Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2020: 1–13

DOI: 10.1177/1729881419898342
journals.sagepub.com/home/arx
SAGE

Efficient hindsight reinforcement learning using demonstrations for robotic tasks with sparse rewards

Guoyu Zuo[1,2], Qishen Zhao[1,2], Jiahao Lu[1,2] and Jiangeng Li[1,2]

Abstract

The goal of reinforcement learning is to enable an agent to learn by using rewards. However, some robotic tasks naturally specify with sparse rewards, and manually shaping reward functions is a difficult project. In this article, we propose a general and model-free approach for reinforcement learning to learn robotic tasks with sparse rewards. First, a variant of Hindsight Experience Replay, Curious and Aggressive Hindsight Experience Replay, is proposed to improve the sample efficiency of reinforcement learning methods and avoid the need for complicated reward engineering. Second, based on Twin Delayed Deep Deterministic policy gradient algorithm, demonstrations are leveraged to overcome the exploration problem and speed up the policy training process. Finally, the action loss is added into the loss function in order to minimize the vibration of output action while maximizing the value of the action. The experiments on simulated robotic tasks are performed with different hyperparameters to verify the effectiveness of our method. Results show that our method can effectively solve the sparse reward problem and obtain a high learning speed.

Keywords

Robot learning, reinforcement learning, sparse reward, CAHER, demonstrations

Date received: 3 October 2019; accepted: 9 December 2019

Topic: Robot Manipulation and Control
Topic Editor: Andrey V Savkin
Associate Editor: Bin He

Introduction

Reinforcement learning (RL)[1] has shown impressive results in numerous simulated tasks ranging from attaining superhuman performance in video games[2,3] and board games[4] to learning complex motion behaviors.[5,6] However, sparse reward has always been a challenging problem for RL on robotic tasks. In robot learning, a sparse reward problem naturally arises when the robot is hoped to be able to achieve some binary goals such as moving an object to a desired location. The form of sparse reward is often easy to be specified, but in many situations it does not guide the robot to effectively explore and obtains the learned policy to realize the desired goal instead of getting stuck in local optima.[7,8]

A common way to deal with sparse rewards in RL is to combine an off-policy RL algorithm with Hindsight Experience Replay (HER).[7] The HER method randomly replaces the original goal with the achieved goal with a certain probability, so that the agent can learn not only from the successful experiences but also from part of the failed experiences. HER greatly accelerates the exploration of agents without reward signals. However, in multi-goal tasks, due to the diversity of the goals and the randomness

[1] Faculty of Information Technology, Beijing University of Technology, Beijing, China
[2] Beijing Key Laboratory of Computing Intelligence and Intelligent Systems, Beijing, China

Corresponding author:
Guoyu Zuo, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China.
Email: zuoguoyu@bjut.edu.cn

of the RL algorithm, HER might take a very long time to discover how to reach specific areas in the state space. An improved HER method[9] combining curiosity with priority mechanism is proposed to improve both the performance and sample efficiency of HER. But this method inherently believes that both the real and hindsight experiences have the same effects and arbitrarily puts more focus on the underrepresented achieved states. Another improvement of HER is ARCHER,[10] which compensates for the bias in HER by giving more rewards to hindsight experiences, but this method may bring harm to the final performance of the algorithm for some complex tasks.

Imitation learning (IL)[11–13] is a well-studied field in robotics. Such methods do not require a reward function, which can significantly speed up the learning of the robot. A wide variety of IL methods have been proposed in the last few decades. The simplest IL method among those is behavior cloning (BC),[14,15] which learns an expert policy in a supervised fashion without environment interaction during training. BC can be the first IL option when a large amount of high-quality demonstrations are available. However, when only a small amount of demonstrations are available or the quality of demonstrations is low, this method fails to imitate expert behavior due to the compound error.[16] Since it is often difficult to provide sufficient high-quality demonstrations in the real-world environment, the application of BC is limited seriously.

Inverse reinforcement learning (IRL)[17–19] is another widely used IL method, which can overcome the problem of compound error. The goal of IRL is to extract a reward function from those demonstrations and then train a policy to maximize the cumulative reward.[20–22] Apprenticeship learning[20] uses the maximum marginal method to solve the reward function, which is to find a hyperplane for dividing the demonstration policy and ordinary policy into two categories and maximize the marginal between them. Subsequently, to solve the problem of artificial base in IRL, neural network is used to replace artificial base in order to improve the expression ability of the reward function.[23] However, since the IRL algorithm needs to solve a RL problem in the inner loop, its huge computational complexity makes it difficult to be applied to complex tasks with high-dimensional spaces.

In recent years, an emerging IL method based on IRL has emerged, namely generative adversarial imitation learning (GAIL).[24] This method combines RL with generative adversarial networks.[25] The generator network uses RL to generate the action policy, and the discriminator network is used to discriminate the generated policy and the expert policy, and finally the generated policy converges to expert policy. Since GAIL has achieved the state-of-the-art performance on a variety of continuous control tasks, the adversarial IL framework has become a popular choice for IL.[26–30] GAIL has achieved amazing results on complex tasks, but it has serious instability during training, which needs a long training time and strong training skills.

In addition, other methods[8,31] combining RL with IL use demonstrations to accelerate the exploration of agents in the environment with sparse rewards, but they are too dependent on the quality of demonstrations, and the final performance is not satisfactory.

In order to address the challenges mentioned above, we put forward a novel RL algorithm based on Twin Delayed Deep Deterministic policy gradient algorithm (TD3).[32] The method use a new experience replay mechanism called Curious and Aggressive Hindsight Experience Replay (CAHER) to improve the sample efficiency and solve the sparse reward problem. Moreover, we leverage the demonstrations to accelerate learning and design an action loss to make the robot action as smooth as possible. We conducted several different experiments in robotic environment to evaluate the proposed algorithm.

The remainder of this article is organized as follows. Some preliminary knowledge is described. In the "Method" section, we introduce the proposed approach in three parts in detail. The experiment evaluation is conducted in the next section. Finally, the last section concludes the article.

Background

Markov decision process and RL

Considering an agent interacting with an environment and assuming that the environment is fully observable, a Markov decision process is defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $p(s_{t+1}|s_t, a_t)$ are transition probabilities, $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. A policy π maps a state to an action, $\pi : \mathcal{S} \to \mathcal{A}$.

At the beginning of each episode, an initial state s_0 is sampled from the distribution $p(s_0)$. Then, at each time step t, the agent performs an action a_t at the current state s_t, which follows the policy $a_t = \pi(s_t)$. A reward $r_t = r(s_t, a_t)$ is produced by the environment E and the next state s_{t+1} is sampled from the distribution $p(\cdot|s_t, a_t)$. The reward might be discounted by the factor γ at each time step. The goal of the agent in RL is to maximize the cumulative reward $G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$ by exploring in different states and learning the policy $a_t = \pi(s_t)$.

So far, many RL algorithms are proposed to solve this problem, most of which involve estimating the action value after taking action a_t

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}[G_t | s_t, a_t] \\ &= \mathbb{E}_{r_t, s_{t+1} \sim E}[r_t + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]] \end{aligned} \tag{1}$$

Equation (1) is the Bellman equation, where Q^{π} is the state-action value function which shows the action value after taking action a_t under state s_t.

Twin Delayed Deep Deterministic policy gradient

Deep deterministic policy gradient (DDPG) includes an actor network $\pi(s_t)$ with the parameters ϕ, a critic network $Q(s_t, a_t)$ with the parameters θ, and a replay buffer R as a set of tuples (s_t, a_t, r_t, s_{t+1}) for each transition experienced. To stabilize learning, a target critic network $Q'(s_t, a_t)$ with the parameters θ' and a target actor network with the parameters ϕ' are used. DDPG alternates between running the policy to collect experiences and updating the parameters. Obviously, the DDPG algorithm can be divided into two parts, actor and critic.

