Application of DDPG in Multidimensional Continuous Action Space

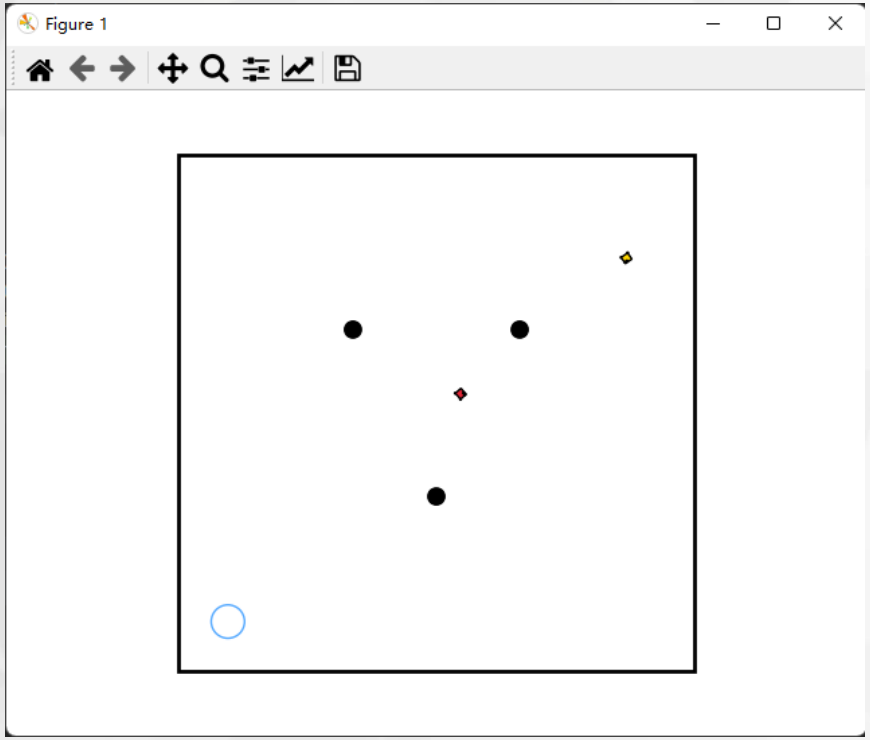
Xu GE
Jialu Xu

Contents

Environment Setup

Environment Setup

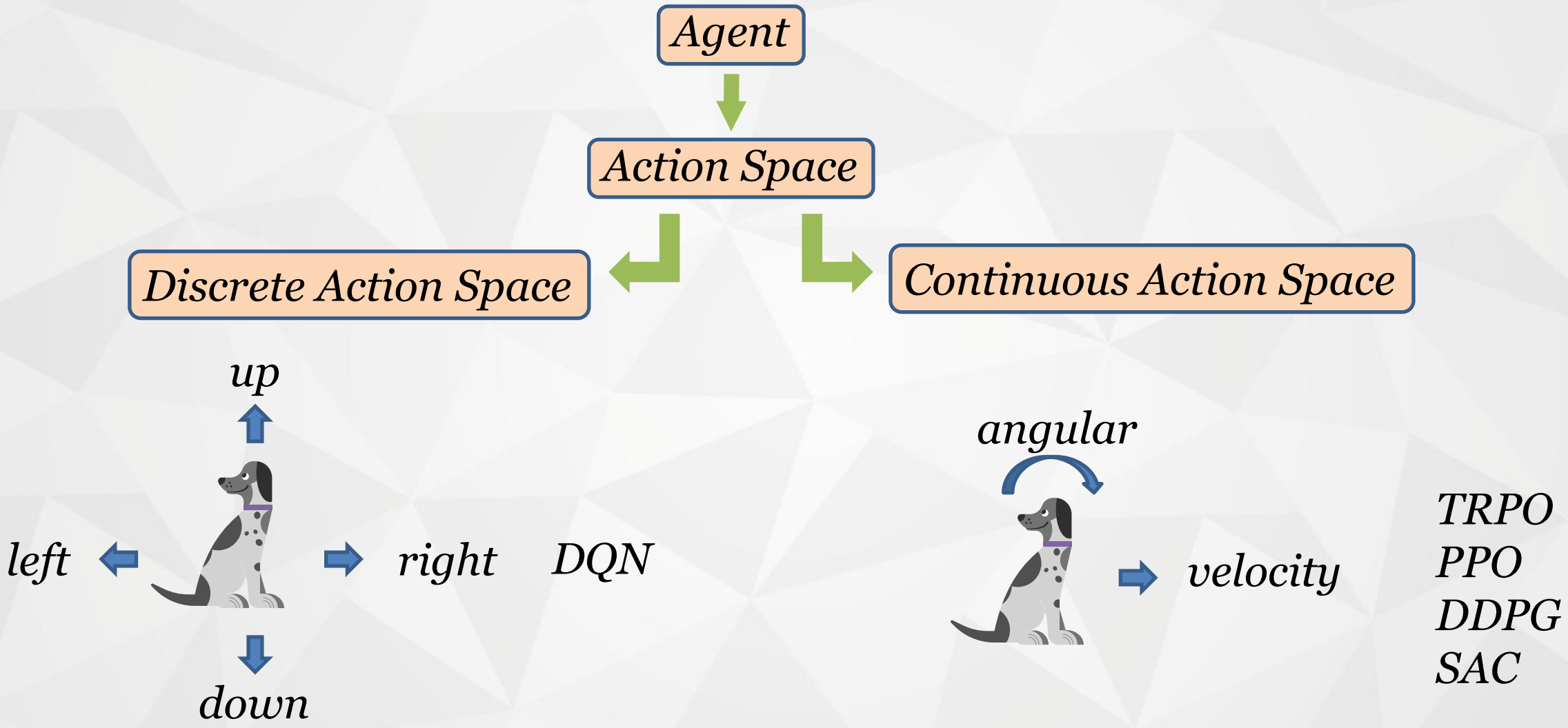
We need to let the prey move from the starting point to the target point, in between try to bypass the obstacles and avoid the hunter's pursuit.



