IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Deep Deterministic Policy Gradient Based on Double Network Prioritized Experience Replay

CHAOHAI KANG[1,2], CHUITING RONG[1,2], WEIJIAN REN[1,2], FENGCAI HUO[1,2,3], and PENGYUN LIU[1]

[1]Colleage of Electrical and Information Engineering, Northeast Petroleum University, Daqing 163318 China (e-mail: kangchaohai@126.com, rong202@139.com, renwj@126.com)

[2]Heilongjiang Provincial Key Laboratory of Networking and Intelligent Control, Northeast Petroleum University, Daqing 163318 China

[3]Bohai-rim Energy Research Institute of NEPU, Qinhuangdao 066004, China

Corresponding author: Weijian Ren (e-mail: renwj@126.com).

This work was supported in part by the key funding project of the Natural Science Foundation of China under Grant 61933007, in part by the Natural Science Foundation of China under Grant 61873058, and in part by the Provincial Science Foundation of Heilongjiang under Grant F2018004 and F2018005.

ABSTRACT The traditional deep deterministic policy gradient (DDPG) algorithm has the disadvantages of slow convergence velocity and ease of falling into the local optimum. From these two perspectives, a DDPG algorithm based on the double network prioritized experience replay mechanism (DNPER-DDPG) is proposed in this paper. Firstly, the value function is approximated by introducing the idea of two neural networks, and the minimum of the action value functions generated by the two networks is selected as the updated value of the actor policy network, which reduces the incidence of local optimal policy. Then, the Q values obtained by the two networks and the immediate reward obtained by the environment are used as the criteria for prioritization, and the importance of the samples in the experience replay mechanism is divided to improve the convergence speed of the algorithm. Finally, the improved method is demonstrated in the classic control environment of OpenAI Gym, and the results show that the proposed method has increased convergence speed and cumulative reward compared with the comparison algorithm.

INDEX TERMS Continuous action space, deep deterministic policy gradient, experience replay mechanism, function approximation error, priority division

I. INTRODUCTION

With the development of the field of artificial intelligence, several achievements in the area of discrete action spaces have been made by reinforcement learning [1][2][3]. In solving discrete action space tasks, exploration algorithms are used to enumerate as many state-actions as possible in some studies. However, when the action space is very large or continuous, it is difficult for algorithms based on discrete action spaces to achieve a good result. At this time, reinforcement learning based on continuous action spaces emerged. How to efficiently handle tasks in the continuous action spaces has become a hot topic and difficult problem in reinforcement learning [4].

At present, the method of dealing with continuous action space tasks is to learn policy directly. The policy is regarded as the function of states and actions. The parameters of a policy function can be obtained by establishing an appropriate loss function and using the reward generated by the interaction between agent and environment. The specific action can be directly generated by the policy function. Under the guidance of such thought, the deep deterministic policy gradient (DDPG) [5] was proposed. Regarding this algorithm, many scholars have conducted in-depth research and have made a series of achievements. An improved DDPG algorithm based on a double-layer back-propagation (BP) neural network was introduced in Ref. [6]. A fuzzy algorithm based on the Armijo-Goldstein criterion and BFGS method were introduced to improve the exploration efficiency of the BP neural network. Experimental results show that the proposed method has greatly improved the BP network's performance. A new sampling idea was proposed in [7], namely that the data in the experience replay mechanism are stored in segments according to the importance of the training data, and the training efficiency and convergence are raised. E. Nikishin et al. [8] proposed an improved reinforcement learning idea. The stochastic weight

averaging method was introduced to reduce the influence of noise in the gradient estimator in training process. It was tested in the continuous action space tasks of Atari and MuJoCo. The stability of the training process is thereby increased. The work presented in [9] shows that high-priority samples are selected for network training. At the same time, similar samples in the replay mechanism are deleted and some rare samples retained. The convergence speed of the method is increased. The method presented in [10] is an improved DDPG algorithm. The parallel actor network was introduced to speed up training efficiency and the prioritized experience replay was introduced to raise sample utilization. Biped robots can be effectively controlled to walk and the training speed was increased through the algorithm. In [11], an improved experience replay sampling method was proposed. The method is based on learning curve theory to achieve the dynamic change effect of the experience replay mechanism. The sample utilization rate and algorithm learning efficiency are improved in the application of robot path planning. Zeshan Li et al. [12] proposed an improved algorithm based on weighted information entropy. Weights are assigned to training data in advance. The data are trained according to the weight ratio and the results are integrated through neural networks. Tested in wireless sensor networks, the training efficiency and convergence of the algorithm were improved. A reinforcement learning method combining prioritized experience replay and DDPG was proposed in [13]. When storing empirical data, the binary tree structure is adopted, and the data sampling training is carried out by using the idea of prioritized experience replay. The algorithm was applied to the automatic driving problem, and the convergence speed of the algorithm improved. To solve the problem of high complexity of the prioritized sampling algorithm, Haoyu Zhang et al. [14] proposed a small sample sorting method. First, a number of times of batch sample data were randomly selected, and then prioritization was carried out on the basis of random samples. Finally, the samples were selected according to priority. Simulation experiments prove that the sampling efficiency of the algorithm was enhanced.

To summarize, the abovementioned reinforcement learning methods applied to the continuous action space problem have made some achievements. The convergence has been raised, but it still needs to be improved. At the same time, the study of local optimum policy still must be advanced further. The main work of this paper is as follows. First, the basic principle of DDPG is introduced, and its network structure and important parameters are elaborated in detail to determine its defects in processing continuous action space tasks. Second, the improved algorithm is proposed to overcome the shortcomings of the DDPG. Two improvements are made. First, to reduce the occurrence of a local optimum, a pair of critic networks are introduced, and the minimum value of the Q value generated by the two critic networks is selected as the target value of updating the parameters of actor network so as to reduce the overestimation bias in network training and realize the solution of optimal policy. Second, to improve the convergence of the algorithm, the priority function of samples in the experience replay mechanism is proposed. The two temporal difference errors (TD-errors) [15] produced by the two critic networks and the sum of the immediate rewards of the samples are set as the priority of samples. The immediate reward of the samples and the absolute opposite value of the TD-errors are linearly related to the importance of the samples. The influence of the two on the importance of the sample can be fully taken into account to achieve the purpose of accelerating the convergence. Finally, the effectiveness of the proposed algorithm in continuous action space tasks is studied. The algorithm is verified by experiments. The rest of this paper is organized as follows. In Section II, the DDPG algorithm is briefly reviewed. The idea of improved algorithm is introduced in Section III. In Section IV, the experiment and result analysis are presented. Section V gives conclusions and describes planned future work.

II. DEEP DETERMINISTIC POLICY GRADIENT ALGORITHM

The DDPG algorithm used to solve the reinforcement learning problem of continuous action space. The flow of the DDPG is as follows: First, the experience data generated by the interaction between the agent and the environment are stored in the experience replay mechanism, and then the sampled data are updated through the actor-critic framework; finally, the optimal policy is obtained. The structure diagram of DDPG is shown in Fig. 1.

