Reward-weighted DHER Mechanism For Multi-goal Reinforcement Learning With Application To Robotic Manipulation Control

Xueyu Wei, Lilong Duan, and Wei Xue*

School of Computer Science and Technology, Anhui University of Technology, Maanshan 243032, China

**Corresponding author. E-mail: xuewei@ahut.edu.cn*

Received: Oct. 26, 2022; Accepted: Mar. 08, 2023

In multi-goal reinforcement learning, an agent learns to achieve multiple goals using a goal-oriented policy, obtaining rewards from positions that have been achieved. Dynamic hindsight experience replay method improves the learning efficiency of the algorithm by matching the trajectories of past failed episodes and creating successful experiences. But these experiences are sampled and replayed by a random strategy, without considering the importance of the episode samples for learning. Therefore, not only bias is introduced as the training process, but also suboptimal improvements in terms of sample efficiency are obtained. To address these issues, this paper introduces a reward-weighted mechanism based on the dynamic hindsight experience replay (RDHER). We extend dynamic hindsight experience replay with a trade-off to make rewards calculated for hindsight experience numerically greater than actual rewards. Specifically, the hindsight rewards are multiplied by a weighting factor to increase the Q-value of the hindsight state–action pair, which drives the update of the policy to select the maximum action for the given hindsight transitions. Our experiments show that the hindsight bias can be reduced in training using the proposed method. Further, we demonstrate RDHER is effective in challenging robot manipulation tasks, and outperforms several other multi-goal baseline methods in terms of success rate.

Keywords: Reinforcement learning, Multi-goal learning, Hindsight experience replay, Hindsight bias, Reward-weighted

http://dx.doi.org/10.6180/jase.202312_26(12).0015

1. Introduction

Reinforcement learning (RL) is the process by which an agent learns to perform a task by interacting with an unknown dynamic environment. In training, the learning algorithm updates the policy parameters of the agent. The goal of the learning algorithm is to find the best policy to maximize the long-term rewards obtained during the training task. Deep reinforcement learning [1] uses neural networks as function approximators [2] for RL, and has been shown in recent years to achieve human-level performance in Go [3–5], video games [6–8], and challenging robotic tasks [9–11].

In most of these tasks, the RL algorithm learns a policy to handle a single task for a specific goal. However, learning from sparse reward signals [12] remains difficult or even impossible to obtain successful policies. Because rewards are only available at the end of completing a task, resulting in too few successful experiences in the replay buffer. Some studies have proposed designing dense reward functions [13], but this approach requires manually designing reward functions with domain-specific knowledge, making it difficult to generalize to different environments as well. One approach is to generalize RL from single-goals to multi-goal [14] settings. In many real-world applications where an agent needs to perform a set of tasks with the same dynamics but different goals, even if each task has few rewards, the agent will reason about the goals and the learning process will be easier. Multi-goal RL enables the agent to find policies to achieve multiple goals by extending typical RL to action-value functions and policies under goal conditions, where the reward function of the environment also depends on the goal.

In the reward function setting for multi-goal RL, the reward is 0 when the distance between the achieved goal and the desired goal is less than a certain threshold, and negative one in all other cases. Thus, the agent receives non-negative rewards only when it successfully achieve the desired goal. In robotic arm tasks, due to the dynamic complexity and lack of exploration, it is usually difficult to achieve the desired goal by stochastic strategies. In order to improve the learning efficiency of goal-oriented RL [15] agents in sparse reward environments, hindsight experience replay (HER) [16] is based on the idea of off-policy data replay, using hindsight goals that allow agents to learn from failures. Specifically, the agent draws from the failed experiences in the replay buffer that have been achieved to replace the original goal as the desired goal in HER, allowing the agent to learn faster and receive positive rewards. The rewards are then recalculated and placed in the experience buffer for training. Since the experiences are sampled and replayed in a random strategy that does not take into account the degree of importance of the sample episodes for learning, using HER not only introduces bias but also decreases efficiency as training proceeds. To improve the efficiency of HER, prioritizing the more valuable episodes becomes the focus of research. Inspired by the working energy principle of physics, HER with energy-based prioritisation (HEBP) [17] assumes that higher-energy trajectories are more valuable to learn, it assigns higher probabilities to trajectory goals with high energy, and then samples transitions uniformly. Bias-reduced HER with virtual goal prioritization (BRHER) [18] then prioritizes the virtual goals to learn more valuable information from the agents and define them by heuristic metrics. It also improves learning algorithm by filtering misleading samples, reducing existing biases in HER. Aggressive rewards to counter bias in HER (ARCHER) [19] extends HER with a trade-off to make rewards calculated for hindsight experiences numerically greater than the real rewards. BHER [20], which handles the hindsight bias by using the bias-corrected hindsight reward in training. Curriculum-guided HER (CHER) [21] samples trajectories uniformly, and then performs hybrid sampling based on proximity to the desired goal and sample diversity.

In this paper, we propose a method for reward-weighted DHER mechanism learning for multi-goal RL (RDHER). New experiences are first collected from the experience of buffer failure, and combined into new goal trajectories to enable the processing of dynamic goals and reduce sample complexity. Then, a weighted trade-off between actual and hindsight rewards is performed to counter the bias introduced in HER and improve the sampling efficiency of the samples. We evaluate RDHER for four challenging robotic manipulation tasks. Experimentally, our proposed method achieves higher success rates than previous work.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries of RL, multi-goal RL and HER, and deep deterministic policy gradient, Section 3 presents other related works in recent years, Section 4 presents the proposed method in detail, and in Section 5 we give some comparative experiments to demonstrate the performance of the proposed method. Finally, we summarize this paper in Section 6.

2. Preliminaries

This section provides some preliminaries to help understanding, including RL, multi-goal RL and HER.

2.1. Reinforcement learning

RL explores the optimal policy for maximizing cumulative rewards by an agent that learns from experiences and continuously optimizes the relationship between the mapping of states and actions. The interaction process between an agent and its environment can be represented by Markov decision process (MDP) [22]. We denote the episode MDP by the tuple (S, A, T, P, r), where S represents the set of discrete spaces, A represents the set of discrete actions, P is the unknown transition dynamics, T is the episode length, and r is the reward function. In a periodic type task, the MDP consists of a series of discrete time steps 0, 1, 2, ..., t, ..., T, where T then denotes the abort time step. At each time step t, the agent observes the state of the environment s_t and then selects an action a_t according to the existing policy $\pi(a_t \mid s_t)$. After executing the action, the MDP reaches the next time step $t+1$. This action transfers the state a_t of the agent to the next state s_{t+1}, and the environment gets a new reward r_{t+1} for performing the action. The reward of an episode T in the MDP is the discounted sum of rewards received by the agent during that episode, defined as:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor, which downplays the influence of future rewards exponentially, effectively reducing the variance of returns. The goal of the agent is to maximize the expected reward by adjusting the policy over a trajectory of length T. The action-value function can be denoted as $Q^{\pi}(s_t, a_t)$ and defined by:

$$Q^{\pi}(s_t, a_t) := \mathbb{E}_{\pi}[R_t] \tag{2}$$

which represents the expected cumulative reward for taking the action a_t starting from state s_t, and following the policy π until the end of the episode.