For such problems we generally use reinforcemen t learning to solve them by designing appropriate reward functions to act on the observations and maximize the reward.

Model Selection

Model Selection



Model Selection

Algorithm 1 DDPG algorithm

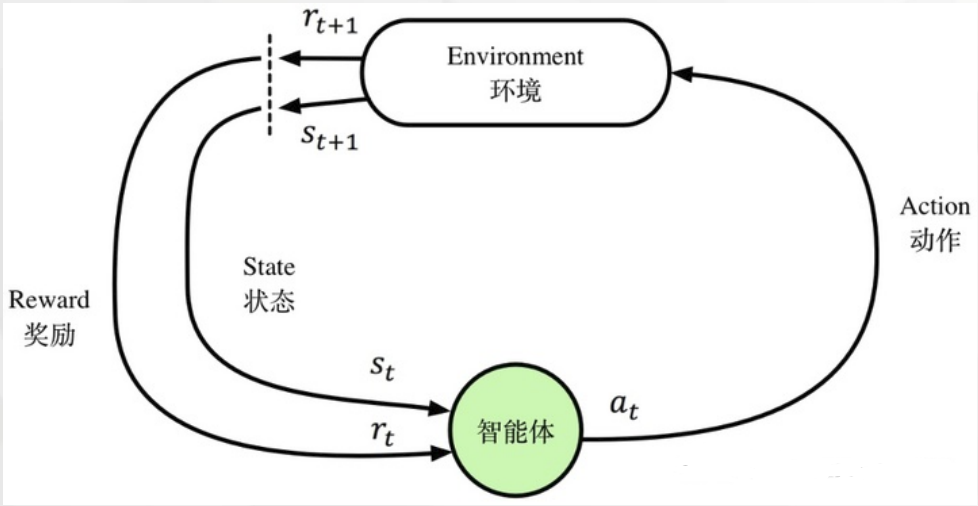
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

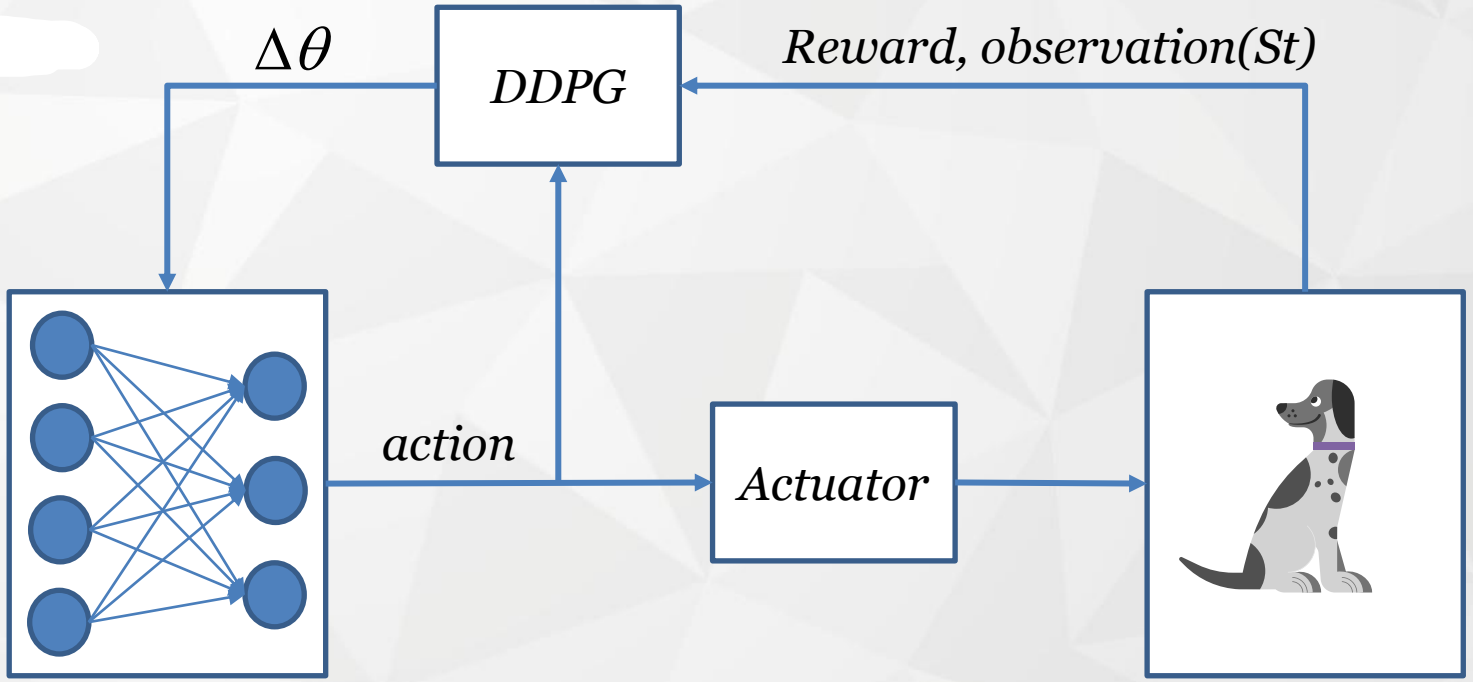
    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

