IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

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

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Enhanced Off-Policy Reinforcement Learning with Focused Experience Replay

SEUNG-HYUN KONG[1,2] (Senior Member, IEEE), I MADE ASWIN NAHRENDRA[2], DONG-HEE PAEK[1]

[1]The Cho Chun Shik Graduate School of Green Transportation, KAIST, Daejeon 34141, Rep. of Korea
[2]The Robotics Program, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Rep. of Korea

Seung-Hyun Kong and I Made Aswin Nahrendra are co-first authors.
Corresponding author: Seung-Hyun Kong (e-mail: skong@kaist.ac.kr).

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by Korea government (MSIT) (No.2020-0-00440, Development of Artificial Intelligence Technology that Continuously Improves Itself as the Situation Changes in the Real World)

ABSTRACT Utilizing the collected experience tuples in the replay buffer (RB) is the primary way of exploiting the experiences in the off-policy reinforcement learning (RL) algorithms, and, therefore, the sampling scheme for the experience tuples in the RB can be critical for experience utilization. In this paper, it is found that a widely used sampling scheme in the off-policy RL suffers from inefficiency due to the inadequate uneven sampling of experience tuples from the RB. In fact, the conventional uniform sampling of the experience tuples in the RB causes a severely unbalanced experience utilization, since experiences stored earlier in the RB is sampled with much higher frequency especially in the early stage of learning. We mitigate this fundamental problem by employing a half-normal sampling probability window that allocates a higher sampling probability to newer experiences in the RB. In addition, we propose general and local size adjustment schemes that determine the standard deviation of the half-normal sampling window to enhance the learning speed and performance and to mitigate the temporary performance degradation during training, respectively. For performance demonstration, we apply the proposed sampling technique to the state-of-the-art off-policy RL algorithms and test for various RL benchmark tasks such as MuJoCo gym and CARLA simulator. As a result, the proposed technique shows considerable learning speed and final performance improvement, especially on the tasks with large state and action space. Furthermore, the proposed sampling technique increases the stability of the considered RL algorithms, verified with less variance of the performance results across different random seeds of network initialization.

INDEX TERMS Reinforcement learning, off-policy, actor-critic, experience replay, replay buffer

I. INTRODUCTION

IN the off-policy reinforcement learning (RL), an agent is trained in a two-step process, exploration step and learning step, where the agent collects and stores experiences through interacting with the environment and then updates the agent's policy using the samples of the stored experiences, respectively. In recent years, various off-policy RL algorithms have been successfully applied and have shown significant performance improvements in the challenging tasks from classic Atari games and Go, [1]–[5] to robotics control environments such as MuJoCo [6]–[13] and real-world implementation of robotic control [9]. However, there still exist two conventional challenges in the off-policy RL; exploration of large state space and efficient utilization of the stored experiences [14], [15]. The exploration focuses on how to make an RL agent encounter new and diverse experiences, and the experience utilization addresses on how to make an RL agent obtain the full knowledge by learning from the stored experiences.

The two-step process has been realized with the replay buffer (RB); in the exploration step, the RB is filled in with experience tuples, and the stored experience tuples are sampled and utilized in the learning step. This sequential process is called experience replay (ER) [16], which has been widely used for the off-policy RL algorithms with buffer size large enough to store various experience samples over

a wide enough interval. However, the application of ER for off-policy RL still suffers from sampling inefficiency [14], [17]; while an agent needs to sample useful experience tuples in the RB at all times for optimal policy development, the sampling could be inefficient when the agent uses the conventional sampling techniques, for example, uniform sampling, especially in the early stage of learning when RB is being filled. In fact, applying uniform sampling results in a relatively high sampling frequency of experience tuples stored early in the RB, because the earlier experience tuples will be included in the sampling window a lot more times than the later experience tuples [18], [19]. Even if the earlier experience tuples are discarded when the RB is full and new experience tuples are collected, this severely unbalanced sampling frequency can be a critical problem when the size of RB is large enough as required for many complex RL problems. And, most importantly, the problem is critical as the policy development by the RL agent starts to show a rapid speed-up well before the RB is fully filled.