2.2. Multi-goal RL and HER

Under conditions of sparse rewards, a single-goal RL task starts with an initialization strategy and requires continuous exploration until a specified location is achieved in order to obtain rewards. RL can be extended to a multi-goal setting: the agent can obtain rewards from the positions already achieved. We use the tuple $(S, A, T, P, r, \mathcal{G})$ to denote the MDP of multi-goal RL. The tuple has an additional goal space $\mathcal{G}$, where the policy of the agent and the reward function $R(s_t, a_t)$ of the environment also depend on the goal $\mathcal{G}$. Each set starts with an original goal $g^o \in \mathcal{G}$ which is randomly drawn from the goal space. A transition in the environment is denoted as: $(s_t, a_t, r_t^o, s_t + 1, g^o)$, and at every time step t, the agent is also provided with the currently achieved goal g_{t+1}^a. The reward function r_t^o is defined as:

$$r_t^o := r\left(g_{t+1}^a, g^o\right) = \mathbb{1}\left\{\|g_{t+1}^a - g^o\|_2^2 < \epsilon\right\} - 1 \tag{3}$$

where ϵ is the tolerance distance. The reward function is defined as whether the state at the next time step reaches the position of the specified goal, and a negative reward will be given when the distance between the achieved goal and the desired goal exceeds a certain threshold. This binary reward setting makes it difficult for the agent to obtain positive feedback and hinder the learning process.

To overcome this, HER is used to solve multi-goal RL problems with binary rewards, which uses a hindsight goal g^h sampled from the replay transition to substitute the original goal, there:

$$g^h = \text{Unif}\left\{g_{t+1}^a, g_{t+2}^a, \cdots, g_{T-1}^a\right\} \tag{4}$$

where the Unif stands for discrete uniform distribution, and then, the hindsight reward is calculated as:

$$r_t^h := r\left(g_{t+1}^a, g^h\right) = \mathbb{1}\left\{\left\|g_{t+1}^a - g^h\right\|_2^2 < \epsilon\right\} - 1 \tag{5}$$

In HER, the original transform tuple is replaced with a new hindsight transform tuple $\left(s_t, a_t, r_t^h, s_{t+1}, g^h\right)$. Her uses the hindsight goal g^h sampled from the replay transitions to replace the original goal, providing the agent with successful experiences to learn.

3. Related work

The need for a large number of samples is more evident when RL are introduced into deep neural networks. If the learning is performed directly using sparse samples sometimes not only fails to enhance the policy, but also leads to the training divergence of the neural network. At present, the research for solving the sparse reward problem mainly contains reward design and learning, experience replay mechanism, exploration and exploitation, multi-goal learning and auxiliary tasks. In this work, we focus on multi-goal RL algorithms that concern the generalization of tasks with the same MDP but different goals. The research covers a broad field of RL methods for robotics control [23, 24] and cybernetics [25]. Methods based on multi-goal RL are expected to be applied to practical applications, including aerial manipulators [26] and humanoid robots [27].

Multi-goal learning algorithms have also been extensively studied in previous work. UVFA [28] extended value functions to multiple goals, while TDM [29] extended them to different time horizons. [30] investigated the effect of size of the replay buffer on performance and proposed Combined Experience Replay (CER) to alleviate the liability of a large replay buffer. [16] proposed HER method based on the goal value function, which has the ability to generalize in the goal space to train the policies of RL algorithms in a sparse and binary reward environment, thus avoiding the need for complex reward engineering. HER can be combined with any off-policy algorithm, e.g., DQN [31], DDPG [32]. Although HER shows good results in multi-goal RL, it does not consider the bias caused by hindsight goals. [33] extended HER to dynamic goals for processing tasks with dynamic goals in the presence of sparse rewards. Sampling rate decay [34] came up with a sampling rate decay strategy that can reduce the number of training hindsight experiences. [35] combined the HER with the policy gradient method, using importance sampling for bias correction, and extending the multi-goal mechanism to the policy algorithm. [36] extended the HER to the task of representing states as images, using variational autoencoder [37] to obtain hidden variable representations of states and goals. Goal replacement and training under the hidden space further extends the application of the multi-goal algorithm.

Hindsight methods can also be considered as an exploration method [38] in solving sparse-reward tasks. [39] introduced a competition between two agents for better exploration. Entropy regularization HER [40] presented a new multi-goal RL based on weighted entropy that encourages agents to maximize expected returns and achieve diverse goals. Multi-goal RL can also be combined with multitask learning [41], imitation learning [42], multiagent learning [43], and planning [44] to solve complex tasks.

4. Proposed method

In this section, we formally describe the two main components of our method, RDHER: (1) dynamic goal-based goal matching and combination module (Fig. 1). New experiences are collected from buffer failures and combined into new goal trajectories to enable the processing of dynamic goals and reduce sample complexity (see Section 4.1); (2) reward-weighted mechanism module for overestimating probability bias based on hindsight experiences (Fig. 2). We introduce a weighted trade-off between the actual reward and the hindsight reward to counter the bias introduced in HER to improve the sampling efficiency of the samples (see Section 4.2).

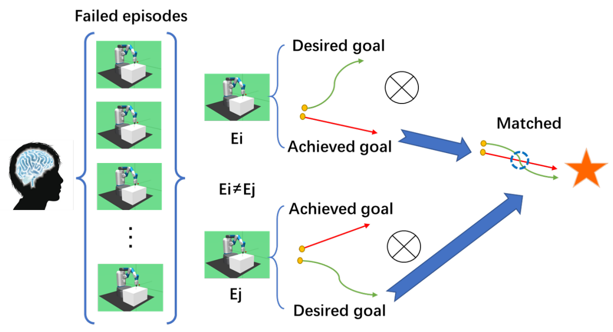


Fig. 1. Dynamic goal-based goal matching and combination module.

4.1. Dynamic goal-based goal matching and combination

To solve the reward sparsity problem, a hindsight experience replay method is proposed, which is a simple and effective method that can operate the replay buffer used for RL algorithms. However, the application of HER algorithm is limited, and with the increasing complexity of the environment, it becomes difficult to achieve changing the position of goal in different time steps within a cycle when the goal is constantly changing. To overcome this limitation, we consider reassembling failed experiences into new episodes for training through cutting and splicing operations.

4.1.1. Goal matching

We create successful trajectory rewards for the agents by efficiently using the failed experiences in the experience replay buffer $\{E_0, E_1, E_2, \cdots\}$. For the achieved goal trajectory, we try to find a matching desired goal trajectory from other episodes. First, we search for two episodes of failures E_i and E_j $(i \neq j)$, where $\exists\, i, j, p, q$, s.t. $g_{i,p}^{ac} = g_{j,q}^{de}$. $g_{i,p}^{ac}$ denotes the goal achieved by the agent at time step p of the episode i, and $g_{j,q}^{de}$ denotes the desired goal of the agent at time step q of the episode j.

