A Manipulator Control Method Based on Deep Deterministic Policy Gradient with Parameter Noise

Haifei Zhang (zhanghf@ntit.edu.cn)
Nantong Institute of Technology
Xu Jian
Nantong University
Liting Lei
Nantong Institute of Technology
Fang Wu
Nantong Institute of Technology
Lanmei Qian
Nantong Institute of Technology
Jianlin Qiu
Nantong Institute of Technology

Research Article

Keywords: Deep reinforcement learning, deep deterministic policy gradient, Manipulator control

Posted Date: November 22nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1016003/v1

A Manipulator Control Method based on Deep Deterministic Policy Gradient with Parameter Noise

Haifei Zhang[1,*], Xu Jian[2], Liting Lei[1], Fang Wu[1], Lanmei Qian[1] and Jianlin Qiu[1,2]

[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

[*]Corresponding Author: Haifei Zhang. Email: zhanghf@ntit.edu.cn

Received: XX Month 202X; Accepted: XX Month 202X

Abstract: Focusing on the motion control problem of two link manipulator, a manipulator control approach based on deep deterministic policy gradient with parameter noise is proposed. Firstly, the manipulator simulation environment is built. And then the three deep reinforcement learning models named the deep deterministic policy gradient (DDPG), asynchronous advantage actor-critical (A3C) and distributed proximal policy optimization (DPPO) are established for training according to the target setting, state variables and reward & punishment mechanism of the environment model. Finally the motion control of two link manipulator is realized. After comparing and analyzing the three models, the DDPG approach based on parameter noise is proposed for further research to improve its applicability, so as to cut down the debugging time of the manipulator model and reach the goal smoothly. The experimental results indicate that the DDPG approach based on parameter noise can control the motion of two link manipulator effectively. The convergence speed of the control model is significantly promoted and the stability after convergence is improved. In comparison with the traditional control approach, the DDPG control approach based on parameter noise has higher efficiency and stronger applicability.

Keywords: Deep reinforcement learning; deep deterministic policy gradient; Manipulator control

1 Introduction

Over recent years, the rapid development of machine learning, especially deep learning and reinforcement learning technology, has improved the automatic production efficiency in the industrial manufacturing process. Different types and functions of mechanical arms play a significant part in the manufacturing process. The two link manipulator is often used in industrial production such as handling and hoisting.

Based on the robot's own perception, decision-making, planning and control ability, the manipulator control system operates the target in the environment within a certain time to make the target reach the target state from the initial state. Traditional control methods, using complex programming, remote control or teaching methods, can make the manipulator have certain operation skills, be better competent for some structured working environment and single fixed task working scenes, and complete repetitive tasks accurately and quickly. However, with the development of robot technology, people have higher and higher requirements for robot's autonomous operation ability, task complexity and intelligent analysis and processing. The manipulator controlled by traditional methods cannot dynamically adapt to this unstructured environment or the workplace with too many uncertain factors. The traditional control system has many problems, such as long development cycle, heavy workload, low efficiency and poor robustness for different environments. The fast development of artificial intelligence technology and the breakthrough of key technologies have brought new ideas to robot control methods. The manipulator control system with autonomous decision-making and autonomous learning ability is developed by using machine learning method. Therefore, the manipulator can learn the execution policy automatically in a complex and dynamic environment, which can make up for the defects of traditional control methods. It greatly improves the environmental adaptability of the manipulator. Autonomous learning ability is one of the important abilities that future robots should have. It is of great importance to develop robot technology in the future. It is also an important basis for the wide application of future robots in various fields.

Intelligent robots are all over the world, and more and more people feel the convenience they bring. Intelligent robots are applied to various social scenes to cultivate a new business form of robot cooperative work. The so-called "intelligent robot" is to give it the same developed "brain" as human beings on the basis of traditional mechanical robot [1]. The reinforcement learning algorithm can better control the motion of the manipulator [2]. Intelligent algorithms like deterministic policy, Q-learning, neural network and other reinforcement learning algorithm have achieved certain results in the control of manipulator, but there are still some problems, such as the training speed, convergence degree and running time of the algorithm need to be further optimized [1, 3, 4, 5, 6, 7]. Based on the research in this field, reinforcement learning algorithm has natural advantages in manipulator control. Training and learning the control ability of manipulator based on reinforcement learning algorithm will play a great part in the development of industrial Internet.

Deep reinforcement learning is a branch of deep learning and artificial intelligence in recent two years, which focusing on solving the problem of computer from perception to decision control so that general artificial intelligence can be realized. Recently, Google, Baidu, Tencent, Alibaba and other companies have proposed a variety of algorithms based on deep reinforcement learning, and have made breakthroughs in the fields of video, games, go, robots and so on. In this paper, three deep reinforcement learning models Deep Deterministic Policy Gradient (DDPG), Asynchronous Advantage Actor-Critic (A3C), Distributed Proximal Policy Optimization (DPPO) are used for training to realize the motion control of two link manipulator. After comparing and analyzing the three models, the DDPG method based on parameter noise is proposed. The experimental results show that the proposed DDPG approach based on parameter noise can control the motion of two link manipulator effectively. The convergence speed of the control model is significantly increased and the stability after convergence is strengthened.

