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

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Ensemble Bootstrapped Deep Deterministic Policy Gradient for Vision-Based Robotic Grasping

WEIWEI LIU*, LINPENG PENG*,JUNJIE CAO, XIAOKUAN FU, YONG LIU,AND ZAISHENG PAN
Institute of Cyber-Systems and Control,Zhejiang University, Hangzhou 310027, China.

Corresponding author: Yong Liu (e-mail: yongliu@iipc.zju.edu.cn)
* Authors contributed equally

ABSTRACT With sufficient practice, humans can grab objects they have never seen before through brain decision-making. However, the manipulators, which has a wide range of applications in industrial production, can still only grab specific objects. Because most of the grasp algorithms rely on prior knowledge such as hand-eye calibration results, object model features, and can only target specific types of objects. When the task scenario and the operation target change, it cannot perform effective redeployment. In order to solve the above problems, academia often uses reinforcement learning to train grasping algorithms. However, the method of reinforcement learning in the field of manipulators grasping mainly encounters these main problems: insufficient sample utilization, poor algorithm stability, and limited exploration. This article uses LfD, BC, and DDPG to improve sample utilization. Use multiple critics to integrate and evaluate input actions to solve the problem of algorithm instability. Finally, inspired by Thompson's sampling idea, the input action is evaluated from different angles, which increases the algorithm's exploration of the environment and reduces the number of interactions with the environment. EDDPG and EBDDPG algorithm is designed in the article. In order to further improve the generalization ability of the algorithm, this article does not use extra information that is difficult to obtain directly on the physical platform, such as the real coordinates of the target object and the continuous motion space at the end of the manipulator in the Cartesian coordinate system is used as the output of the decision. The simulation results show that, under the same number of interactions, the manipulators' success rate in grabbing 1000 random objects has increased more than double and reached state-of-the-art(SOTA) performance.

INDEX TERMS Robotic grasping, reinforcement learning, ensemble learning, Thompson sampling

I. INTRODUCTION

Robotic grasping is one of the most basic robot operation tasks. Many robot interaction problems begin with the need for robots to be able to grab objects. In the current work related to robotic arm grasping, because the data-driven [1] reinforcement learning (RL) method does not require prior knowledge of the grasping object and environment, it can quickly grasp different objects, which meets the needs of the robot to grasp the diversity of objects. Therefore, it has received more and more attention from academia.

However, the neural network needs to fit complex controllers that are difficult to manually design, and the inherent reward sparseness in the field of robotic arm grasping makes the RL algorithm extremely difficult to converge. Therefore, the traditional algorithms require a lot of data for learning algorithm requires a lot of data for learning, but the time for data collection is unbearable. For example, Google [2] used 8 robot arms in parallel and spent 2 months collecting 800k grasping attempts. Lerrel Pinto [3] used a robot for 700 hours of grasping data and collected 50k sets of data to train a convolutional neural network to predict the grabbing pose. Therefore, in order to improve the utilization of sample data and reduce the number of interactions with the environment, it is an important difficulty to promote the development of RL in the field of robot.

This article starts from the contradiction between exploration and exploitation of RL algorithms. In a typical robotic

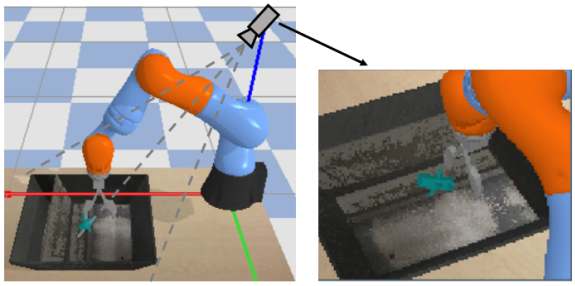


FIGURE 1: Robotic grasping simulation platform [4] and image observation. One object is generated each time in the basket, the success rate of grasping is calculated by one grasping opportunity.

grabbing environment (reward is sparse and requires a lot of interaction data), when the number of interactions with the environment is greatly reduced, each time an object is selected from 1,000 random objects of different shapes and placed in the basket, and the success rate of the robot grabbing this object from the basket exceeds 80%. The main contributions are as follows:

1. The algorithm takes RGB image of only one object instead of multiple objects compared with previous methods each time as an input. Under the same number of interactions with the environment, by grabbing one thousand different random objects, achieving outstanding generalization ability.

2. A multi-evaluation network integration algorithm, Ensemble Deep Deterministic Policy Gradient (EDDPG), is proposed. By using multiple critic network's heads, the Q value of the action is comprehensively evaluated, which suppresses the overestimation of the Q value caused by the Deep Deterministic Policy Gradient(DDPG) algorithm and reduces the fluctuation of the action Q value and improves the stability of the algorithm.

3. Based on the Thompson sampling idea [5] [6], multiple heads are designed for the critic network to output multiple Q-value functions so that the same action can be evaluated from different angles. When the variance of the Q value is large, the action needs to be resampled. Otherwise, the evaluation network is deemed to have evaluated the action correctly, and no further exploration is necessary, thereby improving the sample utilization, reducing the number of interactions with the environment. Therefore, in conjunction with EDDPG, the article proposes Ensemble Bootstrapped Deep Deterministic Policy Gradient (EBDDPG).

II. RELATED WORK

The latest developments in DRL have performed very well in many fields, such as Atari games [7] and Go games [8]. DRL has also had many success stories in manipulator operation control, both model-based [9] and model-free [10].