4.1.2. Goal combination

Next, the achieved goal trajectories are combined into new episodes by matching the achieved goal trajectories with the found desired goal trajectories. It should be known that in a multi-goal RL setup, the state consists of three parts: observation o_t, desired goal g_t^{de} and achieved goal g_t^{ac}, define as $s_t = \left\langle o_t, g_t^{ac}, g_t^{de} \right\rangle$, where g_t^{ac} indicates goals that the agent has achieved and g_t^{de} denotes the desired goal at moment t. If we find two such failed episodes in 4.1.1, we combine the two experience by replacing the desired goals in E_i by $g_{j,t}^{de}$, where j indicates E_j and $t \leq \min\{p, q\}$. On this basis, the new combined goal trajectory is represented as $\left\{g_{j,0}^{de}, \cdots, g_{j,t}^{de}\right\}$. But the problem that arises from doing this is that not all experiences deserve the same degree of learning, and HER introduces significant bias in the replay buffer.

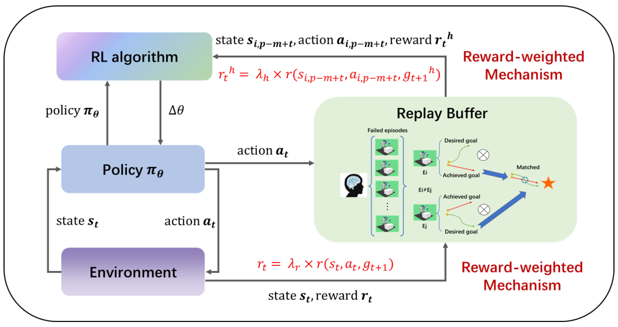

RL algorithm
state $\boldsymbol{s}_{i,p-m+t}$, action $\boldsymbol{a}_{i,p-m+t}$, reward $\boldsymbol{r}_t{}^h$
$r_t{}^h = \lambda_h \times r(s_{i,p-m+t}, a_{i,p-m+t}, g_{t+1}{}^h)$
Reward-weighted Mechanism
policy $\boldsymbol{\pi}_\theta$
$\Delta\theta$
Replay Buffer
Failed episodes
Desired goal
Ei
Achieved goal
Matched
Ei≠Ej
Achieved goal
Ej
Desired goal
action $\boldsymbol{a}_t$
Policy $\boldsymbol{\pi}_\theta$
state $\boldsymbol{s}_t$
action $\boldsymbol{a}_t$
Environment
$r_t = \lambda_r \times r(s_t, a_t, g_{t+1})$
state $\boldsymbol{s}_t$, reward $\boldsymbol{r}_t$
Reward-weighted Mechanism

Fig. 2. Reward-weighted mechanism module.

4.2. Reward-weighted mechanism module for hindsight bias

To use HER more effectively, we consider the degree of importance of the episode samples for learning. Applying a reward-weighted mechanism, two real-valued scalar multipliers λ_r and λ_h are introduced to differentiate between actual rewards r_t and hindsight rewards r_t^h, as is shown in Eqs. (6) and (7):

$$r_t = \lambda_r \times r\left(s_t, a_t, g_{t+1}\right) \tag{6}$$

$$r_t^h = \lambda_h \times r\left(s_{i,p-m+t}, a_{i,p-m+t}, g_{t+1}^h\right) \tag{7}$$

where r$(\cdot)$ denotes the given reward function for the task. Matching the true probability of hindsight experience with the probability of its bias drives the current policy to be consistent with the hindsight experience data in the replay buffer. We use more aggressive hindsight experience replay rewards to increase the Q-value of hindsight state-action pairs to indirectly drive policy updates and improve the sampling efficiency of RL.

It is worth noting that the Vanilla HER is a special case where λ_r and λ_h are equal to 1. We require that $r_t^h \geq r_t$, which yields:

$$\lambda_h \times r\left(s_t, a_t, g^h\right) > \lambda_r \times r\left(s_t, a_t, g\right) \tag{8}$$

however, if the agent is in the domain of functions with negative rewards, i.e. $r(\cdot) \leq 0$, there is $\lambda_r > \lambda_h$. Conversely, if in the domain of functions with positive rewards, i.e. $r(\cdot) \geq 0$, the formula $\lambda_r < \lambda_h$ can be obtained. Using trade-off, the definition of the goal values for actual experience and hindsight experience can take the form of:

$$y_i = \lambda_r \cdot r_i + \gamma Q'\left(s_{i+1} \| g, \mu'\left(s_{i+1} \| g; \theta^{\mu'}\right); \theta^{Q'}\right) \tag{9}$$

$$y_i^h = \lambda_h \cdot r_i^h + \gamma Q'\left(s_{i+1} \| g^h, \mu'\left(s_{i+1} \| g^h; \theta^{\mu'}\right); \theta^{Q'}\right) \tag{10}$$

where the Q' function is the target action-value network parameterized by $\theta^{Q'}$.

The pseudocode of the proposed method is described in Algorithm 1. Note that since RDHER has to search for all failed experiences, in our experiments we use two hash tables to store the trajectories of achieved goals and desired goals, respectively.

5. Experiment

In this section, we first present the experimental setup and simulation environments in order to evaluate the effectiveness of the proposed RDHER method. And also compared with current state-of-the-art HER-based algorithms to solve several challenging robot manipulation tasks for Gym that use the MuJoCo [45] physics simulator. In addition, we perform some sensitivity analyses on the weighting parameters in the experiments.

5.1. Experimental settings

In terms of the experimental settings, we give a brief introduction to the environment and the experimental parameter setting.

5.1.1. Simulation environments

The environment used for the experiments consists of four different tasks, which are FetchReach-v1 (Fig. 3a), FetchPush-v1 (Fig. 3b), FetchSlide-v1 (Fig. 3c) and FetchPickAndPlace-v1 (Fig. 3d).

The Fetch environment is based on the 7-DoF (degrees of freedom) Fetch robotics arm with a two-fingered parallel gripper. In the Fetch environments, the state s_t contains Cartesian positions, rotations, and velocities of joints and positions of objects. The state is a 10-dimensional vector in FetchReach and a 25-dimensional vector in the other Fetch tasks. Each action a_t is a 4-dimensional vector, where the 3-dimensional vector specifies the desired movement of the gripper in Cartesian coordinates and the gripper keeps closing during the process of reaching a certain target location. The desired goal is where the goal wants to be, while the achieved goal is where the goal has been achieved. Each episode is of length $T = 50$. The reward function of the agent is sparse and binary: the agent receives a reward of zero if the goal has been achieved (within the tolerances of certain tasks) and negative one otherwise.

- **FetchReach-v1**: FetchReach is the simplest task in all Fetch environments. The task is to make the Fetch move its gripper to the desired goal position.
- **FetchPush-v1**: The goal of the task is that the robot need to learn how to shift its gripper towards the desired goal.
- **FetchSlide-v1**: FetchSlide is the most difficult task in all Fetch environments. The robot has to hit the slider across a long table so that it slides off and stops on the desired goal.
- **FetchPickAndPlace-v1**: This is even harder, the robot has to grab a box from a table using its fingers and move it to a desired goal above the table as well as in the air.

