# Sampling Rate Decay in Hindsight Experience Replay for Robot Control

Luiz Felipe Vecchietti, Minah Seo, and Dongsoo Har, *Senior Member, IEEE*

***Abstract*—Training agents via deep reinforcement learning with sparse rewards for robotic control tasks in vast state space are a big challenge, due to the rareness of successful experience. To solve this problem, recent breakthrough methods, the hindsight experience replay (HER) and aggressive rewards to counter bias in HER (ARCHER), use unsuccessful experiences and consider them as successful experiences achieving different goals, for example, hindsight experiences. According to these methods, hindsight experience is used at a fixed sampling rate during training. However, this usage of hindsight experience introduces bias, due to a distinct optimal policy, and does not allow the hindsight experience to take variable importance at different stages of training. In this article, we investigate the impact of a variable sampling rate, representing the variable rate of hindsight experience, on training performance and propose a sampling rate decay strategy that decreases the number of hindsight experiences as training proceeds. The proposed method is validated with three robotic control tasks included in the OpenAI Gym suite. The experimental results demonstrate that the proposed method achieves improved training performance and increased convergence speed over the HER and ARCHER with two of the three tasks and comparable training performance and convergence speed with the other one.**

***Index Terms*—Hindsight experience replay (HER), machine learning, reinforcement learning (RL), robot control, sampling rate decay.**

## I. Introduction

REINFORCEMENT learning (RL) is a powerful framework that makes an agent learn a task by taking actions in an environment so as to maximize the accumulated generic rewards over time. The use of deep neural networks for the nonlinear approximation of a policy function led to breakthrough results in simulated game environments, such as learning how to play a range of Atari games [1] and mastering the game of Go without human knowledge [2], [3]. Recently, deep RL has also been widely used for robot control in 3-D space [4]–[6] and applied to online robotic prosthesis personalization [7]. Deep RL presents a key technical advantage since it is able to learn a generalized policy from given examples. Nevertheless, the design of a deep RL framework involves various challenges. One of these challenges is creating a reward function that allows the agent to learn tasks efficiently. Of the various types of reward functions, the binary reward function is the simplest and popularly used for many applications. When the binary reward is 0 or $-1$ and it is sparse, reward 0 is given to target outcome or goal, and reward $-1$ is assigned to other transitional outcomes.

However, in applications where exploration space is vast, such as robotic control tasks in 3-D space, usage of binary sparse rewards requires substantial training time. Because most of the actions are taken randomly at the beginning of learning, the agent is not provided with lots of successful experiences for efficient training. A successful experience is a transition from the current state to the next state when the goal is achieved in the next state. Therefore, overcoming the rare presence of successful experience with binary sparse rewards is critical when training an RL algorithm for tasks in vast state space. This can be achieved by utilizing abundant unsuccessful experiences in the manner of increasing sampling efficiency, taking appropriate rates (proportions) of successful and unsuccessful experiences, during training.

The classic experience replay (ER) framework [8] stores experience collected by the agent in a database known as the *replay buffer*. During training, experiences are sampled uniformly from the replay buffer. Each experience represents a combination of the current state, goal, current action, next state, and reward associated with an environment at each time. However, it is natural to think that the agent can learn more efficiently from some experiences than from others. These experiences should be used more often during training for improved performance. To address this issue, various sampling strategies have lately appeared in the literature. The prioritized ER (PER) [9] samples experiences with high expected learning improvement more frequently. It uses the temporal-difference (TD) error magnitude as the prioritization mechanism to replay experiences. PER is integrated with heuristic dynamic programming in [10]. A method called combined ER is proposed in [11] by which the latest experiences in the replay buffer are added to the batch while training. In [12], the replay buffer is split into two parts, one for successful experiences and the other for unsuccessful experiences. When creating the training batch, successful experiences take higher priority, because they are rare to occur. To maintain a proper rate of successful

Manuscript received July 3, 2019; revised January 7, 2020; accepted April 16, 2020. This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by Korea Government (MSIT) (Development of Artificial Intelligence Technology that Continuously Improves Itself as the Situation Changes in the Real World) under Grant 2020-0-00440. This article was recommended by Associate Editor H. M. Schwartz. *(Corresponding author: Minah Seo.)*

The authors are with the Cho Chun Shik Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: lfelipesv@kaist.ac.kr; minah_seo@kaist.ac.kr; dshar@kaist.ac.kr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.2990722

experience in a batch, they are sampled with a fixed probability. Maximum entropy-based prioritization (MEP) is proposed in [13] with the objective to encourage the sampling of more diverse goals during training. In [14], the type of experience to replay is selected based on age, TD error, and strength of associated exploration noise. ER for real-time control is explored in [15].

The hindsight ER (HER) proposed in [16] attempts to solve the problem of sampling efficiency in multigoal (multiple different goals) tasks. In a traditional RL method, if the agent fails to achieve the goal, the episode is considered a failure and no information is gained from that episode. The main concept of HER comes from the proposition that even though the agent has not succeeded at a specific goal, it achieved a different one at least. Therefore, while substituting the original goal by the achieved goal and recomputing the rewards in hindsight, it is possible to learn not only from successful experiences but also from unsuccessful experiences. It is noted that the goal affects the agent's actions rather than the environment dynamics. Consider a robot arm with the objective to reach goal $P$. When it reaches a different goal $S$ at the end of an episode, it is judged unsuccessful in achieving original goal $P$. On the other hand, it is successful in achieving goal $S$. The agent stores the experience associated with the original goal $P$, corresponding to real experiences, including unsuccessful experiences, in a buffer. To train its policy function, the experiences are taken under the context of achieving goal $S$ in hindsight with the rewards recomputed as if the goal is $S$. Through this manipulation, an unsuccessful experience can play a role as a successful experience and as a result sampling efficiency is increased.

The improvement in sampling efficiency provided by the HER combined with off-policy RL algorithms, such as the deep deterministic policy gradients (DDPG) [17] and deep $Q$-Networks [1], allows the learning of complicated robotic control tasks [16]. The use of hindsight experience in the HER is a type of curriculum learning [18] where the current policy learns to perform goals that it can achieve implicitly. Handcrafted curriculum to train neural networks is presented in [19] and [20]. Automatic task selection is proposed in [21] and [22]. To select a task achievable by the policy, the use of self-play to automatically generate achievable goals is proposed in [23] and [24].

For a given state of the environment, it is assumed by the HER that the agent takes the same action even with the replacement of the original goal with a new goal. However, if the agent is executed with the new goal, it is unlikely that the action of the agent is the same, due to the distinct optimal policy to reach a different goal. Hence, the action value at a state with the new goal is higher than its real value and the probability of taking that action by the trained policy is overestimated as a consequence. The overestimation of the action value introduces bias to the model and affects the learning of optimal policies by the agent. To counter the bias introduced by the use of hindsight experience the aggressive rewards to counter bias in HER (ARCHER) is proposed in [25]. It is of significant importance to counter the bias in the HER and examine the challenge on how and with which frequency hindsight experience should be used. The solution should be able to improve sampling efficiency while countering the bias.

In this article, it is shown that a key factor in learning multigoal tasks with the HER is the (relative) rate of hindsight experience used in each training epoch. The approach described in [16] replaces real experience with hindsight experience at a fixed rate during the entire training process. Our findings, however, suggest that hindsight experiences are more important at the start of training when a robot learns basic sensing skills and subtasks necessary to achieve the goal. Taking this into account, a method on efficient adjustment of the rate of hindsight experience is proposed, using a variable sampling rate between real and hindsight experiences during training. The main contributions of this article are as follows.

1) A sampling rate decay strategy to the HER that decreases the rate of hindsight experience as training proceeds is proposed.
2) An investigation to define guidelines on how to choose the hyperparameters of the sampling rate decay strategy is conducted. It is found that the best pairs of hyperparameters lead to the mean rates of hindsight experience that are close to the best values found in [16].
3) An analysis is presented for the performance evaluation of the proposed method with different sampling rate decay strategies. Experiments are performed with two baseline methods HER and ARCHER, and the prioritization method MEP for three robotic control tasks.

