Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 1117781, 10 pages
https://doi.org/10.1155/2022/1117781

Research Article

Network Architecture for Optimizing Deep Deterministic Policy Gradient Algorithms

Haifei Zhang,[1] **Jian Xu**,[2] **Jian Zhang**,[3] **and Quan Liu**[3]

[1]*School of Computer and Information Engineering, Nantong Institute of Technology, Yongxing Road 211, Nantong 226002, China*
[2]*School of Information Science and Technology, Nantong University, Seyuan Road 9, Nantong 226019, China*
[3]*School of Computer Science and Technology, Soochow University, Shizi Street 1, Suzhou 215006, China*

Correspondence should be addressed to Haifei Zhang; 46462490@qq.com

Received 7 September 2022; Revised 17 September 2022; Accepted 27 September 2022; Published 18 November 2022

Academic Editor: D. Plewczynski

The traditional Deep Deterministic Policy Gradient (DDPG) algorithm has been widely used in continuous action spaces, but it still suffers from the problems of easily falling into local optima and large error fluctuations. Aiming at these deficiencies, this paper proposes a dual-actor-dual-critic DDPG algorithm (DN-DDPG). First, on the basis of the original actor-critic network architecture of the algorithm, a critic network is added to assist the training, and the smallest Q value of the two critic networks is taken as the estimated value of the action in each update. Reduce the probability of local optimal phenomenon; then, introduce the idea of dual-actor network to alleviate the underestimation of value generated by dual-evaluator network, and select the action with the greatest value in the two-actor networks to update to stabilize the training of the algorithm process. Finally, the improved method is validated on four continuous action tasks provided by MuJoCo, and the results show that the improved method can reduce the fluctuation range of error and improve the cumulative return compared with the classical algorithm.

1. Introduction

As artificial intelligence continues to thrive, reinforcement learning (RL), which is a learning process that combines exploration and action, has been well developed in discrete action spaces focusing on decision control. By letting the agents learn continuously in a way of trial and error, RL pursues the overall maximum return while seeking the optimal action policy [1, 2]. However, when high-dimensional inputs or continuous action tasks are involved, traditional RL that relies on maximizing expected returns by performing trial and error may not work well. To tackle these kinds of problems, the concept of deep reinforcement learning (DRL) has been presented. In 2013, DeepMind proposed a method of using deep neural networks to play Atari games. It is the first successful and versatile DRL algorithm, although its scope of application is still limited to low-dimensional discrete action spaces. The topics dealing with continuous action tasks have become a new set of research interests [3, 4].

The basic idea of deep reinforcement learning is to fit the value function and policy function in reinforcement learning through a neural network. Typical algorithms include Deep Q-Network (DQN) [5] based on discrete action tasks and Deep Deterministic Policy Gradients (DDPG) [6] based on continuous action tasks. DDPG and DQN have very high similarities in algorithms. The main difference is that DDPG introduces a policy network to output continuous action values. DDPG can be understood as an extension algorithm of DQN in continuous action. DDPG algorithm has been studied extensively with a series of outcomes obtained. Mnih et al. [7] proposed the concept of two-layer BP neural network and hence improved the DDPG algorithm. The search efficiency of BP network was improved by using Armijo-Goldstein-based criterion and BFGS method [8]. Nikishin et al. [9] reduced the influence of noise on the gradient by averaging methods under the premise of random weights. Parallel actor networks and prioritized experience replay are used and tested in the continuous action space of bipedal robots [10]. The experimental results show that the revised algorithm can effectively improve the training speed. In addition, the storage structure of experience in DDPG is optimized, which improves the convergence speed of the DDPG algorithm through binary tree [11–13].

To sum up, the above methods propose improvements to address the shortcomings of DDPG, and all have achieved good results. Although the performance of the improved algorithms has been significantly improved, the flaws of local optimal solutions and large error fluctuations need to be further addressed.

The main content of this paper is as follows: Firstly, the basic principle of DDPG is introduced, and then, combined with the description of the network structure and its associated parameters, the existing shortcomings are also analyzed. Secondly, an improved algorithm is proposed to tackle the shortcomings of DDPG. The improvement method is mainly divided into two aspects. First, in order to reduce the probability of local optimal solutions, a critic network is added to assist training, and the smallest Q value in the two critic networks is taken as the estimated value of the action. Second, the dual-critic network will select the suboptimal Q value to update each round, and the suboptimal Q value also corresponds to the suboptimal action, which leads to the continuous underestimation of the action value of the agent. In response to this problem, this work introduces a dual-actor network based on the dual-critic network architecture; that is, the most valuable action in the two action networks is selected for training under the minimum Q value, so as to improve the robustness of the network structure. Finally, the effectiveness of the improved method is verified in eight simulated, experimental environments.

The rest of this paper is organized as follows: The basics of DDPG are introduced in Section 2. In Section 3, the idea of improving the algorithm is elaborated. Section 4 includes experimental results and analysis. Section 5 summarizes the work and refers to the future works.

2. Deep Deterministic Policy Gradients

The problem that reinforcement learning needs to solve is how to let the agent learn what actions to take in an environment, so as to obtain the maximum sum of reward values [12–14]. The reward value is generally associated with the task goal defined by the agent. The DDPG algorithm is used to solve the reinforcement learning problem in continuous action space [6, 15–17]. The main process is as follows: Firstly, the experience data generated by the interaction between the agent and the environment is stored in the experience recall mechanism. Secondly, the sampled data is learned and updated through the actor-critic architecture, and finally the optimal policy is obtained. The structure of the DDPG algorithm is shown in Figure 1 [15].

Based on the deterministic policy gradient, the DDPG algorithm uses a neural network to simulate the policy function and the Q function and combines the deep learning method to complete the task training [16]. The DDPG algorithm continues with the organizational structure of the DQN algorithm and uses actor-critic as the basic architecture of the algorithm [17]. The combination of the concepts of the online network and the target network in the DQN algorithm with the actor-critic method makes both actor and critic modules in the DDPG have access to the structure of the online network and the target network [6, 18, 19].

During the training process, the agent in the current state S decides the action A that needs to be performed through the current actor network and then calculates the Q value of the current action and the expected return value $y_i = R + \gamma Q'$ according to the current critic network. Then, the actor target network selects the optimal action A' among the actions that can be performed according to the previous learning experience, and the Q' value of the future action is calculated by the critic target network. The parameters of the target network are periodically updated by the online network parameters of the corresponding module.

