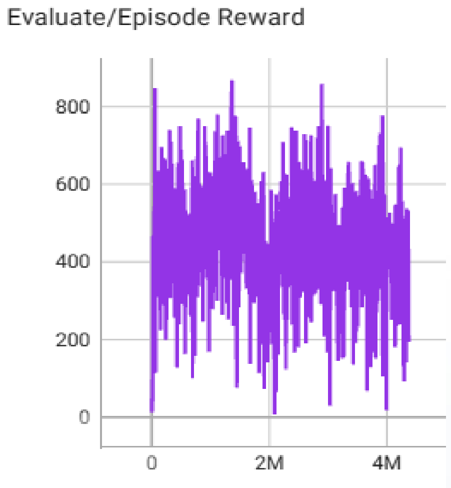
TODO

- Find the #TODO comments and hints, remove the raise NotImplementedError.
- Inherit from the "TD3BaseAgent" and override the "decide_agent_actions" and "update_behavior_network" functions.
- You can try your reward function and network architecture.
- Screenshot of Tensorboard training curve and testing results and put it on the report.

TODO

- Screenshot of Tensorboard training curve and testing results and put it on the report.

Training curve:

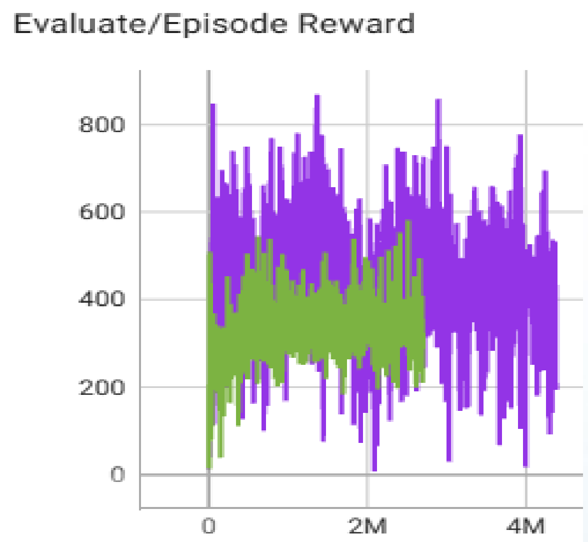


Testing results (10 games):

```
Episode: 1      Length: 999     Total reward: 874.44
Episode: 2      Length: 999     Total reward: 883.05
Episode: 3      Length: 999     Total reward: 797.44
Episode: 4      Length: 999     Total reward: 679.18
Episode: 5      Length: 999     Total reward: 866.78
Episode: 6      Length: 999     Total reward: 888.97
Episode: 7      Length: 751     Total reward: 924.80
Episode: 8      Length: 999     Total reward: 883.33
Episode: 9      Length: 999     Total reward: 614.81
Episode: 10     Length: 999     Total reward: 878.34
average score: 829.1142945389436
```

TODO

- Screenshot of Tensorboard training curve and testing results and put it on the report.



Scoring Criteria

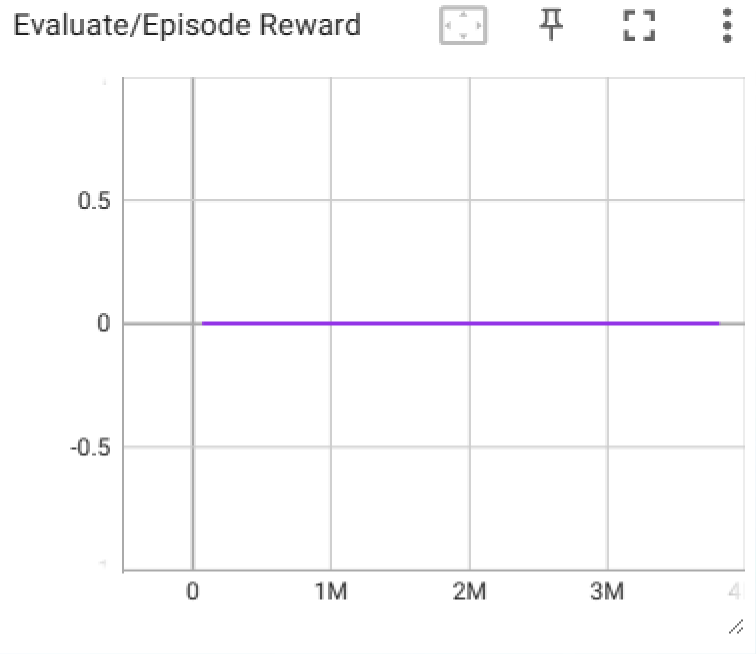
Your Score (130%) = report (30%) + report bonus (30%) + demo performance (50%) + demo questions (20%)

- **Report contains two parts:**
  - **Experimental Results (30%)**
    (1) Screenshot of Tensorboard training curve and testing results on TD3.
  - **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%).
    (2) Screenshot of Tensorboard training curve and compare the impact of enabling and disabling target policy smoothing in TD3, and explain (5%).
    (3) Screenshot of Tensorboard training curve and compare the impact of delayed update steps and compare the results, and explain (5%).
    (4) Screenshot of Tensorboard training curve and compare the effects of adding different levels of action noise (exploration noise) in TD3, and explain (5%).
    (5) Screenshot of Tensorboard training curve and compare your reward function with the original one and explain why your reward function works better (10%).