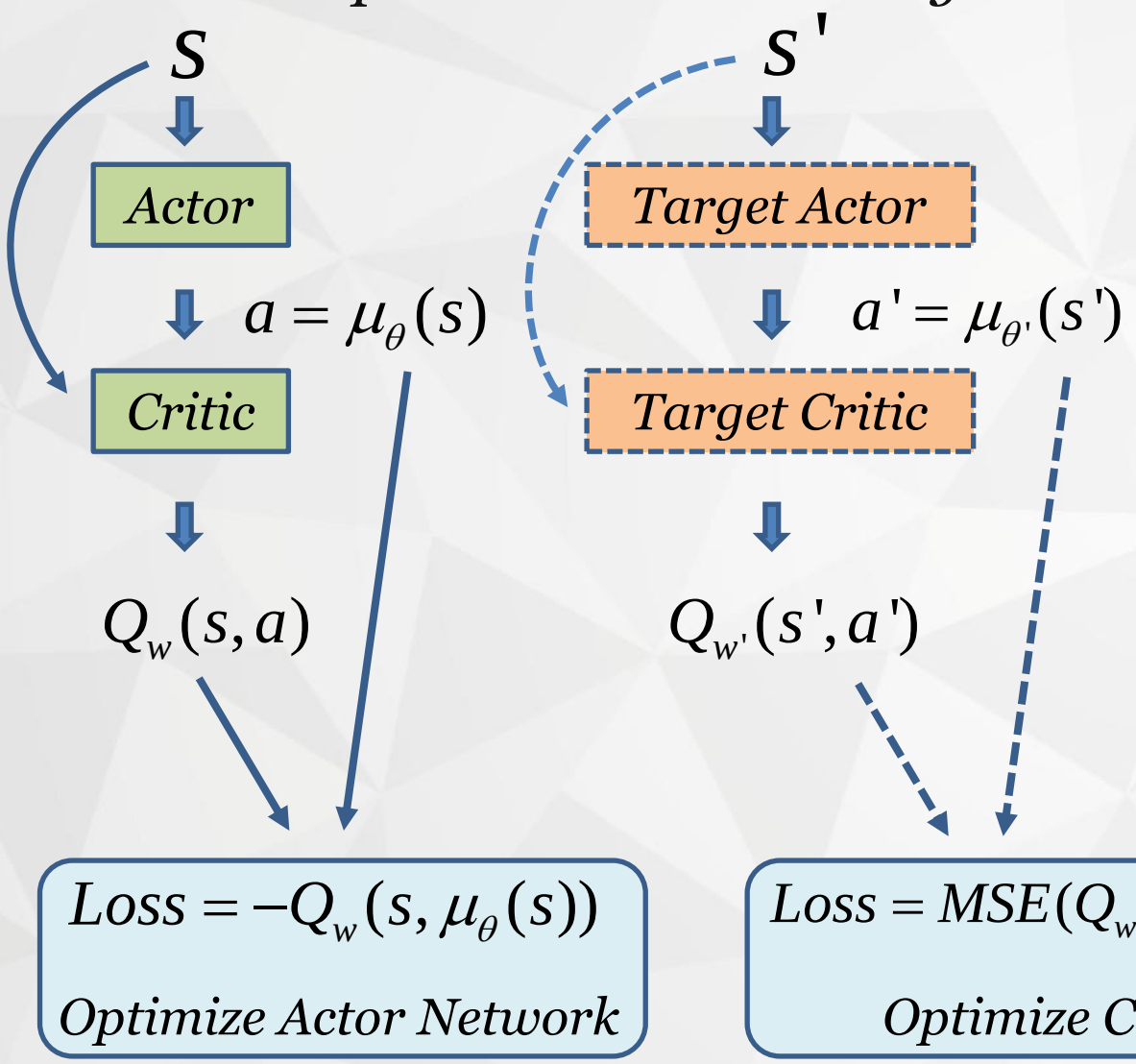


Agent: Network
Environment: Simulation environment and robots
State: Angular velocity, Linear velocity, etc.
Action: output of network
Reward: feedback of the environment
Policy: params of network