DDPG adopts a "soft" method to update the target network parameters; that is, the magnitude of each update of the network parameters is very small, which improves the stability of the training process [20–22]. The update coefficient is denoted as τ, then the "soft" update method can be expressed as

$$w' = \tau w + (1-\tau)w'\theta' = \tau\theta + (1-\tau)\theta'. \quad (1)$$

DDPG makes the decision of using action a_t by the deterministic policy π. It approximates the state-action function via a value network, with the definition of the target function as the accumulated reward with a discounted factor [23, 24] as shown in the following equation:

$$J(\theta) = E_\theta\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots\right]. \quad (2)$$

In the online network of the critic, the update of the network parameters is based on the minimal value of the mean square error of the loss function [10], which can be expressed as

$$J(w) = \frac{1}{\mathrm{m}} \sum_{j=1}^{m} \left(y_{\mathrm{j}} - Q\left(\varnothing\left(S_{\mathrm{j}}\right), A_{\mathrm{j}}, w\right)\right)^2. \quad (3)$$

For the actor online network, the network parameters are updated according to the loss gradients of the policy [10] as shown in the following equation:.

$$\nabla_J(\theta) = \frac{1}{m} \sum_{j=1}^{m} \left[\nabla_a Q\left(s_j, a_j, w\right) \nabla_{\theta_\pi}\right]. \quad (4)$$

3. The DDPG Based on Dual-Actors and Dual-Critics

3.1. Error Analysis. It is an inevitable problem for Q-Learning to tend to overestimate errors [25–28]. In Q-Learning, the update of the estimated value of an action by the learning algorithm is conducted by the ε-greedy policy $y_t = r + \gamma \max\left(Q\left(s_{t+1}, a_{t+1}\right)\right)$, hence the actual maximal value of an action is usually smaller than the estimated

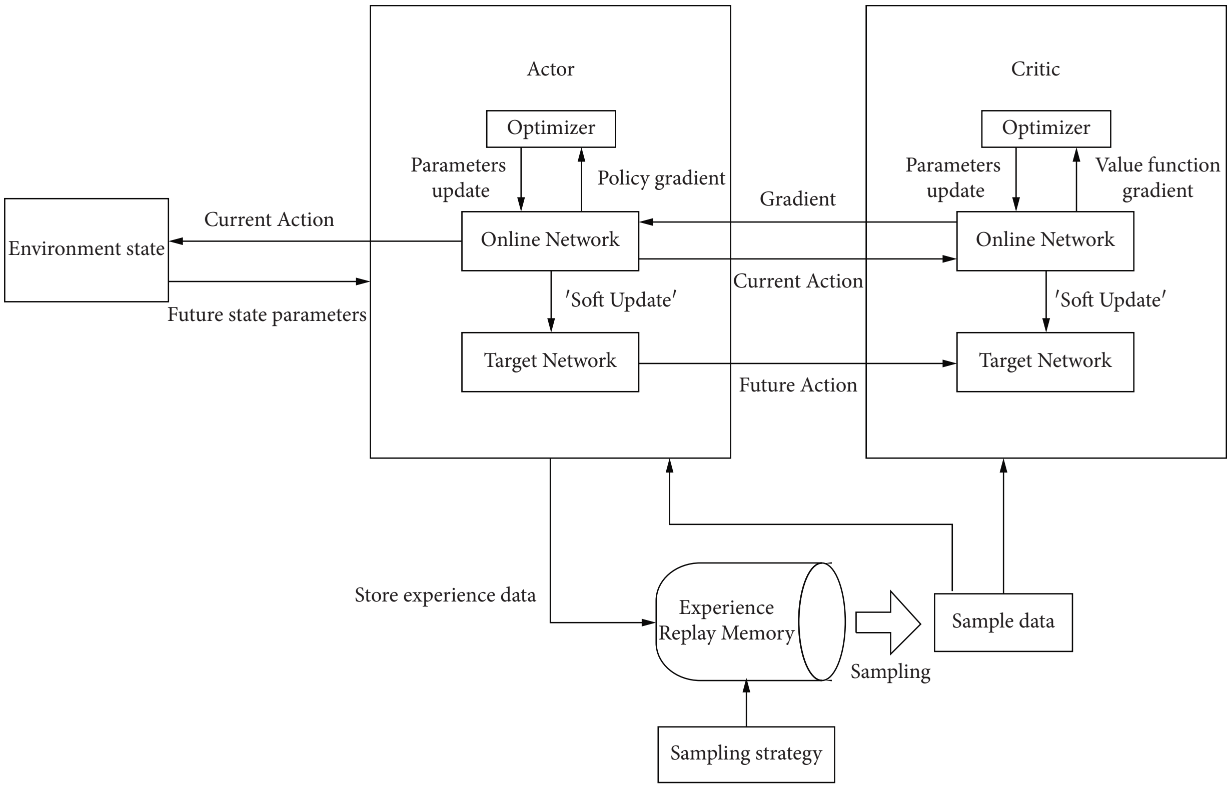


FIGURE 1: The structure of DDPG algorithm.

maximal value of this action as shown in the following equation:

$$\mathbb{E}_{\epsilon}\left[\max_{a'}\left(Q\left(S',a'\right)+\epsilon\right)\right]\geq\max_{a'}Q\left(S',a'\right). \tag{5}$$

Equation (5) has already been proved for its establishment [29, 30]. Even the zero mean error of the initial state will lead to an overestimation of the action value due to the update of the value function, and the adverse effect of this error will be gradually enlarged by the calculation of the Bellman equation.

In the structure of actor-critic, the update of the actor policy depends on the critic value function [31–33]. Given the online network parameter φ, ϕ_{approx} denotes the updated parameter of the actor network calculated by the estimated maximal value function max $(Q_\theta(s,a))$, ϕ_{true} denotes the parameter obtained by using the actual value function $Q_\pi(s,a)$, where $Q_\pi(s,a)$ is unknown in the training process which represents the value function in an ideal state, then ϕ_{approx} and ϕ_{true} can be expressed in the following equation:

$$\begin{aligned}\phi_{\text{approx}}&=\phi+\frac{\alpha}{Z_1}\mathbb{E}_{s\sim p_\pi}\left[\nabla_\phi\pi_\phi(s)\nabla_a Q_\theta s,a\,|_{a=\pi_\phi(s)}\right],\\ \phi_{\text{true}}&=\phi+\frac{\alpha}{Z_2}\mathbb{E}_{s\sim p_\pi}\left[\nabla_\phi\pi_\phi(s)\nabla_a Q^\pi s,a\,|_{a=\pi_\phi(s)}\right].\end{aligned} \tag{6}$$

In Equation (6), $Z_{1,2}^{-1}\|\mathbb{E}[\cdot]\|=1$, which normalizes gradients by using Z_1 and Z_2. Otherwise, highly estimated errors would have been a certain case in a strict constraint if gradient normalization had not been used [34, 35].