Experience replay mechanism: The data used by the algorithm to update the network are collected in the experience replay (ER) [16], which is equivalent to a piece of memory. The experience samples $e_t = (s_t, a_t, r_t, s_{t+1})$ obtained by the agent interacting with the environment are stored in the ER, and then the samples in the ER are sampled to update the parameters of the deep neural network. The production of the ER is mainly due to two reasons. On one hand, because the data for training the neural network must meet the condition of independent and identical distribution, the data samples produced by the interaction between the agent and the environment are not independent and identically distributed. On the other hand, if the sample data are immediately discarded after use, the data cannot be reused, resulting in low learning efficiency of the agent. Because of the existence of ER, the data utilization efficiency and convergence speed of the algorithm are improved.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074535, IEEE Access

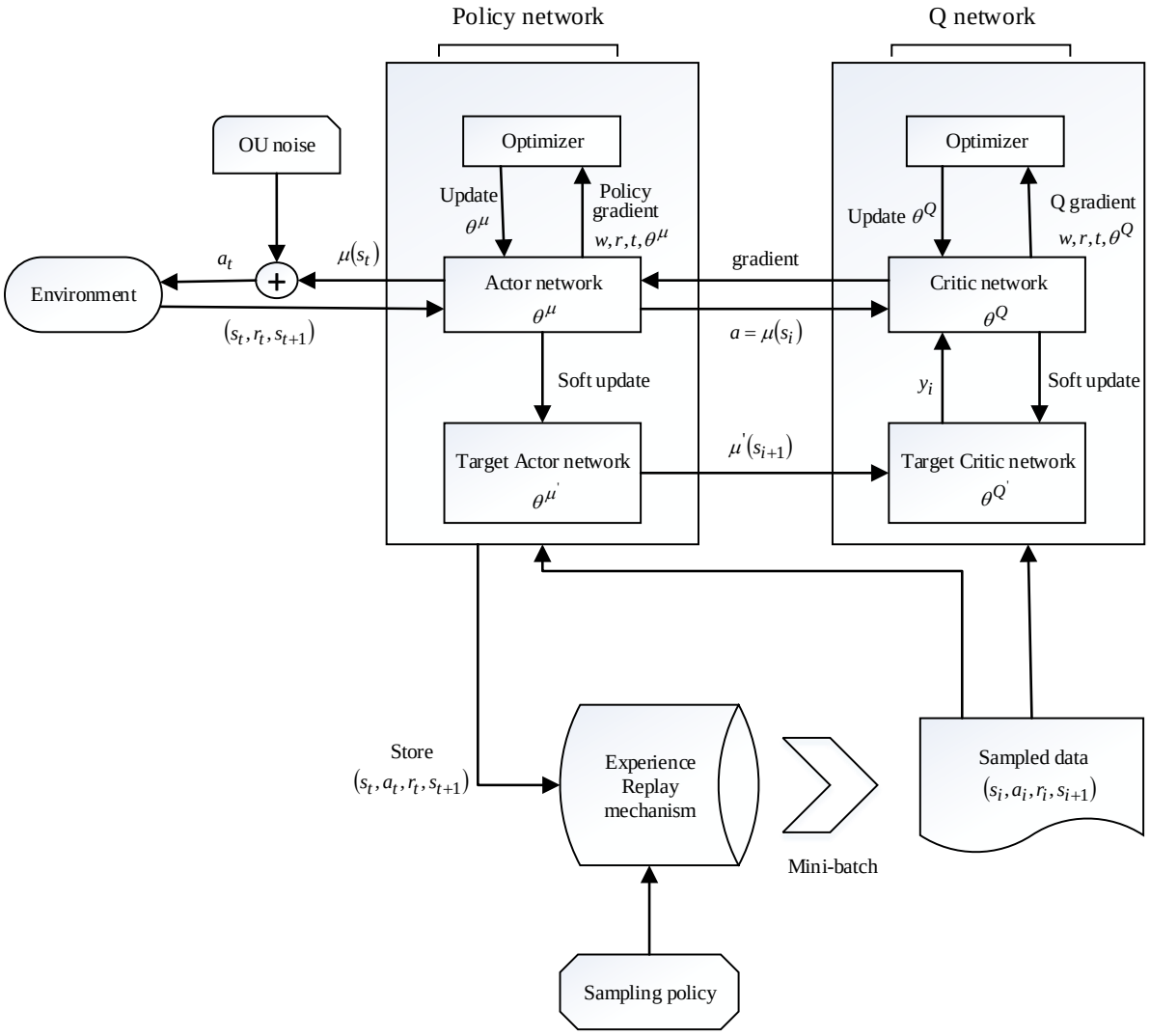

FIGURE 1. The DDPG algorithm structure framework

Actor network: the specific action a_t is generated according to the current state s_t.

$$a_t = \mu(s_t|\theta^{\mu}) + N_t \quad (1)$$

In the Eq. (1), μ function represents the actor network function, θ^{μ} represents the Actor network parameters, and N_t represents the exploration noise. The noise is generated randomly each time.

Target actor network: the follow-up action a_{t+1} used in the estimated value is generated according to the follow-up state s_{t+1} given by the environment.

Critic network：Q value is generated based on state s_t and action a_t .

Target critic network: $Q'(s_{t+1}, a_{t+1})$ is used to calculate the target value $y_t = Q(s_t, a_t)$ that is generated according to the subsequent states s_{t+1}, a_{t+1} . When the network model is trained, the network parameters are updated by minimizing the loss function $L(\theta^Q)$ of the critic network:

$$L(\theta^Q) = \frac{1}{M}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2 \quad (2)$$

where,

$$y_i = r_{i+1} + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right)|\theta^{Q'}\right) \quad (3)$$

r_{i+1} represents the reward generated by the interaction between the agent and environment; γ represents the discount factor, i.e., the discount coefficient between the Q value and the real value; $\theta^{\mu'}$ represents the network parameters of the target-actor network, and θ^Q and $\theta^{Q'}$ represent the network parameters of the critic network and target-critic network, respectively.

In critic network, Monte Carlo method is used to sample to approximate the expected value. The critic network parameters can be approximately updated through the chain rule:

$$\nabla_{\theta^{\mu}}(\theta^{\mu}) \approx \frac{1}{M}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}}(s|\theta^{\mu})|_{s_i} \quad (4)$$

The target network adopts the "soft" update method, and updates the parameters by slowly tracking the learned

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074535, IEEE Access

network

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q} \quad (5)$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu} \quad (6)$$

In the above, parameter τ represents the update coefficient, and the value of τ is small, indicating that the network updates a little bit each time. The "soft" update method makes the instability problem closer to supervised learning. Although it slows down the update speed of the target network parameters, the algorithm can obtain better stability during the learning process.

The DDPG has excellent performance in the reinforcement learning task of processing continuous action space. However, it also has some defects like the low convergence and local optimum.

III. DEEP DETERMINISTIC POLICY GRADIENT ALGORITHM BASED ON DOUBLE NETWORK PRIORITIZED EXPERIENCE REPLAY

In view of the shortcoming of the DDPG in using a neural network to estimate the value function, a function approximation error will occur and fall into a local optimum. Two critic neural networks are introduced to approximate the value function, and the minimum value of the value functions generated by the two networks is selected as the updated value of the actor policy network. In this way, the overestimation bias generated when the function is approximated is reduced, thereby reducing the occurrence of the local optimum. To solve the problem of slow convergence caused by the forgetting of important data when the agent selects the samples in the ER mechanism, the Q value obtained by the two networks and the immediate reward from the environment are taken as the criteria used to divide the sample priority.