However, in many manipulators' training environments, parameters are used as input to speed up the training. For example, [11] [12] completes the tasks of stacking blocks and grabbing blocks in a simulation environment. The input parameters are the specific coordinates of each building block and the coordinates of the target point. Mahmood et al. [13] completed the task of moving the end of the manipulator to the target point on the physical platform of the UR5 robot arm. Its input parameters were also the specific coordinates of the target point. However, parameter input is limited to the simulation environment and the simple physical environment. In the more complex physical grasping environment, accurate information of the object cannot be obtained, so simple image input is a more general reinforcement learning solution. For example, [14] [15] [16] adopts the image observation in simulation or physical environment as the network input. After reinforcement learning and training, the motion of the end of the robot arm can be decided as the output to complete the task. However, the reinforcement learning algorithm for image input requires a lot of data training. Kalashnikov1 [17] also uses the improved version of the Double DQN network. After the training is completed, it can achieve a grasping success rate of more than 70% in the initial task of emptying the objects in the box. It is still the most cutting-edge achievement in the field. However, its experimental results depend on the massive data sets collected, with a total of about 58w grasping tracks, which occupy 4T hard disks, which shows that there are still deficiencies in the balance of exploration and exploitation in the learning process.

In this paper, in order to improve the exploitation of samples, learning from Demonstration(LfD) and Behavior Cloning(BC) and DDPG algorithm is used. Futhermore, for improving the algorithm's ability to explore the environment, based on Thompson's sampling idea, several heads are designed for the critic network to output multiple Q-value, so that the same action can be evaluated from different angles. That is: the distribution of the Q value corresponding to each action in a state is more certain, which means that the action in this state (and its subsequent effects) is certain that no further exploration is necessary. If the distribution of action value changes significantly, it means that the sampling of this action may not be sufficient, and further sampling is required, that is, deep exploration. In addition, in order to reduce the overestimation of the action's Q value in the RL algorithm of AC architecture, and to reduce the fluctuation of the Q value of the estimated action of multiple Q functions, the average Q value estimation method is also used.

III. BACKGROUND

In general, research on reinforcement learning problems is based on Markov decision processes(MDP) [18]. Under the condition of satisfying Markov property, Markov decision process can be described by tuple M = (S, A, P, R), where: S represents the set of all possible states in the environment; A represents the set of all possible actions that the agent can

perform; P represents the state transition probability, a more general expression is S × A → S; R represents the reward or punishment information obtained from the environment after performing an action in state s and reaching state s'. It is more commonly expressed as: S × A × S → R.

The dynamic MDP is as follows: the initial state of the agent is $s_0 \in$ S, the selection action $a_0 \in$ A is executed, and the environment enters the next state s_1 according to the state transfer function P, and at the same time, the reward r_0 is fed back to the agent, and the agent enters next process of decision. We can use τ to describe a state action interaction trajectory:

$$\tau = (s_0, a_0, s_1, a_1, ...) \tag{1}$$

The need to introduce a discount factor of γ (Discount Factor), its role is to assess the future cumulative return expectations:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t, \gamma \in (0, 1) \tag{2}$$

Given a Markov decision process M, the task of reinforcement learning is to find a control policy π: S → A to maximize the expected return.

A. DEEP Q-LEARNING ALGORITHM FOR CONTINUOUS SPACE

Deep Q-Learning(DQN) is a classic reinforcement learning algorithm. Its deep network can fit the Q function in a high-dimensional state very well. It has achieved outstanding results in many discrete action space environments. However, the algorithm needs to process $\max_a Q(s, a)$ when calculating TD-Target, as shown in Equation $y_i = r(s_i, a_i) + \gamma \max_{a_i'} Q_\theta\left(s_i', a_i'\right)$, which makes DQN cannot be directly used in the environment of continuous action space.

Combining the Cross-Entropy Method (CEM) [19] in the evolutionary strategy optimization algorithm with DQN to solve the max operation and realize the DQN algorithm's deployment on the robotic arm grasping platform.

The following two improvements are made: The optimal motion calculation TD-Target is selected in continuous space in combination with the CEM optimization algorithm. The introduction of a guiding strategy to increase the proportion of non-zero reward samples in the experience pool under the premise of ensuring exploratory to solve the problem of low exploration value and slow convergence in the initial stage of the algorithm.

B. DEEP DETERMINISTIC POLICY GRADIENT

Deep Deterministic Policy Gradient is an off-policy AC algorithm. The output of the policy network μ (s $|\theta^\mu$) is a deterministic action. The output of the evaluation network is an optimal state value function. The update process and Q-learning are the same, and the policy network is updated as follows:

$$\begin{aligned} &\bigtriangledown_{\theta^\mu} J \approx \bigtriangledown_{s_t \in \rho^\beta} \left[\bigtriangledown_{\theta^\mu} Q\left(s, a|\theta^Q\right)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right] \\ &= \bigtriangledown_{s_t \in \rho^\beta} \left[\bigtriangledown_a Q\left(s, a|\theta^Q\right)_{s=s_t, a=\mu(s_t|\theta^\mu)} \bigtriangledown_{\theta^\mu} \mu\left(s|\theta^\mu\right)|_{s=s_t}\right] \end{aligned} \tag{3}$$

DDPG refers to the practice of Deep Q-Learning(DQN) and uses experience pool playback for offline training. The data samples $\{s, a, r, s'\}$ generated during the interaction between the agent and the environment are put into the experience pool, and each time Randomly select a batch update value network and policy network from the experience pool. Like DQN, DDPG also uses a separate target value network and target policy network to calculate TD-error(Time Difference).

C. LEARNING FROM DEMONSTRATION AND BEHAVIOR CLONING

Due to the sparseness of the reward function, the agent cannot obtain positive feedback for a long time during the initial interaction. It can only conduct very low-efficiency exploration and rely on the positive feedback that occasionally occurs to guide the agent to optimize towards the goal slowly. In order to improve exploration efficiency, this article introduces imitation learning, also known as learning from teaching [20] [21], which is a very effective supervised learning method that can improve sample utilization efficiency. It has two main advantages: one is to improve the training efficiency by imitating expert samples; the other is to avoid artificially designing complex reward functions. In the imitation learning phase, the strategy can be efficiently trained to a reasonable level without considering the problem of the sparse reward function. At the same time, we also insert expert samples into the experience pool as a permanent experience. It, which is called DDPGfD, will not ablate like the Behavior Cloning(BC) algorithm.