2 Related Work

The control of manipulator has a certain research foundation in China. The control strategies include fuzzy neural network, deep reinforcement learning, iterative learning and other intelligent algorithm control. The strong coupling and nonlinear dynamic characteristics of the manipulator with multi degree of freedom make the learning and verification of this problem a more complex problem [2, 8, 9, 10, 11, 12]. In order to achieve high-precision position tracking, NGO et al. [13] proposed a fuzzy neural network control system with robust self-adaption. This model-free control scheme can ensure stable position tracking and high control accuracy, but it is less intelligent and difficult to adapt to complex environment. Kormushev et al. [14] proposed an approach to learn and reproduce the interaction between human and machine in human-robot interaction environment. The manipulator can obtain the ability of reproducing action through learning by using the imitation learning method in artificial intelligence. Imitation learning makes the robot learn some actions through teaching, which requires manpower to teach in a specific way, but this method has high manpower cost. Zhang et al. [15] designed a manipulator vision control system based on machine learning. Taking the planar manipulator with three degree of freedom as the research object, the model was trained offline by using deep Q network (DQN) [16], and a variety of disturbance terms were set to test the robustness of the algorithm

The combination of reinforcement learning and deep learning will make reinforcement learning algorithm have stronger perception and decision-making ability, and have great advantages in solving the problems in solving high-dimensional state space problems. The DQN algorithm proposed by the Google

DeepMind team allows agents to learn from visual perception input composed of thousands of pixels [17].With its good approximation ability to nonlinear functions, deep reinforcement learning shows great potential in the field of robot control, and has achieved good performance in theoretical analysis and simulation verification, which greatly improves the autonomy and flexibility of robot control [18, 19, 20].Deep reinforcement learning algorithm has been applied widely in computer games, automatic driving, natural language processing, recommendation system and robot control, and achieved good results. DQN algorithm has good advantages in training agents for game confrontation and unmanned driving. In addition, in the area of smart grid, the use of deep reinforcement learning to solve the operation problem of microgrid under uncertainty has made some achievements. Although the algorithm based on deep reinforcement learning has made a series of achievements in solving the complex control problems of manipulator, it has the problems of low sampling efficiency and long model training time. It is a big challenge to apply it to the intelligent control of robot.

In this paper, the motion control task of two link manipulator is divided into two parts: algorithm design and simulation verification. First, the research of deep reinforcement learning is carried out to build an algorithm framework for continuous motion of manipulator. Q-learning algorithm is a traditional reinforcement learning algorithm. It uses q-table to deal with discrete problems, which is not easy to converge in application, and deep Q-learning can only deal with discrete action space. The manipulator motion is a continuous action, and the final result needs to converge. Therefore, DDPG algorithm [20] is used in this paper. DDPG introduces deep Q-learning into the continuous action space to solve the continuous action problem. Set a reasonable action input and reward/punishment value of the manipulator, and build a DDPG algorithm framework suitable for the two link manipulator model. On this basis, reinforcement learning algorithm A3C and DDPO algorithm [21] are introduced to compare with DDPG algorithm. Finally, DDPG method based on parameter noise is proposed for further research to improve its applicability, so as to shorten the debugging time of manipulator model and achieve the target smoothly. It is proved by the experiment that the proposed DDPG method based on parameter noise can control the motion of two link manipulator effectively. The convergence speed of the control model is significantly promoted and the stability after convergence is improved. Compared with the traditional control method, DDPG control method based on parameter noise has higher efficiency and stronger applicability.

3 Deep reinforcement learning

Deep learning is a significant field of machine learning. It mainly simulates the human brain to analyze and solve problems by establishing neural networks with different depths. The neural network framework established through deep learning can be applied to deep reinforcement learning, but deep learning lacks a certain degree of decision-making ability. Reinforcement learning (RL) is a major branch of machine learning, which is applied to describe and solve the problem of maximizing returns or achieving specific goals by learning strategies during the interaction between agent and environment. The main learning contents of agent are action policy and planning. It seeks the optimal behavior policy to obtain the maximum reward value, so as to achieve the task goal. Reinforcement learning has the ability to make decisions, but it lacks perceptual ability. Deep reinforcement learning solves many scientific research problems by combining the perceptual ability of deep learning with the decision-making ability of reinforcement learning and using the neural network framework of deep learning and the decision-making ability of reinforcement learning to.

3.1 Deep Deterministic Policy Gradient (DDPG)

DDPG algorithm [20] is an off-policy and model-free deep deterministic policy gradient algorithm, combining deep learning and reinforcement learning, integrating the advantages of dqn algorithm and actor critical algorithm. DDPG algorithm is the same as AC algorithm framework, but its neural network division is finer. DQN algorithm has good performance in discrete problems. DDPG algorithm learns the experience from dqn for reference to solve the problem of continuous control and realize end-to-end learning. The algorithm flow of DDPG is shown in Figure 1, in which: the actor network accepts the input state, makes action selection and outputs action variables; The critical network evaluates the quality of the selected action and calculates the reward value. The detailed steps of ddpg algorithm are as follows:

(1) Initialize the parameters of the neural network. The actor selects an action according to the behavior policy, adds noise N_t to the action output by the policy network to increase exploration, and transmits it to the environment to execute the action a_t :

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

(2) After the environment is executed, return to reward and new status.

(3) Actor stores the state transition into replay memory as the training set of online network.

(4) DDPG creates two copies of neural networks respectively for policy network and Q network, online network and target network. The updated method to policy network is as follows:

$$\begin{cases} \text{online: } Q(s,a|\theta^{\mu}), & \text{gradient update } \theta^{\mu} \\ \text{target: } Q(s,a|\theta^{\mu'}), & \text{soft update } \theta^{\mu'} \end{cases} \tag{2}$$

Q network update method is as follows:

$$\begin{cases} \text{online: } Q(s,a|\theta^{Q}), & \text{gradient update } \theta^{Q} \\ \text{target: } Q(s,a|\theta^{Q'}), & \text{soft update } \theta^{Q'}. \end{cases} \tag{3}$$

N transition data are randomly sampled from replay memory as mini batch training data of online policy network and online Q network. Single transition data in mini batch is represented by （s_i, a_i, r_i, s_{i+1}） .

(5) In critical, calculate the Q gradient of the online Q network:

The loss of Q network is defined as:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^{Q}))^2;$$

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) . \tag{4}$$

The gradient for L and θ^{Q} can be obtained:$\nabla_{\theta^{Q}} L$, where the calculation uses the of target policy network μ' and target Q network Q' .

(6) Update online Q: update θ^{Q} with Adam optimizer.

(7) In the actor, calculate the policy gradient of the policy network:

$$\nabla_{\theta^{u}} J_{\beta}(\mu) \approx \frac{1}{N} \cdot (\nabla_{\alpha} Q(s,a|\theta^{Q})|_{s=s_i, a=w(s_i)} \cdot \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_i}) . \tag{5}$$

(8) Update online policy network: update θ^{μ} with Adam optimizer.