The remainder of this article is organized as follows. In Section II, the concepts of multigoal RL of the DDPG, HER, and ARCHER algorithms are described. Section III describes the proposed method in details. Section IV presents experimental results obtained with the proposed method, HER, and ARCHER. The details of the hyperparameters used for the experiments are listed in the Appendix. Concluding remarks are in Section V.

## II. System Modeling

In this section, the concept and the mathematical model of the multigoal RL framework for the DDPG, HER, and ARCHER algorithms are presented.

### A. Multigoal Reinforcement Learning

In a multigoal RL framework, the agent is told what to do based on an environmental goal concatenated with the environmental state, both of which are the input to the policy. The objective is to train the goal-conditioned policy to effectively generalize its behavior for multiple environmental goals.

Let $\mathcal{S}$, $\mathcal{G}$, and $\mathcal{A}$ be the state space, goals, and actions, respectively. The input of the RL agent consists of state $s \in \mathcal{S}$ and goal $g \in \mathcal{G}$. For each goal $g \in \mathcal{G}$, the predicate of state $s$ is defined as $f_g : \mathcal{S} \to \{0, 1\}$, representing that the final target of the RL agent is to achieve a state $s$ satisfying $f_g(s) = 1$. It is assumed that a mapping between state $s$ and goal $g$ exists and goal $g$ exists for every state $s$. The mapping $m$ from state to goal can be described as $m : \mathcal{S} \to \mathcal{G}$ s.t. $\forall_{s \in \mathcal{S}} f_{m(s)}(s) = 1$ [16]. Consider an agent that interacts with a

fully observable environment. The environment can be defined by $\mathcal{S}$, $\mathcal{G}$, and $\mathcal{A}$, a probability distribution $p(s_0, g)$ of initial state $s_0$ and goal $g$, a reward function $r\colon \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$, transition probabilities $p(s_{t+1}|s_t, a_t)$ at time $t$, and a discount factor $\gamma \in [0, 1]$. The agent selects actions according to a deterministic policy $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ that maps a 2-tuple of $(s, g)$ to actions.

An episode starts by sampling the initial state and a goal from $p(s_0, g)$. For the current state and goal at time $t$, current action is equal to $a_t = \pi(s_t, g)$. As a consequence of $a_t$ and $s_t$, the reward at time $t$ is equal to $r_t = r(s_t, g, a_t)$. The next state $s_{t+1}$ is sampled according to the transition probability $p(s_{t+1}|s_t, a_t)$. The objective of the agent is to maximize the expected discounted sum of future rewards. The $Q$-function or the action-value function following a policy $\pi$ is defined as

$$Q^{\pi}(s_t, g, a_t) = \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r_i | s_t, g, a_t\right]. \tag{1}$$

For an optimal policy $\pi^*$, the optimal $Q$-function $Q^*$ satisfies the Bellman equation [17] given as

$$Q^*(s_t, g, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, g, a_{t+1})\right] \tag{2}$$

where $Q^*$ is expected sum of reward at time $t$ and discounted maximum $Q^*$ in the next state obtained by a certain future action $a_{t+1}$.

The policy $\pi$ and the $Q$-function $Q^{\pi}$ can be approximated by deep neural networks. The neural networks used to approximate the policy and $Q$-function are called *actor* and *critic* networks, respectively. The DDPG algorithm is used to train these networks. The DDPG is an off-policy and model-free RL algorithm for continuous action space [17]. The critic network can be directly trained by minimizing the loss of the critic network $\mathcal{L}_c$, which can be expressed as

$$\mathcal{L}_c = (r_t + \gamma Q(s_{t+1}, g, \pi(s_{t+1}, g)) - Q(s_t, g, a_t))^2. \tag{3}$$

The actor network is optimized by minimizing the loss of the actor network $\mathcal{L}_a$ as follows:

$$\mathcal{L}_a = -\mathbb{E}_s\big[Q(s_{t+1}, g, \pi(s_{t+1}, g))\big]. \tag{4}$$

For exploration, during training, the action $a_t$ taken by the agent is the sum of the output of the actor network $\pi(s_t, g)$ and a random exploration noise $\mathcal{N}_t$, that is, $a_t = \pi(s_t, g) + \mathcal{N}_t$. To stabilize training, replicas of the actor network and critic network are used. The weights of the replicas are gradually updated with the original network parameters during training.

### B. Hindsight Experience Replay

Let state $s_0$ and goal $g$ be sampled from a distribution $p(s_0, g)$ to initiate an episode. Then, for each environment step, an experience $(s_t, g, a_t, s_{t+1}, r_t)$ is stored in the ER buffer until the end of an episode. A binary reward function returning $-1$ for a failure and 0 for a success can be defined as

$$r_t = r(s_t, g, a_t) = \begin{cases} 0, & \text{when } f_g(s_{t+1}) = 1 \\ -1, & \text{when } f_g(s_{t+1}) = 0. \end{cases} \tag{5}$$

By the HER, a minibatch is sampled from the replay buffer and part of the experiences of the minibatch are substituted by hindsight experiences. The sampling rate $\epsilon$ is the rate of hindsight experience. $\epsilon$ is a hyperparameter that can be changed, as suggested in [16]. A hindsight experience is a 5-tuple in the form of $(s_t, g^h, a_t, s_{t+1}, r_t^h)$, where the hindsight reward $r_t^h$ is recomputed for the hindsight goal $g^h$ as

$$r_t^h = r\Big(s_t, g^h, a_t\Big) = \begin{cases} 0, & \text{when } f_{g^h}(s_{t+1}) = 1 \\ -1, & \text{when } f_{g^h}(s_{t+1}) = 0 \end{cases} \tag{6}$$

for each hindsight experience with the achieved hindsight goal $g^h$.

### C. Aggressive Rewards to Counter Bias in HER

The ARCHER [25] assumes that real and hindsight experiences do not have the same level of importance and, as a result, they cannot be rewarded in the same way. It gives aggressive rewards toward hindsight experiences to increase the $Q$-value of the hindsight state–action pairs. Tradeoff parameters $\lambda_r$ and $\lambda_h$ are used to specify weights of individual rewards of real and hindsight experiences. These hyperparameters are used as follows:

$$r_t = \lambda_r \times r(s_t, g, a_t) \tag{7}$$

$$r_t^h = \lambda_h \times r\Big(s_t, g^h, a_t\Big). \tag{8}$$

Because hindsight experience should be valued higher than real experience, a necessary condition for the ARCHER is that $r_t^h > r_t$. The HER is a special case of the ARCHER with $\lambda_r = \lambda_h = 1$.

## III. Proposed Method

In this section, a novel sampling rate decay strategy for the HER is described. The proposed method controls the rate of hindsight experience in the minibatch, as training proceeds, to train actor and critic networks of the DDPG algorithm. The proposed method acts as a form of implicit curriculum and reduces the bias problem at the end of training, because of the decrease in the number of hindsight experiences used.