For the critic part, its purpose is to evaluate the value of the action taken in the current state, thereby to guide the actor network to select the optimal action. The loss function of critic network is as follows

$$y = r_t + \gamma Q'_{\theta'}\Big(s_{t+1}, \pi(s_{t+1})\Big) \tag{2}$$

$$L = \frac{1}{N}\sum\Big(y - Q_\theta(s_t, a_t)\Big)^2 \tag{3}$$

The main purpose of the actor part is to learn a policy to maximize accumulated rewards. For the purpose of enhancing exploration, the action $a_t = \pi_\phi(s_t) + \mathcal{N}$ is taken as the output of the actor part, where $\mathcal{N}$ is the Ornstein–Uhlenbeck noise. The actor parameters are updated by using the policy gradient

$$\nabla_\phi J = \frac{1}{N}\sum \nabla_{a_t} Q_\theta(s_t, a_t)|_{a_t=\pi(s_t)} \nabla_\phi \pi_\phi(s_t) \tag{4}$$

However, as described in Q-learning,[33] due to the Q function estimation error $\mathbb{E}_\varepsilon[\max_{a'}(Q(s', a') + \varepsilon)] \geq \max_{a'} Q(s', a')$, the algorithm has a high bias after multiple updates. The same problem exists in the AC methods.[34–36] Based on DDPG, Scott Fujimoto proposes TD3,[32] which has the following improvements:

- Two Q functions are learned and the smaller is used when updating the parameters of the critic networks to avoid overestimation of the action values

$$\begin{aligned} y_1 &= r_t + \gamma Q_{\theta'_1}(s_{t+1}, \tilde{a}) \\ y_2 &= r_t + \gamma Q_{\theta'_2}(s_{t+1}, \tilde{a}) \end{aligned} \tag{5}$$

$$y = r_t + \gamma \min_{i=1,2} Q_{\theta'_i}(s_{t+1}, \tilde{a}) \tag{6}$$

- To reduce the errors in multiple updates, the actor network should be updated at a lower frequency than the critic network. The modification is to update the policy and target networks after a fixed number of updates to the critic networks.
- TD3 emphasizes on the notion that similar actions should have similar values. The target policy smoothing, which adds a small amount of random noise to the target policy, is used to reduce the error in the value function estimation process

$$\tilde{a} = \pi_{\phi'}(s_{t+1}) + \varepsilon \tag{7}$$

Hindsight Experience Replay

HER is an experience replay method which can be used to overcome the learning difficulties caused by the use of sparse rewards and avoid complex reward projects. Different from the traditional RL methods, HER is proposed with a new parameter goal which consists of desired goal and achieved goal. The desired goal represents the task that the agent should accomplish. The achieved goal represents the task that the agent has completed at the current time. The key idea of HER is as follows: at some moment, although the agent has not achieved the desired goal, it has completed achieved goal. At this time, the desired goal can be replaced by the achieved goal so that the method can transform failed experiences into successful experiences and learn from them. In HER, even though the desired goal is not completed at present, if the learning process is repeated, the agent will complete the desired goal in the final so as to complete the task with the sparse rewards.

Gaussian mixture model

The Gaussian mixture model (GMM)[37,38] is a probability model that assumes that all data are derived from K Gaussian distributions with unknown parameters, mathematically: $\rho(\mathbf{x}) = \sum_{k=1}^{K} c_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Every Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. c_k are the mixing coefficients. The parameters of GMM can be estimated by the expectation maximization (EM) algorithm.[39]

Method

In this section, we describe the proposed method in detail and give its complete algorithm flow in Algorithm 1.

Curious and Aggressive Hindsight Experience Replay

Motivation. The inspiration of combining curiosity with RL comes from two aspects. First, neurology[40] has shown that curiosity can enhance the ability of learning. For example, when people learn to play basketball, they practice repeatedly in a trial-and-error fashion. In memory, people are more interested in those different episodes and focus more on these episodes. This curiosity mechanism has been proven to speed up learning. Second, the inspiration of adding a curiosity mechanism to RL agents comes from supervised learning. In the face of complex imbalanced datasets, standard supervised learning algorithms, such as neural networks, fail to learn useful information from the datasets. An effective solution is to oversample the samples in the underrepresented class. Zhao and Tresp[9] propose a Curiosity-Driven Prioritization (CDP) framework to focus

Algorithm 1. TD3 from demonstrations based on CAHER.

Given: an off-policy RL algorithm: TD3
a reward function r: $r(s,a,g) =$ -1(fail), 0(success)
demonstration buffer: D
Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer R
Initialize density model GMM
for *epoch* $= 1, M$ **do**
  for *episode* $= 1, N$ **do**
    sample a goal g and initial state s_0
    for $t = 1, T$ **do**
    select action with exploration noise $a \sim \pi_\phi(s_t||g) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r_t, new state s_{t+1}
    calculate the densities ρ for the collected trajectory $(s_t||g, a_t, r_t, s_{t+1}||g)_{t=0}^{T}$ using equation (9)
    store the trajectory $(s_t||g, a_t, r_t, s_{t+1}||g, \rho)_{t=0}^{T}$ in R
    for *step* $= 1, L$ **do**
      sample trajectories $\mathcal{T}1 \ldots \mathcal{T}n$ from R and D according to equation (10)
      sample transitions (s_t, a_t, r_t, s_{t+1}) from $\mathcal{T}1 \ldots \mathcal{T}n$
      sample virtual goals $g^h \in \{s_{t+1} \ldots s_{T-1}\}$ at a future timestep in $\mathcal{T}1 \ldots \mathcal{T}n$
      recalculate reward $r_t^h = \lambda_h \times r(s_t, a_t, g^h)$
      $\tilde{a} \leftarrow \pi_{\phi'}(s_{t+1}) + \varepsilon, \varepsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
      $y \leftarrow r_t^h + \gamma min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \tilde{a})$
      update critics: $\theta_i \leftarrow argmin_{\theta_i} \mathbb{E}_{(s_t,a_t)\sim R\cup D}[(y - Q_{\theta_i}(s_t, a_t))^2]$
      if *step mod d* **then**
        update ϕ by the deterministic policy gradient:
        $\nabla_\phi J = \mathbb{E}_{(s_t,a_t)\sim R\cup D}[\nabla_{a_t} Q_{\theta_1}(s_t, a_t)|_{a_t=\pi(s_t)} \nabla_\phi \pi_\phi(s_t) - 2\lambda \nabla_\phi \pi_\phi(s_t)]$
        update target networks: $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \qquad \phi' \leftarrow \tau\phi + (1-\tau)\phi'$
  train the density model using the collected trajectories in R
  update the densities in R using the trained model

more attention on the underrepresented trajectories. Our method is studied on the basis of the CDP framework.

Curiosity ranking. At the beginning of each episode, the agent uses a policy to explore the environment and store the trajectory in the replay buffer. A complete trajectory $\mathcal{T}$ is represented as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ and contains a lot of s_t in a episode, where t is the time step $t \in \{0, 1, .., T\}$. And the density of a trajectory depends on the states, $s_0, s_1, \ldots, s_T$.

After the trajectories are collected, we can use these trajectories to fit the density model. The model fits on the data in the replay buffer every epoch and refreshes the density for each trajectory. When a trajectory is stored in the replay buffer, the density model predicts its density ρ

$$\rho = \mathrm{GMM}(\mathcal{T}) = \sum_{k=1}^{K} c_k \mathcal{N}(\mathcal{T}|\mu_k, \Sigma_k) \tag{8}$$

where $\mathcal{T}$ contains $s_0, s_1, \ldots, s_T$ and each $\mathcal{T}$ has the same length. GMM can also be replaced with some variants of GMM.[41]

Then the trajectory is normalized as follows

$$\rho_i = \frac{\rho_i}{\sum_{n=1}^{N} \rho_n} \tag{9}$$

where N is the number of trajectories in the replay buffer.