5.1.2. Training parameters

We train different episodes for each agent in different environments. The basic hyperparameters in RDHER are the same as in the HER, and we summarize the relevant hyperparameters in the experiments in Table 1.

5.2. Ablation studies

We perform the following experiments to demonstrate the extent to which the study reward-weighted mechanism affects different environments. For each experiment, we carefully chose trade-off parameters to deeply fit the relative reward optimization between dynamic hindsight and standard experience replay. We set several sets of ratios between hindsight and actual reward weights respectively, and then evaluate our approach by comparing the success rate of the learned policy with the number of cycles required to achieve that performance.

- $\lambda_r, \lambda_h \in \{1, 2\}$: The weight of the hindsight experience reward function is two times or one-half of the actual reward function. We set $\lambda_r = 1$, $\lambda_h = 2$ and $\lambda_r = 2$, $\lambda_h = 1$ separately in our experiments.

```
Algorithm 1: RDHER method
Given:
• an off-policy RL algorithm $\mathbb{A}$                    ▷ e.g. DQN, DDPG, NAF, SDQN
• a strategy $\mathbb{S}$ for sampling goals for replay      ▷ e.g. $\mathbb{S}(s_0,\ldots,s_T) = m(s_T)$
• a reward function $r$, real reward weight $\lambda_r$, hingsight reward weight $\lambda_h$ and replay buffer $R$
for episode = $1, 2, \cdots, M$ do
  Sample an initial goal $g_o$ and an initial state $s_o$
  for $t = 0, \cdots, T-1$ do
    Sample an action $a_t$ using the behavioral policy from $\mathbb{A}$:
    $a_t \leftarrow \pi(s_t \mid g_t)$
    Execute the action $a_t$ and observe a new state $s_{t+1}$ and a new goal $g_{t+1}$
  end for
  for $t = 0, \cdots, T-1$ do
    $r_t := \lambda_r \times r(s_t, a_t, g_{t+1})$
    Store the transitions $(s_t \,|g_t, a_t, r_t, s_{t+1}|\, g_{t+1})$ in $R$ (Standard experience replay)
  end for
  Collected failed episodes to $\epsilon$
  for $E_i \in \mathcal{E}$ do
    Search another $E_j (i \neq j) \in \mathcal{E}$ where $g^{ac}_{i,p} = g^{de}_{j,q}$
    if $E_j \neq \varnothing$ then
      Clone a goal trajectory $\left\{g^h_0, \cdots, g^h_m\right\}_{m=\min}\{p,q\}$ in which $g^h_t = g^{de}_{j,q-m+t}$ from $E_j$
      for $t = \{0, \cdots, m-1\}$ do
        $r^h_t := \lambda_h \times r\left(s_{i,p-m+t}, a_{i,p-m+t}, g^h_{t+1}\right)$
        Store the transition $\left(s_{i,p-m+t} \left| g^h_t, a_{i,p-m+t}, r^h_t, s_{i,p-m+t+1} \right| g^h_{t+1}\right)$ in $R$
      end for
    end if
  end for
  for $t = 0, \cdots, N$ do
    Sample a minibatch $B$ from the replay buffer $R$
    Optimize $\mathbb{A}$ using the minibatch $B$
  end for
end for
```

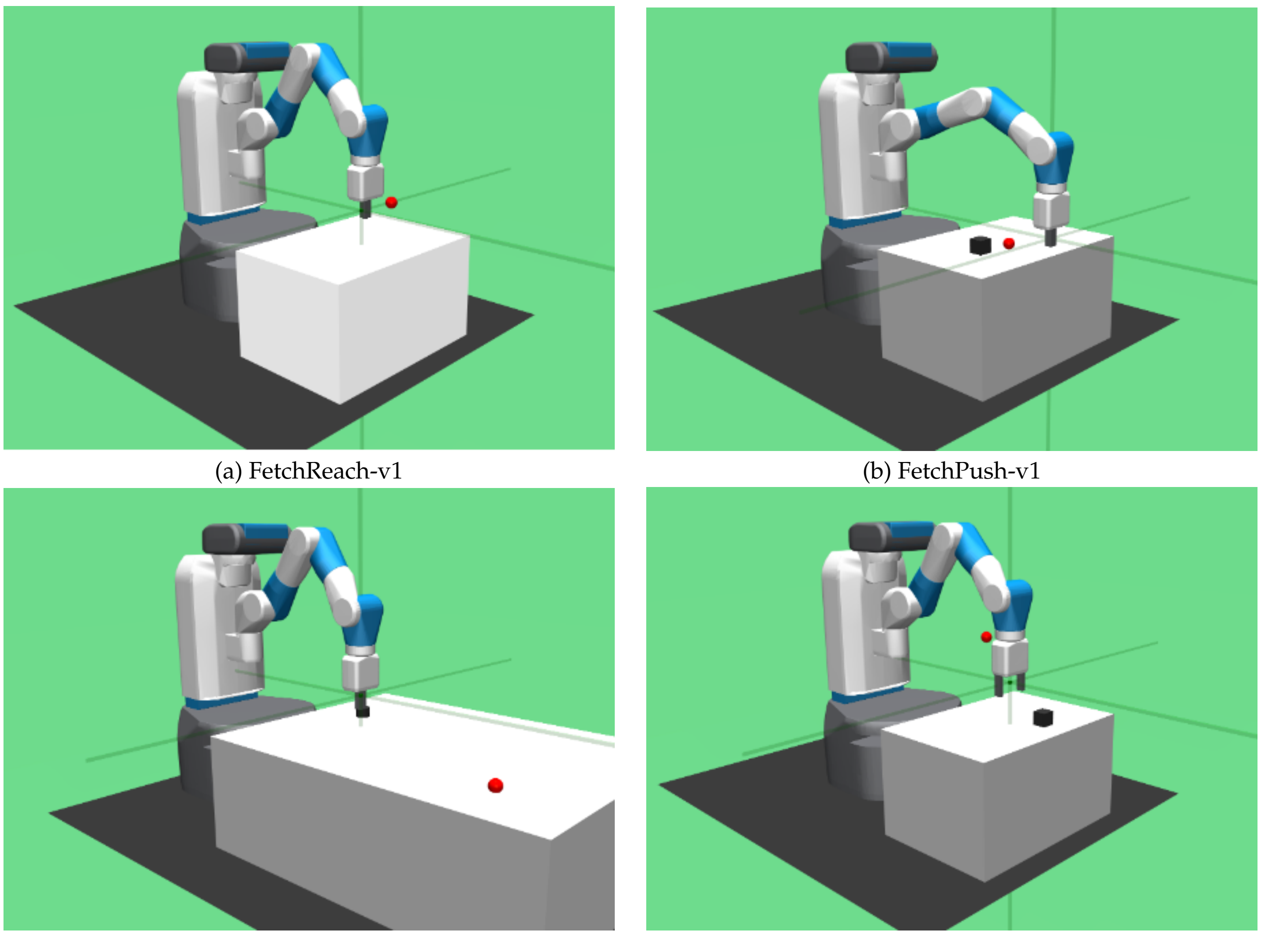
Table 1. Hyperparameters for RDHER.