Recent RL algorithms still need huge number of experiences to develop an optimal policy to solve a real-world problem [6] - [9], [14], [20]. In practice, older experiences often originate from unsuccessful episodes, whereas recent experiences are often from successful episodes and thus have more useful information. For instance, unsuccessful episodes in the initial stage mostly contain environment steps with inappropriate action choices [21] that do not provide useful information. However, an RL agent should not aggressively utilize newer experiences and ignore the older ones, which causes the agent's failure to generalize the knowledge and to experience a catastrophic forgetting [22]. This leads to the stability-plasticity dilemma, in the continual learning domain, about how a learning agent could quickly learn new knowledge while preserving old knowledge [23], [24]. And this explains that RB of a large enough size is necessary for many RL problems.

In this paper, we propose Focused Experience Replay (FER), an experience sampling technique to mitigate the severely unbalanced sampling frequency in the off-policy RL and, thus, to enable faster and stable action policy development. In the proposed technique, we utilize a half-normal sampling probability window, with a linearly increasing standard deviation of the half-normal sampling probability distribution, so that the sampling concentrates more on recent experiences than old ones. In addition, we propose a locally adaptive standard deviation scaling technique to adpatively determine the half-normal sampling probability distribution to prevent the policy from becoming too deterministic prematurely. We demonstrate that the state-of-the-art off-policy RL algorithms, such as Soft Actor-Critic (SAC) [8], [9], Twin Delayed Deep Deterministic (TD3) [7], and Deep Deterministic Policy Gradient (DDPG) [6], are significantly enhanced with the proposed FER in terms of the learning speed (increased up to 2.5 times), which verifies that the proposed FER is a highly efficient RB sampling technique. In addition, the algorithms with the proposed FER show a noticeable improvement on the final learning performance in some complicated MuJoCo locomotion tasks and CARLA simulator for autonomous driving. The main contribution of this work can be summarized as follows.

1) We provide, to the best of our knowledge, the first investigation on the severely unbalanced overall sampling frequency problem in ER for off-policy RL algorithms. We support our findings with theoretical and empirical proofs that testify the unbalanced overall sampling frequency in the conventional sampling techniques for ER.
2) We propose an experience sampling technique that mitigates the unbalanced overall sampling frequency problem in ER. The proposed technique, FER, utilizes a half-normal sampling probability window to focus the experience sampling on newer experiences more to enable evenly distributed overall sampling frequency for all samples.
3) We provide an ablation study to show how the proposed technique can be improved by adjusting the critical hyperparameters. The ablation study consists of a theoretical view of the equations related to the hyperparameters, empirical proof in simulations, and suggestions on future directions for improving the proposed technique.

The organization of this paper is as follows. In Section II, we compare the proposed technique with PER and other sampling techniques in ER for off-policy RL algorithms. In Section III, we provide the problem formulation of general off-policy RL and introduce state-of-the-art off-policy RL algorithms that is used throughout this paper. Section IV provides a detailed description of the proposed technique addressing the unbalanced sampling problem in ER. We demonstrate the performance enhancement when the proposed technique is applied to the state-of-the-art off-policy RL algorithms in comparison to vanilla version and PER-enhanced of those off-policy RL algorithms, in Section V. And finally, we draw a conclusion and provide potential future works of the proposed technique in Section VI.

II. RELATED WORKS

Experience replay (ER) is important for the implementation of the off-policy RL algorithms [23]. The core function of ER includes storing experience tuples, each of which is a set of explored state, action, reward, and the next state, into a large RB and sampling experience tuples from the RB to build a learning data batch.

A breakthrough sampling technique, known as Prioritized Experience Replay (PER) [17], ranks experiences in the RB based on their temporal difference (TD) error that is used to quantify the innovation of the experience compared to the others [14]. Then, PER allocates sampling priority to the experiences with higher TD errors. PER also utilizes weighted importance sampling [25] for annealing the bias induced when experiences with the high TD errors are sampled.

In general, PER improves learning speed and final performance of the off-policy RL algorithms solving problems of discrete states and controls such as Atari games [1], and

shows better performance than the state-of-the-art DQN and DDQN algorithms [3]. In [26], PER is applied to continuous control domain by using PER with deep deterministic policy gradient (DDPG) [6] and shows significant increase in the learning speed on continuous inverted pendulum simulation. However, we observe that, similar to the conventional uniform sampling technique, PER also suffers from severely unbalanced overall sampling frequency for experience tuples in the RB, especially when the RB is being filled in the early stage of learning.