Since the gradient is updated in the direction of the local maximum, there is a very small number $k1$, so that when the learning rate of the neural network is less than $k1$. The parameter π_{approx} based on ϕ_{approx} and the parameter π_{true} based on ϕ_{true} converge to the local optimal value of the corresponding Q function, at this time, the estimation of π_{approx} is restricted to be below π_{true} as shown in the following equation:

$$\mathbb{E}\left[Q_\theta\left(s,\pi_{\text{approx}}(s)\right)\right]\geq\mathbb{E}\left[Q_\theta\left(s,\pi_{\text{true}}(s)\right)\right]. \tag{7}$$

On the contrary, there is an extremely small number $k2$, so that when the learning rate of the neural network is less than $k2$, the parameter π_{approx} and the parameter π_{true} also converge to the local optimal value of the corresponding Q function, and the estimation of π_{true} is limited below π_{approx}.

$$\mathbb{E}\left[Q_\theta\left(s,\pi_{\text{true}}(s)\right)\right]\geq\mathbb{E}\left[Q_\pi\left(s,\pi_{\text{approx}}(s)\right)\right]. \tag{8}$$

If the training effect of the critic network is satisfying, the estimation of the policy value will be at least similar to the actual value of φ_{true} as shown in the following equation:

$$\mathbb{E}\left[Q_\theta\left(s,\pi_{\text{true}}(s)\right)\right]\geq\mathbb{E}\left[Q_\pi\left(s,\pi_{\text{true}}(s)\right)\right]. \tag{9}$$

At this time, if the learning rate of the network is smaller than the smaller one of $k1$ and $k2$, we know by combining Equations (8) and (9), the action value will be overestimated as shown in the following equation:

$$\mathbb{E}\left[Q_\theta\left(s,\pi_{\text{approx}}(s)\right)\right]\geq\mathbb{E}\left[Q_\pi\left(s,\pi_{\text{approx}}(s)\right)\right]. \tag{10}$$

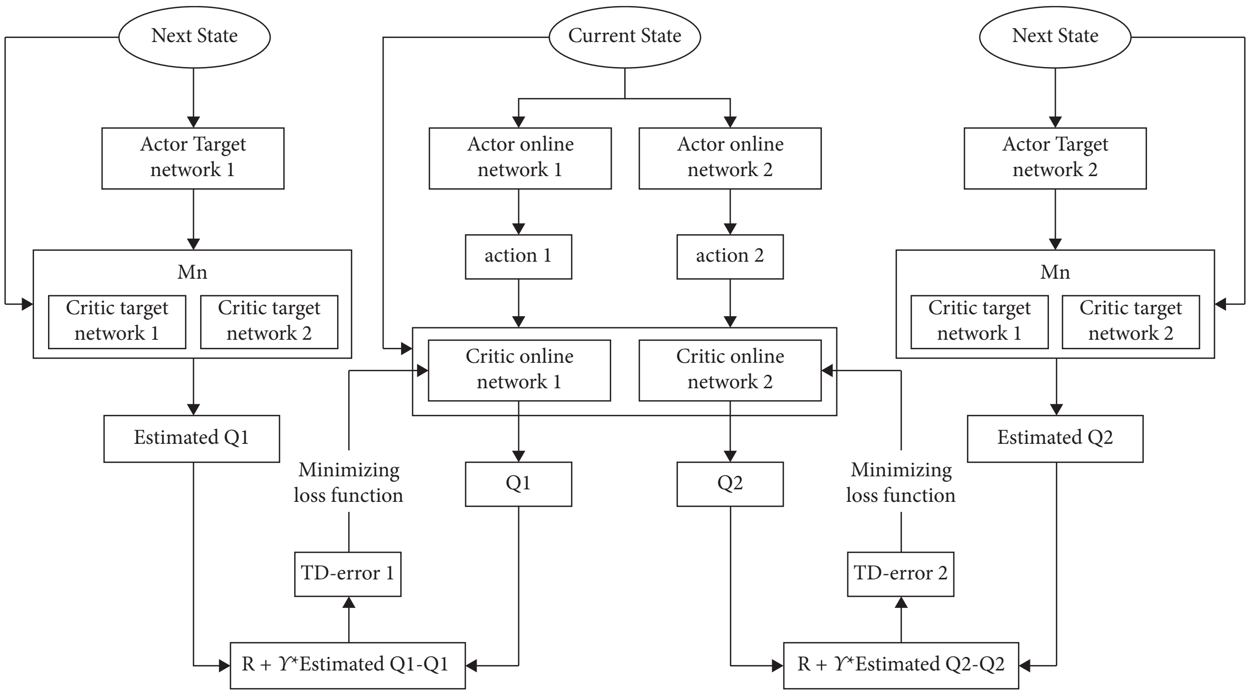


FIGURE 2: Architecture of dual-actors and dual-critics.

The existence of errors will lead to inaccurate estimation of the action value, making the suboptimal policy be taken as the optimal policy output of the online network, thereby affecting the performance of the algorithm.

3.2. Dual-Actors and Dual-Critics Network Structure. Due to the existence of the overestimated error, the estimation of the value function can be used as an approximate upper limit of the estimated value of the future state. If there is a certain error in every Q value update, the accumulation of errors will result in a suboptimal policy. Aiming at this kind of problem, an additional critic network is used in this work. The smallest Q value of the two networks is taken as the estimated value of the action in each update, so as to reduce the adverse effect of the overestimated error.

The process of obtaining the smallest Q value via the dual-critic network is shown in the following equation:

$$y_1 = r_{t+1} + \gamma Q_1'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right)\right) y_2 = r_{t+1} + \gamma Q_2'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right)\right) y = \min\left(y_1, y_2\right). \tag{11}$$

Although the dual-critic network can reduce the overestimated error of the algorithm and reduce the probability of generating a local optimal strategy, in the actual training process, it is rare for the learning rate of the neural network to be less than the minimum value of $k1$ and $k2$. Combined with Section 3.1 analysis, that is, the probability of overestimation is very low. The dual-critic network will select the suboptimal Q value to update in each round. The suboptimal Q value also corresponds to the suboptimal action, which leads to the continuous underestimation of the action value of the agent, and in turn affects the rate of convergence of the critic network [36–38].

Aiming at the problem of underestimation of the dual-critic network, in this work a dual-actor network is presented for training on the basis of the dual-critic network architecture. The network selects the action with the highest value among the two actions under the minimum Q value, which is used to reduce the influence of the Q value underestimation and improve the robustness of the network structure.