The value of the hyperparameter $\epsilon$ of the HER remains fixed during the entire training process. In [16], the learning performance with the values of $\epsilon = \{0.5, 0.667, 0.8, 0.889, 0.941, 1\}$ is analyzed. The best performance is obtained when $\epsilon = 0.8$ or 0.889. It is shown in [16] that $\epsilon$ very close to 1, corresponding to a small rate of real experience, causes degraded training performance. Because the best performance is obtained when the majority of experiences are substituted in hindsight, the bias generated because of the use of hindsight experiences is a problem during the training process. It is proposed by the ARCHER to make the rewards of hindsight experiences greater than those of real experiences. Increasing the rewards of hindsight experiences causes the policy to choose the action for the given hindsight goal [25]. Our hypothesis, on the other hand, comes from the principle that hindsight experience is more important at the beginning of the training process. In this way, the action values of hindsight experiences are maximized, even with overestimated transition probabilities. As training proceeds, the rate of hindsight experience is decayed

**Algorithm 1** Proposed Method: HER With Sampling Rate Decay Strategy

**Given:** an off-policy RL algorithm $\mathbb{A}$, a strategy $\mathbb{S}$ for sampling goals for replay, a reward function $r: \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$, a real reward weight $\lambda_r$, a hindsight reward weight $\lambda_h$, and a sampling rate decay strategy $\mathbb{D}$

Initialize neural networks $\mathbb{A}$
Initialize replay buffer $R$
**for** epoch = 1, K **do**
  Get the sampling rate $\epsilon$ using the sampling rate decay strategy $\mathbb{D}$
  **for** episode = 1, M **do**
    Sample a goal $g$ and initial state $s_0$
    **for** t=0,T-1 **do**
      Sample an action $a_t$ using the behavior policy from $\mathbb{A}$ : $a_t \leftarrow \pi(s_t, g) + \mathcal{N}_t$
      Execute the action $a_t$ and observe new state $s_{t+1}$
      Calculate reward $r_t = \lambda_r \times r(s_t, g, a_t)$
      Store the experience $(s_t, g, a_t, r_t, s_{t+1})$ in replay buffer $R$
    **end for**
    **for** t=1,N **do**
      Sample a minibatch $B$ from the replay buffer $R$
      Sample a set of experiences in $B$ with sampling rate $\epsilon$ to substitute by achieved goals $G$ sampled with $\mathbb{S}$
      **for** $g^h \in G$ **do**
        $r_t^h = \lambda_h \times r(s, g^h, a)$
        Substitute the experience $(s_t, g, a_t, r_t, s_{t+1})$ for the experience $(s_t, g^h, a_t, r_t^h, s_{t+1})$
      **end for**
      Perform one step of optimization using $\mathbb{A}$ and minibatch $B$
    **end for**
  **end for**
**end for**

(decreased), while the rate of real experience is increased. The presence of unsuccessful real experience is also important to ensure that the robot has a large enough sample for the generalization of the policy over the entire state–action space [26]. Therefore, the sampling rate decay strategy for hindsight experience in conjunction with proper rates of successful and unsuccessful real experiences during training can be tested from the perspective of the sampling efficiency and diversity of the sample in the state–action space.

Our hypothesis is tested by introducing a sampling rate decay strategy $\mathbb{D}$ to the HER. The strategy controls the rate of hindsight experience contained in the minibatch $B$. The pseudocode of the proposed method is presented in Algorithm 1.

The sampling rate decay strategy used in experiments can be mathematically shown as

$$\epsilon = 1 - \beta_1 \left(\frac{k}{K}\right)^{\exp(\beta_2)} \tag{9}$$

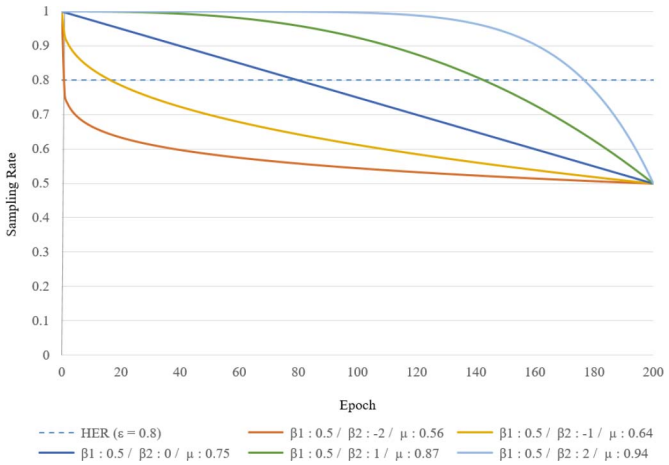


Fig. 1. Sampling rate decay strategy according to hyperparameters $\beta_1$ and $\beta_2$. The hyperparameters $\beta_1$ and $\beta_2$ determine the rate of hindsight experience and decay pattern.

where $k$ is the index of current training epoch and $K$ is the total number of training epochs. The hyperparameters $\beta_1$ and $\beta_2$ mainly determine the rate of hindsight experience around the end of training and the decay pattern, respectively, as shown in Fig. 1. The parameter $\beta_1$ can assume values between 0 and 1 and the parameter $\beta_2$ can assume any real value. When $\beta_2$ is equal to zero, the decay pattern is linear. For $\beta_2 > 0$, the decay pattern follows a concave exponential curve, whereas for $\beta_2 < 0$ it follows a convex exponential curve. Various values of $\beta_1$ and $\beta_2$ are tested during the experiments.

Fig. 2 shows the variation of composition profile of minibatch at each epoch index for the *Push* task. The experiment environment and specification of minibatch are stated in Section IV. The leftmost figure represents the variation of the rate of real experience in a minibatch during training, before substitution in hindsight. Blue color at each epoch index indicates successful real experience and red color denotes unsuccessful real experience. As training proceeds, the rate of successful real experience increases, which means the agent is learning to perform the desired task, and around at the epoch index 150 it is approximately 25%. The yellow curve divides real experience into successful real experience and unsuccessful real experience at each epoch index. Figures on the right are composition profiles obtained by substitutions in hindsight. By the HER, portions of successful real experience and unsuccessful real experience are replaced with hindsight experience. The black vertical line in the second leftmost figure, corresponding to the HER, shows the composition of experiences at the epoch index 150. The rate of hindsight experience is kept at 80%, which is pertinent to the width of the violet section at each epoch index. At the epoch index 150, the section between a1 and a2 represents the hindsight experience replacing successful real experience. Likewise, the section between a2 and a3 represents the hindsight experience substituting unsuccessful real experience. Due to the random sampling for the creation of a minibatch, the portion of the hindsight experience replacing successful real experience gradually rises, as suggested in the leftmost figure, whereas that of the hindsight experience

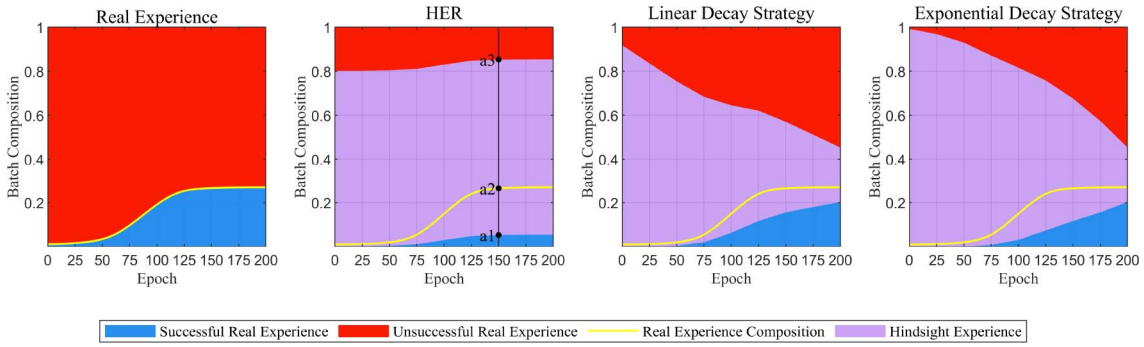


Fig. 2. Composition profile of a minibatch in each epoch with HER and the proposed method. For HER, the rate is fixed at $\epsilon = 0.8$, as suggested in [7]. For linear decay strategy $\beta_1 = 0.75$ and $\beta_2 = 0$ and for exponential decay strategy $\beta_1 = 0.75$ and $\beta_2 = 0.75$.

replacing unsuccessful real experience gradually decreases. As a result, at the epoch index 150, the minibatch is composed of 5.3% of successful real experience, 80% of hindsight experience, and 14.7% of unsuccessful real experience.