The proposed technique differs from PER in that it is not aggressively learning from experiences with large TD errors, but it is allocating uniform overall sampling frequency to old and new experiences in the long run. Therefore, the proposed technique can generalize the learned knowledge by exploting old knowledge and new knowledge in a balanced way.

Recently, there have been studies in the literature that improve or supplement the performance of off-policy learning algorithms with techniques related to experience replay. Hindsight Experience Replay (HER) [27] aims to learn a policy in environments of sparse rewards by collecting experience tuples for sub-goals that are designated by the developer. Combined Experience Replay (CER) [28] observes that a large RB size can degrade the performance, and prevents the degradation by sampling the most recently stored experience tuple at all times. Distributed Prioritized Experience Replay (DPER) [29] shows an improved performance by employing multi-thread (actors) to collect experience tuples and store in a single common RB that is sampled uniformly to train a single learner. Experience Replay Optimization (ERO) [30] proposes an additional experience tuple sampling network of a huge number of weights (e.g., at least 4×10^9) and shows a slight improvement in some continuous control tasks with deep deterministic policy gradient (DDPG) [6].

The proposed technique, FER, differs from most recent studies in multiple ways; FER is a computationally light but efficient sampling technique that does not require additional network and aims to the environments of continuous rewards. FER could be used with off-policy RL algorithms, for instance, deep deterministic policy gradient (DDPG) [6], twin delayed DDPG (TD3) [7], and soft actor-critic (SAC) [8], [9].

III. OFF-POLICY REINFORCEMENT LEARNING

In this section, we define the notations used throughout the paper. And we introduce off-policy RL algorithms, such as DDPG, SAC, and TD3, which can utilize the proposed technique, FER.

In this paper, the concept of actor-critic is used, where an actor's policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is optimized by maximizing the critic's state-action value $Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$, for state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$ at the time step t. The actor's policy $\pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and critic's state-action value $Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)$ are realized using neural networks with parameters ϕ and θ, respectively. When the policy is stochastic, the output of the policy are Gaussian distribution parameters of the action distribution. These outputs are denoted as $\boldsymbol{\mu}_{\boldsymbol{a},t}, \boldsymbol{\sigma}_{\boldsymbol{a},t}$ for mean and standard deviation, respectively. In addition, $\bar{\phi}$ and $\bar{\theta}$ are used to indicate the parameter of the target networks for the actor and the critic, respectively. And $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ indicates the reward from the environment. The experience tuples, $\{\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1}, \boldsymbol{\mu}_{\boldsymbol{a},t}, \boldsymbol{\sigma}_{\boldsymbol{a},t}\}$ are stored into the experience replay buffer, $\mathcal{D}$.

A. DEEP DETERMINISTIC POLICY GRADIENT

The deep deterministic policy gradient (DDPG) [6] is an extension of deep Q-network (DQN) to the continuous space. The DDPG algorithm is based on actor critic, learning a deterministic policy. The critic network of DDPG is trained using the cost function

$$J_Q(\theta) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - y\right)^2\right], \tag{1}$$

where $(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \mathcal{D}$ indicates that the state and action are sampled from the replay buffer, $\mathcal{D}$. And y is the target state-action value defined as

$$y = r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma Q_\theta(\boldsymbol{s}_t, \tilde{\boldsymbol{a}}_t), \tag{2}$$

with γ is the discount factor for future rewards and $\tilde{\boldsymbol{a}}_t$ is the action at $\boldsymbol{s}_t$ with an exploration noise ϵ_t, formulated as

$$\tilde{\boldsymbol{a}}_t = \pi_{\bar{\phi}}(\boldsymbol{s}_t) + \epsilon_t. \tag{3}$$

The policy is updated using policy gradient defined as

$$\nabla_\phi J_\pi(\phi) = \nabla_{\boldsymbol{a}_t} Q_{\theta_1}(\boldsymbol{s}_t, \boldsymbol{a}_t)|_{\boldsymbol{a}_t=\pi_\phi(\boldsymbol{s}_t)} \nabla_\phi \pi_\phi(\boldsymbol{s}_t), \tag{4}$$

where $\nabla_b A$ is the gradient of A with respect to b.