Model Selection

DDPG--Deep Deterministic Policy Gradient



Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

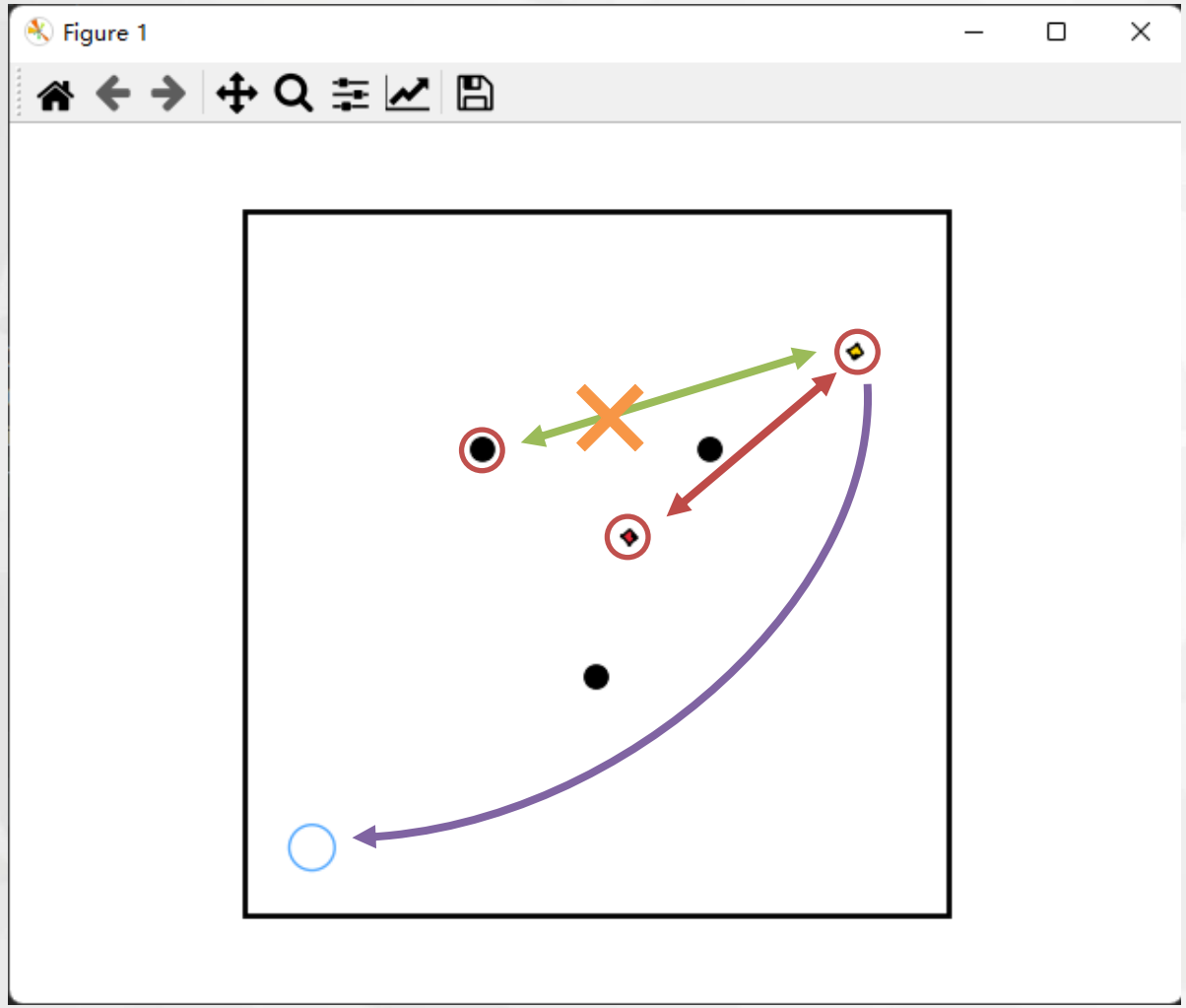
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

WTROBOT
南工问天

Coding Implementation

1.Design Reward and Observation



Observation:

- *Position of Prey*
- *Position of Hunter*
- *Distance of Destination*

Reward:

- *Distance between Prey and Hunter*
- *Distance between Prey and Destination*
- *When the prey and obstacles collide, deduct a certain number of points*

2. State Normalization

The core of state normalization is to maintain a dynamic mean and std of all the states experienced during the interaction with the environment, and then to do normalization on the current acquired states. After normalization, the states conform to a normal distribution with mean=0,std=1. Using such states as input to the neural network is more beneficial to the training of the neural network.

3. Reward Normalization or Reward Scaling

The processing of the rewards is currently done in two ways: rewards normalization and rewards scaling: both of them aim to adjust the scale of the rewards to avoid negative effects on the training of the value function due to too large or too small rewards.

4. Learning Rate Decay

Learning rate decay can enhance the smoothness of the later stages of training to some extent and improve the training effect. Here we use a linear decay of the learning rate so that learning rate decreases linearly from the initial 3e-4, with the number of training steps, to 0.