After normalizing the trajectory density, the density value along with the trajectory is stored in the replay buffer for later prioritization.

During training, the agent puts more focus on the underrepresented achieved states. So we adopt the complementary trajectory density

$$\bar{\rho} \propto 1 - \rho \tag{10}$$

Furthermore, all the trajectories are ranked according to their complementary trajectory density values and the ranked numbers are used as the probability for sampling, which means that the underrepresented trajectories with low density are easier to be replayed. In this way, for off-policy RL, the agent puts more focus on the rare achieved states in the replay buffer and improves its sample efficiency.

Finally, when the transitions of those trajectories are sampled, the HER method is used to update the off-policy RL algorithm.

Aggressive rewards. Although curiosity enhances the exploration ability of the algorithm, it inherently believes that both the real and hindsight experiences have the same values. The cause of bias in HER analyzed in ARCHER[10] is that both experiences cannot be rewarded in the same way. In our method, we

incorporate the idea of aggressive rewards into curiosity to eliminate the bias and increase the sample efficiency of the RL methods. Specifically, in traditional HER methods, the hindsight experience has the form of $(s_t, g^h, a_t, s_{t+1}, r_t^h)$, where the hindsight reward r_t^h is recalculated for the hindsight goal g^h as follows

$$r_t^h = r(s_t, g^h, a_t) = \begin{cases} 0 & \text{when } f_{g^h}(s_{t+1}) = 1 \\ -1 & \text{when } f_{g^h}(s_{t+1}) = 0 \end{cases} \quad (11)$$

where $f_{g^h} : \mathcal{S} \to \{0, 1\}$ indicates whether the goal g^h has been realized in state s_t. The true objective of the agent is to perform action a_t to reach state s_{t+1} so as to let $f_{g^h}(s_{t+1}) = 1$, with which it receives a successful reward signal from the environment.

But in our method, $r_t \neq r_t^h$, and we use the parameter λ_h to weight r_t^h

$$r_t^h = \lambda_h \times r(s_t, g^h, a_t) \quad (12)$$

where the parameter λ_h should guarantee $\lambda_h \times r(s_t, g^h, a_t) > r(s_t, g^h, a_t)$ to make hindsight experiences play a more important role in training process.

TD3 from demonstrations

In traditional TD3, the actor network and the critic network are updated by sampling a mini-batch experiences from the replay buffer. In our method, the trajectories are stored in the replay buffer as $(s_t||g, a_t, r_t, s_{t+1}||g, \rho)$, where the symbol $||$ denotes the concatenation operation. During replay, the agent uses CAHER to change the original goal g in the sampled experiences to some achieved goals with a certain probability. Here CAHER uses the future sampling mode as HER. Furthermore, according to the idea of IL,[13,21,42,43] demonstrations are used to speed up our method. Therefore, the experiences for replay are sampled not only from the replay buffer but also from the demonstration buffer, that is, $(s_t, a_t, r_t, s_{t+1}) \in R \cup D$.

To update the critic network, two one-step off-policy evaluations are used and the smaller is used to avoid the overestimation of action values

$$y = r_t + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \tilde{a})) \quad (13)$$

$$\tilde{a} = \pi_{\phi'}(s_{t+1}) + \varepsilon, \varepsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c) \quad (14)$$

and the critic loss is in the following form

$$L_{critic} = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim R \cup D}[(y - Q_{\theta_i}(s_t, a_t))^2] \quad (15)$$

To update the policy, the policy gradient is used as follows

$$\nabla_\phi J = \mathbb{E}_{(s_t, a_t) \sim R \cup D}[\nabla_{a_t} Q_{\theta_1}(s_t, a_t)|_{a_t = \pi(s_t)} \nabla_\phi \pi_\phi(s_t)] \quad (16)$$

Action loss

In TD3, the main purpose of the actor network is to maximize the value of the output action. However, simply maximizing the action value function will result in dehumanized and excessively stereotyped behaviors. Therefore, the action loss in our method which is the square of the actor network output is added into the loss function of the actor network in order to optimize the output actions when updating the gradients. The loss function of the actor network is as follows

$$L_{actor} = -\mathbb{E}_{(s_t, a_t) \sim R \cup D}[Q_{\theta_1}(s_t, a_t) - \lambda \pi_\phi(s_t)^2] \quad (17)$$

and the gradient step is taken with respect to

$$\nabla_\phi J = \mathbb{E}_{(s_t, a_t) \sim R \cup D}[\nabla_{a_t} Q_{\theta_1}(s_t, a_t)|_{a_t = \pi(s_t)} \nabla_\phi \pi_\phi(s_t) - 2\lambda \nabla_\phi \pi_\phi(s_t)] \quad (18)$$

During training, the actor network maximizes the output of the critic network by minimizing the loss function, that is, maximizing the value of the output action. At the same time, since the action loss is added into the loss function, the vibration of the output action is restrained, that is, the smoothness of the output action is maintained.

Experimental evaluation

In this section, we first describe the experimental environment and the tasks to be completed by the agent in detail. Then, the network architecture and training details are explained. Finally, the push and pick-and-place tasks are conducted in the robotic environment to answer the following three questions:

- Does our method have advantages over other ones in terms of performance and learning speed?
- For our method, what effect does different settings of demonstrations have on the experimental results?
- What is the effect of the action loss?

Experimental environment

The environment we used is the robotic simulations provided by OpenAI Gym,[44,45] using the MuJoCo physics engine.[46] The 7-DOF Fetch robotic arm in the environment has a two-fingered parallel gripper, and there are several manipulation tasks with different goals to be completed. Figure 1 illustrates the push and pick-and-place tasks to be conducted in our experiments.

State s_t: the state $s_t \in \mathbb{R}^{25}$ includes the Cartesian position ($\mathbb{R}^3$) and linear velocity ($\mathbb{R}^3$) of the robot end, and the Cartesian position ($\mathbb{R}^2$) and linear velocity ($\mathbb{R}^2$) of the robot's two-finger gripper, the object's Cartesian position ($\mathbb{R}^3$), rotation ($\mathbb{R}^3$) using Euler angles, linear velocity ($\mathbb{R}^3$) and angular velocity ($\mathbb{R}^3$), as well as the object's position ($\mathbb{R}^3$) relative to the gripper.