(9) The parameters of target network adopt the method of soft update:

$$\begin{cases} \theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{cases} \tag{6}$$

In general, DDPG algorithm uses Actor-Critic framework to iterate the training of policy network and Q network through the interaction among environment, actor and critic.

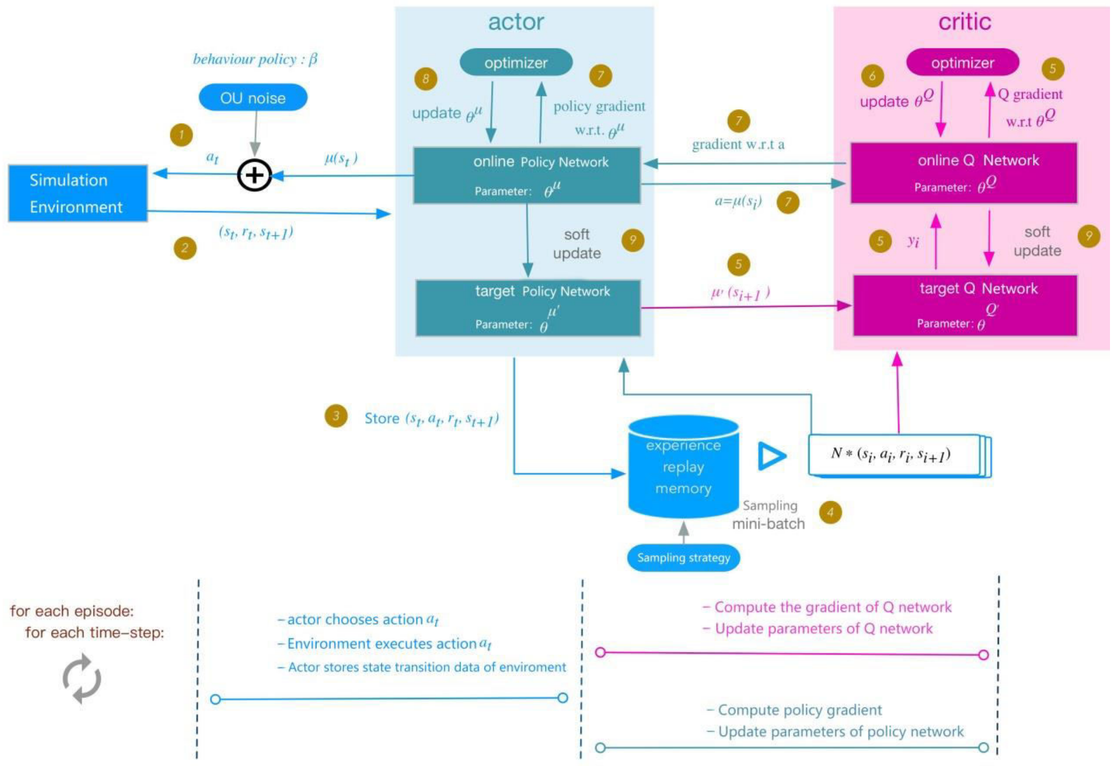
actor
critic
behaviour policy : β
OU noise
optimizer
update θ^{μ}
policy gradient w.r.t. θ^{μ}
optimizer
update θ^{Q}
Q gradient w.r.t θ^{Q}
gradient w.r.t a
a_t
$\mu(s_t)$
Simulation Environment
online Policy Network Parameter: θ^{μ}
online Q Network Parameter: θ^{Q}
$a=\mu(s_i)$
(s_t, r_t, s_{t+1})
soft update
y_i
soft update
target Policy Network Parameter: $\theta^{\mu'}$
$\mu'(s_{i+1})$
target Q Network Parameter: $\theta^{Q'}$
Store (s_t, a_t, r_t, s_{t+1})
experience replay memory
$N*(s_i, a_i, r_i, s_{i+1})$
Sampling mini-batch
Sampling strategy
for each episode:
for each time-step:
- actor chooses action a_t
- Environment executes action a_t
- Actor stores state transition data of enviroment
- Compute the gradient of Q network
- Update parameters of Q network
- Compute policy gradient
- Update parameters of policy network

Figure 1: The algorithm flow chart of DDPG

3.2 Asynchronous Advantage Actor-Critic (A3C)

Reinforcement learning algorithm A3C [20] is to put actor critical into multiple threads for synchronous training, which can make effective use of computer resources to increase training effectiveness and solve the problem of actor critical non convergence; Moreover, A3C can solve the control of continuous action space and is suitable for manipulator control tasks. A3C creates multiple parallel environments and allows multiple agents to update the parameters in the main structure on these parallel environments at the same time. The agents in parallel have no interference with each other, but the parameter update of the main structure is interfered by discontinuity. Hence, the correlation of update is decreased and the convergence is improved. Each core of the server is a thread, which runs the program in multiple cores simultaneously, doubling the running speed. Actor-Critic uses two different networks: actor and critic. A3C puts the two networks together, that is, input state S, output state value V and corresponding policy π. A3C algorithm uses dominance function to accelerate convergence. The expression of dominance function is shown in Eq. (7):

$$A(S,t) = R_t + \gamma R_{t+1} + \cdots + \gamma^{n-1} R_{t+n-1} + \gamma^n V(S') - V(S) \tag{7}$$

The gradient update of policy parameters is shown in Eq. (8):

$$\theta = \theta + \alpha \nabla_\theta \log\pi_\theta(s_t, a_t) A(S,t) + c\nabla_\theta H(\pi(S_t, \theta)) \tag{8}$$

The cumulative total of actor network local gradient is updated as Eq. (9):

$$d\theta = d\theta + \nabla_{\theta'} \log\pi_{\theta'}(s_i, a_i)\big(Q(s,i) - V(S_i, \omega')\big) + c\nabla_{\theta'} H(\pi(S_i, \theta')) \tag{9}$$

The local gradient of cumulative critical is updated as Eq. (10):

$$d\omega = d\omega + \frac{\partial (Q(s,i) - V(s_i, \omega'))^2}{\partial \omega'} \tag{10}$$

Among them, the corresponding parameters of the A3C neural network structure in the common part are θ and ω. The corresponding parameters of the A3C neural network structure of this thread, are θ′ and ω′.In addition, in order to ensure that the policy is fully explored and prevent premature entry into the suboptimal policy, the entropy term H of the policy π is added to the formula, which is the coefficient c.When the entropy value is large, the probability of each action is the same. Therefore, the agent cannot determine which action to perform. When the entropy value is small, an action will have higher probability than other actions, so the agent will choose the action with higher probability. In this way, adding entropy to the loss function will encourage agents to explore further, so as to avoid falling into local optimum.