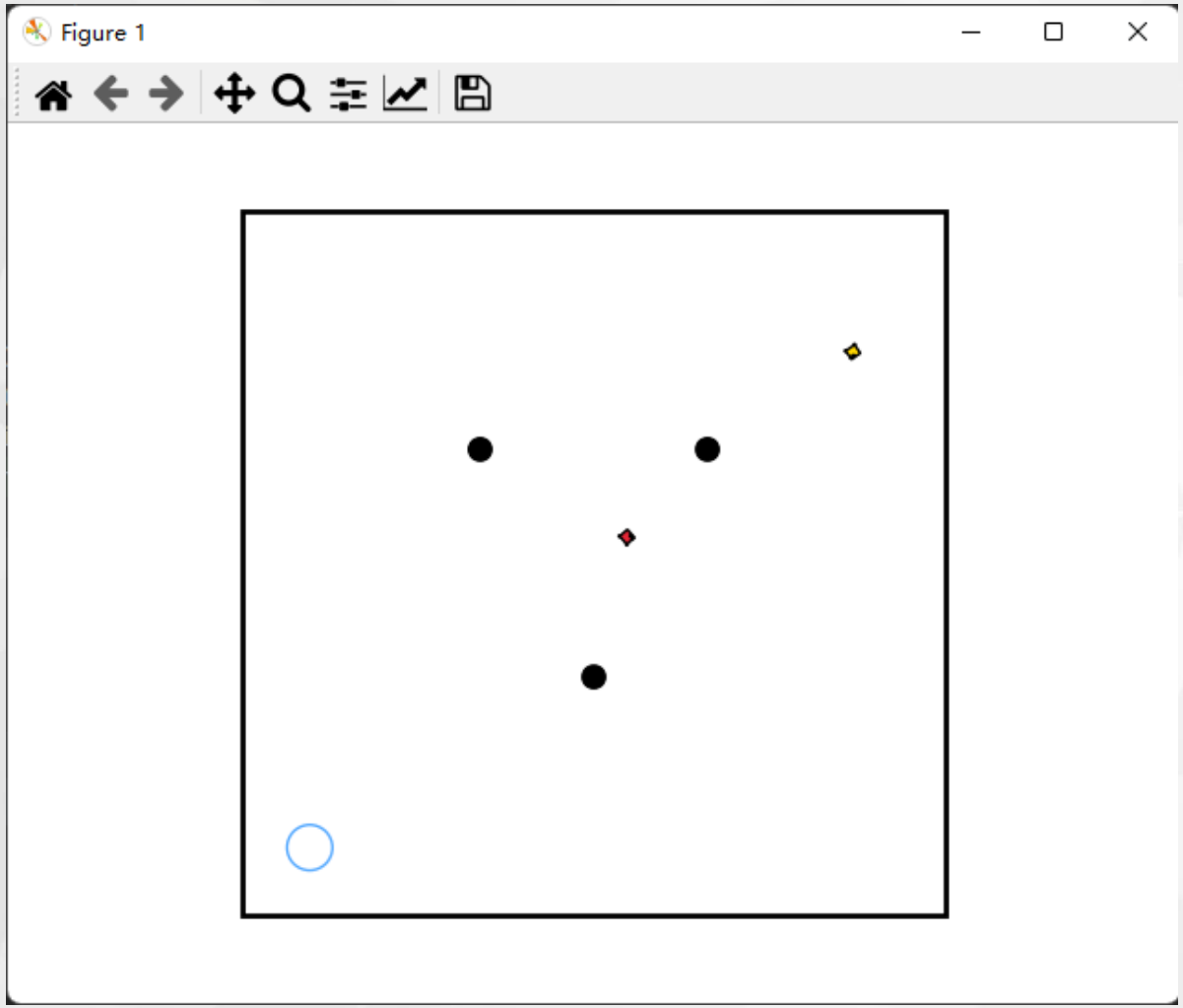
5. Gradient Clip

Gradient clipping is a trick introduced to prevent gradient explosion during training, which also serves to stabilize the training process.