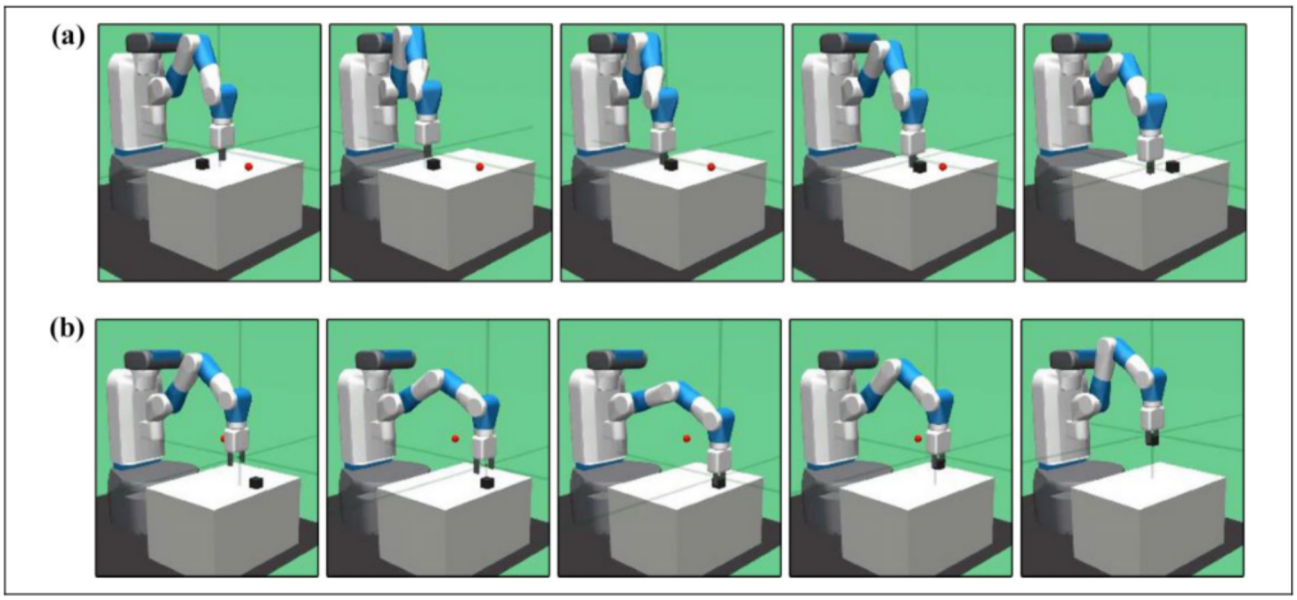

Figure 1. Experimental tasks: (a) Push: the fixture of manipulator is locked and the manipulator needs to use the locked fixture to push the object to the target position. (b) Pick-and-place: the fixture of manipulator is unlocked, and the manipulator needs to clamp the object and place the object to the target position.

Goal g: the goal $g \in \mathbb{R}^3$ is desired position of the object.

Achieved goal ag: the achieved goal ag is the goal that the robot has currently achieved, which is the three-dimensional target position.

Action a_t: the action $a_t \in \mathbb{R}^4$ mainly includes four types of control information. The first three specify the desired gripper movement in the Cartesian coordinates and the last controls the opening and closing of the gripper.

Reward r_t: a sparse reward is used as the reward function, which is shown as follows

$$r_t = \begin{cases} 0 & \text{when } d \leq 5 \text{ cm} \\ -1 & \text{when } d > 5 \text{ cm} \end{cases} \tag{19}$$

This is a binary reward function that shows whether the task has been completed or not. The 5 cm threshold is the default setting in Fetch robotic arm environment and commonly used in the evaluation of the simulated robotic tasks. When the distance between the object and the target is less than the threshold, it is considered that the object is placed on the target position.

Network architecture and training details

The actor and critic networks are designed with three fully connected hidden layers, consisting of 256 *ReLU* neurons in each layer. The tan h activation is used in the output layer of the actor network. To encourage exploration, Gaussian noise is added to the output of the actor network. We use a replay buffer of size 10^5 and randomly sample mini-batches of size 256 for training. The Adam optimizer is used to optimize the actor and critic networks with a learning rate 0.001. The discount factor γ is 0.98, the hindsight reward weight λ_h is 0.5, and the probability of performing CAHER operation is 0.8.

In training, each epoch includes 100 episodes and each episode has 50 steps. In particular, in order to make the success rate curves look smoother, the exponential weighted average processing is carried out, as shown as follows

$$\bar{S}' = \beta \bar{S} + (1 - \beta) S \tag{20}$$

where $\bar{S}'$ represents the exponential weighted average success rate of current epoch, $\bar{S}$ represents the exponential weighted average success rate of last epoch, S represents the real success rate of current epoch, and β is the weight coefficient and set to be 0.9.

The effectiveness of our method

In order to verify that our method can effectively reduce the exploration time and speed up learning, two experiments with the demonstration data of both high and low success rates are conducted to compare our method with other four methods, DDPG with random sampling, DDPG with HER mechanism, DDPG from demonstrations and GAIL, respectively. Here, DDPG with random sampling is the original DDPG method. The number of demonstrations is 2000, the hyperparameter λ of the action loss is 1 and the sampling ratio between expert demonstrations and agent experiences is 1:1.

Demonstrations of high success rate. Figures 2 and 3 show the learning curves of the push and pick-and-place tasks with high success rate demonstrations, respectively. We

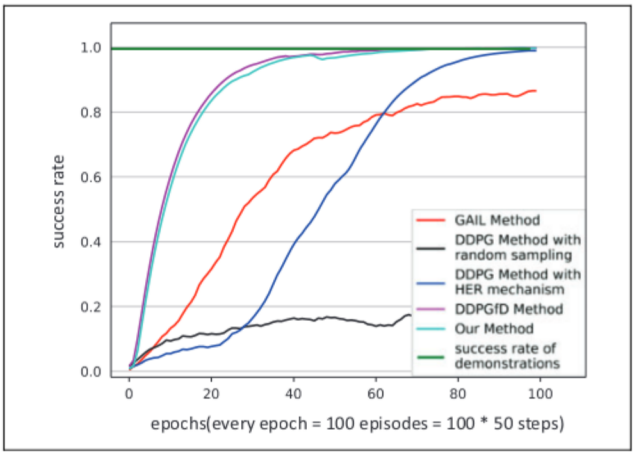

Figure 2. Learning curves of the push task with demonstrations of high success rate.

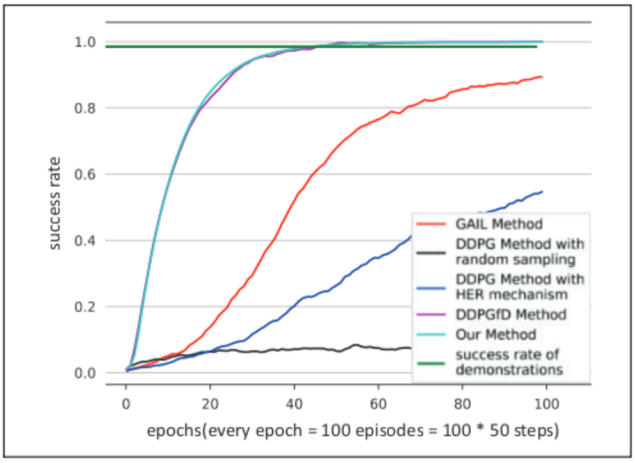

Figure 3. Learning curves of the pick-and-place task with demonstrations of high success rate.

Table 1. The best performance and the corresponding learning time with the demonstrations of high success rate.

	Push		Pick-and-place	
Method	Best performance	Learning time (min)	Best performance	Learning time (min)
GAIL	0.86	600	0.89	630
DDPG with random sampling	0.39	368	0.10	165
DDPG with HER	1.00	285	0.99	482
DDPGfD	1.00	240	1.00	268
Our method	1.00	241	1.00	272

HER: Hindsight Experience Replay; GAIL: generative adversarial imitation learning.

describe the best performance and the corresponding learning time of these methods in Table 1. The success rate of demonstrations for the push task is 0.996 and the success rate of demonstrations for the pick-and-place task is 0.985. The red, black, blue, purple, and cyan curves correspond to GAIL, DDPG with random sampling, DDPG with HER mechanism, DDPGfD and our method, respectively. The green dashed line represents the success rate of demonstrations. The horizontal and vertical coordinates in the figures are the number of epochs and the success rate of robot completing tasks in each epoch, respectively.

Due to lack of clear reward signals guiding the learning of the agent, it is evident that the learning effect of DDPG with random sampling is very poor. DDPG with HER has achieved relatively good results in terms of final performance, but the breadth-first nature of HER makes that some states of the space take a very long time to be learned.[47] This problem is especially serious as the difficulty of the task increases. So, we use demonstrations to overcome the above problem of HER and use the CAHER mechanism to encourage the agent to explore effectively. In fact, we have achieved almost the same learning speed as DDPGfD. Furthermore, because our method does not need to learn a reward function like GAIL, it has a faster learning speed than GAIL.