The expert sample pool is denoted as $De = \{\tau_1, \tau_2, ..., \tau_i, ...\}$, where τ_i is an expert teaching trajectory, which can be decomposed into $\tau_i = \{s_i, a_i, r_i, s_{i+1}\}$. Because the expert sample's data size is minimal, the strategy of directly supervising learning and training is easy to overfit, and its test results are not good. Therefore, it can be performed on the data augmentation operation. Among them, $De \subseteq Daug$. For any $\{s_i, a_i, r_i, s_{i+1}\} \in De$, in addition to adding itself to the augmented data set, it is necessary to randomly add noise to its actions, as follows:

$$a_{aug} = a + N\left(0, \sigma^2\right) \tag{4}$$

And add $\{s, a_{aug}, r, s'\}$ into the data set D_{aug}, where σ is the standard deviation of the noise. For each sample data, formula (4) is processed about ten times, which can increase the original data set ten times, effectively enrich the diversity of the data set, and improve the robustness of the strategic after supervised learning.

In the BC-DDPG phase, the evaluation network is optimized based on TD-error as a loss function. The training of

the strategy network is divided into behavioral clone training and DDPG algorithm joint training. The behavior cloning part supervises and trains the action network in the DDPG network. The network takes states as input and action as output, and L2 loss function is used, which is formulated as follows:

$$L_{BC} = \frac{1}{N} \sum_{i=1}^{N_D} ||\pi(s_i), a_i||^2 \tag{5}$$

Among them, $\pi(s_i)$ is the output of the policy network, $<s_i, a_i>$ is a set of samples in the augmented data set, and N_D is the size of a batch in the behavior cloning process. After combining the two, the gradient of the policy network is:

$$\begin{aligned} \bigtriangledown_\theta L = &-\lambda_1 \bigtriangledown_{\theta_\pi} J(\theta) + \lambda_2 \bigtriangledown_{\theta_\pi} L_{BC} \\ &+ \lambda_3 \bigtriangledown_{\theta_\pi} L_{reg}(\theta_\pi) \end{aligned} \tag{6}$$

In the above formula, the first term is the gradient of the DDPG algorithm policy network. The second term is the gradient of the behavior cloning loss function. The third part is the L2 regularization term gradient of the policy network parameter. Its primary purpose is to prevent some policy network parameters from too large causes the network to overfit. $\lambda_1, \lambda_2, \lambda_3$ are the weight coefficients of the three gradients. In order to better balance the role of the DDPG algorithm and the behavioral cloning algorithm in different stages of training, λ_1, λ_2 in this section satisfy the following relationship:

$$\lambda_1 = 1 - \lambda_2, \lambda_2 = \eta_0 \eta_1^{\frac{t}{T}} \tag{7}$$

λ_2 means exponential decay, η_0 is its initial value, η_1 is the attenuation factor, t is the number of training steps, and T is a constant factor. In this form, behavioral cloning plays an absolute guiding role in the early stage, helping the network to converge and improve performance quickly. Gradually, it decays to 0 in the later stages to maximize the performance of the DDPG algorithm.

IV. ENSEMBLE DETERMINISTIC POLICY ALGORITHM BASED ON IMITATION LEARNING

The BC-DDPGfD(Replace with BC-DDPG below) algorithm proposed in the previous section effectively improves the sample utilization efficiency. Due to the introduction of behavioral cloning algorithms, the algorithm's performance can be quickly improved in the early stage, which significantly improves efficiency. However, there are still two shortcomings:

1. The first issue is stability. The DDPG algorithm is made up of two networks, the strategy network, and the evaluation network. The training of the strategy network depends heavily on the evaluation network's training results, which is very unstable during the training process. The network's variance during training is considerable, which directly affects the final performance of the policy network.

2. The second is exploratory issues. Expert samples are collected according to a specific rule strategy, and the rules can only cover partially successful scenes. Constrained by expert sample imitation learning, the agent's exploratory nature is affected to some extent, which will affect the training efficiency of the algorithm in the middle and late stages. Therefore, it is difficult for the algorithm to fully explore the environment to converge to the optimal solution.

A. ENSEMBLE DEEP DETERMINISTIC POLICY GRADIENT

To overcome the first disadvantage, this section first integrates the evaluation network and designs a single actor multi-critics network structure (Single Actor Multi Critics, SAMC) to improve the stability of network training; Single Actor Multi Critics Network Integration Algorithm Based on behavioral cloning (BC-DDPG: SAMC): The starting point of designing evaluation network integration is to perform gradient calculation on the evaluation network after the policy network integration, which can improve the stability of the algorithm. Assuming that the number of evaluation networks used is K, the K evaluation networks' average value can be used to estimate the state value function. Similarly, the average value of the K target evaluation networks corresponding to the K evaluation networks can be used to calculate TD-error, as follows:

$$\begin{aligned} Q_{avg}(s, a|\theta) &= \frac{1}{K} \sum_{i=1}^{K} Q_i(s, a|\theta_i) \\ Q_{avg}^{tar}(s, a|\theta^-) &= \frac{1}{K} \sum_{i=1}^{K} Q_i^{tar}(s, a|\theta_i^-) \end{aligned} \tag{8}$$