3.3 Distributed Proximal Policy Optimization (DPPO)

DPPO algorithm [21] is a further improvement based on Proximal Policy Optimization (PPO) [2]. Its idea is similar to that of A3C, and it is also learned through multithreading. PPO is an improvement of the Trust Region Policy Optimization (TRPO) [22] algorithm.

TRPO algorithm is a method proposed by Shulman and others to solve the problem that the performance of ordinary policy gradient algorithm can not be monotonous and non decreasing. Shulman et al. Did not start with the update step of the policy gradient, but changed the idea: replace the optimization function. Through theoretical derivation and analysis, Shulman et al. found an alternative loss function, and finally transformed the reinforcement learning policy update step into an optimization problem as shown in Eq. (11):

$$\text{maximize}_{\theta} E_{s\ \pi_{\theta_{old}}, a\ \pi_{\theta_{old}}}\left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s,a), \qquad \text{subject to } E_{s\ \pi_{\theta_{old}}}[D_{KL}(\pi_{\theta_{old}}(\cdot|s)||\pi_{\theta}(\cdot|s))]\right. \leq \delta \tag{11}$$

The standard solution of TRPO is to approximate the objective function to the first order, expand the constraints to the second order by Taylor, and then use the conjugate gradient method to solve the optimal update parameters. However, when the policy is represented by deep neural network, the calculation of the standard solution of TRPO will be very large. Because the conjugate gradient method needs the second-order expansion of the constraints, the calculation of the second-order matrix is very large.

PPO is the first-order approximation of TRPO and can be applied to large-scale policy update. Shulman et al. Proposed a one-step method to solve TRPO, i.e. PPO, in nips in 2016. PPO uses the loss function derived by TRPO and updates the parameters by random gradient descent. The regular term is to control that the updated policy parameters are not too far from the current policy parameters. After that, Shulman et al. Introduced a new PPO algorithm, which further improved the alternative objective function and made the process more concise. The new surrogate loss function is shown in Eq. (12):

$$L(\theta) = \hat{E}_t[\min\left(r_t(\theta)\hat{A}_t\right), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{12}$$

where: $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$.

DPPO algorithm transforms PPO algorithm into multi-thread mode, uses multi-thread to collect data in multiple environments, improves data collection speed, and carries out multi-thread operation to improve the overall computing speed.

4 DDPG algorithm based on parameter noise

In DDPG, noise is added to the output of the policy network to disturb to increase exploration (as shown in Eq. (1)). However, adding noise should not only be limited to the output of the policy network, but also try to add noise to the network parameters θ to increase exploration, as shown in Figure 2. Let the parameters of the original policy network be θ, and then add noise $\epsilon \sim N(0, \sigma^2 I)$ to obtain the disturbed parameter $\tilde{\theta} = \theta + N(0, \sigma^2 I)$. The disturbed policy is recorded as $\tilde{\pi}$.

The action noise is independent of the input state s_t. No matter what the input state is, the noise is added according to certain rules (such as $\epsilon - \text{Greedy}、\text{Gaussion}$); The parameter noise is stable in a cycle. At the beginning of the cycle, add noise under the current policy parameters to get $\tilde{\theta}$ and then use $\tilde{\theta}$ to interact with the environment until the end of the cycle. Therefore, stable exploration is realized. For example, for the same input state s_t, under the action of action noise, the same state s_t will be input many times and the output results will be different; Under the action of parameter noise, if the same state is input many times, the same result can be obtained as long as $\tilde{\theta}$ is fixed.

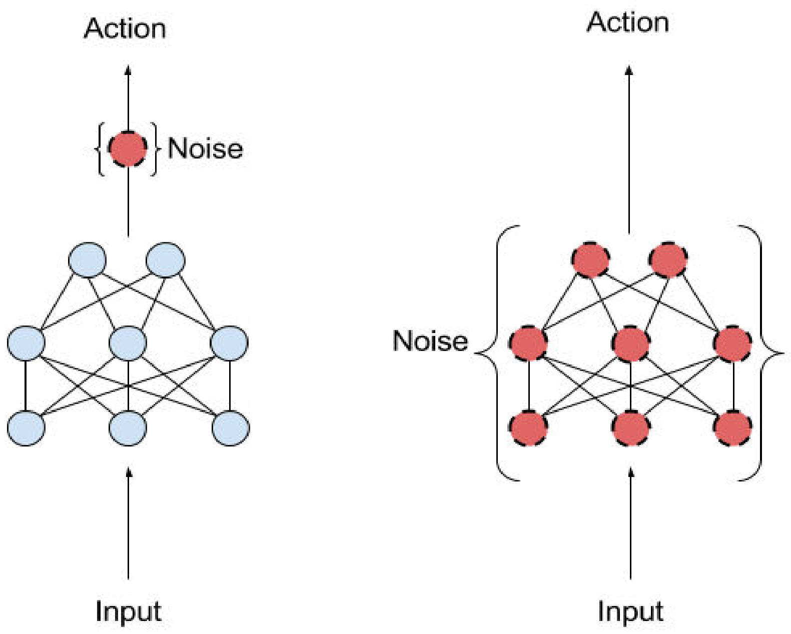

Figure 2: Action space noise (Left) and Parameters Space Noise (Right)

Adding parameter noise to the weight in the depth neural network will change the distribution of activation values in the front and back layers. This paper hopes to add the same noise to each layer, but the noise between each layer does not affect each other. Therefore, it is necessary to add LayerNormalization in each layer in the deep neural network to increase the stability of training.