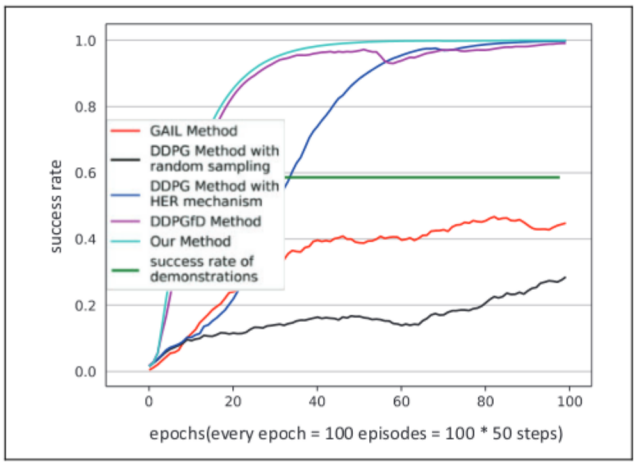

Figure 4. Learning curves of the push task with demonstrations of low success rate.

Demonstrations of low success rate. However, we often have difficulties in acquiring high-quality demonstrations, so we need to consider how to make the algorithm work well with the demonstration data of low success rate. In this experiment, the success rate of demonstrations for the push task is 0.587 and the success rate of demonstrations for the pick-and-place task is 0.529. Figures 4 and 5 show the learning curves of the push and pick-and-place tasks with demonstrations of low success rate, respectively. The best performance and the corresponding learning time of these methods are described in Table 2.

Same as above, the results show that both in the push and pick-and-place tasks, DDPG without the reward signal cannot learn effectively, which further confirms that traditional RL is heavily dependent on the reward function to

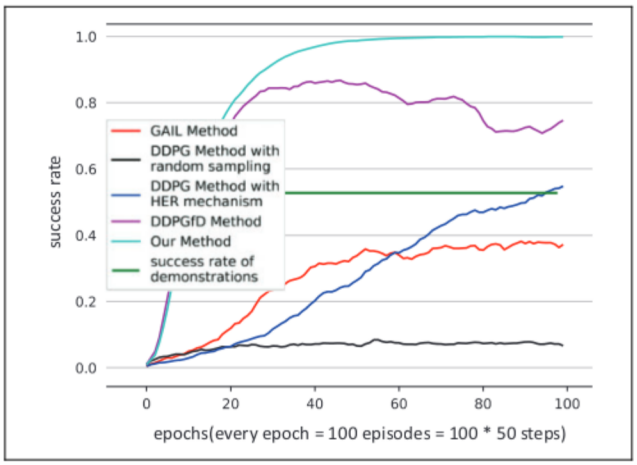

Figure 5. Learning curves of the pick-and-place task with demonstrations of low success rate.

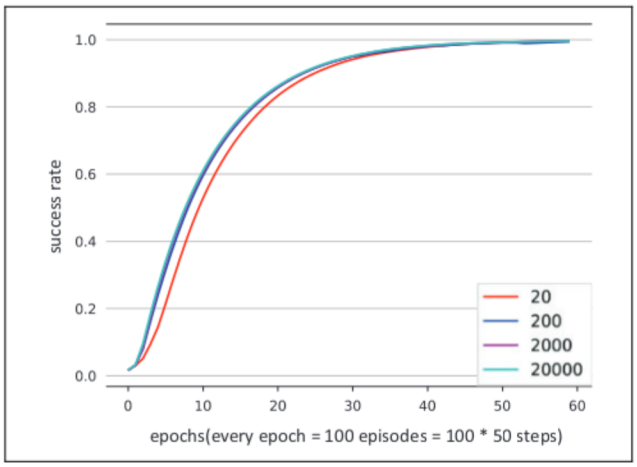

Figure 6. Learning curves of the push task using different number of demonstrations of high success rate.

Table 2. The best performance and the corresponding learning time with the demonstrations of low success rate.

	Push		Pick-and-place	
Method	Best performance	Learning time (min)	Best performance	Learning time (min)
GAIL	0.48	492	0.39	540
DDPG with random sampling	0.39	368	0.10	165
DDPG with HER	1.00	285	0.99	482
DDPGfD	0.99	294	0.87	254
Our method	1.00	245	1.00	277

HER: Hindsight Experience Replay; GAIL: generative adversarial imitation learning.

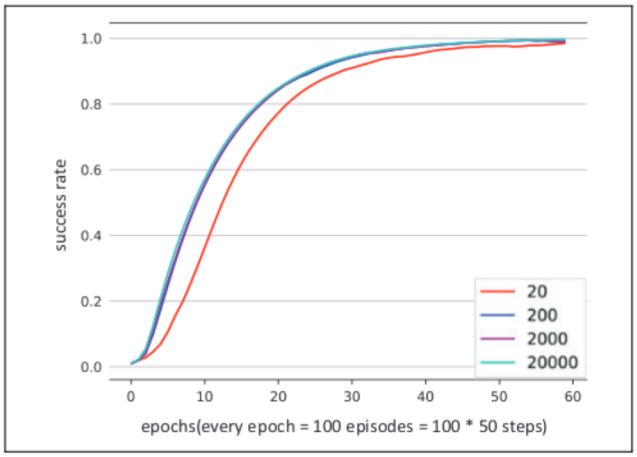

Figure 7. Learning curves of the pick-and-place task using different number of demonstrations of high success rate.

guide the agent to learn. For example, whether the reward function is good or bad, sparse or not, and so on has a great influence on the RL algorithm. In addition, GAIL learns an optimal policy by performing occupancy measure matching. It can achieve similar results on two tasks due to its strong robustness, but it also has a disadvantage that it cannot outperform the demonstrations. Although DDPG with HER is not affected by the quality of demonstrations, its learning speed of complex tasks is significantly slowing. The DDPGfD method, which has similar performance with our method in the above experiment, has a poor stability and even has a performance degradation in the pick-and-place task due to the lack of high-quality demonstrations. In contrast, our method not only accelerates the learning process by leveraging demonstrations but also overcomes the influence of sparse rewards through CAHER, and therefore has good stability and achieves the best performance compared with other methods.

It can be seen from the above experiments that the proposed method can overcome the sparse reward problem and speed up learning. At the same time, compared with other methods that leverages demonstrations, our method reduces the requirement for high quality demonstrations, which is difficult to obtain in some scenarios.

The influence of demonstrations

In our method, demonstrations are used to speed up learning. In this part, two experiments on the number of demonstrations and the sampling ratio between demonstrations and experiences were conducted to analyze the influence of the chief factors of demonstrations on the learning effects.

The number of demonstrations. The experiment was conducted with different numbers of demonstrations, in which the sampling ratio between demonstrations and experiences is 1:1, and the hyperparameter λ of the action loss is 1.

Figures 6 and 7 show that the learning results of the push and pick-and-place tasks with the demonstration data of

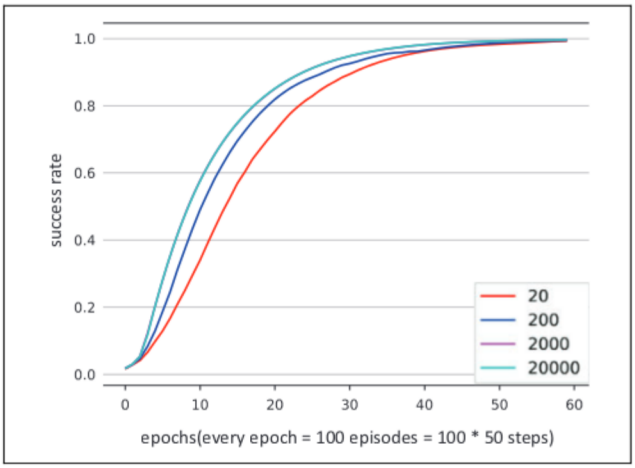

Figure 8. Learning curves of the push task using different number of demonstrations of low success rate.