Among them, $Q_i(s, a|\theta_i)$, $Q_i^{tar}(s, a|\theta_i^-)$ are the output values of the ith critical networks and the target critical networks, and θ, θ^- are all the critical networks and the target critical networks parameters, respectively. The parameter set of the networks, θ_i, θ_i^- are the parameters of the ith evaluation network and the target evaluation network, respectively, $\theta_i \in \theta$, $\theta_i^- \in \theta^-$,Then their average TD deviation and cost function are:

$$\begin{aligned} \delta &= r(s, a) + \gamma Q_{avg}^{tar}(s, a|\theta^-) - Q_{avg}(s, a|\theta), \\ L_{avg}(\theta) &= \left(r(s, a) + \gamma Q_{avg}^{tar}(s', a'|\theta^-) - Q_{avg}(s, a|\theta)\right)^2. \end{aligned} \tag{9}$$

For K evaluation networks, when updating their respective parameters, the cost function can be obtained by weighing the average TD deviation of all the evaluation networks and their respective TD deviations. This fully considers the network's average performance to reduce network training time fluctuations while taking into account the differences between the evaluation networks. Also, it is necessary to consider that when the differences between the networks are too significant, additional unstable factors will be brought to the training of the strategic network and other evaluation networks. Therefore, an additional penalty term needs to be added to the cost function. The value is the mean square error between each evaluation network's output value and the average output value of all evaluation networks to ensure that the networks have relatively similar output results. In

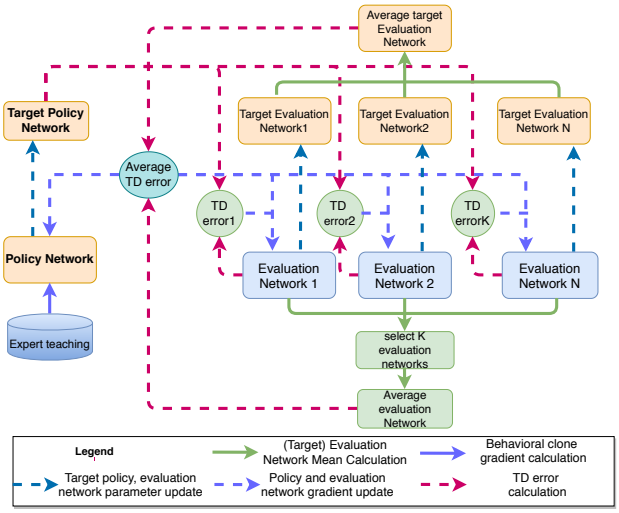


FIGURE 2: EBDDPG overall network structure diagram

summary, the cost function of the i-th evaluation network is as follows:

$$L(\theta_i) = \alpha L_{avg}(\theta) + \beta L_{td_i}(\theta_i) + \omega \left(Q_i(s,a|\theta_i) - Q_{avg}(s,a|\theta)\right)^2 \quad (10)$$

where α, β are weight coefficients, and their size is constrained to $\alpha + \beta = 1$. ω is the penalty factor of the penalty term, which can be set to a smaller value than α, β. L_{td_i} is the cost function formed by the TD error of the i-th critical network. The gradient calculation formula of the policy network is still shown in Equation 6, but the $Q(s,a,\theta)$ in the first part $\nabla_{\theta_\pi} J(\theta)$ needs to be replaced by $Q_{avg}(s,a,\theta)$, which can be changed to the following form:

$$\nabla_{\theta_\pi} J(\theta) = E_{s_t \in \rho^\beta}[\nabla_\alpha Q_{avg}\left(s,a|\theta^Q\right)|_{s=s_t,a=\mu(s_t|\theta^\mu)}, \nabla_{\theta_\mu}\mu(s|\theta^\mu)|_{s=s_t}]. \quad (11)$$

B. ENSEMBLE BOOTSTRAPPED DEEP DETERMINISTIC POLICY GRADIENT

The ensemble critics' network proposed in the previous chapter improves the stability of the algorithm. However, in the environment of extremely sparse reward, the exploratory nature of the RL algorithm is more important. Without changing the stability of the algorithm (still integrating the multi-critics network structure), in order to improve the explorability of the algorithm, similar to [5], based on Thompson sampling, an Ensemble Bootstrapped Deep Deterministic Policy Gradient algorithm was proposed.

According to the Bootstrapped DQN idea, an approximate prior distribution over Q-values function in DDPG. At the beginning of training, the algorithm will samples several critical networks from its approximate posterior as the best network to evaluate the action given by the actor-network, that is, the Q value. This is the Thompson Sampling heuristic's natural adaptation to the RL, allowing for extended (or deep) exploration. In this regard, we have established multiple critical networks to evaluate the same action from different angles. As shown in Figure 2, it is essential that each critical network is independent of each other and has its own parameters to train. This means that every critical network in the algorithm provides a temporally extended (and consistent) estimate of the value uncertainty via TD estimates.

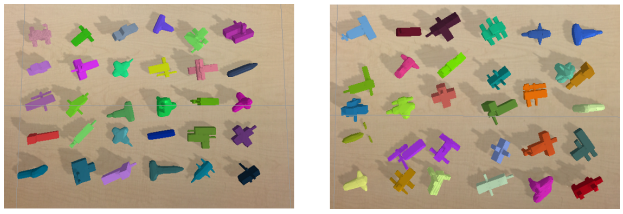


FIGURE 3: Examples of training objects (30/900) and testing objects (30/100)

Unlike Bootstrapped DQN, all our neural networks use the same dataset. However, several theories can explain that even without data bootstrapping, the multi-critic network still maintains significant diversity and can still help the algorithm to deep exploration. First, each critical network is independent, and its initialization parameters are random. Even with the same data set, it can still maintain enough diversity for deep exploration because the target network for each critic network is not the same, together with stochastic minibatch and flexible nonlinear representations, this means that even small initial differences become large due to different TD errors. As DDPG takes many frames to update its policy, many of its data points are stored in the same memory pool, so they have a lot of the same data. Even if the same memory pool is used, the algorithm's performance will not be significantly affected.