By the proposed method, the rate of hindsight experience is controlled by (9). For the linear and exponential decay strategies, the rate of hindsight experience is higher than that of the HER at the beginning of training and keeps decreasing as training proceeds. Comparing with the HER, the proposed method takes more real experience, more specifically unsuccessful real experience, which is helpful for the agent to avoid undesired actions when it is trained up to a certain level. At the end of the training, the rate of real experience is significantly higher when compared with the HER, alleviating the bias problem as well as improving learning efficiency.

In order to compare the impacts of the fixed sampling rate and variable sampling rate, the mean value of $\epsilon$ and $\mu_\epsilon$, over $K$ epochs is calculated with different values of $\beta_1$ and $\beta_2$. $\mu_\epsilon$ is evaluated as follows:

$$\mu_\epsilon = \frac{1}{K}\sum_{k=1}^{K}\left(1-\beta_1\left(\frac{k}{K}\right)^{\exp(\beta_2)}\right). \tag{10}$$

In (9), the fraction $k/K$ can be approximated by a continuous variable ranging from 0 to 1, when $K$ is in the order of hundreds. This approximation of $k/K$ leads to the approximated value of $\mu_\epsilon$ in (9) as follows:

$$\mu_\epsilon \approx \int_0^1 \left(1-\beta_1(x)^{\exp(\beta_2)}\right)dx = 1-\frac{\beta_1}{1+\exp(\beta_2)}. \tag{11}$$

$\mu_\epsilon$ is used for the comparison presented in Section IV-C to select the values of $\beta_1$ and $\beta_2$.

## IV. Experiments

This section describes the experiment environment and presents experimental results of the HER, ARCHER, and proposed method.

### A. Experiment Environment

Experiments are conducted adopting the multigoal RL environment described in [27] for continuous control tasks. The experiment environment is developed with an integration of the OpenAI Gym environment [28] and the MuJoCo physics engine [29]. Experiment environment considered in this article is the *Fetch* environment and the experiment is to control a 7-DoF robot arm, which has a gripper of two parallel fingers.

The proposed method is tested with three tasks described in [27] as follows.

1) *Push (FetchPush):* Robot moves a box placed on a table to a target location. The robot achieves the goal by pushing and rolling the box. An episode is successful if the distance between the box and the desired target is less than 7 cm.
2) *Pick and Place (FetchPickAndPlace):* Robot grabs a box and moves it to a target location. This location can be positioned in the air or on a table surface. An episode is successful if the distance between the box and the desired target is less than 7 cm.
3) *Slide (FetchSlide):* Robot achieves the goal by hitting the puck. After the puck is hit, it slides and stops at the desired position because of friction. An episode is successful if the distance between the puck and desired target is less than 20 cm.

Fig. 3 shows an illustration of the experiment environment when running an episode for each task. More details about the tasks are referred to [27]. The multigoal RL framework for given tasks can be described as follows.

1) *States:* State consists of angles and velocities of the robot joints. For the object, puck, or box, it consists of position, rotation, and (linear and angular) velocity.
2) *Goals:* Goal describes the target position of the object in 3-D space. There exists a tolerance that can be depicted as a sphere with radius $\xi$ around the target position. $\xi$ is 7 cm for Push and Pick and Place tasks and 20 cm for the Slide task. Therefore, the predicate of $f_g(s)$ can be defined as

$$f_g(s) = \begin{cases} 1, & \text{when } \left|g - s_{\text{object}}\right| \le \xi \\ 0, & \text{when } \left|g - s_{\text{object}}\right| \ge \xi \end{cases} \tag{12}$$

where $g$ is the target position and $s_{\text{object}}$ is the object position in state $s$.
3) *Rewards:* Binary sparse rewards are used in the experiments as follows:

$$r(s_t, g, a_t) = \begin{cases} 0, & \text{when } f_g(s_{t+1}) = 1 \\ -1, & \text{when } f_g(s_{t+1}) = 0. \end{cases} \tag{13}$$

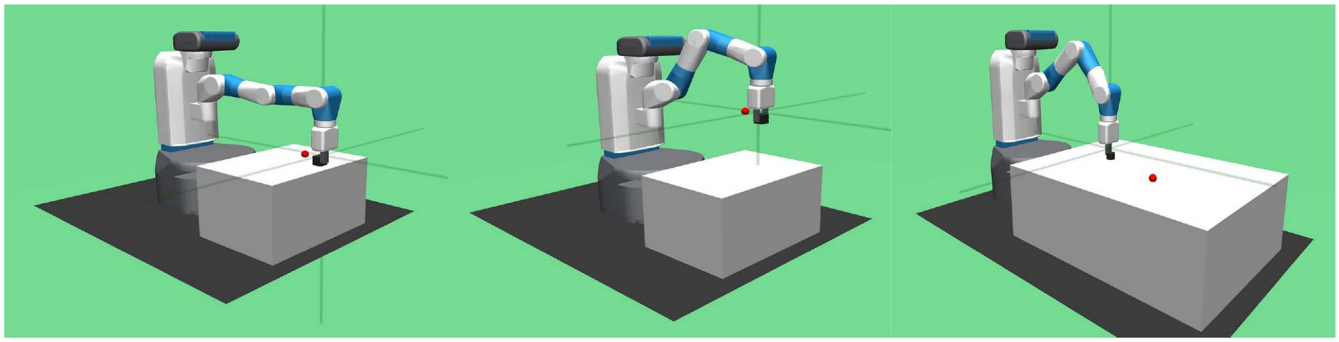


Fig. 3. Illustration of three tasks considered in experiments. (a) Push. (b) Pick and Place. (c) Slide.

4) *State–Goal Distributions:* Initial state of robot joints, including the joint of the gripper, is fixed and goal is chosen randomly. Initial state–goal pairs already satisfying the goal are discarded.
5) *Observations:* Observation consists of the position and linear velocity of the gripper, position and (linear and angular) velocity of the object, relative position of the object to the gripper position, and distance between the fingers of the gripper and target position.
6) *Actions:* Action space is 4-D, with one dimension corresponding to the distance between both fingers of the gripper and the other three dimensions pertinent to the gripper movement.
7) *Strategy for Sampling Goals for Replay: Future* strategy is used, as described in [16]. A goal achieved between the sampled experience and the end of the episode is randomly selected to be substituted in hindsight.
8) *Sampling Rate Decay Strategy:* The linear or exponential decay strategy is used as described in Section III.

The actor and critic networks use a multilayer perceptron architecture with rectified linear units (ReLUs) per layer as nonlinear activation functions. They are trained by backpropagation using the ADAM [30] as the optimizer. The DDPG algorithm [17] is used as the off-policy RL algorithm. Other values used for training are adopted from [27] and listed in the Appendix.

### B. Results

The experimental results obtained from the proposed sampling rate decay strategy are presented together with the results of the HER and ARCHER. For the HER, the parameters $\lambda_h$ and $\lambda_r$ are set to 1 and the rate of hindsight experience in sampling is kept at $\epsilon = 0.8$. For the ARCHER, the parameters $\lambda_h$ is set to 0.5 and $\lambda_r$ is set to 1, following those values in [25]. The rate of hindsight experience in sampling is also kept at $\epsilon = 0.8$. The training is performed for 200 epochs. For each training epoch, 50 episode cycles are executed by two parallel agents in the experiment environment and stored in the replay buffer, followed by 40 steps of optimization of the policy performed by the ADAM. After each training epoch, 20 test episodes are executed and the success rate is calculated. The success rate is defined as the rate of the successful episode that ends in achieving the respective goal. Each goal is given by the target position in 3-D space, as described in Section IV-A. To reduce granularity at each epoch index, the moving average of the success rate across 50 epochs is taken. The foregoing process is repeated with five random seeds and as a consequence, five smoothed curves are obtained. For each epoch index, the median of five success rates of the five smoothed curves is selected and presented as the result.