Hyperparameters	Value	Description
optimizer	Adam	Adam optimizer is used for training.
layers	3	Number of layers in the critic/actor networks.
hidden	256	Number of neurons in each hidden layers.
learning rate	0.001	The learning rate for both the actor and the critic.
buffer_size	int(1E6)	For experience replay.
polyak factor	0.95	Polyak averaging coefficient.
rollout_batch_size	2	Per mpi thread.
n_batches	40	Training batches per cycle.
batch_size	256	Per mpi thread, measured in transitions and reduced to even multiple of chunk_length.
n_test_rollouts	10	Number of test rollouts per epoch, each consists of rollout_batch_size rollouts.
random_eps	0.3	Percentage of time a random action is taken.
noise_eps	0.2	Std of gaussian noise added to not-completely-random actions as a percentage of max_u.
norm_eps	0.01	Epsilon used for observation normalization.
norm_clip	5	Normalized observations are cropped to this values.

- $\lambda_r, \lambda_h \in \{1, 4\}$: The weight of the hindsight experience reward function is four times or one-fourth of the actual reward function. We set $\lambda_r = 1$, $\lambda_h = 4$ and $\lambda_r = 4$, $\lambda_h = 1$ separately in our experiments.

- $\lambda_r, \lambda_h \in \{0.5, 4\}$: The weight of the hindsight experience reward function is eight times or one-eighth of the actual reward function. We set $\lambda_r = 0.5$, $\lambda_h = 4$ and $\lambda_r = 4$, $\lambda_h = 0.5$ separately in our experiments.

We train the policy by using one CPU core and set the above different weighting factors in different environments to train the policy of the agent. Fig. 4 shows the performance plots in the four Fetch environments. From these



(a) FetchReach-v1 (b) FetchPush-v1

(c) FetchSlide-v1 (d) FetchPickAndPlace-v1

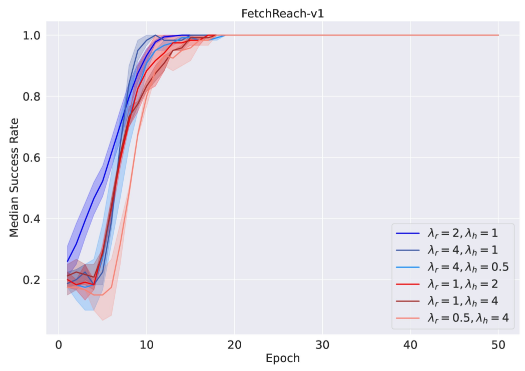
Fig. 3. Robot Fetch environments.

figures, we observe that the overall blue line part is superior to the red line, which means we can learn and enhance the performance proportionally from hindsight and actual experience. Since the reward function is binary negative -1/0, when $\lambda_r > \lambda_h$, it corresponds to increasing the Q-value of the hindsight transition state-action pair. Following this, the effective influence of the reward-weighted mechanism on the agent's search for the optimal policy.

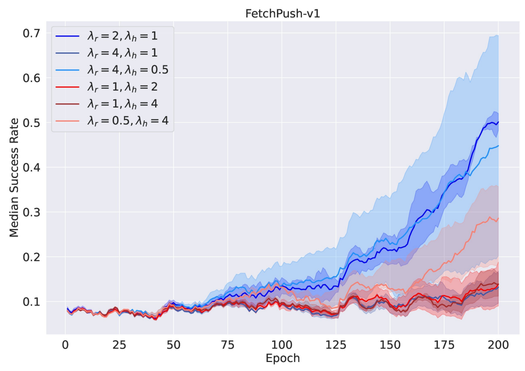
Figs. 5 to 8 show the comparative effects of these different weighting factors and Vanilla HER. In the FetchReach environment, we can observe that the final results of the RDHER are better than the baseline Vanilla HER regardless of the ratio set by λ_r and λ_h. Of course, when λ_r is 2, 4, 8 times λ_h is always better than λ_h is 2, 4, 8 times λ_r. In the FetchPush environment, the algorithm works best when λ_r is set to 8 times λ_h, but the variance is larger, and when $\lambda_r, \lambda_h \in \{1, 4\}$, the effect is not as good as the baseline Vanilla HER, and when λ_r is set to 2 times λ_h, the effect is not only better than the baseline but also the variance is smaller. In the FetchSlide environment, similar to the FetchReach environment, the final result of the algorithm is better than the baseline Vanilla HER regardless of the ratio of λ_r and λ_h settings. When $\lambda_r, \lambda_h \in \{1, 4\}$, the algorithm is comparable, and when λ_r is set to 2 and 8 times of λ_h, both are stronger than when λ_h is greater than λ_r. In the FetchPickAndPlace environment, the training effect of the algorithm is not only comparable when $\lambda_r, \lambda_h \in \{0.5, 4\}$, but also comparable to the Vanilla HER. The effect is better than the baseline when $\lambda_r, \lambda_h \in \{1, 4\}$ and is best when λ_r is four times λ_h. When $\lambda_r, \lambda_h \in \{1, 2\}$, the curves are on the upper and lower sides of the baseline algorithm respectively, and the best results are obtained when λ_r is two times of λ_h.

5.3. Benchmark results

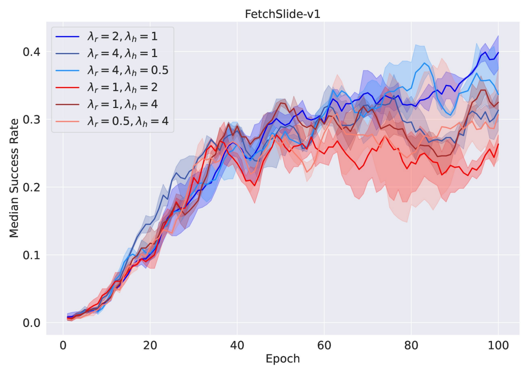
Our evaluation of different methods is based on DDPG. We use different methods to sample hindsight experiences to replay and train policies on the environments issuing sparse rewards. We compared the following baselines in