V. EXPERIMENTS

The simulation grabbing system is built based on PyBullet's existing Kuka 7DOF robotic arm. Control the attitude of the end of the robot arm relative to the base in Euclidean Space, which can be converted to the joint space displacement using its inverse kinematics solution function. The end of the robot arm uses two-finger parallel grippers, which can be closed by controlling the grippers' angle. The algorithm uses the standardized interface of OpenAI Gym [22] to obtain environmental observation values, reward values, output actions, etc. The reward is sparse, in each episode, if the robot grabs an object, it will get a reward of 1, otherwise it will be 0. Otherwise, it is zero. The Pybullet [23] physics engine generates an RGB image of all visible objects in the range by calculation. The image size set in this article is 64 * 64. In order to realize the grasping of general objects by the robotic arm, this article takes 1,000 kinds of objects in the Pybullet environment as the target grasping objects, of which 900 are used as objects in the training process, and the remaining 100 are used as objects that appear during the

Algorithm 1 BC-EBDDPG

Input: Expert teaching data experience pool R_D, evaluation network number K
Randomly initialize actor and critic network $\mu\left(s_t|\theta^\mu\right)$, $Q\left(s,a|\theta^Q\right)$ with weights θ^μ, θ^Q, and $Q\left(s,a|\theta^Q\right)$ has N heads $\{Q_n\}_{n=1}^N$. Masking distribution M
Initialize target network μ^-, Q^- with weigths $\theta^{\mu^-} \leftarrow \theta^\mu, \theta^{Q^-} \leftarrow \theta^Q$
Initialize replay buffer R, batch size N, N_D, weight parameters
for episode=1, E **do**
    Initialize environment and initial state s_0, initialize noise function $\mathcal{N}$
    Pick K value function to act using $K \sim \text{Uniform}\{1,\ldots,N\}$
    for steps t=1, T **do**
        Output action $a_t = \mu\left(s_t|\theta^\mu\right) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute the action a_t, interact with the environment to get the next observation s_{t+1} and reward r_t
        Sample bootstrap mask $m_t \sim M$
        Interactive samples $<s_t, a_t, r_t, s_{t+1}, m_t>$ are stored in the experience pool R
    end for
    for steps t=1, train-steps **do**
        Collect N samples from the experience pool R, and N_D samples from the teaching experience pool R_D
        Calculate the ensemble evaluation network according to (9), and update K critic head according to (10)
        Calculate the gradient of the policy network according to (6) and (11), and update its parameters
        Update target policy network parameters and K target evaluation network parameters
    end for
end for

test. Some training and testing objects are shown in Figure 3.

The training and testing of the algorithm are performed simultaneously. The evaluation criterion of the grabbing performance is the success rate of grabbing random objects in the test set (such as $\frac{N}{M}$, M grabs, and N successes). Because the interaction between the robot arm and the environment is time-consuming each episode in the simulation environment, it is not realistic to perform a success rate test at each step in the training process. In the actual test, select the grab success rate test every 5k steps, take M = 50 in each test and use the success rate of 50 grabs to approximately replace the current policy's grab success rate. Randomly select 100 objects from the test set, and the initial pose of the objects is also randomly generated.

A. COMPARISON OF DQN AND DDPG ALGORITHM PERFORMANCE UNDER DIFFERENT INITIAL DATA SETS

The purpose of the experiments in this section are as follows:

1. Test the impact of the number of initial empirical samples of different sizes on the training efficiency and final training results of the DQN algorithm;

2. Test DDPG and modified DQN algorithm performance.

Figure 4 shows that in the DQN algorithm with the increase of the data volume of the initial data set, the success rate of the robot arm capture increased. However, the increase is not significant, the data set from 10K to 100K, and the average grasp success rate only increased from 28% to 40%. Compared with the DDPG algorithm, in the case of 10K initial data set, the DDPG algorithm's performance is better than DQN algorithm under the 100K data set. Although the DQN algorithm has been modified so that it can be applied to continuous actions, the modified DQN algorithm only selects the action with the highest Q value among 10 actions at a time, which is the continuous actions are discretized. Not all continuous actions can be evaluated, which leads to the algorithm easily falling into a local optimum. However, the initial success rate of the DDPG algorithm is low, and it takes a long time to interact with the environment and collect data to improve the success rate of grasping.

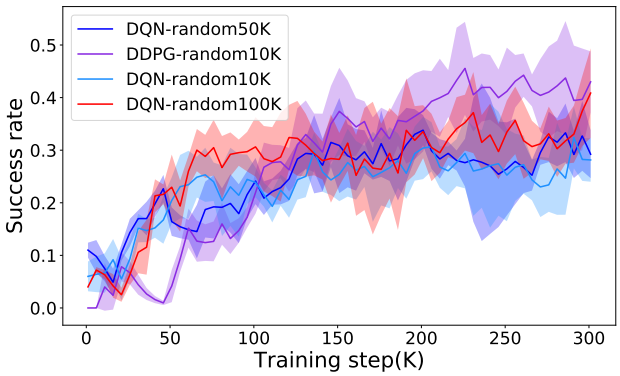


FIGURE 4: Improved DQN algorithm capture success rate under different initial data sets, and comparison with the DDPG algorithm. random nK means that the amount of data randomly obtained in the initial experience database is $n*1000$ interaction data with the environment.