To choose the set of hyperparameters $\beta_1$ and $\beta_2$ used in the experiments a grid searching is performed. The range of $\beta_1$ is between 0 and 1 with the grid spacing 0.25. For the linear decay strategies, $\beta_2$ is set to 0. For the exponential decay strategies, an exhaustive search is performed to find the best value of $\beta_2$ with the grid spacing 0.25 for each $\beta_1$. The best values are to be found as the ones satisfying $\mu_\epsilon$ in the optimal region presented in Fig. 12. The sets of hyperparameters $\beta_1 = \beta_2 = 0.25$ and $\beta_1 = \beta_2 = 0.5$ leading to $\mu_\epsilon$ falling into the optimal region, and the sets of hyperparameters $\beta_1 = \beta_2 = 0.75$ and $\beta_1 = \beta_2 = 1.0$, producing $\mu_\epsilon$ outside the optimal region, are used in the experiments. Some results obtained from the sets of hyperparameters leading to $\mu_\epsilon$ outside the optimal region are also presented to confirm inferior learning performance.

Figs. 4 and 5 show the results with the *Push* task. Comparative performance of the proposed method to the HER is seen in Fig. 4. It is evident in Fig. 4 that the use of the sampling rate decay strategy significantly increases the convergence speed of the RL algorithm. With the linear decay strategies, corresponding to Fig. 4(a), the best performance is obtained when $\beta_1$ is equal to 0.25. With the exponential decay strategies, corresponding to Fig. 4(b), the results of the proposed method for $\beta_1 = \beta_2 = 0.25$ and 0.5, and 0.75 are close to each other and seem to be improved over the HER. When $\beta_1 = \beta_2 = 1$, training performance of the proposed method is marginally better than that of the HER over the range of epoch index 50–200. For this particular task, the variation of the training performance with different random seeds is seen to be smaller with the sampling rate decay strategy. Fig. 5 shows the comparison between the ARCHER and the proposed method. It is seen in Fig. 5 that the sampling rate decay strategy leads to the performance improved over the ARCHER regardless of values of $\beta_1$ and

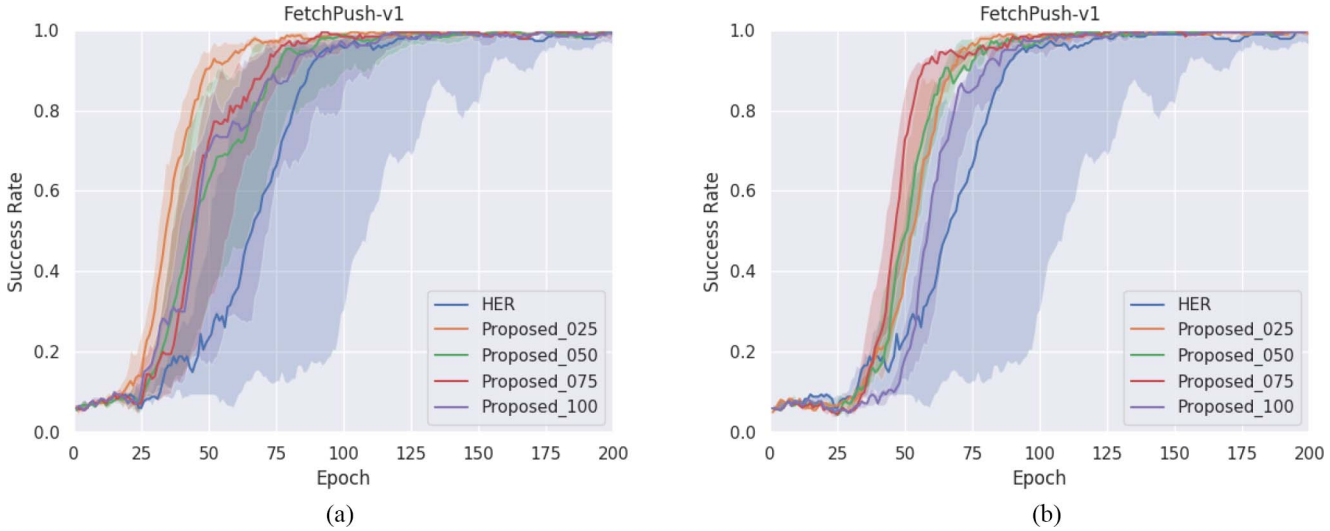


Fig. 4. Learning curves of the *Push* task obtained by the proposed method and HER [16]. Curves represent the variation of the median success rate and shaded areas represent the range of the success rate estimated with five random seeds. The "Proposed_025," "Proposed_050," "Proposed_075," and "Proposed_100" for the linear decay denote $\beta_1 = 0.25, 0.5, 0.75, 1$, and $\beta_2 = 0$, respectively. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the exponential decay denote $\beta_1 = \beta_2 = 0.25, 0.5, 0.75, 1$, respectively. For both the proposed method and HER, $\lambda_h = \lambda_r = 1$. (a) Linear decay. (b) Exponential decay.

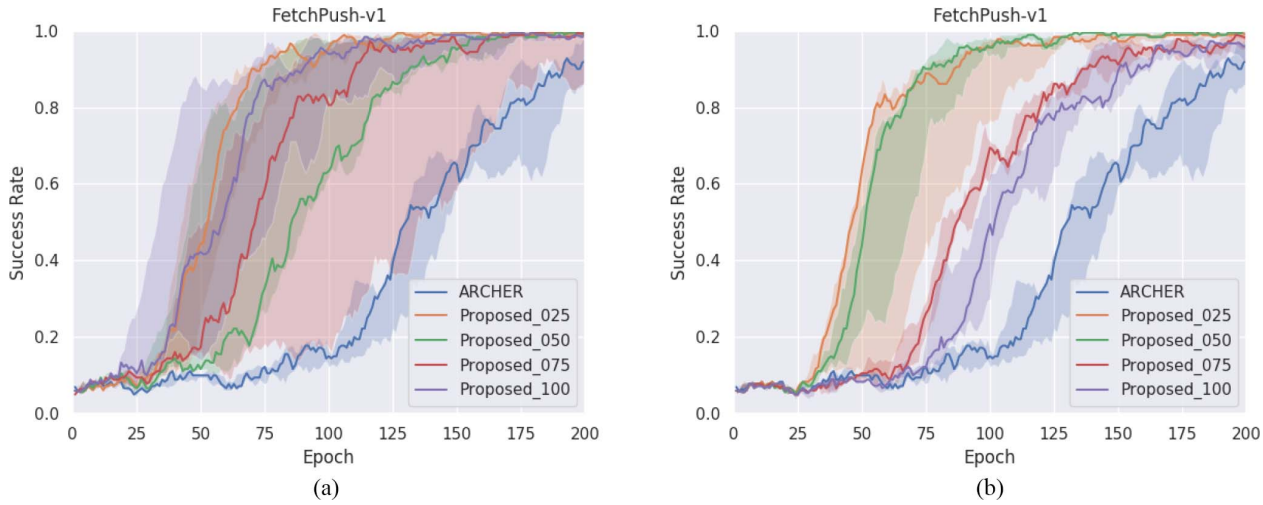


Fig. 5. Learning curves of the *Push* task obtained by proposed method and ARCHER [25]. Curves represent the variation of the median success rate and shaded areas represent the range of the success rate estimated with five random seeds. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the linear decay denote $\beta_1 = 0.25, 0.5, 0.75, 1$, and $\beta_2 = 0$, respectively. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the exponential decay denote $\beta_1 = \beta_2 = 0.25, 0.5, 0.75, 1$, respectively. For both the proposed method and ARCHER, $\lambda_h = 0.5$ and $\lambda_r = 1$. (a) Linear decay. (b) Exponential decay.

$\beta_2$. The best result is obtained in Fig. 5(a) when using a linear decay strategy with $\beta_1 = 0.25$.