When adding noise, it is necessary to determine the variance σ^2 of noise. If the parameters are adjusted manually, it will be very difficult. This paper measures the divergence between the policy $\tilde{\pi}$ corresponding to the parameter $\tilde{\theta}$ after adding noise and the original policy π. In training, if the value is less than a certain threshold, increase the variance: $\sigma_{k+1} = \alpha \cdot \sigma_k$; Otherwise, reduce the variance:$\sigma_{k+1} = \frac{1}{\alpha} \cdot \sigma_k$. Set $\alpha = 1.01$ in the experiment.

In ddpg, we associate the noise caused by parameter space disturbance with the noise caused by additive Gaussian noise. The way to measure the policy divergence $d(\pi, \tilde{\pi})$ before and after disturbance is as follows: the output of the policy network is the value of continuous action, and the square error is directly used to measure the difference between the output actions of the two strategies. The formula is shown in Eq. (13):

$$d(\pi, \tilde{\pi}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} E_s[(\pi(s)_i - \tilde{\pi}(s)_i)]^2} \tag{13}$$

Where $E_s[\cdot]$ is estimated according to a batch of states in the playback buffer（replay buffer）,and N represents the dimension of the action space. If noise $N(0, \sigma^2 I)$ is used in the action space, then $\tilde{\pi} \sim \mathcal{N}(\pi, \sigma^2 I)$. The square error is:$E[(\tilde{\pi} - \pi)^2] = E[(\tilde{\pi} - \mu(\tilde{\pi}))^2] = \mathrm{Varianse}(\tilde{\pi}) = \sigma^2$, therefore, the threshold is set to $\delta = \sigma$, resulting in an effective action space noise with the same standard deviation as the conventional Gaussian action space noise.

5 Control of two-link manipulator based on deep reinforcement learning

5.1 System structure

This system is divided into two parts: deep reinforcement learning algorithm and experimental simulation. The neural network in the system is trained by deep reinforcement learning, so that the algorithm can control the movement of the two link manipulator and finally reach the target position. The environment of the simulation part includes manipulator and target. Firstly, the control signal of the algorithm is received to make the manipulator move; Then, the motion is transmitted to the control algorithm, and the state variables and reward values are obtained by deep reinforcement learning according to the received information. With the continuous training, the parameters of the neural network are also updated, and the reward value obtained is constantly changing, as shown in Figure 3.

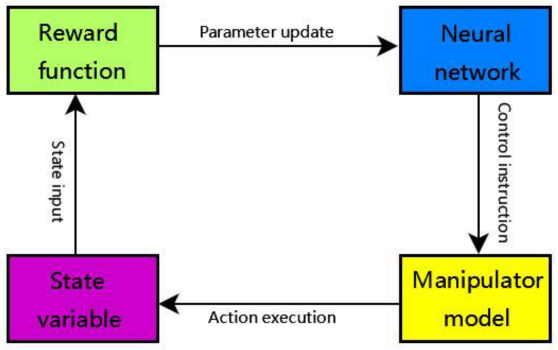

Figure 3: Structure diagram of manipulator control system

5.2 Two-link manipulator model

The model of the two-link manipulator can be established by D-H (Denavit, Hartenberg) method, and the model parameters are shown in Tab. 1.

Table 1: DH parameters of manipulator

link parameters	link 1	link 2
θ	θ_1	θ_2
d	0	0
a	100	100
α	$\pi/2$	0

According to the parameters, a two-link manipulator model can be established. The two links of this model are connected by a rotating pair, and the base part is a fixed rotating pair. The specific model is shown in Figure 4.

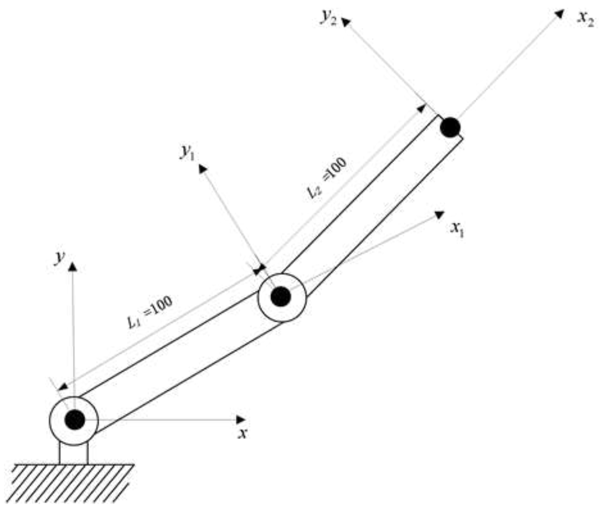

Figure 4: Two-link manipulator model

5.3 Deep reinforcement learning control method

Based on the characteristics of the two-link manipulator, we set up a suitable environment and build a control model of deep reinforcement learning algorithm. By setting the ideal state variable (state) and outputting the specified action (action), the quality of the action is analyzed to get the reward (reward). Then the parameters of the neural network should be updated to continue training.

(1) Reward

When the reinforcement learning controls the arm, it is necessary to manually set a better reward form. The reward is of great importance and will involve convergence problems. For the arm environment, it involves the position of the target point, the arm end and obstacles. The target position is (x, y), the arm end position is (x_2, y_2), then the opposite of the distance from the arm end to the target point is recorded as r_1, and its expression is as Eq. (14).

$$r_1 = -\sqrt{(x - x_2)^2 - (y - y_2)^2} \tag{14}$$

According to whether the end of the arm reaches the target, rewards are given. ε is used to indicate whether the goal is reached, and its value is {true, false}. When reaching the target position, the reward is 10, which is expressed as Eq. (15).

$$r_2 = \begin{cases} 10;\ \varepsilon = \text{true} \\ 0;\ \varepsilon = \text{false} \end{cases} \tag{15}$$

The final reward for the execution of this action equals to the sum of the above two rewards, and it is recorded as R, so the expression of reward is as follows.

$$R = r_1 + r_2 \tag{16}$$

(2) State

The relevant parameters of the robotic arm are selected as the state parameters of the environmental motion, and the convergence of the algorithm can be improved by selecting the appropriate parameters. The state variables here can be expressed as follows:, the distance between the end points of the two arms to the target, the four coordinate values of the end points of the two arms, and the distance from the end of the second arm to the target. According to whether the end point reaches the target, the state value 1 means that the target has been reached, on the contrary,0 means that the target has not been reached The state values observed by the design model this time are the above 7 states. By putting the target reward parameter and state parameter into the established environment model, the DDPG algorithm is used to control the movement of the established manipulator in the environment.