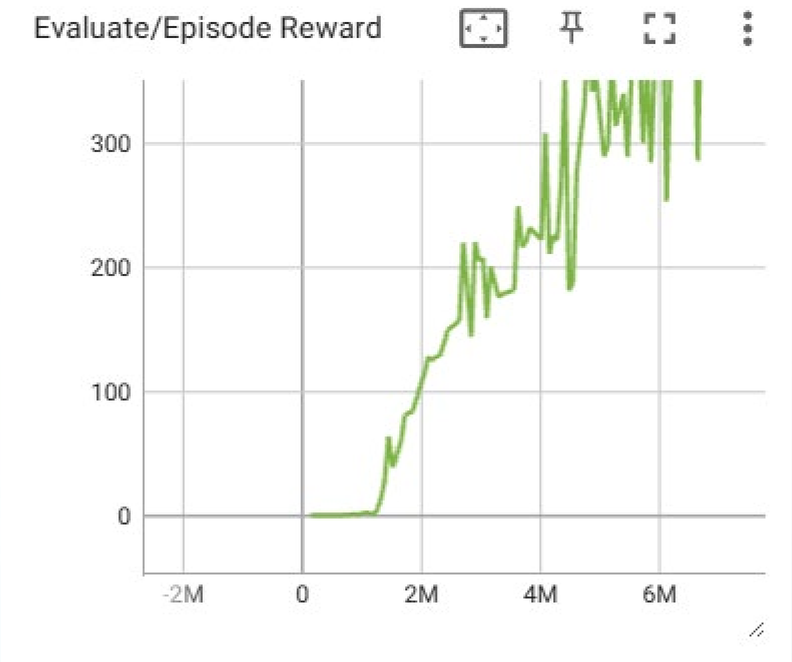
Scoring Criteria

- Screenshot of Tensorboard training curve and testing results and put it on the report.

No score



Get score (30%)



Scoring Criteria - Demo Performance

- Demo performance score = baseline (15%) + ranking (35%)
- Test your best model for five race tracks. Seeds of five tracks will be given on demo day.
- You have to show the video while testing. You can use env.render() or save video function to achieve this.
- You can use a fixed random seed to reproduce your best game score.
- If you outperform the baseline, you will get 15%. Other 35% will be based on your rank in all students.

Five race tracks avg.

Reward	Points (15%)
0~100	0
100~199	5
200~299	10
>=300	15

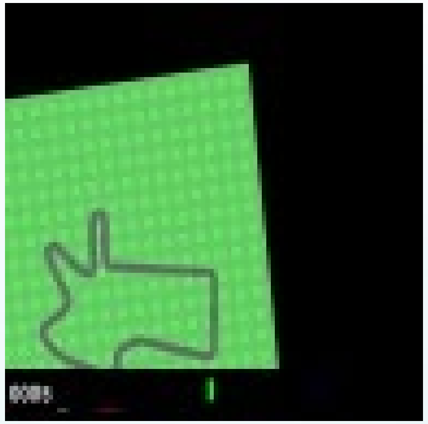
Scoring Criteria - Demo Performance

- Test your best model for five race tracks. Seeds of five tracks will be given on demo day.
- You can decide the race track by giving seed in env.reset().

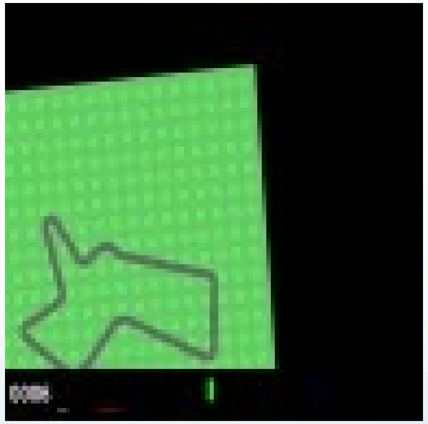
  For example:

  obs, info = self.env.reset(seed=?)

self.env.reset(seed=2)



self.env.reset(seed=5)



Tensorboard Remote Server

- ssh -p [your port] -L 6006:localhost:6006 pp037@140.113.215.196
- tensorboard --logdir log/dqn
- Open your browser locally and input 127.0.0.1:6006

Recommended Package Version

gym	0.26.2
numpy	1.25.2
pytorch	2.0.1
tensorboard	2.14.0
opencv-python	4.8.0.76
moviepy	1.0.3

Reminders

- Your network architecture and hyper-parameters can differ from the defaults.
- Ensure the shape of tensors all the time especially when calculating the loss.
- with no_grad() : scope is the same as xxx.detach()
- Be aware of the indentation of hints.

References

1. Lillicrap, Timothy P. et al. "Continuous control with deep reinforcement learning." CoRR abs/1509.02971 (2015).
2. Silver, David et al. "Deterministic Policy Gradient Algorithms." ICML (2014).
3. OpenAI. "OpenAI Gym Documentation." Retrieved from Getting Started with Gym: https://gym.openai.com/docs/.
4. PyTorch. "Reinforcement Learning (DQN) Tutorial." Retrieved from PyTorch Tutorials: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html.
5. Dankwa, Stephen, and Wenfeng Zheng. "Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent." Proceedings of the 3rd international conference on vision, image and signal processing. 2019.
6. My results: https://youtu.be/FAqATf_k5fI?si=p_bwyjt4RDchLmJ5