Figs. 6 and 7 present the results for the *Pick and Place* task. Fig. 6 shows that the training performance of the proposed method employing the sampling rate decay strategy is improved over the HER with $\beta_1 = 0.25$ and 0.5. With the linear decay strategy for $\beta_1 = 0.75$ and 1 the performance is comparable to the HER. It is seen in Fig. 6 that the proposed method increases the median success rate at the epoch index 200 by approximately 20% with $\beta_1 = 0.25$ and 0.5 for linear decay strategies and for all values of $\beta_1$ and $\beta_2$ for exponential decay strategies. The comparison between the proposed method and ARCHER is seen in Fig. 7. The performance of the proposed method is improved for all the values of $\beta_1$ and $\beta_2$ with exponential decay strategies.

Figs. 8 and 9 present the results for the *Slide* task. Fig. 8 shows the comparison between the proposed method and HER. It is seen that both methods achieve a similar level of performance when a linear decay strategy with $\beta_2 = 0$ is used for the proposed method. The proposed method and the HER achieve a similar level of a median success rate at the end of 200 epochs, with the best median success rate being achieved by the proposed method when $\beta_1 = 0.5$ and 0.75 and $\beta_2 = 0$. The comparison between the proposed method and ARCHER is seen in Fig. 9. Comparable performance of the proposed method is obtained when linear decay strategies are used, while the ARCHER seems marginally better than the proposed method with exponential decay strategies. The median success rate is around 0.5 at the end of 200 epochs, with the


Fig. 6. Learning curves of the *Pick and Place* task obtained by the proposed method and HER [16]. Curves represent the variation of the median success rate and shaded areas represent the range of the success rate estimated with five random seeds. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the linear decay denote $\beta_1 = 0.25, 0.5, 0.75, 1$, and $\beta_2 = 0$, respectively. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the exponential decay denote $\beta_1 = \beta_2 = 0.25, 0.5, 0.75, 1$, respectively. For both the proposed method and HER, $\lambda_h = \lambda_r = 1$. (a) Linear decay. (b) Exponential decay.

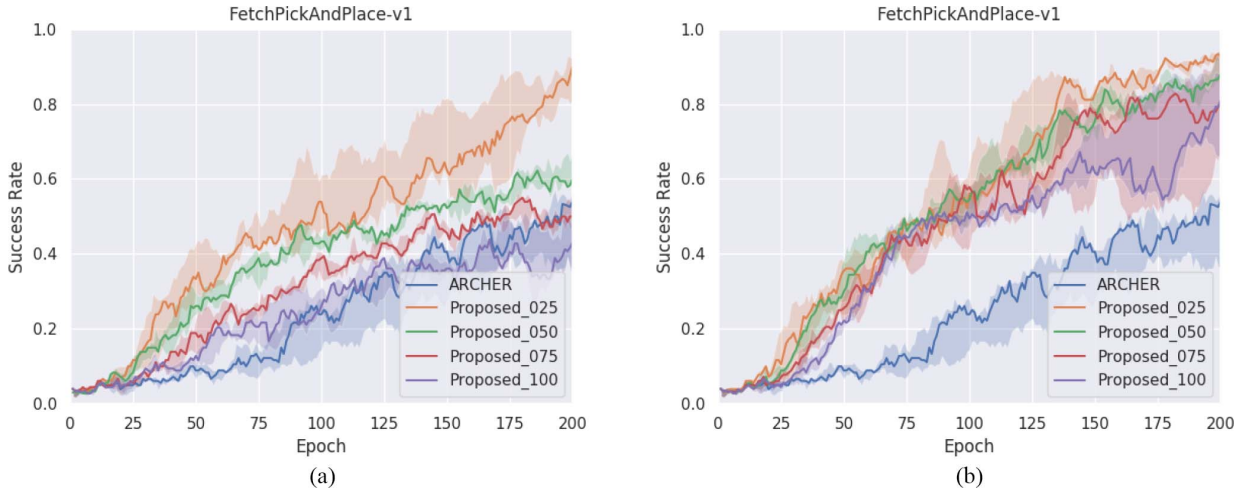


Fig. 7. Learning curves of the *Pick and Place* task obtained by the proposed method and ARCHER [25]. Curves represent the variation of the median success rate and shaded areas represent the range of the success rate estimated with five random seeds. The Proposed_025, Proposed_050, Proposed_075, and "Proposed_100 for the linear decay denote $\beta_1 = 0.25, 0.5, 0.75, 1$, and $\beta_2 = 0$, respectively. The "Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the exponential decay denote $\beta_1 = \beta_2 = 0.25, 0.5, 0.75, 1$, respectively. For both the proposed method and ARCHER, $\lambda_h = 0.5$ and $\lambda_r = 1$. (a) Linear decay. (b) Exponential decay.

best result being obtained when $\beta_1 = 0.5$ and 0.75 and $\beta_2 = 0$.

### C. Variation of the Mean Q-Value During Training

Variation of the mean $Q$-value during training is plotted in comparison with HER. The mean $Q$-value is the average $Q$-value in a training batch. A linear sampling rate decay with $\beta_1 = 0.25$ and $\beta_2 = 0$ and an exponential sampling rate decay with $\beta_1 = \beta_2 = 0.25$ are taken for the three tasks and consequent results averaged over multiple random seeds are shown in Fig. 10. The "linear_025" denotes the linear sampling rate decay with $\beta_1 = 0.25$ and $\beta_2 = 0$ and the "exp_025" represents the exponential sampling rate decay with $\beta_1 = \beta_2 = 0.25$.

It is seen in Fig. 10(a) and (b) that the overestimation of the mean $Q$-value for the *Push* task and *Pick and Place* task at the beginning of training is diminished over time by the proposed method significantly faster than HER. The variation pattern of the mean $Q$-value obtained suggests that the proposed method is more likely to lead to good actions and consequentially higher convergence speed, as compared to HER. It is shown in Fig. 10(c) that the mean $Q$-value for the *Slide* task is not significantly diminished over time, regardless of methods. A similar variation pattern of the mean $Q$-value is observed with other combinations of $\beta_1$ and $\beta_2$ considered for Figs. 4–9.

### D. Combination With ER Prioritization Methods

An important aspect of the proposed method is the possibility of integration with curriculum learning and ER prioritization methods. As an enhancement to HER, the MEP is proposed in [13]. The MEP in [13] utilizes the


Fig. 8. Learning curves of the *Slide* task obtained by the proposed method and HER [16]. Curves represent the variation of the median success rate and shaded areas represent the range of success rate estimated with five random seeds. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the linear decay denote $\beta_1 = 0.25, 0.5, 0.75, 1$, and $\beta_2 = 0$, respectively. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the exponential decay denote $\beta_1 = \beta_2 = 0.25, 0.5, 0.75, 1$, respectively. For both the proposed method and HER, $\lambda_h = \lambda_r = 1$. (a) Linear decay. (b) Exponential decay.


Fig. 9. Learning curves of the *Slide* task obtained by the proposed method and ARCHER [25]. Curves represent the variation of the median success rate and shaded areas represent the range of the success rate estimated with five random seeds. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the linear decay denote $\beta_1 = 0.25, 0.5, 0.75, 1$, and $\beta_2 = 0$, respectively. The Proposed_025, Proposed_050, Proposed_075, and Proposed_100 for the exponential decay denote $\beta_1 = \beta_2 = 0.25, 0.5, 0.75, 1$, respectively. For both the proposed method and ARCHER, $\lambda_h = 0.5$ and $\lambda_r = 1$. (a) Linear decay. (b) Exponential decay.

same set of environments considered to get the results in Figs. 4–9. Fig. 11 shows the success rate of the HER, HER combined with MEP (HER-MEP), proposed method (Proposed), and proposed method combined with MEP (Proposed-MEP).