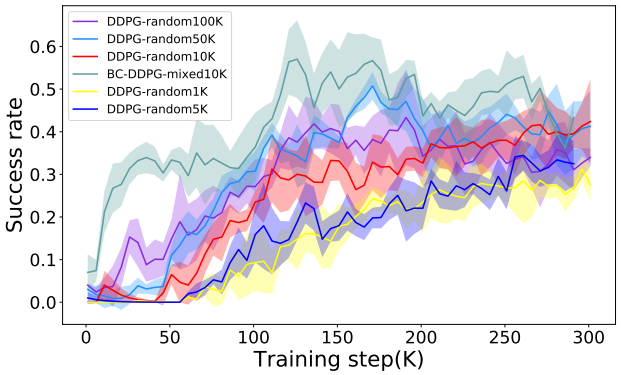


FIGURE 5: Impact of BC and different initial experience pools on the DDPG algorithm. mixed 10K means 2K expert samples and 8K random samples in the data set.

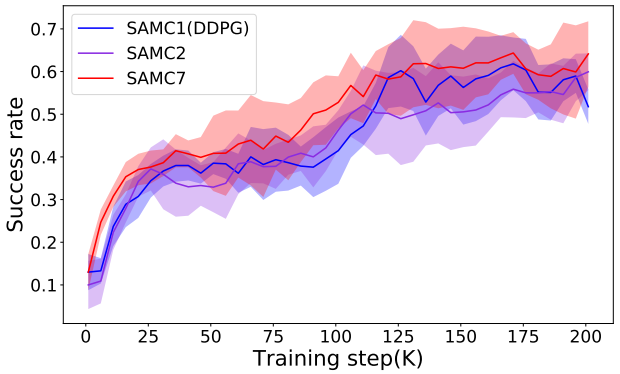


FIGURE 6: The impact of integrating different critics on the algorithm, SAMC7 represents an algorithm ensemble by a single actor and 7 critics, other representations are similar.

B. PERFORMANCE COMPARISON BETWEEN BC-DDPG AND DDPG ALGORITHMS WITH DIFFERENT INITIAL EXPERIENCE POOLS

The purpose of the experiments in this section is to test performance between BC-DDPG and DDPG algorithms with different initial experience pools. The DDPG algorithm and DQN algorithm experience pool samples are stored in the form $\{s_i, a_i, r_i, s_{i+1}\}$, so 100w interactive samples collected using the strategy π_E are directly reused, and from which 1K, 5K, 10K, 50K, 100K interaction trajectories are selected as the initialized experience samples. The average return during the training process is shown in Figure 5.

As can be seen from Figure 5, first, similar to the training of the DQN algorithm, the final performance of the model is positively related to the size of the initial sample in the experience pool. With the increase of the amount of data in the initial experience pool, from 1K to 10K, the success rate of the algorithm has increased from 25% to 40%. However, the amount of data in the initial experience pool has subsequently increased (such as 50K and 100K data)), which has little effect on the effectiveness of the algorithm; Secondly, although a large amount of data in the initial experience pool is reduced to 10K. Since it contains 2K expert data, the BC-DDPG algorithm can still enhance the validity of the data and improve the final effect of the DDPG algorithm with a faster speed. Note that using mixed data here is that purely using expert data is very easy to cause the algorithm to overfit. The initial data sets of all algorithms below are the same mixed data sets as here.

C. THE INFLUENCE OF THE NUMBER OF ENSEMBLE CRITICS ON THE EDDPG ALGORITHM

The purpose of the experiments in this section is:

1. Verify whether the integration of the evaluation network can improve network training stability under a single strategy?

2. How does the number of different evaluation networks affect the training speed and final performance of the network?

During the experiment, the evaluation network quantity K was set to 1, 2, 7 for testing. When the quantity is 1, BC-DDPG: SAMC degenerates into the BC-DDPG algorithm. During the training process, the same initialization parameters are selected between different evaluation networks, and only the number of networks of the critic is changed. The success rate curve of the algorithm is shown in Figure 6.

As can be seen from Figure 6, the BC-DDPG algorithm's performance has been further improved by the integration of multiple evaluation networks. It maintains the advantage of behavior cloning training at the beginning of training, and its training efficiency is very high. In the middle and late stages of training, the integration of the multi-evaluation network makes the update of the policy network more stable. Fewer or more network integrations will reduce the effectiveness of the algorithm. The reason is that when the number of evaluation heads in the integrated network is small, the performance of a single network will directly affect the overall performance when it is weak. when the number of evaluation heads is large, what it learns is the average performance of each evaluation network, which may fall into a local optimum.

D. EBDDPG ALGORITHM PERFORMANCE

Although the EDDPG algorithm proposed in the previous section has enhanced the stability of the algorithm, the search performance of the algorithm has not improved. In order to improve this situation, this paper designs the EBDDPG algorithm.

1. Does EBDDPG improve the explorability of the algorithm without changing the algorithm's stability and improve the success rate of grasping?

2. Is the more choice possible, the better the exploration performance of the algorithm?

3. Is there a relationship between the number of criticals and the number of criticals chosen to evaluate the actor?

From Figure 7, it can be seen that, firstly, the algorithms SAMC7-C1 are more exploratory than the algorithms SAMC7-C7. However the former has only one critical actu-

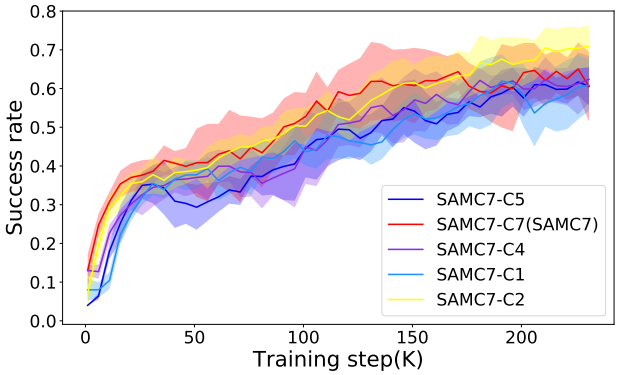


FIGURE 7: Impact of the change in the number of selected critics on the performance of the algorithm with the same number of designed critics, SAMC7-C5 represents an algorithm ensemble by a single actor and 7 critics, but only 5 of them are ensemble to evaluate the action in each iteration, other representations are similar.