B. TWIN DELAYED DEEP DETERMINISTIC POLICY GRADIENT

The twin delayed deep deterministic policy gradient (TD3) [7] builds on the deep deterministic policy gradient (DDPG) and added an additional critic, Q_{θ_2} as a twin of the first critic, Q_{θ_1}. In addition, TD3 delays the update of the actor's policy and the critic's state-action value to stabilize the learning. Each critic network is trained using the cost function

$$J_Q(\theta_i) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_{\theta_i}(\boldsymbol{s}_t, \boldsymbol{a}_t) - y\right)^2\right], \tag{5}$$

where i is the index of the critic that is being updated and y is the target state-action value defined as

$$y = r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma \min_{i=1,2} Q_{\bar{\theta}_i}(\boldsymbol{s}_t, \tilde{\boldsymbol{a}}_t). \tag{6}$$

and $\tilde{\boldsymbol{a}}_t$ is the action at $\boldsymbol{s}_t$ with a standard-normal noise ϵ_t, formulated as

$$\tilde{\boldsymbol{a}}_t = \pi_{\bar{\phi}}(\boldsymbol{s}_t) + \epsilon_t. \tag{7}$$

The policy is updated using policy gradient defined as

$$\nabla_\phi J_\pi(\phi) = \nabla_{\boldsymbol{a}_t} Q_{\theta_1}(\boldsymbol{s}_t, \boldsymbol{a}_t)|_{\boldsymbol{a}_t=\pi_\phi(\boldsymbol{s}_t)} \nabla_\phi \pi_\phi(\boldsymbol{s}_t), \tag{8}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access

where $\nabla_b A$ is the gradient of A with respect to b. In the deterministic policy gradient update, only the gradient of Q_{θ_1} is used to update the policy to make the policy learning more stable. When training the Q networks, using the smaller Q-value helps reducing overestimation. However, when training the policy, alternating between Q_{θ_1} and Q_{θ_2} may lead to unstable learning.

C. SOFT ACTOR-CRITIC

The soft actor-critic algorithm [8], [9] addresses the RL problem using entropy maximization and soft Q-function, where the soft Q-function is trained to minimize the soft Bellman residual by using a cost function. The cost function $J_Q(\theta)$ is the expectation of mean squared error between the Q value of a state-action pair sampled from the RB, i.e., $Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)$, and the target Q value

$$J_Q(\theta) = \mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\mathcal{D}}\Bigg[\frac{1}{2}\Bigg(Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t) - \Big(r(\boldsymbol{s}_t,\boldsymbol{a}_t) + \gamma\mathbb{E}_{(\boldsymbol{s}_{t+1})\sim p}\left[V\left(\boldsymbol{s}_{t+1}\right)\right]\Big)\Bigg)^2\Bigg], \tag{9}$$

where $V(.)$ is the soft state-value function, $\mathbb{E}_{a\sim A}[b]$ is the expectation of b by taking argument a following the distribution A. $\mathcal{D}$ denotes the replay buffer which stores experience tuples $\{\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \mu_t, \sigma_t\}$, where $\boldsymbol{\mu}_{\boldsymbol{a},t}$ and $\boldsymbol{\sigma}_{\boldsymbol{a},t}$ are the mean and standard deviation of the the policy distribution given $\boldsymbol{s}_t$. The target Q value is the summation of current reward with the expected soft state-value in the next state $\boldsymbol{s}_{t+1}$ discounted by factor γ, given $\boldsymbol{s}_{t+1}$ sampled from the state transition probability p.

In the initial version of SAC in [8], a separate neural network is used to parameterize the soft state-value function $V(\cdot)$. Later, it is found that the network for $V(\cdot)$ is not necessary [9], and it can be expressed directly with the combination of target state-action value network and policy network as

$$V\left(\boldsymbol{s}_{t+1}\right) = Q_{\bar{\theta}}\left(\boldsymbol{s}_{t+1}\boldsymbol{a}_{t+1}\right) - \alpha\log\left(\pi_\phi\left(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1}\right)\right), \tag{10}$$

where $0 < \alpha < 1$ is the entropy temperature to weight the importance of entropy maximization. The resulting objective function for the optimization of the policy network is defined as