From the observation of Fig. 11, the HER-MEP is seen to achieve improved performance over HER for *Pick and Place* task. With three tasks, the proposed method or MEP combined with the proposed method demonstrates the highest success rate and convergence speed. As a disadvantage of the proposed method, a time-consuming grid search is needed to find the best values of the hyperparameters $\beta_1$ and $\beta_2$. On the other hand, the MEP simply uses rank-based prioritization to be robust over changes of the hyperparameters. The proposed method is advantageous over the MEP in the sense that the proposed method just indicates the rate of the experience in the minibatch that will be substituted in hindsight while the MEP needs to calculate the prioritized sampling distribution at each epoch using the entire ER. The use of the prioritization mechanism like the MEP can increase the time for training by more than ten times as specified in [13].

### *E. How to Select $\beta_1$ and $\beta_2$*

The experimental results shown in Figs. 4–9 confirm our hypothesis that controlling the rate of hindsight experience during training is effective for improved training performance and increased convergence speed.

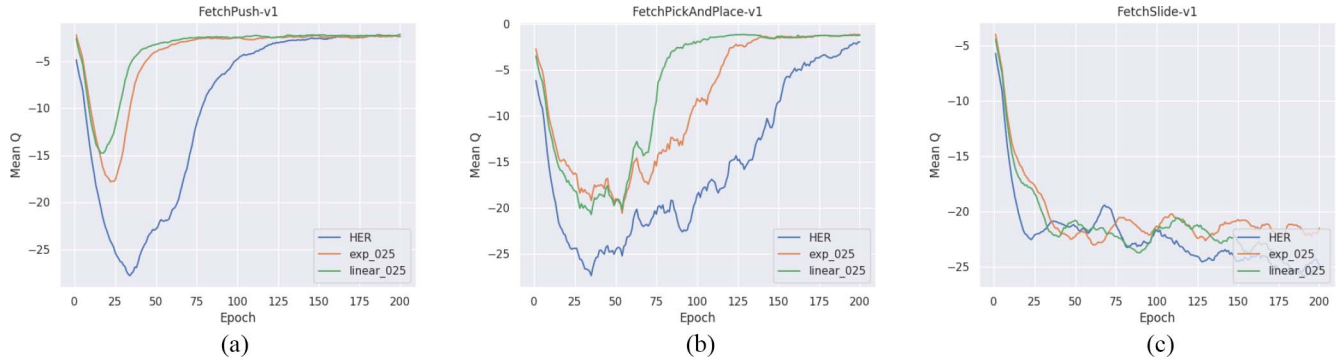


Fig. 10. Variation of the mean $Q$-value during training. The "linear_025" denotes the linear sampling rate decay with $\beta_1 = 0.25$ and $\beta_2 = 0$, and the "exp_025" represents the exponential sampling rate decay with $\beta_1 = \beta_2 = 0.25$. (a) Push. (b) Pick and Place. (c) Slide.

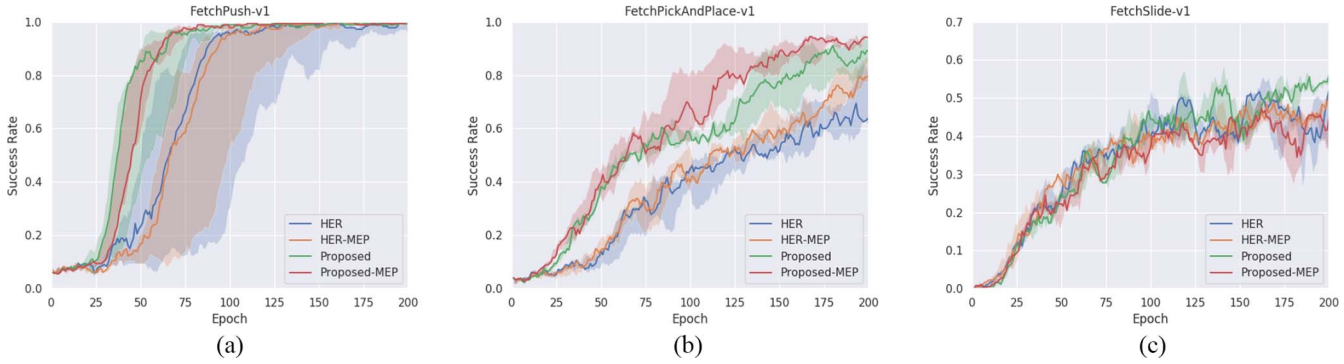


Fig. 11. Comparison between the proposed method and the prioritization method MEP. The "Proposed" denotes the proposed method with $\beta_1 = 0.5$ and $\beta_2 = 0$. The "HER-MEP" denotes the MEP [13] combined with HER. The "Proposed-MEP" denotes MEP combined with the proposed method with $\beta_1 = 0.5$ and $\beta_2 = 0$. (a) Push. (b) Pick and Place. (c) Slide.

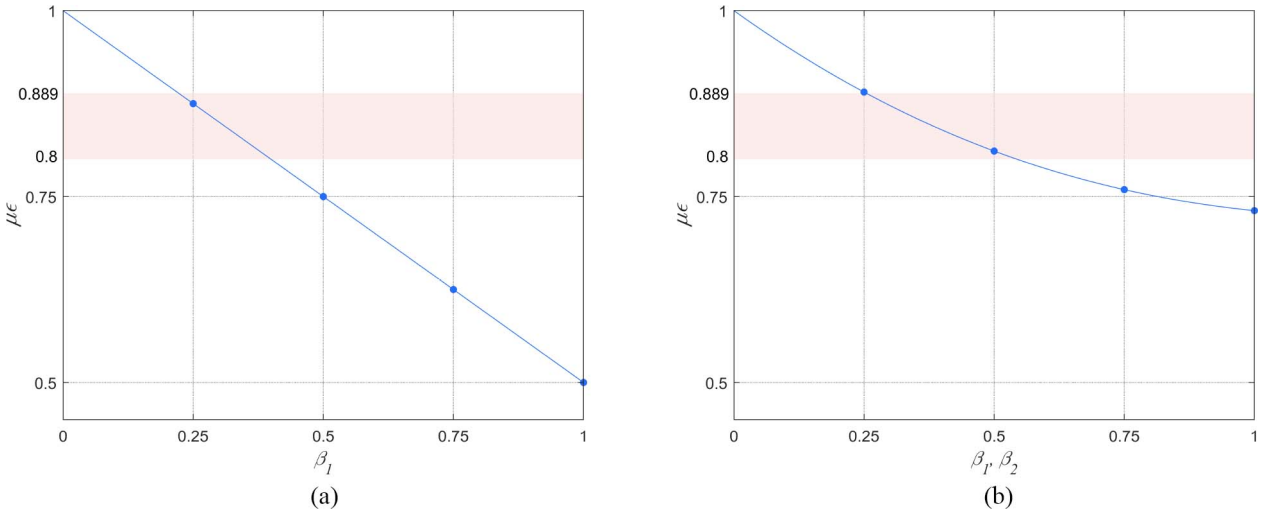


Fig. 12. Comparison between the best values of $\epsilon$ in [16] marked by the shaded area and the best values of $\mu_\epsilon$ obtained from the proposed method. The values of $\mu_\epsilon$, when $\beta_1$ is 0.25 or 0.5 for linear decay strategies and when $\beta_1 = \beta_2 = 0.25$, 0.5 for exponential decay strategy, are located near or inside the shaded area. (a) Linear decay. (b) Exponential decay.

The HER in [16] achieves the best results when the rate of hindsight experience is equal to 0.8 and 0.889, indicating that the majority of training data in the minibatch comes from hindsight experience. Comparison between the expected $\epsilon$, $\mu_\epsilon$, and the best values of $\epsilon$ when using a fixed $\epsilon$ obtained in [16] is shown in Fig. 12. The shaded area corresponds to the region between the two best values of $\epsilon$ in [16], 0.8 and 0.889. The line and curve represent $\mu_\epsilon$ for different values of $\beta_1$ and $\beta_2$. The markers are placed on the values of hyperparameters used in experiments. The best values of the proposed method, when $\beta_1 = 0.25, 0.5$ and $\beta_2 = 0$ for linear decay strategies and when $\beta_1 = \beta_2 = 0.25$, 0.5 for exponential decay strategies, correspond to $\mu_\epsilon$ located near or inside the shaded area. Both methods

highlight the importance of having the prevalence of hindsight experience.