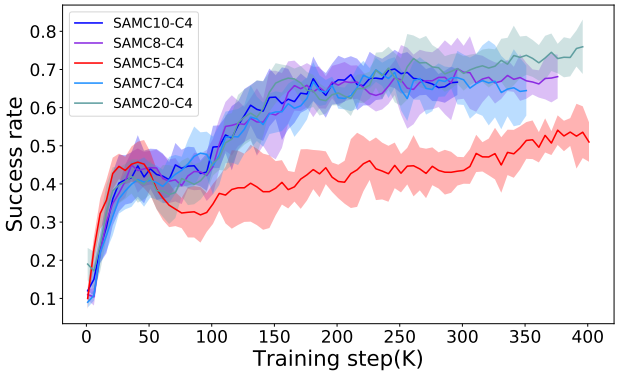


FIGURE 8: Impact of the change in the number of designed critics on the performance of the algorithm with the same number of selected critics

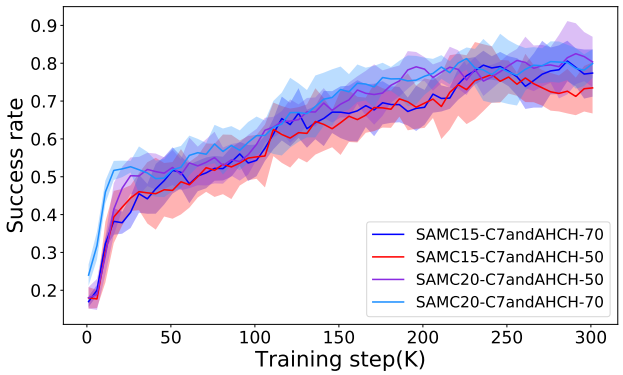


FIGURE 9: Impact of the change in the number of designed critics on the performance of the algorithm with the same number of selected critics, AHCH-50 indicates that the actor and critic network undergo a hard update every 50 steps.

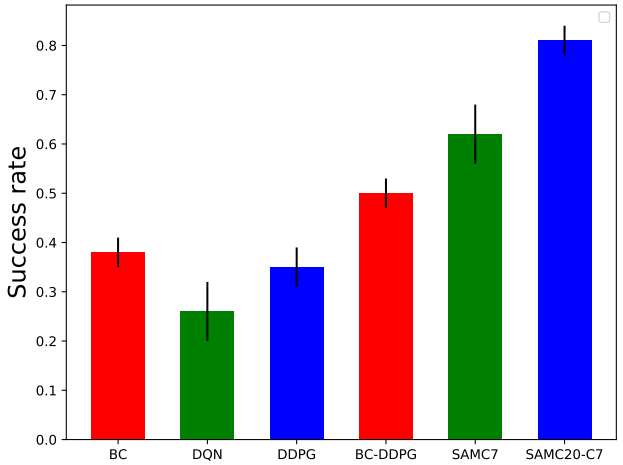


FIGURE 10: Performance comparison of different algorithms

ally to evaluate the Q value of the actor, it is easy to make the Q value oscillate, and then affect the overall stability of the algorithm. Secondly, the algorithm SAMC7-C2 is more exploratory and stable than the algorithm SAMC7-C1, so the former has a higher success rate than the latter. Although the algorithm SAMC7-C2 is less stable than SAMC7-C7, explorability is better than the latter. If the algorithm only needs to maintain specific stability, improving the explorability of the algorithm will improve the performance of the algorithm. Thirdly, although the algorithm SAMC7-C2 is less exploratory than SAMC7-C4 or the same as SAMC7-C5, its performance is better than the latter two, because the more the number of critics selected, the more it learns for the average performance of each evaluation network, will fall into local optimal.

From Figure 8, all algorithms only select 4 critics, that is, when the stability of the algorithm is relatively consistent, as the number of designed critics increases, the number of combinations is C_5^4, C_7^4, and C_8^4, the number of choices increases from 5, 35 to 70. At this time, the algorithm's exploration performance is continuously increasing, and its improvement effect is also getting larger; however, after the number of critics is increased to 10, 20, the algorithm's improvement effect is reduced, which is no longer apparent. It shows that the exploration ability of the algorithm has a limit.

As can be seen from Figure 9, when the number of critics designed is sufficient, other variables of the algorithm (such as updating parameter steps or increasing the number of critics selected) have little effect on the algorithm, i.e., the algorithm has better robustness and improves the stability of the algorithm.

E. COMPARISON OF DIFFERENT ALGORITHMS

The result of Table I is that the algorithm uses different random seed experiments to obtain the average value after five experiments. Where Ieds represents the initial empirical

TABLE 1: Final performance comparison of each algorithm

Algorithm	Ieds	Ts	Srogtrs	Srogtes
DQN	1W	30W	30.2%	26.2%
	5W	30W	34.3%	31.3%
	10W	30W	39.5%	40.5%
DDPG	1K	30W	26.5%	28.1%
	5K	30W	35.5%	30.6%
	1W	30W	34.2%	35.3%
	5W	30W	39%	41%
	10W	30W	43.7%	39.4%
BC	1W	30W	40.1%	38%
BC-DDPG	1W	30W	52.2%	50%
EDDPG(SAMC7)	**1W**	**30W**	**67.8%**	**62%**
IEDDPG(SAMC20-C7)	**1W**	**30W**	**87.5%**	**81.1%**

data scale, Ts represents the training step, Srogtrs represents the training set object success rate, and Srogtes represents the test set object success rate. In general, the SAMC20-C7 algorithm has the best effect. As can be seen from Figure 10 and Table I, compared with the original DDPG algorithm, under the same number of interactions between the robot and the environment, the average grasping success rate has increased from about 35% to 81%, which has more than doubled. This shows that the EBDDPG algorithm can still greatly reduce the interactive samples when achieving SOTA effects.