The network structure of the dual-actors and dual-critics is shown in Figure 2.

For a two-actor network, the training of this network is subject to the same issues upon the use of the same sample data and processing methods. In order to ameliorate this kind of problems, the update of the parameters of the two-actor network is based on different policy gradients, which helps to reduce the coupling between the two-actors and further improves the convergence rate of the algorithm [39, 40].

If the policies of the two-actors are defined as π_1 and π_2, and the parameters of the dual-critic network are θ_1 and θ_2, we will have two actions $a_1 = \mu(s|\pi_1)$ and $a_2 = \mu(s|\pi_2)$, then we can select the action with the maximal value based on this dual-actor network by using the following equation:

```
     Input: $(\theta_1^Q, \theta_2^\mu, \theta_2^Q, \theta_2^\mu, D, N_t, \gamma)$
     Output: $(\theta_Q', \theta_\mu')$
(1)  Randomly initialize the actor-critic network for their parameters $\theta_1^Q$, $\theta_2^\mu$ and $\theta_2^Q$, $\theta_2^\mu$
(2)  Initialize the target network $Q'$ and $\mu'$, and copy the online network parameters to the target network
(3)  Initialize the experience playback buffer D, noise coefficient $N_t$, and discount rate $\gamma$
(4)  Set up external loop, the round number = 1, M
(5)  Initialize State S as the current state, and obtain the start state $s_1$
(6)  Set up internal loop, the round number = 1, T
(7)  Select action $a_t$: $a_t = \text{argmax}_a [Q_1 (s_t, a_t, \theta_1^\mu), Q_2 (s_t, a_t, \theta_2^\mu)] + N_t$
(8)  Conduct action $a_t$, and obtain the reward $r_t$ and the new state $s_{t+1}$
(9)  Save the experience data $(s_t, a_t, r_t, s_{t+1})$ in an experience pool
(10) Randomly select a certain number of samples $(s_i, a_i, r_i, s_{i+1,})$ from the experience pool
(11) Calculate the target value Q: $y_1 = r_{t+1} + \gamma Q_1' (s_{t+1}, \mu' (s_{t+1} | \theta^{\mu'})) y_2 = r_{t+1} + \gamma Q_2' (s_{t+1}, \mu' (s_{t+1} | \theta^{\mu'})) y = \min (y_1, y_2)$
(12) Calculate the square error of the loss function and update the critic network: $J(w) = 1/m \sum_{j=1}^{m} (y_j - Q(\o (S_j), A_j, w))^2$
(13) Update the actor network via the gradients of the sample data: $\nabla_{\theta^\mu} J \approx 1/N \sum \nabla_a Q(s, a | \theta^Q) |_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu (s | \theta^\mu)|_{s_i}$
(14) Regularly update the parameters of the target network: $\theta_Q' = \tau\theta_Q + (1-\tau)\theta_Q' \overset{i}{\bullet} \theta_\mu' = \tau\theta_\mu + (1-\tau)\theta_\mu'$
(15) End internal loop
(16) End external loop.
```

Algorithm 1: The DN-DDPG process.

$$a = \text{argmax}_a \max [Q_1 (s, \mu (s | \pi_1); \theta_1), Q_2 (s, \mu (s \rfloor \pi_2); \theta_2)]. \quad (12)$$

3.3. Modeling the Algorithm. Combining the ideas proposed in Section 3.2, this paper proposes a dual-actor and dual-critics based DDPG algorithm (DN-DDPG). The process of the DN-DDPG algorithm is shown in Algorithm 1.

4. Experiments

4.1. Software and Hardware Setup. The software environment used in this work is Anaconda3 4.8.3 (Python 3.8), the integrated development environment (IDE) is Pycharm, TensorFlow-GPU 1.8.0 is used as the learning framework. Python virtual environment is run in Anaconda3. NVIDIA GeForce GTX 1650 + CUDA 11.1 is the hardware environment.

4.2. Experimental Setup. In this paper, the Arm environment is written based on the Pyglet module. Two classical controls on the OpenAI GYM [20] platform and four continuous control tasks in the Mujoco physics simulator [21] are used as the experimental environment. OpenAI GYM is an open source toolkit that provides a variety of game environment interfaces to facilitate the research and development of artificial intelligence experiments.

The Arm environment used in this work includes the following items:

(1) Arm_easy. 400 * 400 2-dimensional space is constructed in the Arm environment. One end of a robot arm is fixed in the middle of the environment. The goal of the training is to make the other end of the robot arm find the blue target point as shown in Figure 3.

(2) Arm_hard. This is similar to the Arm_easy environment, the only difference is that the target point is randomly generated in each round.

Two classical, continuous control task used in this work are shown below.

(1) Pendulum. The pendulum starts at a random position, the aim is to push it swing upwards and keep erected.

(2) Mountain Car Continuous. This task is to drive a car to reach the top of a hill; however, the power of the car is not sufficient to drive it directly to reach the top, it needs to rise and drop on the left and right sides repeatedly so that it can accumulate power to reach the top. It is shown in Figure 4.

The 4 Mujoco continuous control tasks include:

(1) Half Cheetah. Train a bipedal agent to learn running as shown in Figure 5.

(2) Hopper. Train a single legged robot to learn jumping forward.

(3) Humanoid. Train a 3-dimensional bipedal agent to learn standing without falling down.

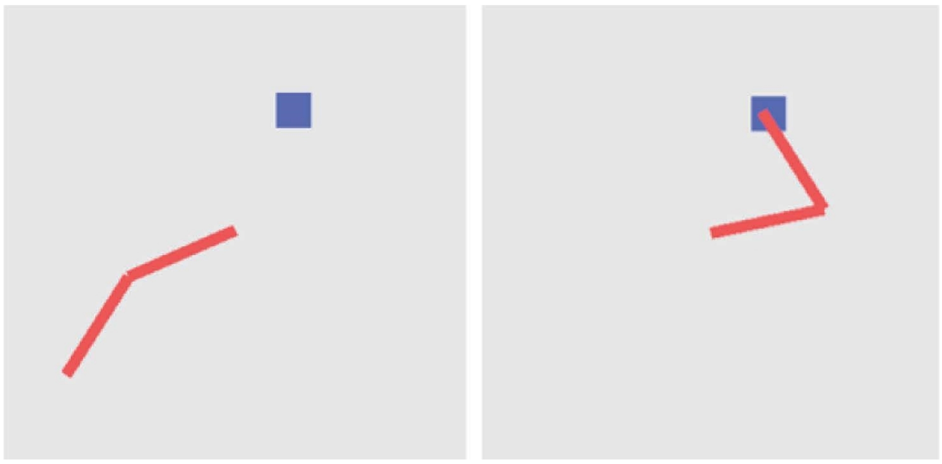


Figure 3: Arm_easy environment and task.