A. USING DOUBLE CRITIC NETWORKS TO REDUCE OCCURRENCE OF LOCAL OPTIMUM POLICY

A neural network is used by the DDPG to approximate the value function, which will produce function approximation errors, value overestimation and sub-optimal phenomena. In the actor-critic structure of the DDPG, the policy is updated based on the estimate of the value of the approach to the critic function. Here, the algorithm uses the estimation of subsequent states to update the estimation of value function, which means that inaccurate estimation is used in each update, which will lead to the accumulation of such errors. Owing to the existence of overestimation bias, some bad states will be overestimated, leading to sub-optimal strategies and even divergent behavior. Thrun S. et al. [17] proved that the overestimation bias caused by the function approximation error will be propagated through the Bellman equation [18], and the error is inevitable. Scott Fujimoto et al. [19] proved that the value estimation in the deterministic policy gradient will be overestimated, and an overestimation bias exists. The TD-error that emerged in the neural network is expressed as follows:

$$\delta = r_{t+1} + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right) + N_t\right) - Q(s_t, a_t|\theta^{Q}) \quad (7)$$

where r_{i+1} represents the reward generated by the agent interacting with the environment, γ is the discount factor, N_t represents the noise, Q denotes the target value generated by the neural network in the current state, and Q' is called the target value expected by the agent. In reinforcement learning, the agent always uses the greedy policy to select actions when learning strategies. The greedy policy is based on TD-error as the standard. In formula (7), it can be seen that Q' is an estimated value, not a true value, so deviations will occur. This bias will continue to increase with the learning of the agent, resulting in overestimation bias, which is very unfavorable for algorithm learning.

In reducing the problem of overestimation bias, scholars have proposed several methods. In double Q-learning [20], the greedy policy is distinguished from the value function by maintaining two independent value estimates, and each value estimate is used to update the corresponding value function. In double deep q network [21], the author uses the target network as the value estimation and maximizes the current value network instead of the target network through the greedy policy, and then obtains the policy. Inspired by this idea, two critic networks are introduced on the original basis of the DDPG algorithm in order to update it. The minimum Q value generated by the two critic networks is selected as the target network update value. Because of overestimation bias, the value estimate can be used as the approximate upper limit of the target Q value. If the Q value selected each time is a Q value that produces overestimation bias, then the overestimation bias will be gradually increased and eventually the incorrect policy will be produced. In this subsection, the minimum of the two target Q values is used to update the policy. This underestimation idea is helpful to reduce overestimation bias. Even though the phenomenon of overestimation bias is inevitable, the idea of underestimation has a positive effect on its reduction. In reinforcement learning, the underestimation idea is usually not spread [19]. In reinforcement learning, the agent always learns in the direction of increasing reward value with a greedy policy. Therefore, the underestimation idea will not be spread as the agent continues to learn, which can effectively reduce overestimation bias. The improved critic network is shown in Fig. 2.

By selecting the smallest target Q value in the two critic networks as the updated value of the network parameters, the target value formulas generated by the two critic networks are expressed as follows:

$$y_1 = r_{t+1} + \gamma Q_1'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right) + N_{t_1}\right) \quad (8)$$

$$y_2 = r_{t+1} + \gamma Q_2'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right) + N_{t_2}\right) \quad (9)$$

The TD-error value and critic network update value become, respectively,

$$\delta_t = |y_1 - Q_1(s_t, a_t|\theta^{Q})| + |y_2 - Q_2(s_t, a_t|\theta^{Q})| \quad (10)$$

$$y = min_{i=1,2} y_i \quad (11)$$

$Q_1(s_t, a_t|\theta^{Q})$ and $Q_2(s_t, a_t|\theta^{Q})$ represent the respective Q values generated by the critic network in each iteration, and N_{t1} and N_{t2} represent noise. The noises N_{t1} and the noise N_{t2} in the y_1 and y_2 target value formulas are different.

The two noises are randomly generated when the network is initialized, and the two noise values are different, so two different target values are obtained, and the minimum of the two target values is selected as the updated value of the actor network. This method of reducing overestimation bias has a drawback: the variance of the algorithm becomes larger. To reduce the variance, appropriately reducing the discount factor γ of the algorithm can achieve the purpose of balancing the deviation and variance of the algorithm.