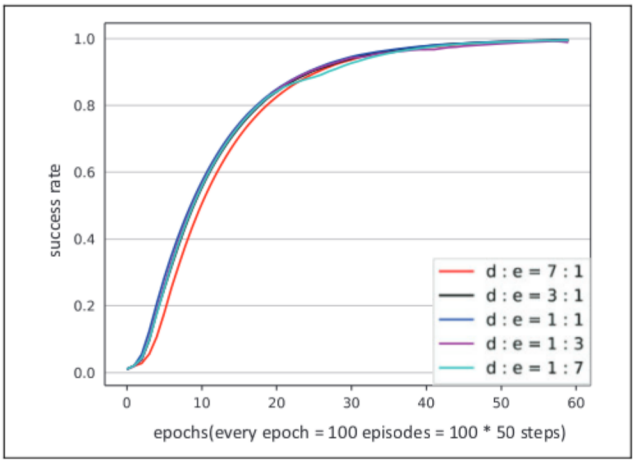

Figure 11. Learning curves of the pick-and-place task using different sampling ratios between demonstrations of high success rate and experiences.

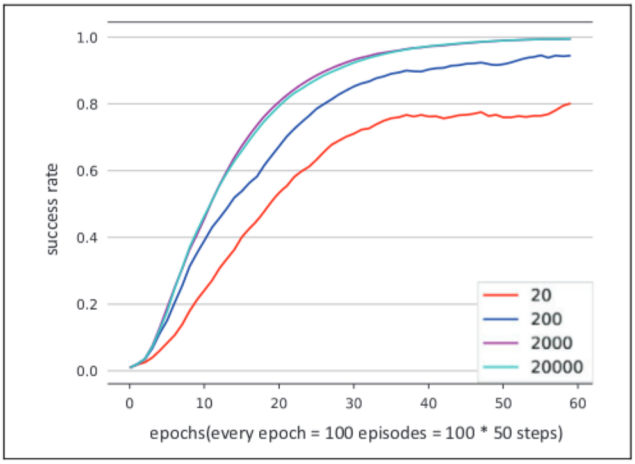

Figure 9. Learning curves of the pick-and-place task using different number of demonstrations of low success rate.

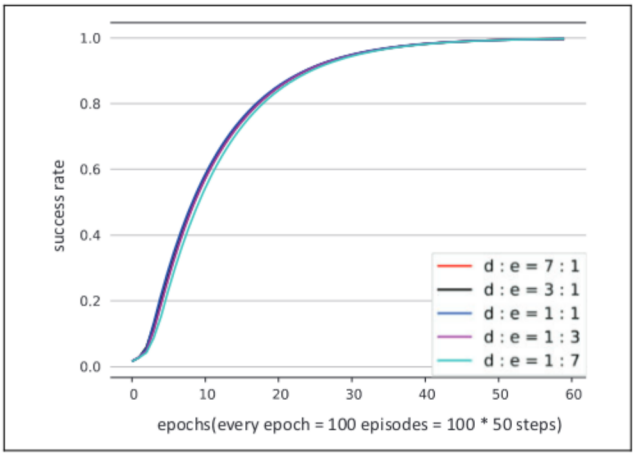

Figure 12. Learning curves of the push task using different sampling ratios between demonstrations of low success rate and experiences.

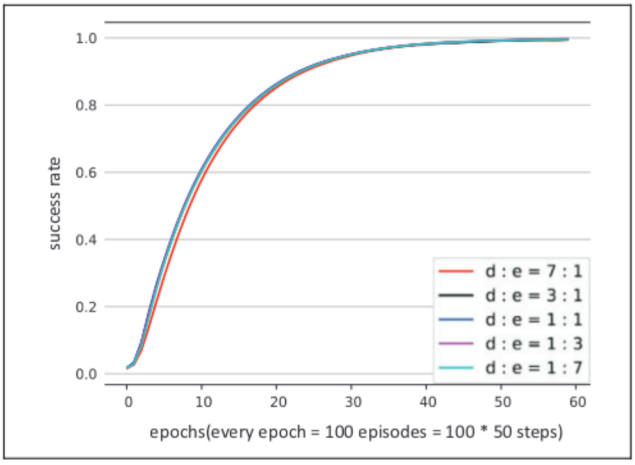

Figure 10. Learning curves of the push task using different sampling ratios between demonstrations of high success rate and experiences.

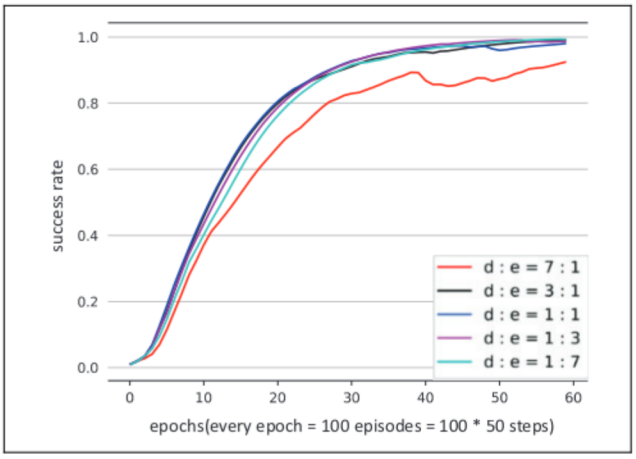

Figure 13. Learning curves of the pick-and-place task using different sampling ratios between demonstrations of low success rate and experiences.

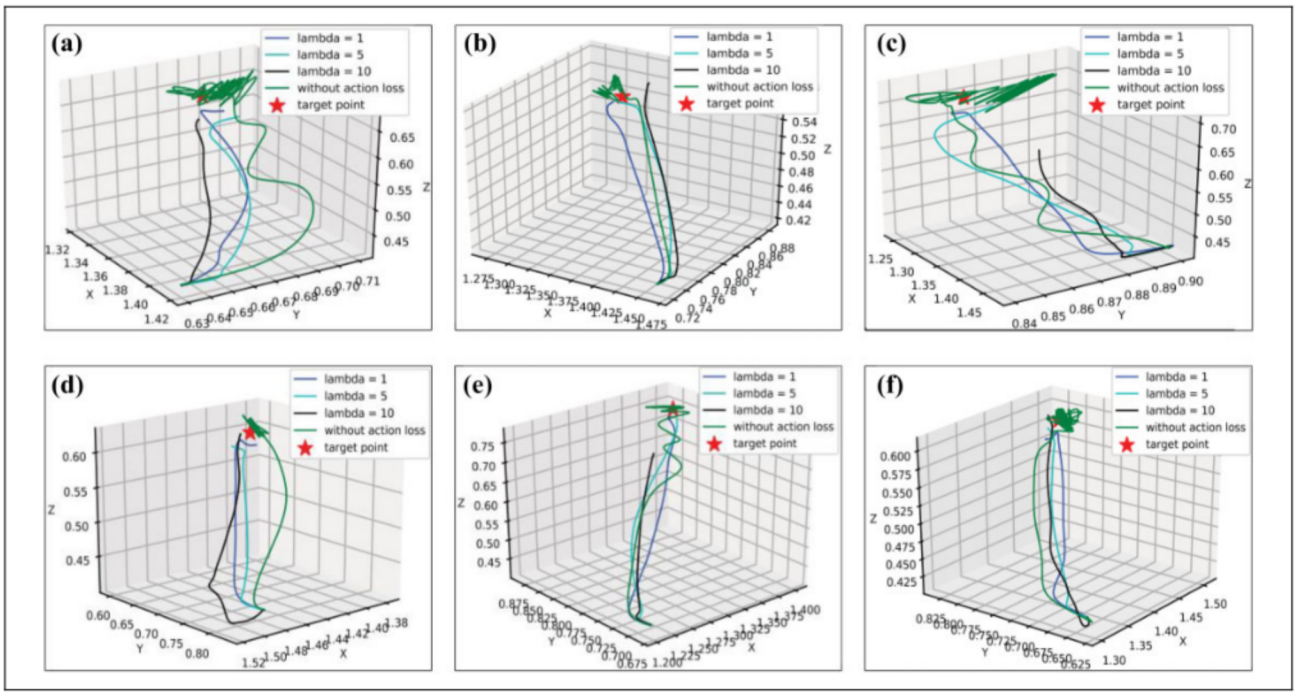

Figure 14. Motion trajectories of the object.

high success rate. When the number of demonstrations is 200, 2000, 20000, three learning curves rise rapidly with almost identical shapes. In contrast, the rising speed of the learning curve slows down obviously when the number is 20.