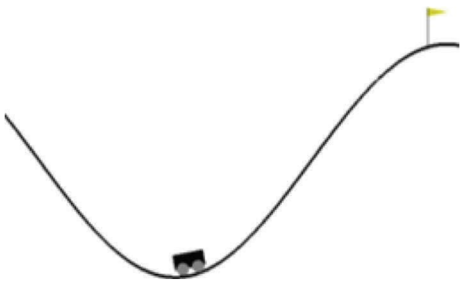


Figure 4: Mountain car continuous.

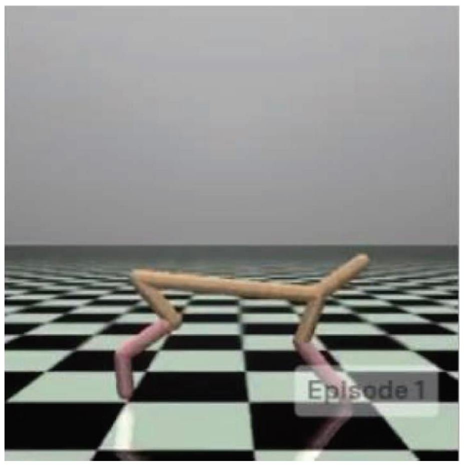


Figure 5: Half cheetah.

(4) Walker2d. Train a 3-dimensional bipedal agent to walk forward as fast as possible.

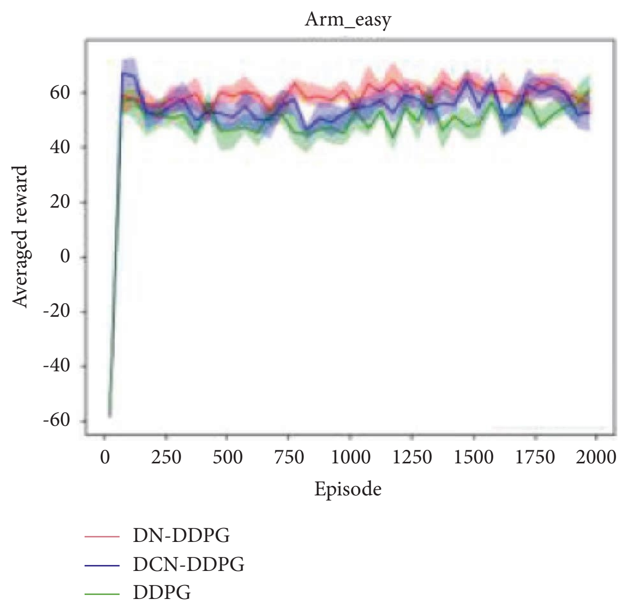
This work compares the performance of DN-DDPG and the original DDPG algorithm. In order to study the improvement effect of the dual-critic network and the dual-actor network, the DCN-DDPG algorithm which is the single-actor and dual-critic network is included for comparison. The outcomes of the comparison are shown intuitively through experiments.

Table 1: Mujoco environment model hyperparameters.

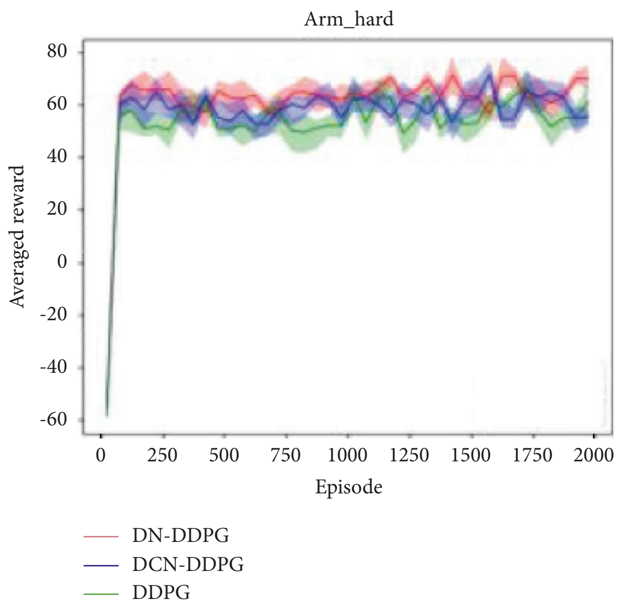
Order	Parameter	Value
1	Decay rate	0.9
2	Actor net learning rate	0.0001
3	Critic net learning rate	0.0001
4	Neuron number in 1st layer	400
5	Neuron number in 2nd layer	300
6	Experience pool volume	100000
7	Batch data size	256
8	Soft update coefficient	0.01
9	Action reward discount rate	0.99
10	Critic net output distribution low limit	−20
11	Target net parameters update round number	10

4.3. Parameter Setting. To ensure the accuracy and fairness of the experimental results, the common parameter values of different algorithms are the same. The training rounds for both the Arm environment and the two Gym classic control tasks are set to 2000 times, and the maximum number of training steps per round is 300 times. The training rounds of 4 kinds of Mujoco continuous control tasks are set to 5000 times, and the number of training steps per round is the maximum number of round steps in the Gym environment. The agent continuously learns and explores in the environment. If the preset task in the environment is successfully completed or the number of training times per round exceeds the maximum number of times, the scene will be reset and a new round will be started. Some parameters in the MuJoCo task are shown in Table 1.