6. Adam Optimizer

7. Tanh Activation Function

Coding Implementation



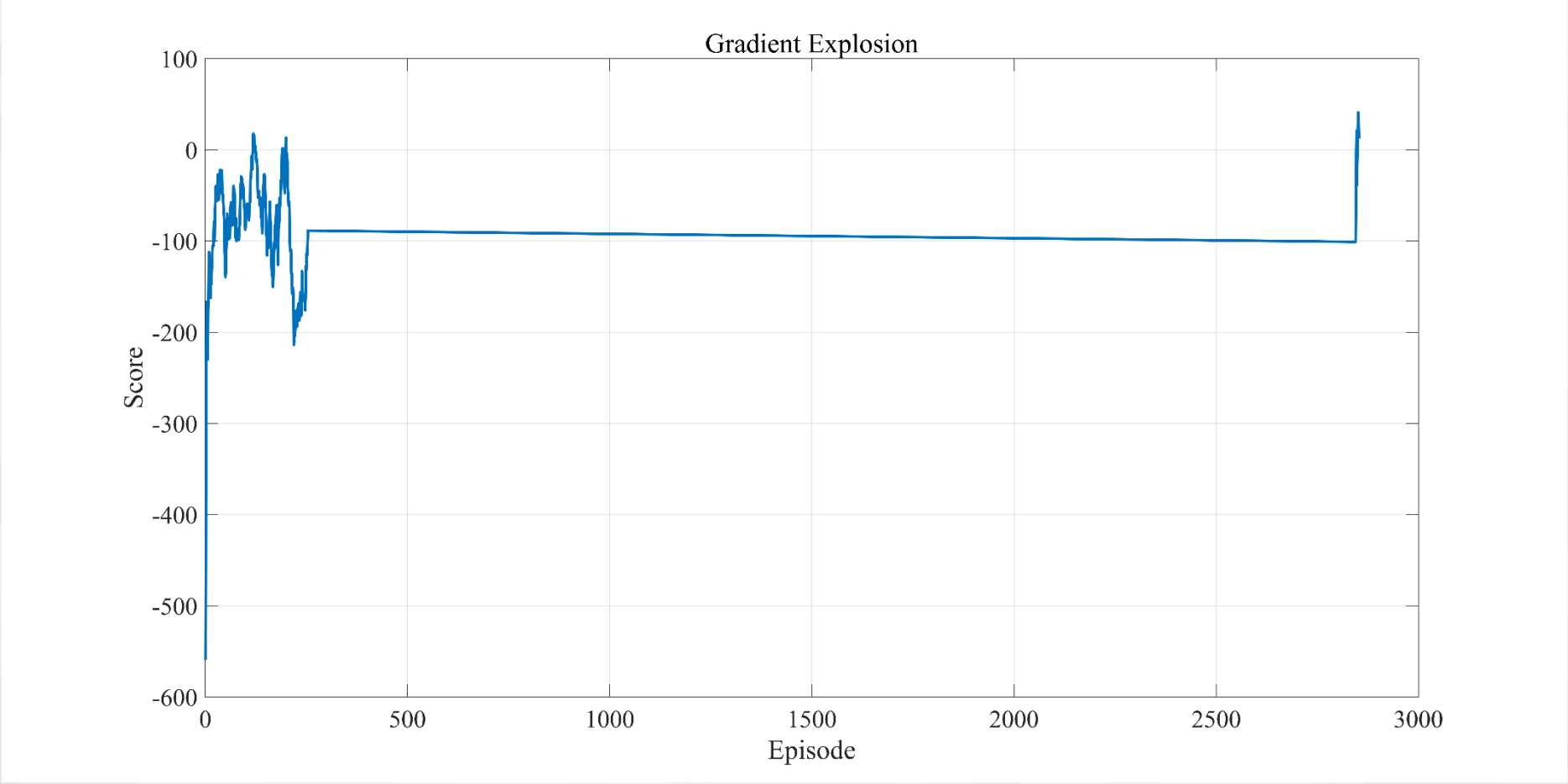
In order to ensure that each turn the prey can be fully explored, when the prey and the hunter is very close to the end of the turn does not end, but to deduct the reward instead, and set the maximum step of each turn for 1500