Figures 8 and 9 are the learning curves of the push and pick-and-place tasks with demonstrations of low success rate. When the number of demonstrations is 2000, 20000, these two learning curves are almost identical and rise rapidly. In contrast, the rising speed of learning curve slows down obviously when the number is 20. And when the number of demonstrations is 200, the rising speed lies between the 20 and 2000 groups.

We can get the following conclusions from above: as the number of demonstrations increases, the learning effect of the agent gradually improves, especially when the success rate of demonstrations is low. However, the training time decreases logarithmically. For example, for the results with 2000 and 20000 demonstrations, we can see that even if the number of demonstrations differs by an order of magnitude, the difference in learning effect can be negligible. Therefore, for this kind of robotic tasks, 2000 demonstrations are used in our method.

The sampling ratio between demonstrations and experiences. The experiment was conducted with different sampling ratios between demonstrations and experiences, in which the number of demonstrations is 2000, and the hyperparameter λ of the action loss is 1.

Figures 10 and 11 show the learning curves of the push and pick-and-place tasks with demonstrations of high success rate. The curves show that when the sampling ratio between demonstrations and experiences is between 3:1 and 1:7, there is almost no difference in the learning curves. Only when the ratio is 7:1, the rising speed of the learning curve decreases slightly, but this difference can be negligible.

Figures 12 and 13 show the learning results of the push and pick-and-place tasks with low success rate demonstrations. The results show that for the push task, changing the sampling ratio between demonstrations and experiences does not affect the learning process of the agent. For the more difficult task, pick-and-place, when the ratio is 7:1, the learning curve rises at the slowest speed, and there is still a certain gap of success rate compared with the others.

We can get the following conclusions from above: generally, the sampling ratio between demonstrations and experiences does not affect the learning results of the agent. However, when the success rate of demonstrations is low, the agent still learns from a large amount of demonstrations, which will have an impact on the learning speed. Therefore, in the learning process, the sampling ratio should be kept between 1:3 and 3:1.

The effect of the action loss

In order to verify that the loss function can maintain the motion stability of the manipulator effectively, the methods both with and without action loss are performed in this experiment, in which the number of demonstrations is 2000, and the sampling ratio between demonstrations and experiences is 1:1.

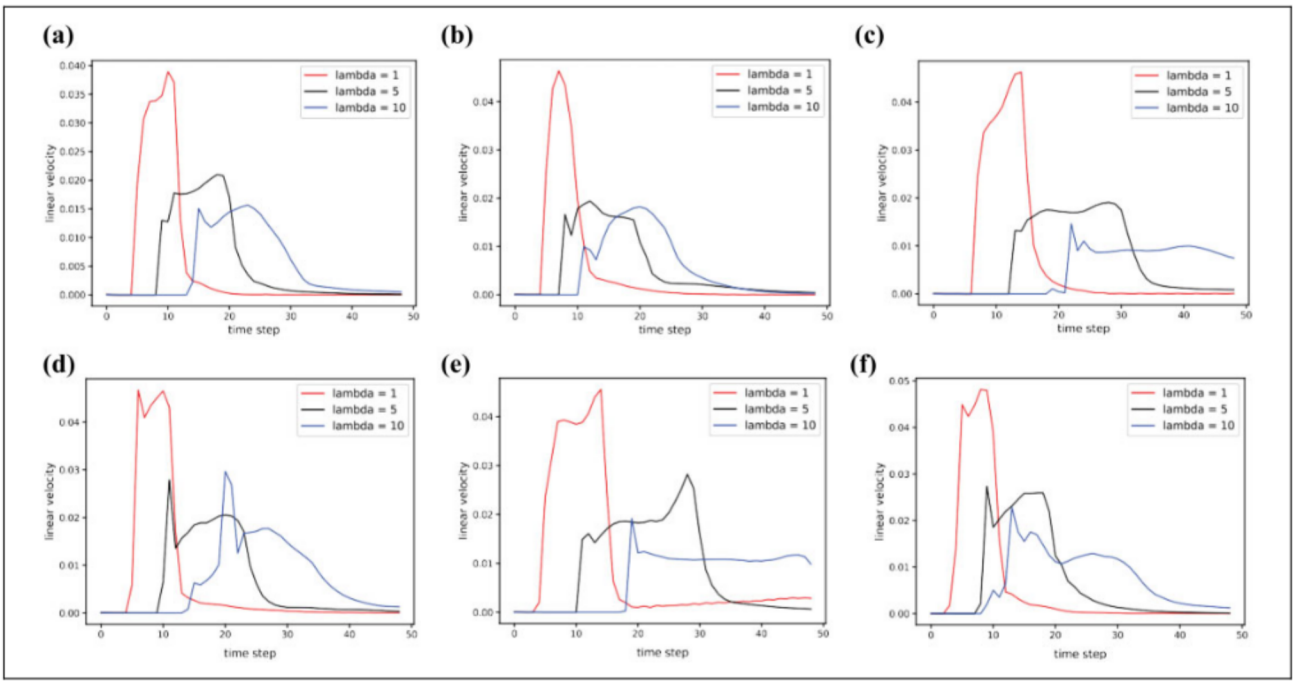

Figure 15. Linear velocity curves of the object.

Figure 14 shows the motion trajectories of the object which is grabbed by the manipulator in the pick-and-place task. The blue, cyan, black, and green curves represent the object trajectories when the hyperparameter λ of the action loss is 1, 5, 10, and 0, respectively. The red pentagram represents the target location where the object is to arrive. When the distance between the target position and the object position is less than 5 cm, the task is considered completed.

It is evident that the method without action loss has obvious vibrations during the operation process, especially in the target area. Although the vibration of the manipulator is very serious, the agent only knows that it has accomplished the task if the object is within 5 cm of the target point. In contrast, the method with action loss has no serious vibration during the operation process as the action loss we designed is to minimize the vibration amplitude of the output action, and larger the action loss parameter λ is, the smoother the action is. But the action loss parameter λ cannot be as big as possible. When λ is too big, the end of the manipulator moves too slowly to complete the task within the expected number of steps. Figure 15 plots the linear velocity of the object which is grabbed by the manipulator in the pick-and-place task. When the curve rises sharply, it means that the agent starts to perform the task, and when the curve falls sharply, it means that the agent has completed the task. We can see that as the parameter λ increases, the moving speed of the object gradually slows down. Sometimes, as shown in Figure 15(c) and (e), there is a possibility that the object cannot be placed on the target point within the expected number of steps.

Conclusion

This article proposes an off-policy model-free RL algorithm which uses demonstrations to accelerate learning the robotic tasks with sparse rewards. Firstly, we design a new experience replay mechanism, CAHER, which introduces the curiosity and aggressive rewards into HER, in order to enable the agent to learn the robotic tasks only with sparse rewards effectively. Secondly, based on an off-policy RL algorithm, TD3, we use demonstration data as part of off-policy training data to accelerate learning. Finally, the action loss is added into the actor loss to make the robot action as smooth as possible. The experimental results show that our method is comparable to, even better than the other current RL methods in robotic manipulation tasks with sparse rewards. The action loss can also effectively reduce the jitter of the output action of the robot and make the output action look more like the action of human beings. The major limitation of this work is the sample efficiency of completing the harder tasks. In the next work, we will improve the sample efficiency of our method on header tasks and test it on the real robot.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by the National Science Foundation of

China (61873008) and Beijing Natural Science Foundation (4182008).