Finally, the reward parameter and the state parameter that have been set are added to the established environment, and the algorithm DDPG, A3C and DPPO control the movement of the robotic arm through the established environment.

6. Simulation experiment and result analysis

6.1 Simulation environment

The simulation environment of this article is built under Python, the pygelt module in gym is used to build a two-link manipulator model, and Python is used to complete the construction of the deep reinforcement learning model. The construction environment consists of a two-link robotic arm and a target. The top of the robotic arm is controlled by a deep reinforcement learning algorithm to reach the target. The built environment is shown in Figure 5.

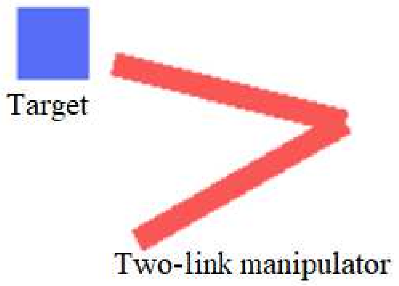

Figure 5: Motion control simulation environment of two-link manipulator

6.2 Experimental setup

The deep reinforcement learning framework of this article is TensorFlow, which is widely used in the field of deep learning, and the network model builds two fully connected layers.

The input of the neural network is the state, and the output is the action. The starting position of the robotic arm is random, the arm length of the two sections is 100, the arm width is 5, and the side length of the target is 20. The action angle ranges from -180° to 180°, the rotation speed is set to 1, and the freedom degree is 2.

Comparison of DDPG, A3C, and DPPO algorithms, the training parameters are set as follows: the maximum number of DDPG iterations (Episodes) is 2000, the maximum number of steps per episode is set to 300, the reward discount factor is set to 0.9, the learning rate of the actor part is 0.0001, the learning rate of the critic part is 0.0002, BATCH_SIZE is set to 16, and the number of neurons is set to 200. The A3C and DPPO training parameters are the same as the DDPG settings. To compare the action noise-based DDPG algorithm with the parameter noise-based DDPG algorithm, the training parameters are set as follows: the maximum number of episodes (Episodes) is 2000, the maximum number of steps per episode is set to 300, the reward discount factor is set to 0.9, and the learning rate of the actor part It is 0.0001, the learning rate of the critic part is 0.0002, BATCH_SIZE is set to 128. the number of neurons is set to 128 in 1st and 2nd layer , and 64 in 3rd and 4th layer.

When the top of the robotic arm reaches the target, the number of reaching the target is increased by 1. When it is continuously maintained within the target for less than 50 steps, the round ends, which will be regarded as a success.

6.3 Result analysis

At the beginning of training, the robotic arm conducts random search in all directions, occasionally reaching the target. In the middle of training, the end of the robotic arm can already grasp the target satisfactorily. For targets that are relatively off-site, the robotic arm still needs a certain amount of time to find the target, but it performances much faster than in the initial stage of training. In the later stage of training, the robotic arm has been able to quickly grasp the target, the grasping point is basically in the center of the target, and the reward target value of the action is also relatively high.

After training, the average value of rewards for every 100 episodes is taken, and the reward value change of the reinforcement learning model is shown in Figure 5, which reflects the convergence of the DDPG algorithm under the current settings for training the robotic arm task. It indicates that the reward value in the seek phase is unstable after the robotic arm is trained under the DDPG algorithm. In this process, the robotic arm will move at will, and the variance of the reward value is relatively large, resulting in convergence after about 150 episodes. When converging, the robotic arm reaches the target smoothly, and the reward value is stable at about 75. When the convergence continues, the reward value will also change slightly, and finally stable convergence is achieved. After continuing to establish the deep reinforcement learning A3C algorithm and the DPPO algorithm to train the robotic arm tasks, the results obtained according to the three algorithms are shown in Figure 6.

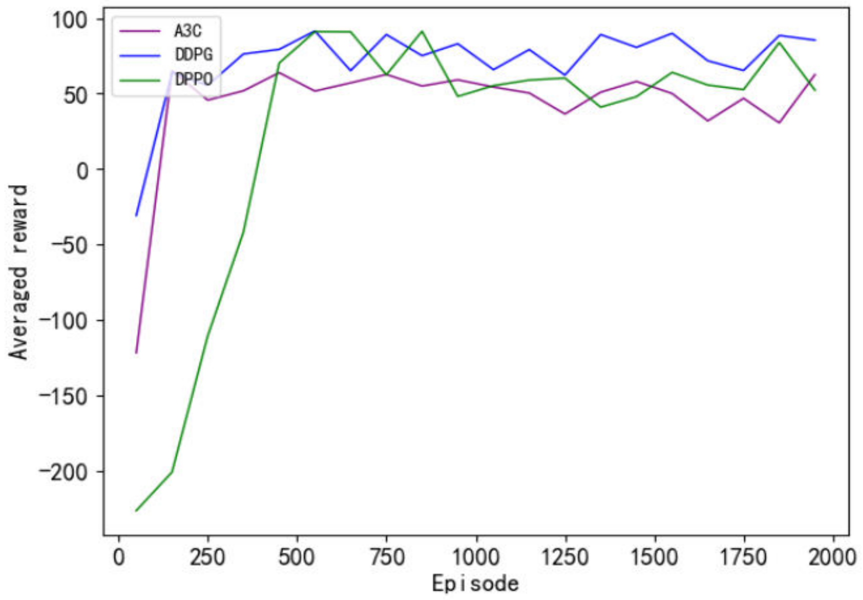

Figure 6: Reward values of different algorithms with different iteration times

In Figure 6, the reward results obtained from the A3C algorithm training show an upward trend after the task starts. After 150 iterations, the robotic arm gradually approaches the target, and its reward value begins to decrease. During the approach to the target, the reward value keeps increasing. It fluctuates around 60, the fluctuation frequency is small, and the number of times the manipulator reaches the target is small, and the obtained convergence is not ideal. Continue to use the DPPO algorithm for training.