VI. CONCLUSIONS

In this paper, the end-to-end decision-making process of universal object grasp is studied using the method of deep reinforcement learning. Under the above constraints, the method of intensive learning will mainly encounter four problems: insufficient sample utilization, significant fluctuation of grasp success rate, and limited exploration. The research of this paper focuses on these three problems.

For the first problems, this paper designs a pilot strategy with a success rate of about 20% to replace completely random exploration improves sampling efficiency, combine BC and LfD algorithm, BC-DDPG can achieve about 45% grasp success rate; Given the last two problems, the BC-DDPG: SAMC(EDDPG algorithm) network structure is designed, through the integration of the critics' network, inhibits the problem of overestimation of Q value inherent in the AC architecture algorithm, and thus improves the stability of the algorithm, and the article design BC-EBDDPG algorithm, based on Thompson sampling thought, evaluates its Q value from different angles for the same action, and realizes the function of exploration.

In this paper, some improvement algorithms are proposed for the end-to-end decision-making process of universal object grasping in the basic intensive learning algorithm. Although the final grasp success rate has been improved, and the amount of interactive data and interaction time have been reduced, when the model is tested, the majority of the grasp process follows the correct grasp trend, because the monolith camera is missing depth information, it is difficult to judge the relative attitude relationship between an unknown size object and the end of the robotic arm from the image.

In the future, a depth camera can be used to replace a monocular camera for experiments, or the observation of a multi-view camera can be used as input. Furthermore, the texture information of the grabbing object used in the simulation environment is not rich enough, and it needs more color and shape information. The data set of the object can be operated subsequently to enhance its texture information. In addition, we will set up sim-to-real grasp interaction platform and do more elaborate real grasping experiments. Meanwhile, we will refine our method to reduce the interaction steps while still to keep the best performance in the simulation scenario when to expand our method to real scenario.

REFERENCES

[1] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," arXiv preprint arXiv:1703.09312, 2017.

[2] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 421–436, 2018.

[3] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in 2016 IEEE international conference on robotics and automation (ICRA), pp. 3406–3413, IEEE, 2016.

[4] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6284–6291, IEEE, 2018.

[5] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in Advances in neural information processing systems, pp. 4026–4034, 2016.

[6] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling," arXiv preprint arXiv:1707.02038, 2017.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," nature, vol. 518, no. 7540, pp. 529–533, 2015.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., "Mastering the game of go with deep neural networks and tree search," nature, vol. 529, no. 7587, pp. 484–489, 2016.

[9] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in International Conference on Machine Learning, pp. 2829–2838, 2016.

[10] S. Bansal, R. Calandra, K. Chua, S. Levine, and C. Tomlin, "Mbmf: Model-based priors for model-free reinforcement learning," arXiv preprint arXiv:1709.03153, 2017.

[11] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in Advances in neural information processing systems, pp. 5048–5058, 2017.

[12] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6292–6299, IEEE, 2018.

[13] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," arXiv preprint arXiv:1809.07731, 2018.

[14] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al., "Scalable deep reinforcement learning for vision-based robotic manipulation," in Conference on Robot Learning, pp. 651–673, 2018.

[15] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al., "Reinforcement and imitation

learning for diverse visuomotor skills," arXiv preprint arXiv:1802.09564, 2018.
[16] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4238–4245, IEEE, 2018.
[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255, Ieee, 2009.
[18] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," in Advances in neural information processing systems, pp. 345–352, 1995.
[19] R. Y. Rubinstein and D. P. Kroese, The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer Science & Business Media, 2013.
[20] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al., "Deep q-learning from demonstrations," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[21] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," Robotics and autonomous systems, vol. 57, no. 5, pp. 469–483, 2009.
[22] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," arXiv preprint arXiv:1606.01540, 2016.
[23] E. Coumans, Y. Bai, and J. Hsu, "Pybullet," Available on: https://pypi.org/project/pybullet/. Retrieved, vol. 5, p. 30, 2019.

LIU WEIWEI was born in Hefei city. In 2019, he entered the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University, to pursue the Ph.D. degree in electronics and information.

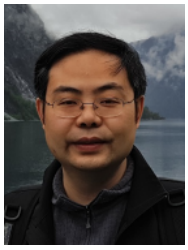
At present, his main research directions are artificial intelligence, decision-making and control.

LINPENG PENG is currently working toward the M.S. degree in control engineering with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China.

His current research interests include robotic grasping, computer vision and 5G network applications.

XIAOKUAN FU received the B.S. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2017, where he is currently pursuing the M.S. degree with the Department of Control Science and Engineering, Institute of CyberSystems and Control.

His current research interests include robotic manipulation, robotic vision, and reinforcement learning.



YONG LIU received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively. He is currently a Professor with the Department of Control Science and Engineering, Institute of Cyber Systems and Control, Zhejiang University.

He has published more than 30 research papers in machine learning, computer vision, information fusion, and robotics. His latest research interests include machine learning, robotics vision, information processing, and granular computing.

ZAISHENG PAN born in July 1971, from Deqing, Zhejiang, senior engineer. In July 1993, he graduated from Zhejiang University with a bachelor's degree in production process automation (automation). In March 1996, he graduated from Zhejiang University with a master's degree in industrial automation.

Since April 1996, he has been in the Department of Control Science and Engineering, Zhejiang University and engaged in scientific research and product development.