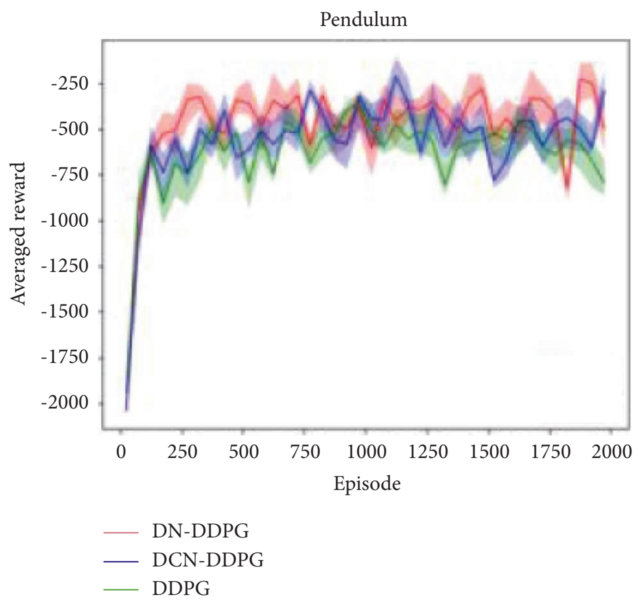
4.4. Experimental Outcomes. In this work, the performances of three algorithms, DN-DDPG, DCN-DDPG and original DDPG, are compared in terms of their performance in two Arm environments, two Gym classical control environments, and four continuous tasks in Mujoco. DN-DDPG and DCN-DDPG are both based on the improvement of the DDPG method, the difference is that DCN-DDPG is based on the original DDPG with addition of an extra critic network, while DN-DDPG is based on the DCN-DDPG with



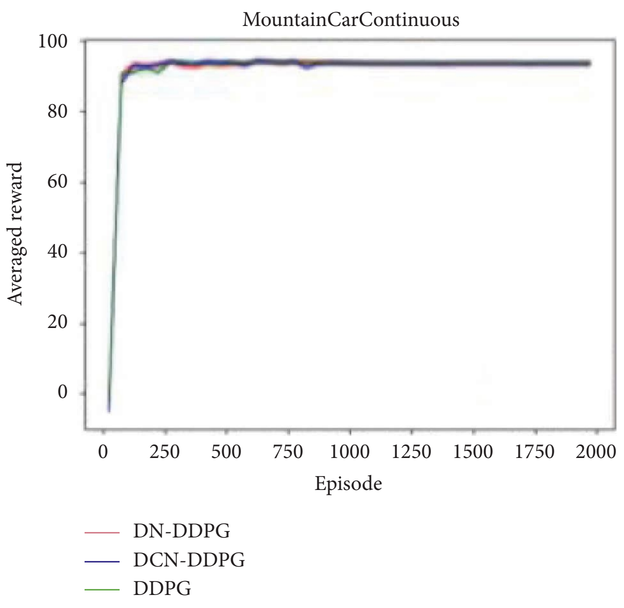
(a)



(b)

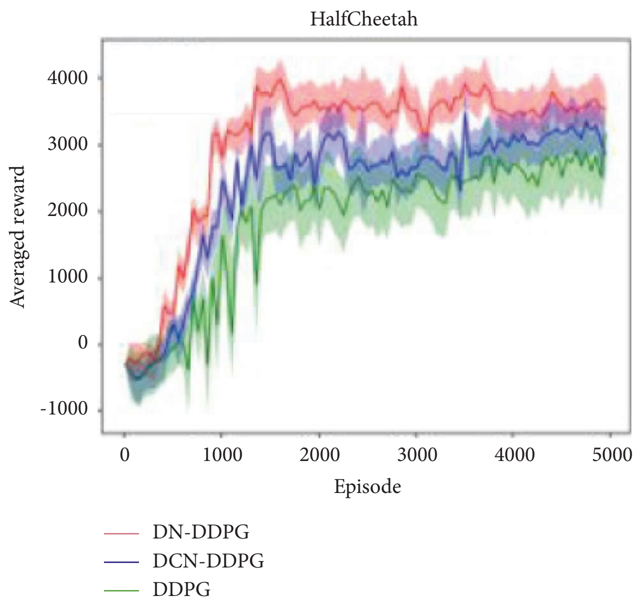


(c)

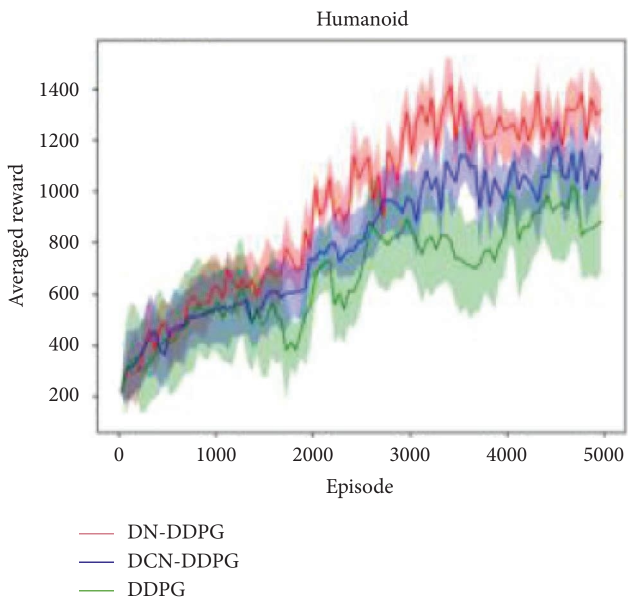


(d)

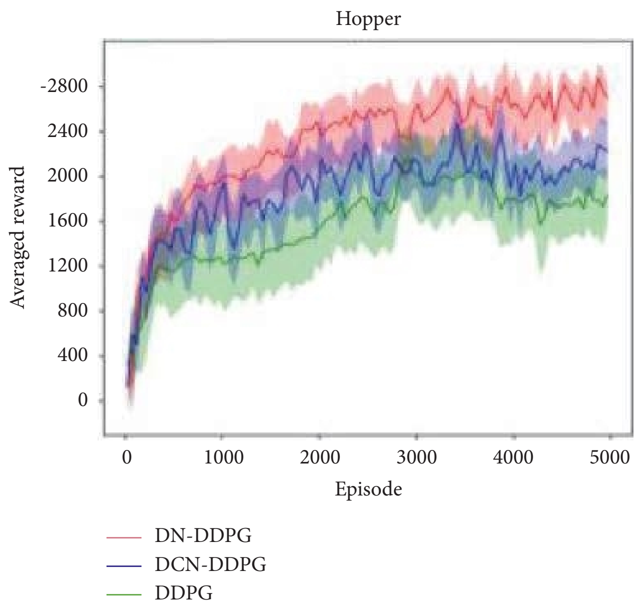
Figure 6: Continued.