After the task of training the robotic arm, the reward return value obtained by the DPPO algorithm changes as follows: at the beginning of training, the robotic arm carries on behavior search, and the reward return value slowly increases; after more than 500 iterations, the robotic arm explores the target position, and the reward return value occurs change suddenly, the reward value converges around 50. At this time, the robotic arm reaches the target. After a period of convergence, the reward value changes drastically, and it continues to converge.

Comparing the results obtained after training of the three algorithms, it can be found that the A3C algorithm has fast training speed and fast exploration of the target due to multi-threaded calculations, but the convergence of reaching the target position is not that good; during the robot arm movement task, the DPPO algorithm has a better performance in the pre-training situation, it can show better convergence, but the reward value fluctuates greatly in the later stage, and it takes more iterations to stabilize. The DDPG algorithm has a slower training speed and a longer exploration time in the early stage in the training process, but Stable convergence will occur in the later period, and the convergence period is shorter. Through comparison, it can be seen that although the DDPG algorithm has unstable convergence during training, it has better convergence than the other two algorithms, and the obtained convergence return value is more stable, so the DDPG algorithm is more suitable for experimental models. Finally, the DDPG algorithm is chosen to continue training the robotic arm environment.

The change of the reward value of the DDPG algorithm based on action noise and the DDPG algorithm based on parameter noise (taking the average value of rewards per 100 episodes) is shown in Figure 7. Before about 400 episodes, after the two DDPG algorithms were trained on the manipulator, the reward value in the search target phase was unstable. In this process, the robotic arm will move at will, and the variance of the reward value will be large. After about 400 episodes, the convergent speed of the reward value of the DDPG algorithm based on parametric noise is much higher than that of the DDPG algorithm based on motion noise. The manipulator using the DDPG algorithm of parametric noise reaches the target faster.

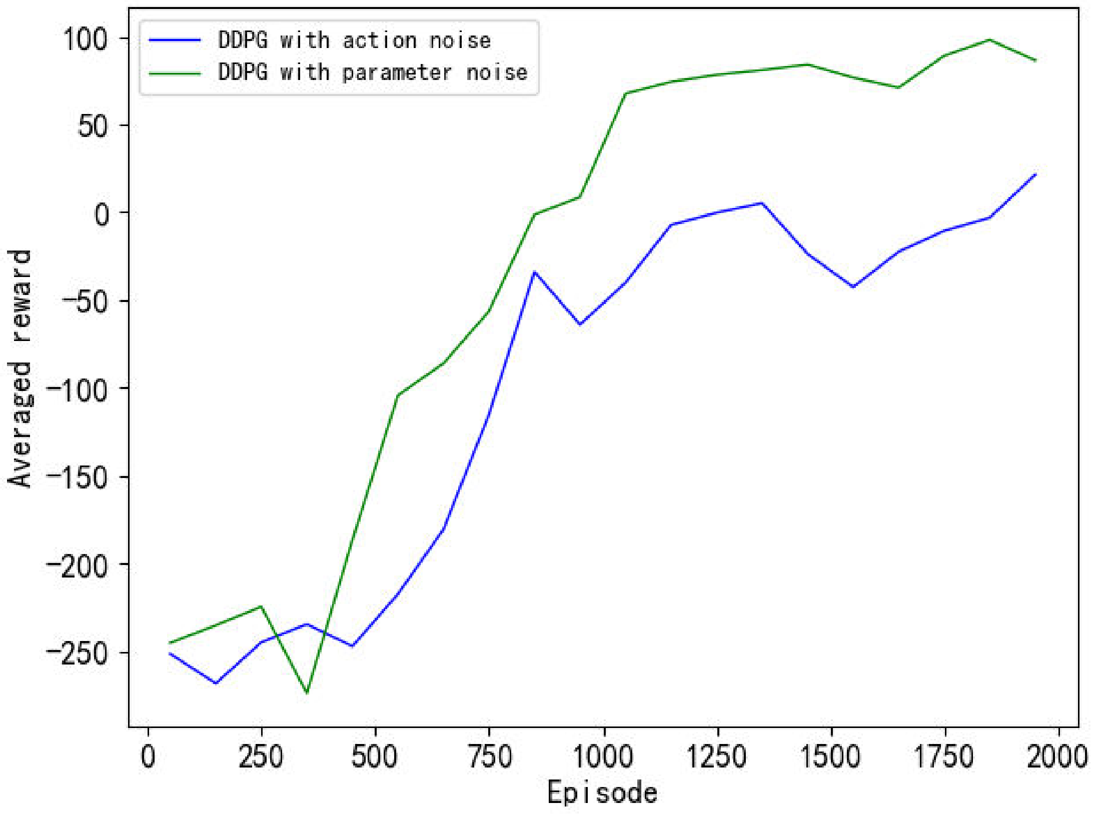

Figure 7: Reward values of DDPG with different noise

7 Conclusions

This paper takes the two-link manipulator as the research object, transforms the two-link manipulator's target search problem into the manipulator control problem, and proposes a deep deterministic strategy gradient algorithm based on the parameter noise to solve the manipulator control problem. This paper establishes the control system and manipulator model, establishes the simulation model and the reinforcement learning model in gym and TensorFlow respectively, and uses the motion noise-based DDPG algorithm, A3C algorithm and DPPO algorithm for training comparison. Finally the DDPG algorithm is chosen for further research. The parameter-based noise DDPG algorithm is creatively proposed to improve the applicability and stability of the motion noise-based DDPG algorithm for the tasks of the manipulator. The simulation experiment results show that through continuous algorithm debugging and model training, the two-link manipulator can reach the target smoothly. In comparison with the traditional control method, the gradient algorithm based on the parameter noise depth deterministic strategy proposed in this paper is more efficient. What's more, it is suitable for a variety of environments, and has wider applicability. However, this paper we only verifies the control of the parametric noise DDPG algorithm on the manipulator in a two-dimensional plane, and it can be further verified in a three-dimensional space in the future.