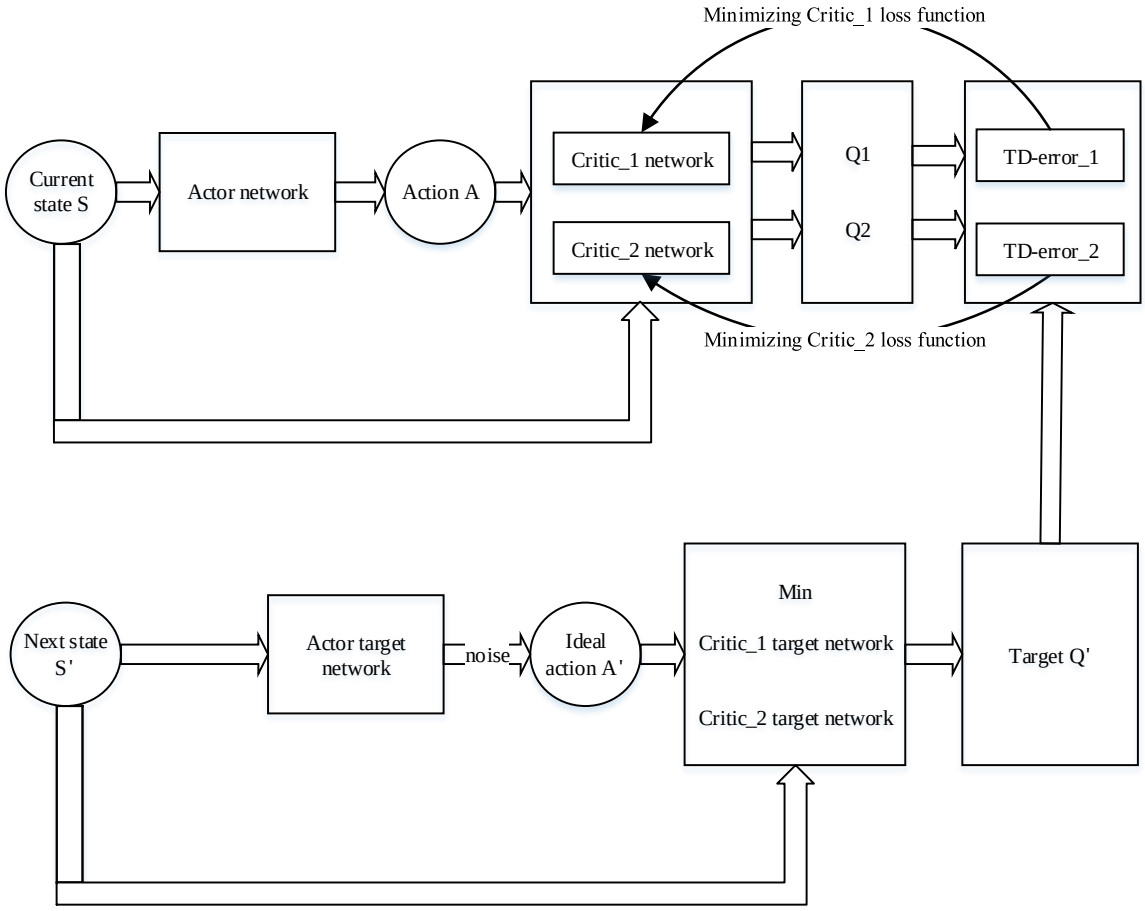

FIGURE 2. **The improved Critic network framework**

B. SETTING SAMPLE PRIORITY IN REPLAY MECHANISM TO IMPROVE CONVERGENCE SPEED

When the data in the experience replay mechanism are sampled, the data utilization rate will be low, which will cause the algorithm to converge slowly. In this subsection, the importance of samples is divided on the basis of the introduction of prioritized ER thinking: the sum of two TD-errors generated by double networks and the immediate reward is used as the basis for dividing the importance.

1) PRIORITIZED EXPERIENCE REPLAY THOUGHTS

The emergence of ER has led scholars to invest in new research directions. Mnih et al. [2] used ER to store experience samples and randomly selected experience samples for network training, which could break the correlation between experience samples. The training efficiency of the algorithm is improved. Although the correlation between experience samples is broken through their method, the importance of different samples is not considered. In [22], Schaul et al. gave each sample a different priority according to the importance of the experience sample, and designed a prioritized ER (PER) mechanism, which improved the convergence speed of the algorithm. In Ref. [23], the PER idea was applied to the DDPG algorithm, and the convergence speed of the improved algorithm increased.

The core idea of PER is to replay more frequently experiences related to very successful or very bad performance. Therefore, determining the standard of the empirical data being played back is the core issue. In most reinforcement learning algorithms, TD-error is usually used to update the action value function $Q(s,a)$. The TD-error value can be used as the correction of the estimation and can reflect the extent to which the agent can learn from experience. The greater the absolute value of TD-error, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074535, IEEE Access

more positive the correction of the expected action value. In this case, experience with higher TD-error is more likely to be of high value and is related to very successful attempts. In addition, experience with a large negative TD-error is a condition for the agent to behave badly, and the state of these conditions cannot be fully understood by the agent. Replaying these experiences more frequently will help the agent gradually understand the true results of the incorrect behavior in the corresponding state, and avoid recurrence of the incorrect behavior under these conditions, thereby improving the overall performance. Therefore, these inexperienced samples are also considered to be of high value. The probability of sampling experience is defined as

$$P(j) = \frac{D_j}{\sum_k D_k^\alpha} \tag{12}$$

where $D_j = \frac{1}{rank(j)} > 0, rank(j)$ is the level of experience j in the replay buffer, which is based on the absolute TD-error value. The parameter α represents the degree of control usage priority. The definition of sampling probability can be seen as a method of adding random factors when selecting experience, because even those samples with lower TD-error can still be played back, thus ensuring the diversity of sampling experience. This diversity is very helpful in preventing neural networks from overfitting.

Since the agent is more inclined to replay the experience with higher TD-error more frequently, there is no doubt that it will change the frequency of state access. The change may cause the training process of the neural network to easily oscillate or even diverge [19]. To solve this problem, importance sampling weights are used in the calculation of weight changes:

$$W_j = \frac{1}{S^\beta \cdot P(j)^\beta} \tag{13}$$

where S is the size of the replay buffer, $P(j)$ is the probability of sampled experience j, and the parameter β controls the degree of correction used.

Although the algorithm's convergence speed is accelerated by this method, the sample utilization rate in the ER mechanism is low, and its convergence speed is still far behind the speed required for processing the actual continuous motion space tasks.

2) SETTING OF SAMPLE IMPORTANCE

The DDPG based on prioritized ER uses TD-error as a measure of the importance of samples in the ER mechanism, which can improve the efficiency and quality of data sampling. Defects still exist in the classification of the importance of samples, i.e., insufficient sample division leads to a slower learning speed of the algorithm. Neuroscience research has shown that rodents will replay previously experienced sequences in the hippocampus during waking or sleep, and reward-related sequences will be replayed more frequently [24][25]. Some samples have small TD-error, but their immediate rewards are high. Such samples are also very useful for the learning of agents, but they cannot be sampled. Since the immediate reward obtained by the agent interacting with the environment at each moment is different, there is a difference between the TD-error it generates and the immediate reward value. On this basis, in this paper an improvement has been made; that is, the sum of the two TD-error values generated by the double networks in Section III.A and the immediate reward generated by the interaction between the agent and the environment are used as the criterion for judging the importance of the sample.

The improved priority evaluation criterion is

$$p = |y_1 - Q_1(s_t, a_t|\theta^Q)| + |y_2 - Q_2(s_t, a_t|\theta^Q)| + r + \varepsilon \tag{14}$$

where y_1 and y_2 are the expected values calculated from the target Q values generated by the two networks. The two Q values are estimated values obtained under the current policy, γ is an immediate reward, and ε is a small constant, the increase of which prevents some samples from being sampled when the sample priority p is 0, which, in turn, helps increase the diversity of sampled samples.

By using Eq. (14) as the priority criterion for evaluating experience data, the importance of the data in the ER mechanism can be further divided. Taking the errors obtained in the two networks into account is more accurate than considering only one of the errors, and the importance of empirical data can be more accurately judged. Immediate reward refers to the reward that the agent acquires when it reaches a certain state. Therefore, the immediate reward can portray the importance of experience data, so the immediate reward and the two TD-errors are taken into consideration as priority evaluation criteria. When the agent is learning, minimum batch sampling is performed according to the priority standard, which can make full use of the important data in the ER mechanism. This is helpful in improving the sample utilization rate in the replay mechanism and, thus, the convergence speed of the algorithm.

C. THE PSEUDOCODE OF IMPROVED ALGORITHM

Combining the improvement ideas presented in Sections III.A and III.B, the improved DDPG is obtained. The pseudocode of the improved algorithm is shown in Table I.