$$J_\pi(\phi) = \mathbb{E}_{\boldsymbol{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\Big[\log\pi_\phi(f_\phi(\epsilon_t;\boldsymbol{s}_t)|\boldsymbol{s}_t) - Q_\theta(\boldsymbol{s}_t, f_\phi(\epsilon_t;\boldsymbol{s}_t))\Big], \tag{11}$$

where $f_\phi(\epsilon_t;\boldsymbol{s}_t)$ is the reparameterization of the action $\boldsymbol{a}_t$ with noise ϵ_t in the state $\boldsymbol{s}_t$, and the noise ϵ_t is sampled from the standard normal distribution $\mathcal{N}$.

IV. FOCUSED EXPERIENCE REPLAY

A. OVERALL SAMPLING FREQUENCY OF CONVENTIONAL TECHNIQUES

When an RL agent interacts with the environment, a new experience is collected and stored into the RB, and some of the experience tuples in the RB are sampled. The uniform sampling technique samples the experience tuples with uniform probability as shown in Figure 1.

Let N_{max} be the size of an RB that is currently filled with N experiences. And let the batch size for the uniform sampling be b. The probability of selecting an arbitrary experience tuple x_m at index m is

$$P(x_m) = \frac{b}{N}. \tag{12}$$

Since each experience at any index m will be sampled with the same probability in uniform sampling, the overall accumulated sampling frequency $f(x)$ for experience tuple x_m at sampling instance up to N can be found as

$$\begin{aligned} f(x_m) &= b\cdot\left(\frac{1}{m} + \frac{1}{m+1} + ... + \frac{1}{N-1} + \frac{1}{N}\right) \\ &\approx b\cdot\int_m^N \frac{1}{r}dr \\ &= b\log\frac{N}{m}. \end{aligned} \tag{13}$$

Equation (13) shows that the experience at the smaller index m will have much larger overall sampling frequency. We verify this mathematical expression from a preliminary simulation and show the overall sampling frequency of each experience as depicted in Figure 2, where the older experience is located at the smaller index of the buffer. The histogram clearly shows that old experiences are selected more frequently and the sampling frequency for the newer experiences is decreasing almost exponentially, following equation (13). The severely unbalanced overall sampling frequency distribution could make the learning focuses too much on old experiences, which results in a difficulty in learning new knowledge from the newer experiences. We also evaluate the overall sampling frequency of PER, and the result is also shown in Figure 2. The result shows that, even though PER assign a higher sampling probability for experiences with larger TD error, the overall sampling frequency is still severely unbalanced. Therefore, PER still suffers from sampling inefficiency problem, similar to the vanilla off-policy RL algorithm using uniform sampling probability.

B. HALF-NORMAL SAMPLING PROBABILITY WINDOW

To mitigate the severely unbalanced overall sampling frequency problem with the conventional sampling techniques, we propose a half-normal sampling probability window that has focused experience replay (FER) sampling on recent experience tuples in the RB. A half-normal sampling probability distribution is a folded Gaussian distribution as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access

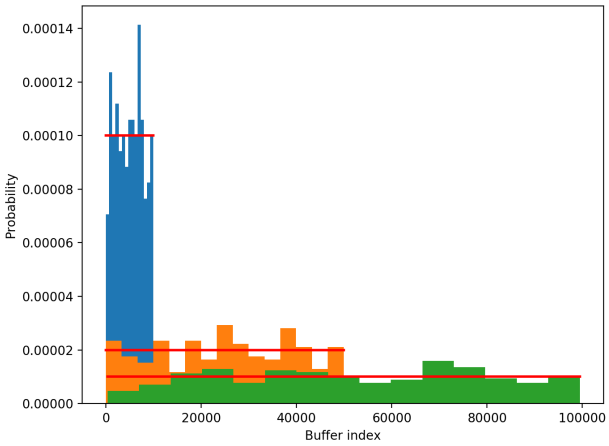


FIGURE 1: Three (blue, orange, and green) histograms as results of uniform sampling when RB is filled with 10000, 50000, and 100000 experience tuples, respectively. It can be found that the accumulation of the three histograms and other possible histograms at different buffer indices will result in an overall exponential sample distribution.