ORCID iD

Guoyu Zuo https://orcid.org/0000-0002-7624-4728

References

1. Arnold B. Reinforcement learning: an introduction (adaptive computation and machine learning). *IEEE Trans Neural Netw* 1998; 9(5): 1054.
2. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.
3. Vinyals O, Babuschkin I, Chung J, et al. AlphaStar: mastering the real-time strategy game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019, accessed 24 January 2019).
4. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge. *Nature* 2017; 550(7676): 354–359.
5. Heess N, Dhruva TB, Sriram S, et al. Emergence of locomotion behaviours in rich environments. 2017. ArXiv preprint arXiv:1707.02286.
6. Florensa C, Duan Y, and Abbeel P. Stochastic neural networks for hierarchical reinforcement learning. In: *International conference on learning representations*, Toulon, France, 24–26 April 2017.
7. Andrychowicz M, Wolski F, Ray A, et al. Hindsight experience replay. In: *Neural information processing systems*, Long Beach, CA, USA, 4–9 December 2017, pp. 5048–5058.
8. Večerík M, Hester T, Scholz J, et al. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. 2017. ArXiv preprint arXiv:1707.08817.
9. Zhao R and Tresp V. Curiosity-driven experience prioritization via density estimation. 2019. ArXiv preprint arXiv:1902.08039.
10. Lanka S and Wu T. ARCHER: Aggressive Rewards to Counter Bias in Hindsight Experience Replay. 2018. ArXiv preprint arXiv:1809.02070.
11. Kalakrishnan M, Buchli J, Pastor P, et al. Learning locomotion over rough terrain using terrain templates. In: *Intelligent robots and systems*, St Louis, Missouri, 11–15 October 2009, pp. 167–172.
12. Ross S, Gordon GJ, and Bagnell JA. A reduction of imitation learning and structured prediction to no-regret online learning. In: *International conference on artificial intelligence and statistics*, Ft. Lauderdale, FL, USA, 11–13 April 2011, pp. 627–635.
13. Bojarski M, Testa DD, Dworakowski D, et al. End to end learning for self-driving cars. 2016. ArXiv preprint arXiv:1604.07316.
14. Pomerleau DA. ALVINN: An Autonomous Land Vehicle in a Neural Network. *Advances in Neural Information Processing Systems* 1988; 89(77): 305–313.
15. Pomerleau DA. Efficient training of artificial neural networks for autonomous navigation. *Neural Comput* 1991; 3(1): 88–97.
16. Ross S and Bagnell D. Efficient reductions for imitation learning. In: *International conference on artificial intelligence and statistics*, Sardinia, Italy, 13–15 May 2010, pp. 661–668.
17. Russell S. Learning agents for uncertain environments (extended abstract). In: *Conference on learning theory*, Madison, Wisconsin, USA, 24–26 July 1998, pp. 101–103.
18. Ng AY and Russell S. Algorithms for inverse reinforcement learning. In: *International conference on machine learning*, vol. 67, Stanford, CA, USA, 29 June–2 July 2000, pp. 663–670.
19. Abbeel P and Ng AY. Apprenticeship learning via inverse reinforcement learning. In: *International conference on machine learning*, Banff, Alberta, Canada, 4–8 July 2004, p. 1.
20. Ziebart BD, Maas AL, Bagnell JA, et al. Maximum entropy inverse reinforcement learning. *Springer Verlag* 2008; 36(5): 823–834.
21. Finn C, Levine S, and Abbeel P. Guided cost learning: deep inverse optimal control via policy optimization. 2016. ArXiv preprint arXiv:1603.00448.
22. Fu J, Luo K, and Levine S. Learning robust rewards with adversarial inverse reinforcement learning. 2017. ArXiv preprint arXiv:1710.11248.
23. Xia C and Kamel AE. Neural inverse reinforcement learning in autonomous navigation. *Robot Auton Syst* 2016; 84: 1–14.
24. Ho J and Ermon S. Generative adversarial imitation learning. In: *Neural information processing systems*, Barcelona, Spain, 4–9 December 2016, pp. 4565–4573.
25. Goodfellow IJ, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets. In: *Neural information processing systems*, Montreal, Canada, 8–13 December 2014, pp. 2672–2680.
26. Baram N, Anschel O, Caspi I, et al. End-to-end differentiable adversarial imitation learning. In: *International conference on machine learning*, Long Beach, CA, USA, 4–9 December 2017, pp. 390–399.
27. Hausman K, Chebotar Y, Schaal S, et al. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In: *Neural information processing systems*, Long Beach, CA, USA, 4–9 December 2017, pp. 1235–1245.
28. Li Y, Song J, and Ermon S. Infogail: interpretable imitation learning from visual demonstrations. In: *Neural information processing systems*, Long Beach, CA, USA, 4–9 December 2017, pp. 3812–3822.
29. Sasaki F, Yohira T, and Kawaguchi A. Sample efficient imitation learning for continuous control. In: *International conference on learning representations*, New Orleans, USA, 6–9 May 2019.
30. Kostrikov I, Agrawal KK, Dwibedi D, et al. Discriminator-actor-critic: addressing sample inefficiency and reward bias in adversarial imitation learning. In: *International conference*

on learning representations, New Orleans, USA, 6–9 May 2019.
31. Hester T, Vecerik M, Pietquin O, et al. Learning from demonstrations for real world reinforcement learning. 2017. ArXiv preprint arXiv:1704.03732.
32. Fujimoto S, Hoof HV, and Meger D. Addressing function approximation error in actor-critic methods. In: *International conference on machine learning*, Stockholm, Sweden, 10–15 July 2018, pp. 1587–1596.
33. Watkins CJ and Dayan P. Q-learning. *Machine learning* 1992; 8(3–4): 279–292.
34. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. In: *International conference on learning representations*, San Juan, Puerto Rico, 2–4 May 2016.
35. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: *International conference on machine learning*, New York City, USA, 19–24 June 2016.
36. Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *International conference on machine learning*, Stockholm, Sweden, 10–15 July 2018, pp. 1856–1865.
37. Duda RO and Hart PE. *Pattern classification and scene analysis*. Hoboken: Wiley, 1973.
38. Robert C. *Machine learning, a probabilistic perspective*. Cambridge, Massachusetts: The MIT Press, 2012.
39. Dempster AP, Laird NM, and Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc* 1977; 39(1): 1–38.
40. Gruber MJ, Gelman BD, and Ranganath C. States of curiosity modulate hippocampus-dependent learning via the dopaminergic circuit. *Neuron* 2014; 84(2): 486–496.
41. Blei DM and Jordan MI. Variational inference for Dirichlet process mixtures. *Bayesian Anal* 2006; 1(1): 121–143.
42. Chengchen Z, Cai Y, and Tang Z. A novel dynamic obstacle avoidance algorithm based on collision time histogram. *Chin J Electron* 2017; 26(3): 522–529.
43. Giusti A, Guzzi J, Dan C, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot Autom Lett* 2017; 1(2): 661–667.
44. Brockman G, Cheung V, Pettersson L, et al. OpenAI gym. 2016. ArXiv preprint arXiv:1606.01540.
45. Plappert M, Andrychowicz M, Ray A, et al. Multi-goal reinforcement learning: challenging robotics environments and request for research. 2018. ArXiv preprint arXiv:1802.09464.
46. Todorov E, Erez T, and Tassa Y. Mujoco: a physics engine for model-based control. In: *Intelligent robots and systems*, Algarve, Portugal, 7–12 October 2012, pp. 5026–5033.
47. Held D, Geng X, Florensa C, et al. Automatic goal generation for reinforcement learning agents. In: *International conference on machine learning*, Stockholm, Sweden, 10–15 July 2018, pp. 1514–1523.