## V. Conclusion

Training agents via deep RL for robotic control tasks with binary sparse rewards is challenging due to the rare presence of successful experiences. In a multigoal environment, the HER increases sampling efficiency by randomly substituting experiences in hindsight. In this article, a novel method on efficient adjustment of the rate of hindsight experience is proposed. The proposed method is validated with three robotic control tasks included in the OpenAI Gym suite. For the *Push* task, decaying the sampling rate significantly increases the convergence speed, outperforming HER and ARCHER. The best performance of the proposed method is obtained when a linear strategy with $\beta_1 = 0.25$ and $\beta_2 = 0$ is used. In the *Pick and Place* task, decaying the sampling rate increases the success rate by 20% over HER and ARCHER achieving the best performance with a linear decay strategy with $\beta_1 = 0.25$ and $\beta_2 = 0$ and an exponential strategy with $\beta_1 = \beta_2 = 0.25$. For the *Slide* task, the proposed method achieves a similar level of performance when compared with HER and ARCHER. It is important to note that one of the most critical findings with the proposed method is that different combinations of $\beta_1$ and $\beta_2$ can be used, regardless of the decay strategy, as long as the combined pairs satisfy the condition of $\mu_\epsilon$ in (11) $0.8 < \mu_\epsilon < 0.889$. Since the optimal decay strategy does not have to follow a deterministic function, it is extremely time consuming to find out the optimal portion of hindsight experience for each epoch index. In this article, a systematic approach based on deterministic functionalities, for example, linear function and exponential function, is presented to show the effect of the time-varying sampling rate to improve learning performance.

## Appendix
## Network Architecture and Hyperparameters

The hyperparameters used for experiments are adopted from [27] describing the experiment environment of the HER. The list of the hyperparameters used for experiments is as follows.

1) *Actor and Critic Networks:* Three layers with 256 units each and ReLU nonlinearities after each layer.
2) ADAM optimizer [30] with learning rate $10^{-3}$ for training both actor and critic networks.
3) *Buffer Size:* $10^6$ experiences.
4) *Polyak-Averaging Coefficient:* 0.95.
5) *Action L2 Norm Coefficient:* 1.0.
6) *Observation Clipping:* $[-200, 200]$.
7) *Batch Size:* 256.
8) *Rollouts per MPI Worker:* 2.
9) *Number of MPI Workers:* 1.
10) *Cycles per Epoch:* 50.
11) *Batches per Cycle:* 40.
12) *Test Rollouts per Epoch:* 10.
13) *Probability of Random Actions:* 0.3.
14) *Scale of Additive Gaussian (Exploration) Noise:* 0.2.
15) *Normalized Clipping:* $[-5, 5]$.

The details of the hyperparameters are described in details in [16] and [27]. An episode is defined as 50 environment timesteps. One environment timestep is consisted of 20 MuJoCo (simulator) steps with $\Delta t = 0.002$ s. The inputs of the neural networks are normalized to have zero mean and unit standard deviation. Normalized clipping is also used.

## References


[1] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[2] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[3] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[4] S. Levine *et al.*, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.
[5] S. Gu *et al.*, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3389–3396.
[6] M. Seo, L. F. Vecchietti, S. Lee, and D. Har, "Rewards prediction-based credit assignment for reinforcement learning with sparse binary rewards," *IEEE Access*, vol. 7, pp. 118776–118791, 2019.
[7] Y. Wen, J. Si, A. Brandt, X. Gao, and H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, early access, Jan. 3, 2019, doi: 10.1109/TCYB.2019.2890974.
[8] L. J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol.8, nos. 3–4, pp. 293–321, 1992.
[9] T. Schaul *et al.*, "Prioritized experience replay," 2015. [Online]. Available: arXiv:1511.05952.
[10] Z. Ni, N. Malla, and X. Zhong, "Prioritizing useful experience replay for heuristic dynamic programming-based learning systems," *IEEE Trans. Cybern.*, vol. 49, no. 11, pp. 3911–3922, Nov. 2019.
[11] S. Zhang and R. Sutton, "A deeper look at experience replay," 2017. [Online]. Available: arXiv:1712.01275.
[12] K. Narasimhan, T. D. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," in *Proc. Conf. Empirical Meth. Nat. Lang. Process.*, 2015, pp. 1–11.
[13] R. Zhao, X. Sun, and V. Tresp, "Maximum entropy-regularized multi-goal reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, 2019, pp. 7553–7562.
[14] T. de Bruin, J. Kober, K. Tuyls, and R. Babuska, "Experience selection in deep reinforcement learning for control," *J. Mach. Learn. Res.*, vol. 19, no. 9, pp. 1–56, 2018.
[15] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 385–398, Mar. 2012.
[16] M. Andrychowicz *et al.*, "Hindsight experience replay," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5055–5065.
[17] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.
[18] Y. Bengio *et al.*, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.
[19] W. Zaremba and I. Sutskever, "Learning to execute," 2014. [Online]. Available: arXiv:1410.4615.
[20] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.
[21] J. Schmidhuber, "PowerPlay: Training an increasingly general problem solver by continually searching for the simples still unsolvable problem," *Front. Psychol.*, vol. 4, p. 313, Jun. 2013. Accessed: Dec. 23, 2019. doi: 10.3389/fpsyg.2013.00313.
[22] Z. Ren, D. Dong, H. Li, and C. Chen, "Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2216–2226, Jun. 2018.
[23] S. Sukhbaatar *et al.*, "Intrinsic motivation and automatic curricula via asymmetric self-play," 2017. [Online]. Available: arXiv:1703.05407.
[24] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1514–1523.

[25] S. Lanka and T. Wu, "ARCHER: Aggressive rewards to counter bias in hindsight experience replay," 2018. [Online]. Available: arXiv:1809.02070.
[26] T. de Bruin *et al.*, "The importance of experience replay database composition in deep reinforcement learning," presented at the Deep Reinforcement Learn. Workshop Conf. Neural Inf. Process. Syst. (NIPS), 2015.
[27] M. Plappert *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," 2018. [Online]. Available: arXiv:1802.09464.
[28] G. Brockman *et al.*, "OpenAI Gym," 2016. [Online]. Available: arXiv:1606.01540.
[29] E. Todorov, T. Erez, and Y. Tassa, "MujoCO: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2012, pp. 5026–5033.
[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: arXiv:1412.6980.

**Luiz Felipe Vecchietti** received the B.Sc. degree in electronics and computer engineering and the M.Sc. degree from the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Cho Chun Shik Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology, Daejeon, South Korea.

His research interests include machine learning, digital signal processing, and applied deep learning.

**Minah Seo** received the B.S. degree in mechanical engineering and logistics administration from Inha University, Incheon, South Korea, in 2017. She is currently pursuing the M.S. degree with the CCS Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology, Daejeon, South Korea.

She is interested in machine learning and data preprocessing.

**Dongsoo Har** (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1986 and 1988, respectively, and the Ph.D. degree in electrical engineering from Polytechnic University, Brooklyn, NY, USA, in 1997.

He is currently a Faculty Member of KAIST, Daejeon, South Korea. He has authored and published more than 100 articles in international journals and conferences. He also presented invited talks and keynote in international conferences. His main research interests include optimization of communication system operation and transportation system development with embedded artificial intelligence.

Dr. Har was a recipient of the Best Paper Award (Jack Neubauer Award) from the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2000. He was the member of the advisory board, a program chair, a vice chair, and a general chair of international conferences. He is an Associate Editor of IEEE SENSORS JOURNAL.