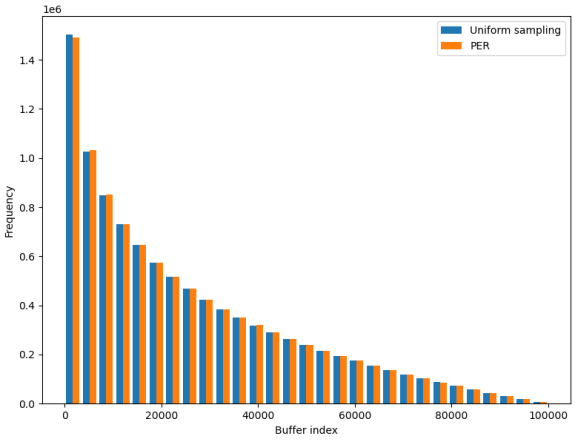


FIGURE 2: Overall (accumulated) sampling frequency for different sampling schemes and $N = 10^5$.

$$P(x_m) = \frac{2}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{m-\mu}{\sigma}\right)^2} = \frac{2n}{N\sqrt{2\pi}} \cdot e^{-\frac{n^2}{2}\left(\frac{m-N}{N}\right)^2}, \quad (14)$$

where μ and σ are the mean and standard deviation, respectively, and $0 \leq m \leq N$. In the implementation, $\mu = N$ and $\sigma = N/n$ are used, where $n(> 1)$ is a positive real number. Therefore, the probability of selecting a newer experience $\underset{m\to N}{P}(x_m)$ is the probability when m is close to N in (14) as

$$\underset{m\to N}{P}(x_m) = \frac{2n}{N\sqrt{2\pi}}. \quad (15)$$

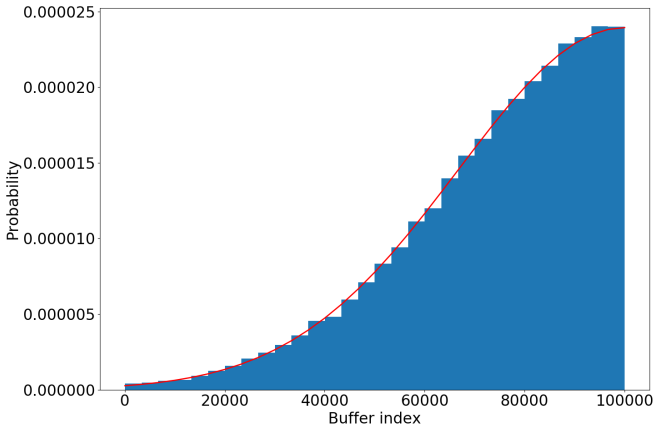


FIGURE 3: Half-normal probability sampling window

And assuming that m is close to 0 in (14), the probability of selecting an old experience $\underset{m\to 0}{P}(x_m)$ becomes

$$\underset{m\to 0}{P}(x_m) = \frac{2n}{N\sqrt{2\pi}} \cdot e^{-\frac{n^2}{2}}. \quad (16)$$

For N large enough, we can let n be a positive constant and it becomes that $\underset{m\to 0}{P}(x_m) \ll \underset{m\to N}{P}(x_m)$, which results in a higher probability for a newer experience (i.e., $m \to N$) to be selected. On the other hand, older experience (i.e., $m \to 0$) has given chances to be sampled from the beginning when N was small. Therefore, the old experience's overall sampling frequency will not dominate too much as what happened when using uniform sampling. And at the same time, it is not losing generalization as well, since the older experience still has enough probability to be sampled. Figure 3 illustrates an example of the half-normal sampling probability window when applied to a monotonously increasing number of experience tuples in the RB, where the current number of experiences in the RB is 10^5. The half-normal sampling probability window is a folded version of a normal distribution, $\mathcal{N}(\mu, \sigma)$, to the lower half, where the mean μ is set to the current size of the RB N, and the standard deviation σ is linearly increasing with respect to N with a factor β_1 $(0 < \beta_1 < 1)$.

1) General sizing factor for σ.

In this paper, we introduce a general sizing factor β_1 $(0 < \beta_1 < 1)$ for the standard deviation, such that $\sigma = \beta_1 N$, which is to scale σ linearly with N. The quantity β_1 is a hyperparameter that needs to be tuned in advance to achieve a good learning performance.