TABLE I
IMPROVED DDPG ALGORITHM PSEUDOCODE

Improved DDPG algorithm
Initialize the critic network $Q(s,a\|\theta^Q)$ and actor network randomly with weights θ^Q and θ^μ
Use weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$ to initialize the target network Q' and μ' respectively
Initialize the experience replay mechanism R capacity to **S**
For episode=1, M do
Initialize a random noise N_t as the noise of the actor's choice of action
Get the initial observation state s
For t = 1, T do
Select action $a = \mu(s_t\|\theta^\mu) + N_t$ through current strategies and exploring noise
Perform action a_t , get immediate reward r_t and next state s_{t+1}
Store experience (s_t, a_t, r_t, s_{t+1}) into R
If t > **S** then
For j = 1,K do
Sampling experience j with priorities
Calculate the corresponding importance sampling weight W_j and priority p according to Eq. (13) and (14)

Update the priority of sample data j according to the priority
End for
Use Eq. (2) to minimize the loss function to update the action-value network
Use Eq. (4) to calculate the policy gradient to update the actor network
Adjust the parameters of the critic function and actor function with the update rate λ
End for
End for

IV. PENDULUM CONTROL BASED ON IMPROVED ALGORITHM

To evaluate the proposed algorithm, three different continuous action space tasks in the classic control problem of OpenAI Gym [26] are used to test the effectiveness of the improved algorithm: Pendulum-v0, MountainCarContinuous-v0 and LunarLanderContinuous-v2. In Pendulum-v0, assuming that the pendulum starts at a random position, the goal is to swing up and keep the pendulum upright. The observed values are the sine and cosine values of the angle between the pendulum and the vertical direction and angular velocity of the pendulum respectively. The output action value is a continuous value of (-2,2), which represents the magnitude of the left and right-hand forces exerted on the pendulum. The formula of the reward function is $-(\theta^2 + 0.1 \times \theta_{dt}^2 + 0.001 \times action^2)$. The gravitational field is the gravitational field of Earth and the direction is toward the center of Earth. The pendulum environment is shown in Fig. 3.

FIGURE 3. OpenAI gym "Pendulum-v0" environment used for testing

In the MountainCarContinuous-v0 environment, the car is located on a one-dimensional track between two “mountains”. The goal is to drive to the mountain on the right. However, the car’s engine is not strong enough for a single pass. Therefore, the only way to succeed is to keep moving forward and accumulate momentum. If the car spends less energy to reach the target point, the reward will be higher. Environmental observations are the position and speed of the car. The position interval is (-1.2, 0.6), and the speed (-0.07, 0.07). The output action value means pushing the car to the left (negative value) or right (positive value). The reward function is the reward for reaching the target on the hill on the right minus the sum of squares of actions from the start to the target. The termination status is reaching the set episodes or reaching the mountain on the right. The environment is shown in Fig. 4.

FIGURE 4. OpenAI gym "MountainCarContinuous-v0" environment used for testing

In the LunarLanderContinuous-v2 environment, the landing pad is always at coordinates (0,0) and the coordinates are the first two numbers in the state vector. The reward for moving from the top of the screen to the landing spot and zero speed is approximately 100-140 points. If the lander is far from the landing site, it will lose the reward. If the lander crashes or comes to a standstill, the episode ends and an extra -100 or +100 points are awarded. The ground contact bonus for each leg is +10 points. Firing the main engine costs -0.3 points in each frame. The reward for achieving the goal is 200 points. Landing outside the landing pad is possible. The actions are two real-valued vectors from -1 to +1. First control the host, -1-0 means off, and 0-+1 means the throttle is increased from 50% to 100%. The power of the engine cannot be lower than 50%. The second value reaches -1.0 to -0.5 to ignite the left-hand engine, +0.5 to +1.0 to ignite the right-hand engine, and -0.5 to +0.5 to turn off the engine. The ultimate goal is to enable the lander to successfully land on the landing site. The condition for the end of the experiment is to reach the set episodes. The experimental environment is shown in Fig. 5.

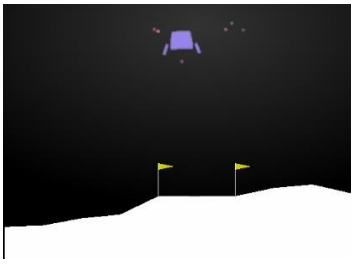

FIGURE 5. OpenAI gym "LunarLanderContinuous-v2" environment used for testing

A. SOFTWARE AND HARDWARE ENVIRONMENT

The software environment used in this paper is Anaconda 3-2.3.1 (Python 3.7), the IDE is PyCharm, and the programming framework is PyTorch 1.4.0. The python virtual environment is established in Anaconda and added to the compiler. The hardware environment is Lenovo's rescuer R720-15IKBN laptop, using NVIDIA GEFORCE GTX 1050 + CUDA10.1 + CUDNN7.0 GPU for computing acceleration.

B. PARAMETER SETTING

To ensure the fairness of the experimental results, the values of the common parameters of the five algorithms are set to the same. For the parameters used in the experiment, the control variable method is used to select the experimental parameters. When exploring the optimal value of a certain variable, the value of other variables is fixed. The optimal value of the variable is determined by changing the size of the variable and measuring the optimal reward. The noise added to actor network is Ornstein-Uhlenbeck noise distribution with the same parameters. The noise has a relatively high exploration efficiency for inertial systems [27]. The number of batch samples is equal. The capacity of the experience replay mechanism is set to 10000, the reward discount factor is 0.99, and the number of batch samples is 64. The learning rate of the actor network is 0.0005, that of the critic network is 0.001, and the τ parameter used when updating the target network is set to 0.005.

In this subsection, two important parameter values, i.e., the number of batch samples (batch size) and the ER mechanism capacity (buffer limit), are experimentally determined. The experimental results are obtained by testing with four comparison algorithms (DDPG, PER-DDPG, TD3 and PPO [28]). To clarify the experimental results, network training was set to 500 episodes and selected according to the reward value. The greater the reward value, the better the corresponding parameter value.

Batch sampling size (batch size) plays an important role in the training time of the model and convergence state of the algorithm. When the batch size is too large, to achieve the same accuracy the training time of the algorithm will be greatly increased. If the batch size is too small, the randomness will be greater and the algorithm will have difficulty reaching the state of convergence [29]. Therefore, the batch size is very important for evaluating the pros and cons of the algorithm. To facilitate the experiment, two cases, of size 32 and 64, were selected for parameter determination; the reward values are shown in Table II.

TABLE II
REWARD VALUE OF BATCH SIZE WITH DIFFERENT ALGORITHMS

Algorithm	Batch Size	Reward Value
DDPG	32	-332.1
	64	**-240.29**
PER-DDPG	32	-266.7
	64	**-231.32**
TD3	32	-1537.24
	64	**-1532.19**
PPO	32	-353.86
	64	**-344.34**
DNPER-DDPG	32	-210.38
	64	**-175.92**

It can be seen from Table II that these five algorithms have different reward values corresponding to different batch sizes. When the batch size is 64, the reward value obtained is the largest. Therefore, the batch sampling value was set to 64 for experimentation.

The buffer limit of the ER mechanism also plays an important role in the convergence of the algorithm. If the buffer limit selected is too small, the data diversity sampled by the agent becomes smaller and the algorithm does not easily converge to the global optimal state. If the selection is too large, it will occupy a significant amount of running memory and reduce the speed of the algorithm. Therefore, the size of buffer limit plays an important role in the evaluation of the algorithm. The capacity buffer limit of the ER mechanism was therefore selected as 5000, 10000, and 15000 for parameter determination. The reward values are shown in Table III.

It can be seen from Table III that as the capacity of the ER mechanism increases, the rewards of these five algorithms first become larger and then smaller. Each algorithm has the largest reward value when the buffer limit is 10000, so the capacity of the ER mechanism was selected as 10000 to test the algorithm.