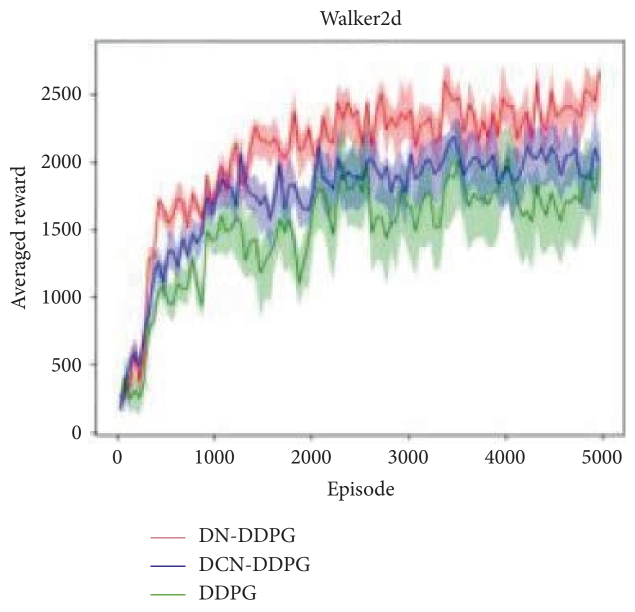
(e)



(f)



(g)



(h)

FIGURE 6: Comparative experiments of three algorithms in eight different continuous action tasks. (a) Arm_easy. (b) Arm_hard. (c) Pendulum. (d) Mountain car continuous. (e) Half cheetah. (f) Humanoid. (g) Hopper. (h) Walker2d.

addition of an extra actor network to optimizing training. The comparison of these three algorithms can make a more intuitive display of the two improved methods mentioned in this article: dual-critics and dual-actors. The experimental results are shown in Figure 6.

The shaded part in the figure represents the standard deviation during training, that is, when using the same hyperparameters and network model, different random number seeds are used to achieve random exploration. The shaded upper limit is the optimal result. The x-axis represents the number of rounds of agent training, the y-axis represents the cumulative reward obtained per round, and the experiment recorded the average reward value per 100 rounds.

In the environments of Arm easy and Arm hard, the average rewards from three algorithms stay around a same value. In some cases, the rewards from both DCN-DDPG and DDPG are superior to that of DN-DDPG. However, from the point of view of overall training effects, DN-DDPG performs better than the other two algorithms, while DCN-DDPG is slightly better than DDPG. In Pendulum experiment, the overall performance of the DN-DDPG is the best, which is due largely to the fact that dual-critics network is able to reduce the error while dual-actors network selects the action of higher value. In cases of Mountain Car Continuous, the average rewards from these three algorithms tend to be the same. However, during the process of 200 time steps, DN-DDPG has a better convergence speed than the rest two algorithms. In addition, in Half Cheetah, Humanoid, Hopper and Walker2d, DN-DDPG has a worse starting performance than DCN-DDPG and DDPG, which could be due to the fact that DCN-DDPG and DDPG have relatively simpler network structure able to deal with complex environment easier than DN-DDPG. The DN-DDPG needs a period for training, and after this initial training period the average reward from DN-DDPG becomes obviously better than the rest two algorithms. Again, the overall performance of DCN-DDPG is better than DDPG. Finally, the shaded areas of different algorithms are compared, with the outcomes that the area of DN-DDPG is smaller than those of DCN-DDPG and DDPG, which reflects that the training of DN-DDPG is more stable.

From the experimental results in Figure 6, the dual-critics method is able to increase the performance of DDPG algorithm, but to a limited extent. By introducing dual-actors method, the DN-DDPG network, based on the DCN-DDPG, is able to further increase the overall performance and training stability of the algorithm. Hence, compared to the original DDPG, the DN-DDPG which is based on dual-actors and dual-critics, has the best increased performance.

5. Conclusion

A deep deterministic policy gradient algorithm is proposed based on a dual-actors and dual-critics network. In order to reduce the overestimated error in the original actor-critic network, a dual-critics target network is introduced into the algorithm, and the minimum action estimate generated by the two networks is selected to update the policy network. In order to alleviate the problem of underestimation caused by the dual-critics network, a dual-actors network is added on the basis of the original network, and the action with the highest value among the two actions generated by the dual-actors network is selected. The experimental results show that, compared with the original DDPG algorithm, and the DDPG algorithm based on the single-actor and two-critics network, the novel DN-DDPG algorithm based on the dual-actors and dual-critics network has a higher cumulative reward and a smaller standard deviation of training.

There is more to be explored in future work. First, in order to improve the optimization ability of the algorithm, more suitable deep learning methods can be explored and applied to neural networks. Second, for the experience replay mechanism in the DDPG algorithm, it is viable to explore whether there is a better method to determine the sample priority to improve the convergence speed during training.

Data Availability

The dataset can be accessed upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest to report regarding the present study.

Acknowledgments

This work was supported by (1) Training Project of Top Scientific Research Talents of Nantong Institute of Technology (XBJRC2021005); (2) the Universities Natural Science Research Projects of Jiangsu Province (17KJB520031, 21KJD210004, 22KJB520032, 22KJD520007, 22KJD520008, and 22KJD520009); and (3) the Science and Technology Planning Project of Nantong City (JC2021132, JCZ21058, JCZ20172, JCZ20151, and JCZ20148).

References

[1] M. Dörpinghaus, R. É, I. Neri, H. Meyr, and F. Jülicher, "An information theoretic analysis of sequential decision-making," in *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 3050–3054, IEEE, Aachen, Germany, September 2017.

[2] Q. Liu, J. W. Zhai, Z.-Z. Zhang, S. Zhong, Q. Zhou, and P. Zhang, "A survey on deep reinforcement learning," *Chinese Journal of Computers*, vol. 41, no. 1, pp. 1–27, 2018.

[3] H. V. Hasselt and M. A. Wiering, "Using continuous action spaces to solve discrete problems," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1149–1156, Atlanta, GA, USA, October 2009.

[4] W. Zhang, Q. Chen, J. Yan, S. Zhang, and J. Xu, "A novel asynchronous deep reinforcement learning model with adaptive early forecasting method and reward incentive mechanism for short-term load forecasting," *Energy*, vol. 236, Article ID 121492, 2021.

[5] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020.

[6] Q. Zhou, "A novel movies recommendation algorithm based on reinforcement learning with DDPG policy," *International Journal of Intelligent Computing and Cybernetics*, vol. 13, no. 1, pp. 67–79, 2020.

[7] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[8] M. Zhang, Y. Zhang, Z. Gao, and X. He, "An improved DDPG and its application based on the double-layer BP neural network," *IEEE Access*, vol. 8, no. 99, pp. 177734–177744, 2020.

[9] E. Nikishin, P. Izmailov, and B. Athiwaratkun, "Improving stability in deep reinforcement learning with weight averaging," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Monterey, USA, July 2018.

[10] X. Wu, S. Liu, and T. Zhang, "Motion control for biped robot via DDPG-based deep reinforcement learning," in *Proceedings of the 2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pp. 40–45, IEEE, Beijing, China, June 2018.