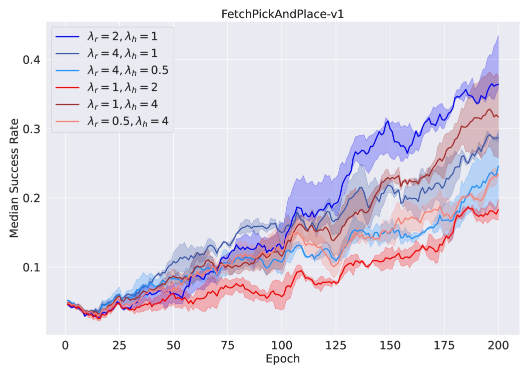
(a) FetchReach-v1



(b) FetchPush-v1

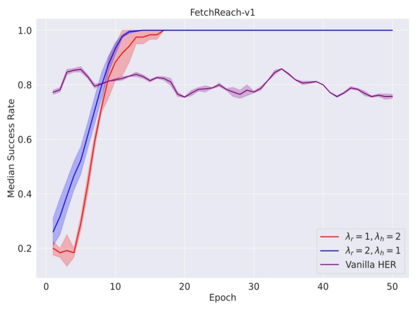


(c) FetchSlide-v1

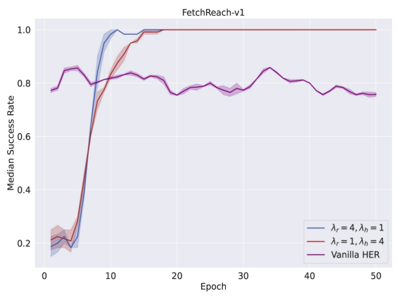


(d) FetchPickAndPlace-v1

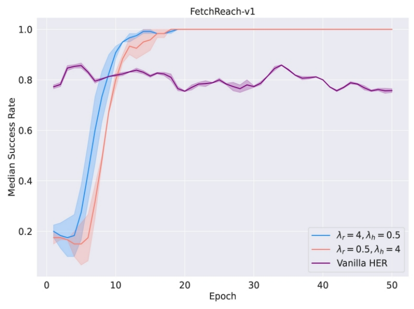
Fig. 4. Robot Fetch environments.



(a) $\lambda_r, \lambda_h \in \{1, 2\}$



(b)$\lambda_r, \lambda_h \in \{1, 4\}$



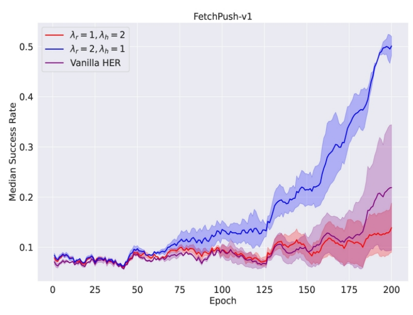
(c) $\lambda_r, \lambda_h \in \{0.5, 4\}$

Fig. 5. Comparison of different weights with Vanilla HER in FetchReach-v1.

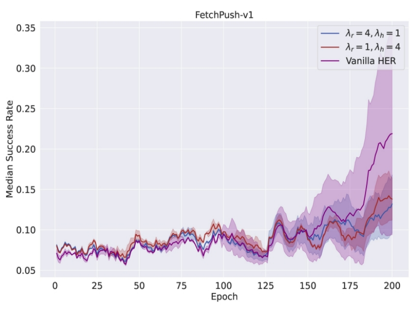
our experiments.

- **DDPG** [32]: which is a model-free RL algorithm for continuous control that simultaneously learns Q-functions and policies. It learns a deterministic policy by using a stochastic counterpart to explore in the training.
- **HER** [16]: which uses hindsight goals to obtain hindsight rewards without adjustment to the bias.
- **CHER** [21]: which controls the exploration–exploitation trade-off in HER by the difficulty and diversity of hindsight goals via hindsight experience selection.
- **DHER** [33]: which is able to address the tasks with sparse rewards and dynamic goals.

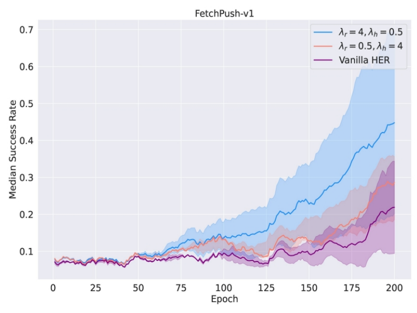
We sequentially selected the curves with the best weight fits for the different baseline algorithms in the ablation



(a) $\lambda_r, \lambda_h \in \{1, 2\}$

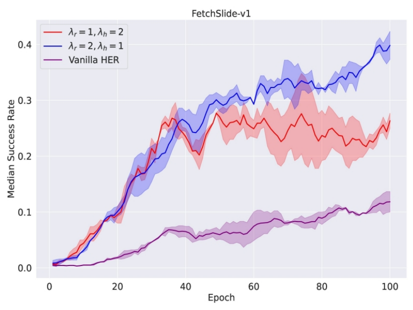


(b)$\lambda_r, \lambda_h \in \{1, 4\}$

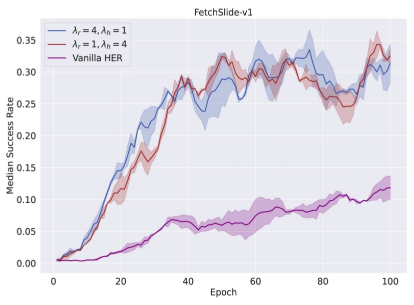


(c) $\lambda_r, \lambda_h \in \{0.5, 4\}$

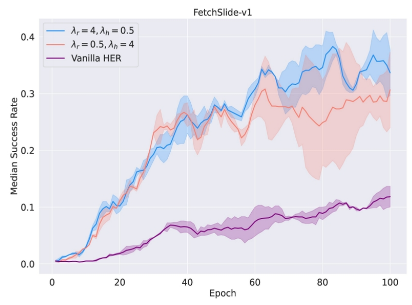
Fig. 6. Comparison of different weights with Vanilla HER in FetchPush-v1.



(a) $\lambda_r, \lambda_h \in \{1, 2\}$

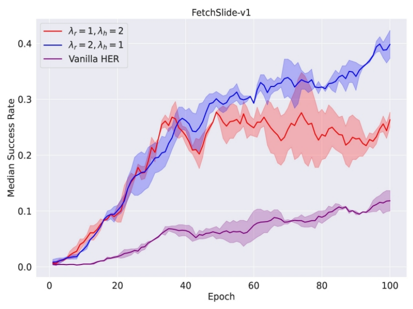


(b)$\lambda_r, \lambda_h \in \{1, 4\}$

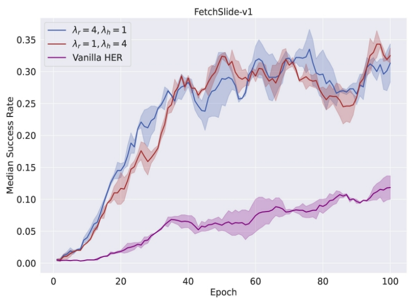


(c) $\lambda_r, \lambda_h \in \{0.5, 4\}$

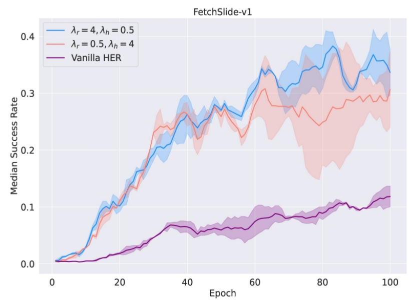
Fig. 7. Comparison of different weights with Vanilla HER in FetchSlide-v1.