TABLE III
REWARD VALUE OF BUFFER LIMIT WITH DIFFERENT ALGORITHMS

Algorithm	Buffer Limit	Reward Value
DDPG	5000	-400
	10000	**-250**
	15000	-380
PER-DDPG	5000	-300
	10000	**-250**
	15000	-400
TD3	5000	-1400
	10000	**-1200**
	15000	-1550
PPO	5000	-900
	10000	**-700**
	15000	-800
DNPER-DDPG	5000	-450
	10000	**-200**
	15000	-300

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074535, IEEE Access

C. EXPERIMENTAL RESULTS AND ANALYSIS

Because there is no training data set and validation set for deep reinforcement learning, it is difficult to evaluate the training situation of the algorithm online. Therefore, two main ways exist to evaluate the training effect. One is by convergence time. The less time the algorithm takes to reach the maximum reward value, the better the algorithm performance. The second is average reward value. After a certain period of network training, the larger the reward value, the better the policy and the better the effect [30]. Through these two methods, the performance of the improved algorithm can be tested.

Figure 6 shows the performance of the five algorithms in the three experimental environments of Pendulum-v0, MountainCarContinuous-v0 and LunarLanderContinuous-v2. The comparison algorithms are the DDPG, PER-DDPG, TD3, and PPO algorithms. To make the comparison between the improved algorithm DNPER-DDPG and other algorithms clearer, the method of comparing one by one was selected. To turn the pendulum upside down and realize the stable effect of the algorithm as much as possible, the number of training steps set in the experiment should be relatively large; here, it was set to 5,000 episodes, each episode containing 200 time-steps, and the five algorithms all used the same parameters. The training reward values are shown in Fig. 6.

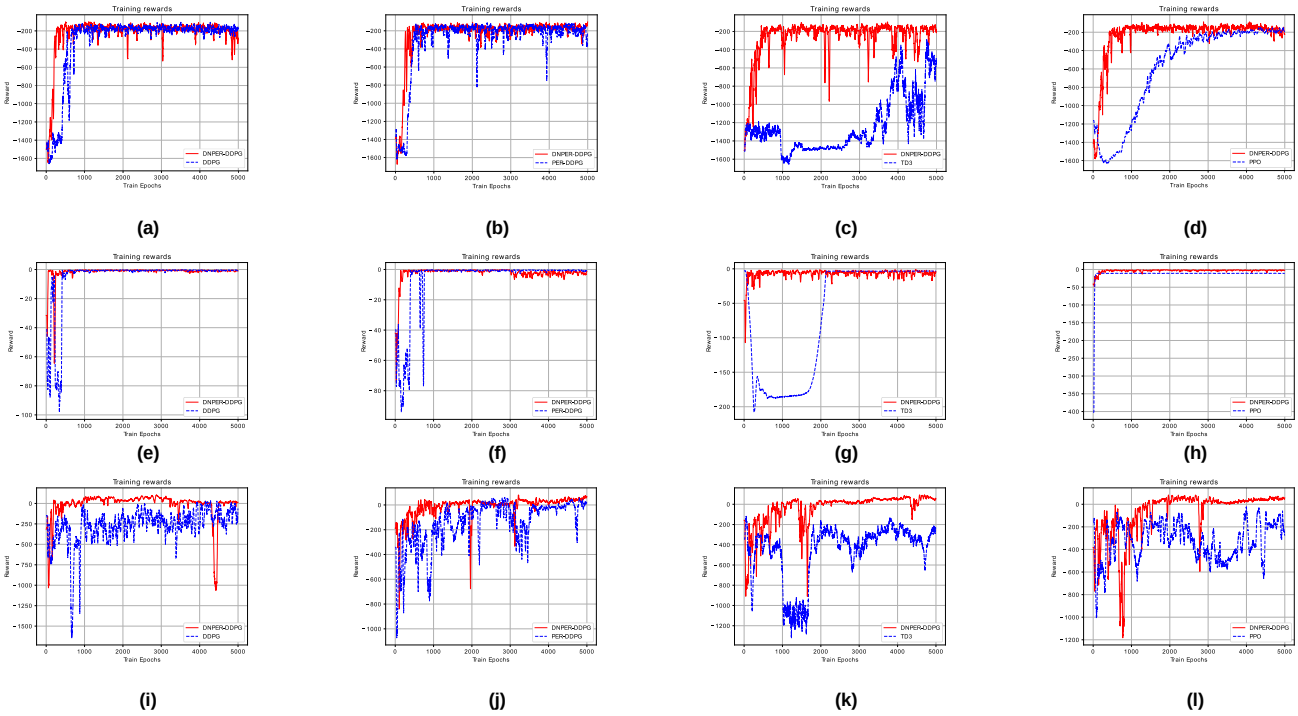

FIGURE 6. Comparison of cumulative rewards and convergence of different algorithms. Among them, (a)-(d) are the experimental results of five algorithms in Pendulum-v0; (e)-(h) are the experimental results in MountainCarContinuous-v0; and (i)-(l) are the experimental results in LunarLanderContinuous-v2.

Figure 6 is an experimental effect diagram of the training time and training reward value of the algorithm. The abscissa represents the number of episodes of algorithm training and the ordinate the training reward values of the algorithm. The reward curve in Fig. 6 shows an upward trend, and finally stabilizes near a certain reward value (the phenomenon of the curve finally stabilizing). The final stable phenomenon of the curve indicates that the pendulum is finally maintained at a certain angle after learning. The higher the reward value, the better the completion of the pendulum task. The episode number used for the curve to reach a stable reward value represents the convergence time of the algorithm. The smaller the episode number, the faster the convergence, and vice versa.

Owing to the randomness of the single experiment, five tests were conducted in each experiment. The convergence times and reward values of each test are recorded in Tables IV and V, respectively. The average cumulative rewards and average convergence time of the five experiments were selected as the final experimental results. The ratio of the average convergence times of the improved algorithm and the difference between the comparison algorithm and average convergence time of the comparison algorithm are used to determine the rate of improvement of the convergence speed of the improved algorithm. The difference between the average reward of the improved algorithm and the comparison algorithm is used as the added value of the reward of the improved algorithm. The decrease rate of convergence and reward increment value are calculated and entered into the quantitative analysis tables, as shown in Tables IV to IX.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074535, IEEE Access

TABLE IV
CONVERGENCE TIME OF PENDULUM-V0

	The first time	The second time	The third time	The fourth time	The fifth time	Average Convergence time (episode)	Decrease Rate(%)
DDPG	1460	1450	1460	1465	1480	1463	
DNPER-DDPG	410	400	420	430	400	412	**71.8**
PER-DDPG	1650	1680	1670	1680	1600	1656	
DNPER-DDPG	500	520	530	520	500	514	**69.0**
TD3	4000	4050	4080	4050	4000	4036	
DNPER-DDPG	1000	990	1010	1000	1050	1010	**75.0**
PPO	900	950	940	960	950	940	
DNPER-DDPG	300	340	320	330	340	326	**65.3**