[11] J. Tang, L. Li, and Y. Ai, "Improvement of End-To-End Automatic Driving Algorithm Based on Reinforcement Learning," in *Proceedings of the 2019 Chinese Automation*

Congress (CAC), pp. 5086–5091, IEEE, Hangzhou, China, November 2019.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, Cambridge, MA, USA, 2018.

[13] Y. Zhang, B. Zhao, and D. Liu, "Deterministic policy gradient adaptive dynamic programming for model-free optimal control," *Neurocomputing*, vol. 387, pp. 40–50, 2020.

[14] Y. Chu, J. Fei, and S. Hou, "Adaptive global sliding-mode control for dynamic systems using double hidden layer recurrent neural network structure," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1297–1309, 2020.

[15] H. Zhang, J. Xu, and L. Lei, *A Manipulator Control Method Based on Deep Deterministic Policy Gradient with Parameter Noise*China, 2021.

[16] J. Liu, F. Gao, and X. Luo, "Survey of deep reinforcement learning based on value function and policy gradient," *Chinese Journal of Computers*, vol. 42, pp. 1406–1438, 2019.

[17] E. Mizutani and S. Dreyfus, "Totally model-free actor-critic recurrent neural-network reinforcement learning in non-Markovian domains," *Annals of Operations Research*, vol. 258, no. 1, pp. 107–131, 2017.

[18] Y. Xiang, J. Wen, W. Luo, and G. Xie, "Research on collision-free control and simulation of single-agent based on an improved DDPG algorithm," in *Proceedings of the 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 552–556, IEEE, Zhanjiang, China, October 2020.

[19] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the Fourth Connectionist Models Summer School*, pp. 255–263, Stanford, CA, June 1993.

[20] G. Brockman, V. Cheung, and L. Pettersson, "OpenAI gym. CORR," 2016, https://arxiv.org/abs/1606.01540.

[21] E. Todorov, T. Erez, and Y. T. MuJoCo, "A physics engine for model-based control," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, Vilamoura-Algarve, Portugal, October 2012.

[22] X. Fu, J. Zhu, Z. Wei, H. Wang, and S. Li, "A UAV pursuit-evasion strategy based on DDPG and imitation learning," *International Journal of Aerospace Engineering*, vol. 2022, Article ID 3139610, pp. 1–14, 2022.

[23] L. Li, J. Hang, H. Sun, and L. Wang, "A conjunctive multiple-criteria decision-making approach for cloud service supplier selection of manufacturing enterprise," *Advances in Mechanical Engineering*, vol. 9, no. 3, 2017.

[24] L. h Li, J. c Hang, Y. Gao, and C. Y. Mu, "Using an integrated group decision method based on SVM, TFN-RS-AHP, and TOPSIS-CD for cloud service supplier selection," *Mathematical Problems in Engineering*, vol. 2017, Article ID 3143502, pp. 1–14, 2017.

[25] P. Li, X. Ding, H. Sun, S. Zhao, and R. Cajo, "Research on dynamic path planning of mobile robot based on improved DDPG algorithm," *Mobile Information Systems*, vol. 2021, Article ID 5169460, 12 pages, 2021.

[26] L. Li, B. Lei, and C. Mao, "Digital twin in smart manufacturing," *Journal of Industrial Information Integration*, vol. 26, no. 9, Article ID 100289, 2022.

[27] L. Li, C. Mao, H. Sun, and B. YuanLei, "Digital twin driven green performance evaluation methodology of intelligent manufacturing: hybrid model based on fuzzy rough-sets AHP, multistage weight synthesis, and PROMETHEE II," *Complexity*, vol. 2020, no. 6, Article ID 3853925, p. 1–24, 2020.

[28] Y. Du, X. Zhang, Z. Cao et al., "An optimized path planning method for coastal ships based on improved DDPG and DP," *Journal of Advanced Transportation*, vol. 2021, Article ID 7765130, p. 1–23, 2021.

[29] Z. Yao, Y. Wang, L. Meng, X. Qiu, and P. Yu, "DDPG-based energy-efficient flow scheduling algorithm in software-defined data centers," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6629852p. 1–10, , 2021.

[30] R. Wu, F. Gu, H.-L. Liu, and H. Shi, "UAV path planning based on multicritic-delayed deep deterministic policy gradient," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 9017079, p. 1–12, 2022.

[31] L. Zhang, Z. Pan, Yu Pan et al., "A hidden attack sequences detection method based on dynamic reward deep deterministic policy gradient," *Security and Communication Networks*, vol. 2022, Article ID 1488344, 2022.

[32] Y. Li and L. H. Li, "Enhancing the optimization of the selection of a product service system scheme: a digital twin-driven framework," *Strojniški vestnik - Journal of Mechanical Engineering*, vol. 66, no. 9, pp. 534–543, 2020.

[33] L. H. Li and H. G. Wang, "A VVWBO-BVO-based GM (1, 1) and its parameter optimization by GRA-IGSA integration algorithm for annual power load forecasting," *PLoS One*, vol. 13, no. 5, p. e0196816, May 16 2018.

[34] H. Zhang, J. Xu, and J. Qiu, "An automatic driving control method based on deep deterministic policy gradient," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 7739440, pp. 1–9, 2022.

[35] W. Yuan, Z. Xiwen, Z. Rong, T. Shangqin, Z. Huan, and D. Wei, "Research on UCAV maneuvering decision method based on heuristic reinforcement learning," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 1477078, p. 1–17, 2022.

[36] J. Chen, Y. Wang, J. Ou et al., "ALBRL: automatic load-balancing architecture based on reinforcement learning in software-defined networking," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 3866143, p. 236, 2022.

[37] L. Li, T. Qu, Y. Liu, and C. ZhongXuSunGaoLeiMaoPanWangMa, "Sustainability assessment of intelligent manufacturing supported by digital twin," *IEEE Access*, vol. 8, pp. 174988–175008, 2020.

[38] L. Li and C. Mao, "Big data supported PSS evaluation decision in service-oriented manufacturing," *IEEE Access*, vol. 8, no. 99, pp. 154663–154670, 2020.

[39] Zi-J. Wang, X.-M. Chen, P. Wang, M.-Xi Li, Y.-J.-X. Ou, and H. Zhang, "A Decision-making model for autonomous vehicles at urban intersections based on conflict resolution," *Journal of Advanced Transportation*, vol. 2021, Article ID 8894563, pp. 1–12, 2021.

[40] Xi-L. Chen, L. Cao, Z.-X. Xu, J. Lai, and C.-Xi Li, "A study of continuous maximum entropy deep inverse reinforcement learning," *Mathematical Problems in Engineering*, vol. 2019, Article ID 4834516, 36 pages, 2019.