(a) $\lambda_r, \lambda_h \in \{1, 2\}$



(b)$\lambda_r, \lambda_h \in \{1, 4\}$

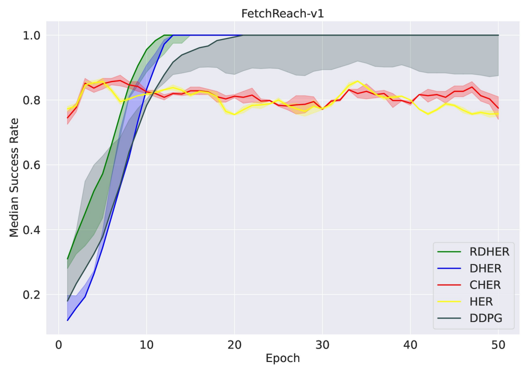


(c) $\lambda_r, \lambda_h \in \{0.5, 4\}$

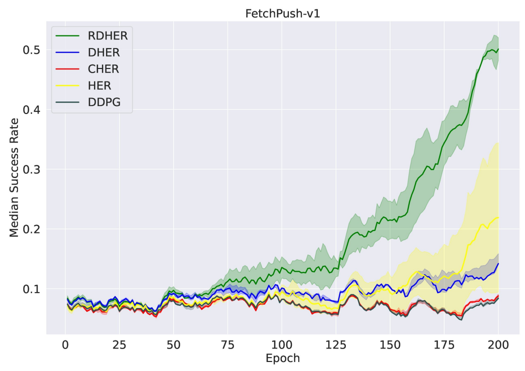
Fig. 8. Comparison of different weights with Vanilla HER in FetchPickAndPlace-v1.

experiments (Section 5.2), where the parameters selected in the FetchReach and FetchPickAndPlace environments were λ_r=2, λ_h=1, the parameters selected in the FetchPush and FetchSlide environments were λ_r=4, λ_h= 0.5. Fig. 9 depicts the median test success rates of the RDHER method and baseline methods on the four Fetch tasks. We use the best-learned policy for evaluation and test it in the environment. The testing results are the mean success rates. The best trained success rate of each algorithm in the set number of episodes is shown in Table 2, and the time spent in a single run is shown in Table 3. In the FetchReach and FetchSlide environments, RDHER method takes the least training time under the premise of ensuring the success rate. In the other two environments, although it is not as time-consuming as DDPG and HER, it is still slightly less time-consuming than DHER. Compared to all other methods, our method learns relatively faster and has a higher success rate. In particular, without using HER, DDPG is almost unable to solve any task other than FetchReach.

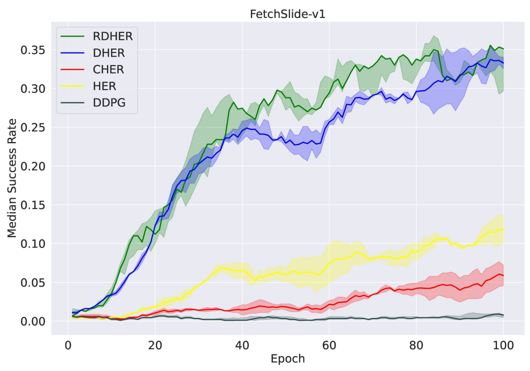
In summary, we fit different weighting factors to different environments to train the algorithm, and derive the optimal ratio of hindsight rewards to actual rewards in different environments by ablation studies.



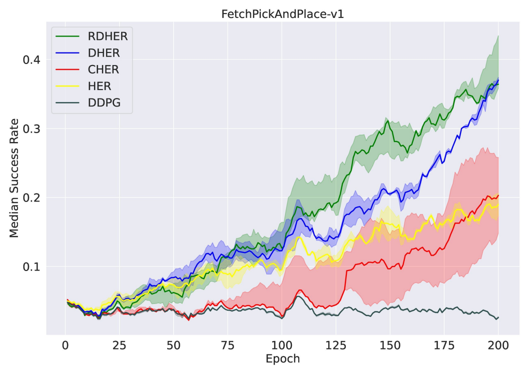
(a) FetchReach-v1



(b) FetchPush-v1



(c) FetchSlide-v1



(d) FetchPickAndPlace-v1

Fig. 9. Shows the training curves for the five algorithms. To make a fair comparison for each algorithm, we repeated the experiment five times for each algorithm in all tasks to exclude the effect of randomness on the comparison results. The reward curves for the algorithms are plotted after taking the sliding average of the five experiments.

Table 2. Comparison of mean success rate over seeds of the best policy. Bold scores signify the best score out of all methods.

	DDPG	HER	CHER	DHER	RDHER(ours)
FetchReach	100%	88%	89%	100%	100%
FetchPush	15%	39%	14%	21%	**58%**
FetchSlide	2%	24%	17%	41%	**50%**
FetchPickAndPlace	8%	25%	33%	45%	**47%**

Table 3. Training time (hours:minutes:seconds) in all four environments (single run).

	DDPG	HER	CHER	DHER	RDHER(ours)
FetchReach	00:50:24	00:46:55	01:45:13	00:11:27	00:11:16
FetchPush	03:35:59	03:30:32	06:43:20	04:10:14	03:53:16
FetchSlide	01:53:53	01:49:18	03:41:51	01:57:47	01:43:58
FetchPickAndPlace	03:34:50	03:28:32	07:50:02	04:06:21	03:50:06

6. Conclusion

In this paper, aiming at the hindsight bias caused by random strategies in dynamic hindsight experience replay, a reinforcement learning algorithm based on a weighted hindsight experience replay structure is proposed. We use numerically larger hindsight rewards to weaken the bias and improve the sampling efficiency of the algorithm. Our ablation experiments demonstrate that: (1) when the specified hindsight weights are opposite to the reward function of the environment, the bias is amplified and the reward is reduced; (2) learning exclusively from only hindsight experiences decreases the performance of the algorithm; (3)

in addition, for benchmark comparison experiments, the RDHER algorithm is more effective and advantageous in four challenging robotic tasks compared to several previous baseline algorithms.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61806004) and the Anhui Provincial Natural Science Foundation (Grant No. 2208085MF168).

References

[1] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, (2018) *"An introduction to deep reinforcement learning"* **Foundations and Trends in Machine Learning *11***(3-4): 219–354. DOI: 10.1561/2200000071.

[2] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

[3] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, (2020) *"Mastering Atari, Go, chess and shogi by planning with a learned model"* **Nature *588***(7839): 604–609. DOI: 10.1038/s41586-020-03051-4.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, (2017) *"Mastering the game of Go without human knowledge"* **Nature *550***(7676): 354–359. DOI: 10.1038/nature24270.

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, (2018) *"A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play"* **Science *362***(6419): 1140–1144. DOI: 10.1126/science.aar6404.

[6] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al., (2019) *"Dota 2 with large scale deep reinforcement learning"* **arXiv preprint arXiv:1912.06680**:

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, (2015) *"Human-level control through deep reinforcement learning"* **Nature *518***(7540): 529–533. DOI: 10.1038/nature14236.