WTROBOT
南工问天

Final Test

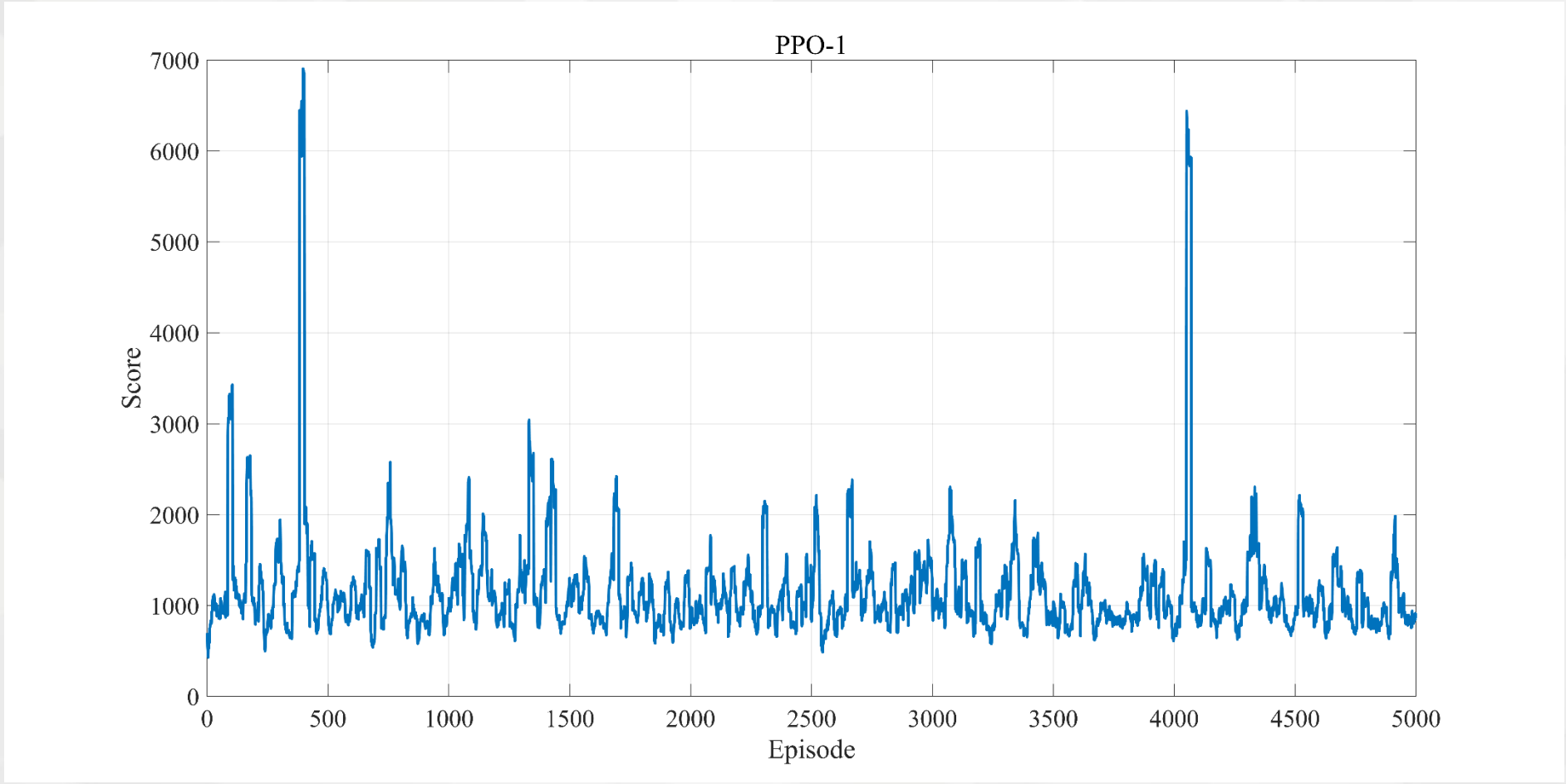
Final Test

1.Failed:Because of Gradient Explosion



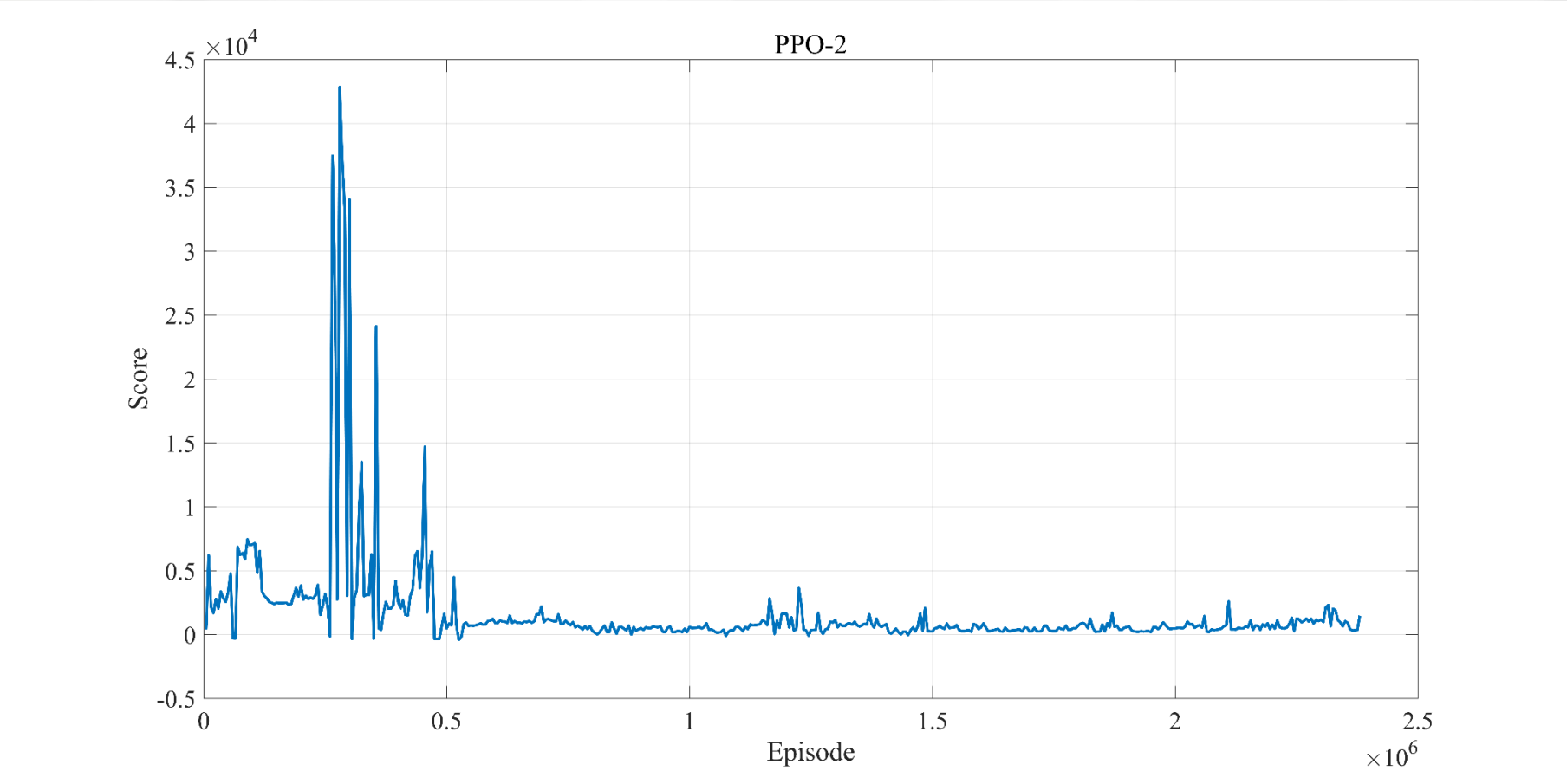
Final Test

2.PPO Failed: The Network didn't converge