TABLE V
CONVERGENCE TIME OF MOUNTAINCARCONTINUOUS-V0

	The first time	The second time	The third time	The fourth time	The fifth time	Average Convergence time (episode)	Decrease Rate(%)
DDPG	450	430	460	450	440	446	**45.3**
DNPER-DDPG	230	240	260	240	250	244	
PER-DDPG	850	840	830	850	870	848	**76.7**
DNPER-DDPG	200	180	210	190	210	198	
TD3	2100	2150	2100	2200	2250	2160	**76.0**
DNPER-DDPG	500	520	510	530	530	518	
PPO	100	120	130	120	120	118	**15.3**
DNPER-DDPG	100	110	90	100	100	100	

TABLE VI
CONVERGENCE TIME OF LUNALANDERCONTINUOUS-V2

	The first time	The second time	The third time	The fourth time	The fifth time	Average Convergence time (episode)	Decrease Rate(%)
DDPG	1000	1020	960	980	1060	1192	**74.2**
DNPER-DDPG	320	310	320	300	290	308	
PER-DDPG	2000	1900	2050	1950	2100	2000	**62.2**
DNPER-DDPG	700	780	750	800	750	756	
TD3	2000	2050	2080	1900	2100	2026	**50.6**
DNPER-DDPG	1000	1100	950	1000	950	1000	
PPO	1500	1600	1550	1500	1400	1510	**28.5**
DNPER-DDPG	1100	1000	1200	1000	1100	1080	

TABLE VII
CUMULATIVE REWARDS OF PENDULUM-V0

	The first time	The second time	The third time	The fourth time	The fifth time	Average Reward	Value Added
DDPG	-190	-200	-210	-210	-200	-202	**22**
DNPER-DDPG	-170	-180	-180	-190	-180	-180	
PER-DDPG	-200	-205	-220	-210	-200	-207	**20.4**
DNPER-DDPG	-185	-190	-180	-188	-190	-186.6	
TD3	-600	-650	-640	-645	-640	-635	**424**
DNPER-DDPG	-200	-220	-210	-220	-205	-211	
PPO	-200	-220	-230	-220	-240	-222	**61**
DNPER-DDPG	-160	-170	-160	-165	-150	-161	

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074535, IEEE Access

TABLE VIII
CUMULATIVE REWARDS OF MOUNTAINCARCONTINUOUS-V0

	The first time	The second time	The third time	The fourth time	The fifth time	Average Reward	Value Added
DDPG	0	20	0	10	20	10	**5**
DNPER-DDPG	5	25	10	15	20	15	
PER-DDPG	0	10	20	0	15	9	**5**
DNPER-DDPG	0	20	15	10	25	14	
TD3	-5	-10	-5	-8	-10	-7.6	**3**
DNPER-DDPG	-8	0	-10	-5	0	-4.6	
PPO	-15	0	-20	-15	-20	-14	**19**
DNPER-DDPG	0	10	0	15	0	5	

TABLE IX
CUMULATIVE REWARDS OF LUNARLANDERLANDERCONTINUOUS-V2

	The first time	The second time	The third time	The fourth time	The fifth time	Average Reward	Value Added
DDPG	-250	-260	-250	-240	-260	-252	**290**
DNPER-DDPG	30	50	40	40	30	38	
PER-DDPG	-50	-80	-70	-90	-60	-70	**121**
DNPER-DDPG	50	60	50	40	55	51	
TD3	-250	-200	-260	-200	-230	-228	**328**
DNPER-DDPG	100	80	110	100	110	100	
PPO	-200	-240	-220	-240	-250	-230	**61**
DNPER-DDPG	60	40	80	50	60	58	

1) CONVERGENCE TIME ANALYSIS

It can be seen from Tables IV-VI that the convergence speed of the improved algorithm has been improved. In different experiments, the improvement of the convergence speed is also different. In Pendulum-v0, the convergence speed of the algorithms proposed in this paper is increased by more than 50%. In Mountaincarcontinuous-v0, compared with the PPO algorithm, the convergence time is increased by 15%, and compared with the other three algorithms, the convergence speed is increased by more than 45%. In Lunarlandercontinuous-v2, the algorithm is 28% faster than the PPO algorithm and more than 50% faster than the other three algorithms. This shows that the priority function set in this paper plays an important role in raising the convergence speed of the algorithm. Using this priority-setting method, experience samples that are more effective for network model learning can be selected. In the same training phase, the algorithm can learn policies with higher cumulative rewards. The introduction of two networks to set the priority of samples can prevent the algorithm from repeatedly selecting those useless samples in the ER mechanism. The sum of the TD-errors generated by the two networks and the immediate reward are used as the index by which the priority of the sample is evaluated, so that the importance of each sample is more detailed. At the same time, samples with high importance can be sampled preferentially during sampling. This method is conducive to obtaining the global optimal value and greatly improves the convergence of the algorithm.

2) CUMULATIVE REWARD ANALYSIS

It can be seen from Tables VII-IX that the cumulative reward of the algorithm proposed in this paper has been improved. In Pendulum-v0 and Lunarlandercontinuous-v2, when the reinforcement learning task is trained to a steady state, its cumulative rewards are increased. In Mountaincarcontinuous-v0, when the algorithm is trained to a steady state, the increase in cumulative reward is not obvious. This is because the task of that particular environment is easier than the other two environments, so when the algorithm is trained to the end, the increase in cumulative reward is not obvious. However, in the three reinforcement learning tasks, the cumulative reward of the proposed algorithm was increased. This shows that the algorithm proposed in this paper can obtain a better policy than the other four algorithms. The introduction of double networks has certain advantages for algorithm improvement. In this paper, the minimum target value produced by the double network selected here is used as the updated value, which can reduce overestimation bias in the network. It is helpful for the algorithm to move in the direction of outputting more realistic actions. It is also conducive to the update of the policy, so that the algorithm can acquire a higher cumulative reward, thereby obtaining the global optimal policy.

3) STABILITY ANALYSIS

It can be seen from Fig. 6 that in the three reinforcement learning tasks the reward value fluctuations when the reward

value curve of the five algorithms reaches a stable trend are different. In Pendulum-v0, comparing the proposed algorithm with the TD3 and PPO algorithm, it can be found that the stability of the algorithm was enhanced and its fluctuation is more stable than that of those two algorithms. In Mountaincarcontinuous-v0, compared with the DDPG, PER-DDPG and TD3 algorithm, the proposed algorithm has better stability. In Lunarlandercontinuous-v2, the proposed algorithm has increased stability compared to the four comparison algorithms. From the above analysis, it can be known that the idea of prioritizing the samples in the ER machine based on the introduction of double networks can improve the stability of the algorithm.

V. CONCLUSIONS

A deep deterministic policy gradient algorithm based on a double network prioritized ER mechanism is proposed in this paper. To reduce the probability of getting stuck in a local optimum, two critic networks are introduced into the structure of the algorithm. When the policy network is updated, the minimum Q value produced by the two networks is selected for learning. To solve the problem of slow convergence of the algorithm, based on the introduction of two networks, the ideas of prioritized ER and immediate reward are used as the criteria with which to judge the importance of samples, and the sample data in the experience replay mechanism are prioritized. The experimental results show that compared with four other methods, the improved DDPG exhibits an increased convergence rate and cumulative reward, and the probability of getting stuck in a local optimum is reduced.