[8] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, (2019) *"Grandmaster level in StarCraft II using multi-agent reinforcement learning"* **Nature *575***(7782): 350–354. DOI: 10.1038/s41586-019-1724-z.

[9] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, (2018) *"Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection"* **International Journal of Robotics Research *37***(4-5): 421–436. DOI: 10.1177/0278364917710318.

[10] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, (2020) *"Learning dexterous in-hand manipulation"* **International Journal of Robotics Research *39***(1): 3–20. DOI: 10.1177/0278364919887447.

[11] S. Gu, E. Holly, T. Lillicrap, and S. Levine. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: Cited by: 777; All Open Access, Green Open Access. 2017, 3389–3396. DOI: 10.1109/ICRA.2017.7989385.

[12] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Van De Wiele, V. Mnih, N. Heess, and T. Springenberg. "Learning by playing - Solving sparse reward tasks from scratch". In: ***10***. Cited by: 45. 2018, 6910–6919.

[13] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, (2019) *"Deep Reinforcement Learning with Optimized Reward Functions for Robotic Trajectory Planning"* **IEEE Access *7***: 105669–105679. DOI: 10.1109/ACCESS.2019.2932257.

[14] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al., (2018) *"Multi-goal reinforcement learning: Challenging robotics environments and request for research"* **arXiv preprint arXiv:1802.09464**:

[15] J. Tarbouriech, E. Garcelon, M. Valko, M. Pirotta, and A. Lazaric. "No-regret exploration in goal-oriented reinforcement learning". In: ***PartF168147-13***. Cited by: 5. 2020, 9370–9379.

[16] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. "Hindsight experience replay". In: ***2017-December***. Cited by: 616. 2017, 5049–5059.

[17] R. Zhao and V. Tresp. "Energy-based hindsight experience prioritization". In: *Conference on Robot Learning*. PMLR. 2018, 113–122.

[18] B. Manela and A. Biess, (2021) *"Bias-reduced hindsight experience replay with virtual goal prioritization"* **Neurocomputing *451***: 305–315. DOI: 10.1016/j.neucom.2021.02.090.

[19] S. Lanka and T. Wu, (2018) *"Archer: Aggressive rewards to counter bias in hindsight experience replay"* **arXiv preprint arXiv:1809.02070**:

[20] C. Bai, L. Wang, Y. Wang, Z. Wang, R. Zhao, C. Bai, and P. Liu, (2023) *"Addressing Hindsight Bias in Multigoal Reinforcement Learning"* **IEEE Transactions on Cybernetics *53***(1): 392–405. DOI: 10.1109/TCYB.2021.3107202.

[21] M. Fang, T. Zhou, Y. Du, L. Han, and Z. Zhang. "Curriculum-guided hindsight experience replay". In: ***32***. Cited by: 43. 2019.

[22] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[23] Y. Wen, J. Si, A. Brandt, X. Gao, and H. H. Huang, (2020) *"Online Reinforcement Learning Control for the Personalization of a Robotic Knee Prosthesis"* **IEEE Transactions on Cybernetics *50***(6): 2346–2356. DOI: 10.1109/TCYB.2019.2890974.

[24] S. B. Niku. *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2020.

[25] X. Wang, L. Ke, Z. Qiao, and X. Chai, (2021) *"Large-Scale Traffic Signal Control Using a Novel Multiagent Reinforcement Learning"* **IEEE Transactions on Cybernetics *51***(1): 174–187. DOI: 10.1109/TCYB.2020.3015811.

[26] Y.-C. Liu and C.-Y. Huang, (2022) *"DDPG-Based Adaptive Robust Tracking Control for Aerial Manipulators With Decoupling Approach"* **IEEE Transactions on Cybernetics 52**(8): 8258–8271. DOI: 10.1109/TCYB.2021.3049555.

[27] M. Patacchiola and A. Cangelosi, (2022) *"A Developmental Cognitive Architecture for Trust and Theory of Mind in Humanoid Robots"* **IEEE Transactions on Cybernetics 52**(3): 1947–1959. DOI: 10.1109/TCYB.2020.3002892.

[28] T. Schaul, D. Horgan, K. Gregor, and D. Silver. "Universal value function approximators". In: **2**. Cited by: 364. 2015, 1312–1320.

[29] V. Pong, S. Gu, M. Dalal, and S. Levine, (2018) *"Temporal difference models: Model-free deep rl for model-based control"* **arXiv preprint arXiv:1802.09081**:

[30] S. Zhang and R. S. Sutton, (2017) *"A deeper look at experience replay"* **arXiv preprint arXiv:1712.01275**:

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, (2013) *"Playing atari with deep reinforcement learning"* **arXiv preprint arXiv:1312.5602**:

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, (2015) *"Continuous control with deep reinforcement learning"* **arXiv preprint arXiv:1509.02971**:

[33] M. Fang, C. Zhou, B. Shi, B. Gong, J. Xu, and T. Zhang. "Dher: Hindsight experience replay for dynamic goals". In: Cited by: 21. 2019.

[34] L. F. Vecchietti, M. Seo, and D. Har, (2022) *"Sampling Rate Decay in Hindsight Experience Replay for Robot Control"* **IEEE Transactions on Cybernetics 52**(3): 1515–1526. DOI: 10.1109/TCYB.2020.2990722.

[35] P. Rauber, A. Ummadisingu, F. Mutz, and J. Schmidhuber, (2017) *"Hindsight policy gradients"* **arXiv preprint arXiv:1711.06006**:

[36] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. "Visual reinforcement learning with imagined goals". In: ***2018-December***. Cited by: 137. 2018, 9191–9200.

[37] D. P. Kingma and M. Welling, (2013) *"Auto-encoding variational bayes"* **arXiv preprint arXiv:1312.6114**:

[38] C. Bai, L. Wang, L. Han, J. Hao, A. Garg, P. Liu, and Z. Wang. "Principled exploration via optimistic bootstrapping and backward induction". In: *International Conference on Machine Learning*. PMLR. 2021, 577–587.

[39] H. Liu, A. Trott, R. Socher, and C. Xiong, (2019) *"Competitive experience replay"* **arXiv preprint arXiv:1902.00528**:

[40] R. Zhao, X. Sun, and V. Tresp. "Maximum entropy-regularized multi-goal reinforcement learning". In: ***2019-June***. Cited by: 8. 2019, 13022–13035.

[41] C. Colas, P. Founder, O. Sigaud, M. Chetouani, and P.-Y. Oudeyer. "CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning". In: ***2019-June***. Cited by: 9. 2019, 2372–2387.

[42] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel. "Goal-conditioned imitation learning". In: ***32***. Cited by: 43. 2019.

[43] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, (2018) *"Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning"* **arXiv preprint arXiv:1809.05188**:

[44] S. Nasiriany, V. H. Pong, S. Lin, and S. Levine. "Planning with goal-conditioned policies". In: ***32***. Cited by: 38. 2019.

[45] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control". In: Cited by: 1643. 2012, 5026–5033. DOI: 10.1109/IROS.2012.6386109.