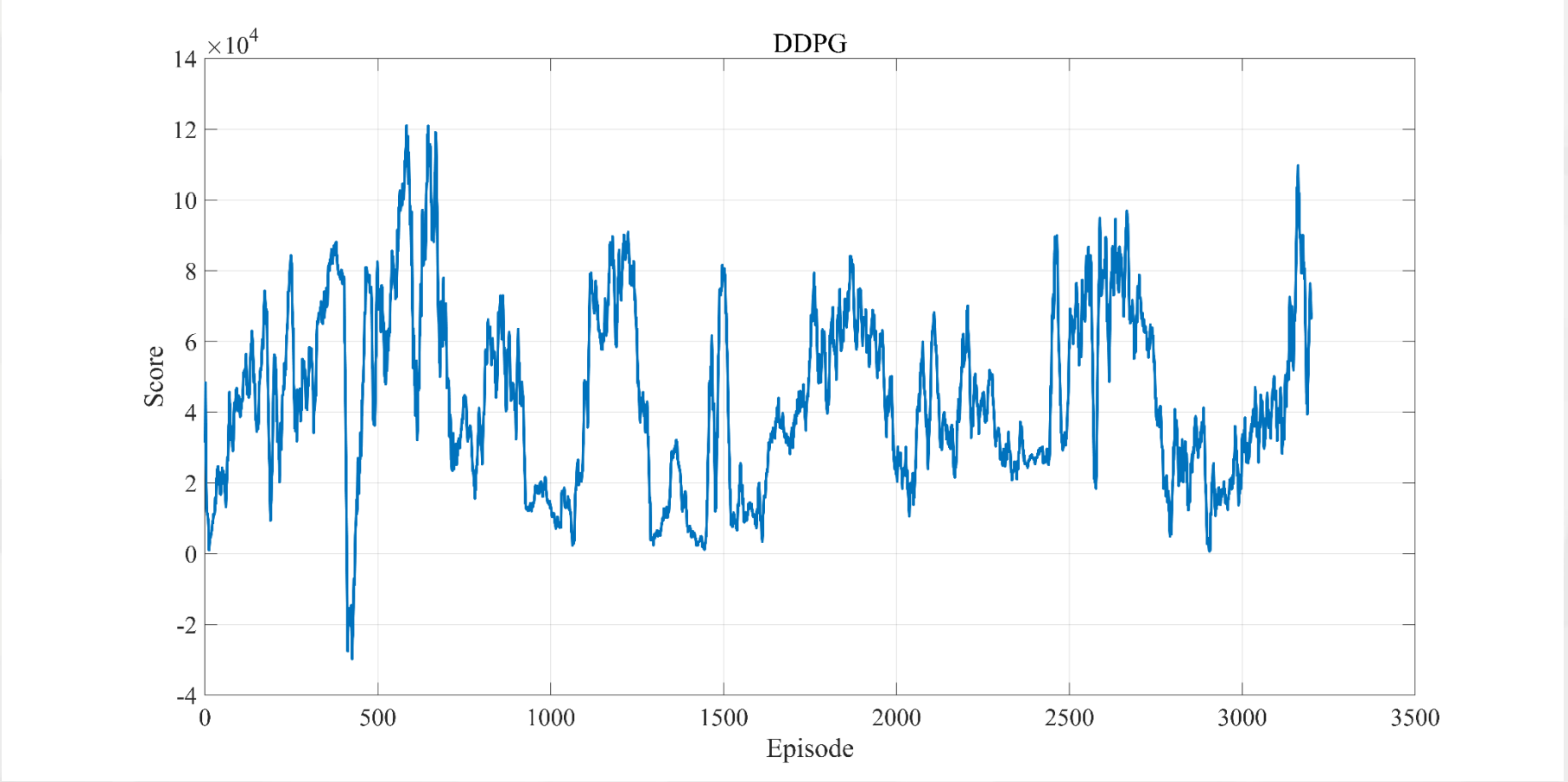
Final Test

3.PPO Failed Again: The Network became Even Worse



Final Test

2.DDPG has Succeeded: Good performance in the middle stage



Thanks