Planned future research using this method includes the following. First, the deep-learning method can be combined with a neural network to improve the algorithm's optimization ability. Second, to further increase the rate of convergence, more appropriate criteria can be used to evaluate the priority of samples in the ER mechanism.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, and D. Silver, et al. "Playing atari with deep reinforcement learning," in NIPS 2013, Barcelona, Spain, 2013.

[2] V. Mnih, K. Kavukcuoglu, and D. Silver, et al., "Human-level control through deep reinforcement learning," Nature, pp. 529-533, Feb. 2015.

[3] M. Riedmiller, T. Gabel, and R. Hafner, et al., "Reinforcement learning for robot soccer," Auton. Robot., vol. 27, no. 1, pp. 55–73, May, 2009, 10.1007/s10514-009-9120-4.

[4] H. V. Hasselt, and M. A. Wiering, "Using continuous action spaces to solve discrete problems," in IJCNN, Atlanta, GA, USA, 2009, pp. 1149-1156.

[5] T. P. Lillicrap, J. J. Hunt, and A. Pritzel, et al., "Continuous control with deep reinforcement learning," CoRR. vol. 8, no. 6, Sep. 2015.

[6] M. Zhang, Y. Zhang, and Z. Gao, et al., "An improved DDPG and its application based on the double-layer BP neural network," IEEE Access. vol. 8, pp. 177734-177744, Aug, 2020, 10.1109/ACCESS.2020.3020590.

[7] Z. Tian, X. Zuo, and X. Li, "Autopilot Policy Based on Improved DDPG Algorithm," SAE. Technical Paper, Jan. 2019, 10.4271/2019-01-5072.

[8] E. Nikishin, P. Izmailov, and B. Athiwaratkun, et al., "Improving stability in deep reinforcement learning with weight averaging," in UAI, Monterey, CA, USA, 2018.

[9] H. Xiang, J. Cheng, Q. Zhang, et al., "Synthesized Prioritized Data Pruning based Deep Deterministic Policy Gradient Algorithm Improvement," in ICIA., Wuyi Mountain., China, 2018, pp. 121-126.

[10] X. Wu, S. Liu, and T. Zhang, et al. "Motion Control for Biped Robot via DDPG-based Deep Reinforcement Learning," in WRC SARA, Beijing, China, 2018, pp. 40-45.

[11] Y. Liu, W. Zhang, and F. Chen, et al. "Path planning based on improved Deep Deterministic Policy Gradient algorithm," in IEEE ITNEC, Chengdu, China, 2019, pp. 295-299.

[12] Z. Li, and G. Guo. "Research on Optimization Policy Based on Improved DDPG Algorithm in WSN," in ICDSP, Chengdu, China, 2020.

[13] J. Tang, L. Li, and Y. Ai, et al. "Improvement of End-to-end Automatic Driving Algorithm Based on Reinforcement Learning," in CAC, Hangzhou, China, 2019, pp. 5086-5091.

[14] H. Y. Zhang, K. Xiong, and J. Bai. "Improved Deep Deterministic Policy Gradient Algorithm Based on Prioritized Sampling," in CISC'18, Wenzhou, China, 2018, pp. 205-215

[15] A. W. Moore, and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," Mach Learn, vol. 1, no. 13, pp. 103-130, Oct. 1993.

[16] M. Fang, Y. Li, and T. Cohn, "Learning how to active learn: A deep reinforcement learning approach," in EMNLP, Copenhagen, Denmark, 2017, pp. 595-605.

[17] S. Thrun, and A. Schwartz, "Issues in using function approximation for reinforcement learning," in Proc. The 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum. 1993.

[18] R. Bellman, "Dynamic programming," Science, vol. 3731, no. 153, pp. 34-37, Jul. 1966, 10.1126/science.153.3731.34.

[19] S. Fujimoto, H. V. Hoof, and D. Meger., "Addressing function approximation error in actor-critic methods," in ICML, Stockholm, Sweden, 2018, pp. 1587-1596.

[20] H. Hasselt, "Double q-learning," In Advances in Neural Information Processing Systems, no. 23, pp. 2613–2621, 2010.

[21] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 1, no. 30, 2016.

[22] T. Schaul, J. Quan, and I. Antonoglou, et al., "Prioritized experience replay," in ICLR, San Juan, PUR, USA, 2016.

[23] Y. Hou, L. F. Liu, and Q. Wei, et al., "A novel ddpg method with prioritized experience replay," in IEEE SMC, Budapest, Hungary, 2017, pp. 316-321.

[24] L. A. Atherton, D. Dupret, and J. R. Mellor, "Memory trace replay: the shaping of memory consolidation by neuromodulation," Trends in neurosciences, vol. 9, no. 38, pp. 560-570, 2015.

[25] H. F. Ólafsdóttir, C. Barry, and A. B. Saleem, et al., "Hippocampal place cells construct reward related sequences through unexplored space," Elife, no. 4, Jun. 2015.

[26] G. Brockman, V. Cheung, and L. Pettersson, et al., "OpenAI Gym," Mach Learn, Jun. 2016.
[27] G. E. Uhlenbeck, and L. S. Ornstein, "On the theory of the Brownian motion," Phys. Rev, vol. 36, no. 5, pp. 823, 1930.
[28] J. Schulman, F. Wolski, and P. Dhariwal, et al., "Proximal policy optimization algorithms," Mach Learn, Jul. 2017.
[29] P. Goyal, P. Dollar, and R. B. Girshick, et al., "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," CVPR, Computer Vision and Pattern Recognition, Jun, 2017.
[30] M. G. Bellemare, Y. Naddaf, and J. Veness, et al., "The arcade learning environment: An evaluation platform for general agents," J Artif Intell Res, no. 47, pp. 253-279, Jun, 2013, 10.1613/jair.3912.

CHAOHAI KANG received the M.S. degree in control theory and control engineering in 2005 from Northeast Petroleum University, Daqing, China. He is currently an Associate Professor at Northeast Petroleum University, Daqing, China. His research interests include intelligent control and pattern recognition.

CHUITING RONG received the B.S. degree in automation from Shandong University of Technology, Zibo, China, in 2016. He is currently pursuing the M.S. degree in control science and engineering of Northeast Petroleum University, Daqing, China. His current research interests include reinforcement learning, path planning, and robot control.

WEIJIAN REN received the M.S. degree in control theory and control engineering in 1990 and Ph.D. degree in oil storage and transport engineering in 2006, both from Northeast Petroleum University, Daqing, China. She is currently a Professor at Northeast Petroleum University, Daqing, China. Her research interests include modelling and control of complex systems, fault diagnosis and simulation.

FENGCAI HUO received the Ph.D. degree in oil and natural gas engineering from Northeast Petroleum University, Daqing, in 2015. He has been with School of Electrical Information Engineering in Northeast Petroleum University, Daqing, China, where he is currently an associate professor. His research interests and areas of publication include artificial intelligence, intelligent control and image processing.

PENGYUN LIU received the B.S degree in computer network from Jiangxi Normal University, Nanchang, China, in 2018. He is currently pursuing the M.S. degree in control science and engineering of Northeast Petroleum University, Daqing, China. His current research interests include deep learning, quantitative trading, and reinforcement learning.