Acknowledgements: This work is sponsored by: (1) the team for science & technology and local development service of Nantong Institute of Technology under Grant No. KJCXTD312; (2) the science and technology planning project of Nantong City under Grant No. JC2021132, JCZ21080，JCZ21058，JCZ20186，JCZ20172, JCZ20151, JCZ20148, JCZ20144, MS22021028; (3) the project of philosophy and social science research in colleges and universities in Jiangsu Province under Grant No. 2021SJA1629 ; (4) the key project of college students' innovation and entrepreneurship training program of Jiangsu Province in 2021 under Grant No. 202112056003Z; (5) the college students' innovation and entrepreneurship training program of Nantong Institute of technology in 2021 under Grant No. XDC2021036; (6) the scientific research backbone training project of Nantong Institute of Technology under Grant No. ZQNGG109.

I. Ethical approval

The problems and experiments studied in this paper do not involve ethical issues.

II. Funding details (In case of Funding)

(1) The team for science & technology and local development service of Nantong Institute of Technology under Grant No. KJCXTD312; (2) the science and technology planning project of Nantong City under Grant No. JC2021132, JCZ21080, JCZ21058, JCZ20186, JCZ20172, JCZ20151, JCZ20148, JCZ20144, MS22021028; (3) the project of philosophy and social science research in colleges and universities in Jiangsu Province under Grant No. 2021SJA1629 ; (4) the key project of college students' innovation and entrepreneurship training program of Jiangsu Province in 2021 under Grant No. 202112056003Z; (5) the college students' innovation and entrepreneurship training program of Nantong Institute of technology in 2021 under Grant No. XDC2021036; (6) the scientific research backbone training project of Nantong Institute of Technology under Grant No. ZQNGG109.

III. Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

IV. Informed Consent

The authors' research institutions are known and have approved the authors to do this study.

References

1. Guo X. (2018). *The Study of Robotic Arm Control Policy Based on DQN (MS thesis)*. Beijing JiaoTong University, Beijing, China.
2. Li H. Y., Lin T. Y., Zeng B., Shi G Q. (2020). Control Method of Space Manipulator by Using Reinforcement Learning. *Aerospace Control*, V.38, No.188(06):40-45.
3. Leng S., Wu K., Ju H H. (2019). Overview of Manipulator Kinematics Modeling and Solving Method. *Journal of Astronautics*, 040 (011):1262-1273.
4. Yao J., Ke L T., Ren J. (2020). Adaptive gain control algorithm based on deep reinforcement learning. *Journal of Zhejiang Sci-Tech University*, 43(5):647-652.
5. Li N. (2006). *Study of Multi-agent Learning Problem Based on Reinforcement Learning (MS thesis)*. Jiangnan

University, Wuxi, China, 2006.

6. Liu Q. Y. (2019). *Deep Reinforcement Learning Based Object Grasping of Dual-Arm Robot (MS thesis)*. Shandong University, Jinan, China.
7. Bai X. N. (2019). Research on robot arm control method based on hierarchical reinforcement learning. *Automation and Instrumentation*, 10:121-123.
8. Fu X. K. (2020). *The Grasp Manipulation Strategy of Robotic Arm Based on Deep Reinforcement Learning (MS thesis)*. Zhejiang University, Hangzhou, China.
9. Wang H. T. (2019). *Self-Learning Control of Mechanical Arm Based on Reinforcement Learning (MS thesis)*. Harbin Institute of Technology, Harbin, China.
10. Liu Y.. *Robotic Arm Grasping Policy Based on Deep Reinforcement Learning (MS thesis)*. Beijing JiaoTong University, Beijing, China, 2020.
11. Zhou Q. J., Liu M. L., Li X. M., Z H. (2020). Research on solid radioactive waste grasping method. *Application Research of Computers*, Vol.37, No. 349(11): 169-173.
12. Li H. Y., Zhao Z. L., Gu L. ,et al. (2019). Robot Arm Control Method Based on Deep Reinforcement Learning. *Journal of System Simulation*, Vol.31 (11):278-283.
13. NGO T Q, WANG Y N, MAI T L, et al. (2012). Robust adaptive neural-fuzzy network tracking control for robot manipulator. *International Journal of Computers Communications & Control*, 7(2):341-352.
14. KORMUSHEV P, CALINON S, CALDWELL D G. (2011). Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input. *Advanced Robotics*, 25(5):581-603.
15. ZHANG F Y., LEITNER J., MILFORD M., et al. (2020). Towards vision-based deep reinforcement learning for robotic motion control. https://arxiv.org/pdf/1511.03791.pdf.
16. MNIH V., KAVUKCUOGLU K., SILVERD, et al. (2020). Playing Atari with deep reinforcement learning. https://arxiv.org/pdf/1312.5602.pdf.
17. Volodymyr M., Koray K., David S., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533.
18. NAKANISHI J, CORY R, MISTRY M, et al. (2008). Operational space control: a theoretical and empirical comparison. *International Journal of Robotics Research*, 2008, 27(6) : 737-757.
19. KOBER J, BAGNELL J A, PETERS J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32 (11): 1238-1274.
20. LILLICRAP T P, HUNT J J, PRITZEL A, et al. (2020). Continuous control with deep reinforcement learning. https://arxiv.org/pdf/1509.02971v2.pdf.
21. SCHULMAN J, WOLSKI F, DHARIWAL P, et al. (2020). Proximal policy optimization algorithms. https://arxiv.org/pdf/1707.06347.pdf.
22. Schulman J, Levine S , Moritz P, et al. (2015). Trust Region Policy Optimization. *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, V.37:1889–1897. https://arxiv.org/pdf/1502.05477.pdf.