2) Local adjustment factor for σ.

In addition to the linearly changing σ, we use a local adjustment factor to enhance the performance of FER by adding an offset value β_2, so that $\sigma = \beta_1 N + \beta_2$. The factor β_2 is designed such that it increases when the agent's policy starts converging. We measure the policy convergence using KL-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access

divergence (D_{KL}), which measures the difference between a probability distribution compared to another probability distribution. When the policy converges, the old and the current policy should have a similar distribution and have a small KL-divergence $D_{\text{KL}}\left(\pi_{\text{current}}(\cdot|\boldsymbol{s}_t), \pi_{\text{old}}(\cdot|\boldsymbol{s}_t)\right)$. The locally adjustment factor β_2 is defined as

$$\beta_2 = \beta_1 N \left[\frac{D_{\text{KL,mid}}}{D_{\text{KL}}\left(\pi_{\text{current}}(\cdot|\boldsymbol{s}_t) \| \pi_{\text{old}}(\cdot|\boldsymbol{s}_t)\right)} - 1 \right], \quad (17)$$

where $D_{\text{KL}}(\pi_{\text{current}}(\cdot|\boldsymbol{s}_t) \| \pi_{\text{old}}(\cdot|\boldsymbol{s}_t))$ measures how similar the current policy $\pi_{\text{current}}(\cdot|\boldsymbol{s}_t)$ with the policy of the sampled experience $\pi_{\text{old}}(\cdot|\boldsymbol{s}_t)$ with a given $\boldsymbol{s}_t$ and $D_{\text{KL,mid}}$ is a hyperparameter that indicates the median value of D_{KL} between the current and old policies from the batch of experiences. When the current policy is perfectly similar with the old one, no local adjustment is applied, i.e., $\beta_2 = 0$.

Note that, equations (15) and (16) prove that utilizing the half-normal sampling probability at every sampling instance can result in a more uniform overall experience sampling frequency when σ is set to a smaller value. For example, for $\sigma = 0$ the sampling window will only allow the latest experience to be sampled at each sampling instance and, therefore, the overall sampling frequency will be perfectly uniform to all experiences. On the contrary, when σ is set to a very large value, the instantaneous sampling probability window becomes flat, i.e., more like the conventional uniform sampling, and the resulting overall sampling frequency will be similar to that in Figure 2.

The overall sampling frequency distributions for different choice of σ agree with the mathematical derivation as in equation (14), and the simulation results in Figure 4 show that smaller σ leads to more uniform-like distribution of the overall experience sampling frequency. Figure 4a shows that the histogram of overall experience sampling frequency is severely unbalanced for $\sigma = N/2$, whereas, as shown in Figure 4b and 4c, decreasing the standard deviation smaller than $N/2$ results in a more balanced (i.e., uniform) overall experience sampling frequency distribution. However, it should be noted that a very small σ leads to a learning based on the newer experiences only, which results in a poor learning performance, due to the loss of generalization.

C. ENCOURAGING DIFFERENT FUTURE POLICY

The proposed FER mitigates the severely unbalanced overall sampling frequency problem by learning more from newer experiences. However, focusing too much on the newer experiences can cause the agent's performance to converge prematurely. This is because of the deceptive local optima in the newer experiences. Therefore, when the agent falls into local optima, a more diverse experiences are needed for the agent to learn and escape from that local optima. To handle this problem, n in (14) should not be too large, and the agent should be trained to explore more by maximizing a distance between the current and old policies. A distance measurement, i.e., KL-Divergence is added to the policy loss function [31]. For instance, the policy loss function of SAC [8], [9] with policy distance maximization is defined as

$$J_\pi(\phi) = \mathbb{E}_{\boldsymbol{s}_{t\sim\mathcal{D}}, \epsilon_{t\sim\mathcal{N}}} \Big[\log \pi_\phi(f_\phi(\epsilon_t; \boldsymbol{s}_t)|\boldsymbol{s}_t) - Q_\theta(\boldsymbol{s}_t, f_\phi(\epsilon_t; \boldsymbol{s}_t)) - D_{\text{KL}}(\pi_{\text{current}}(\cdot|\boldsymbol{s}_t) \| \pi_{\text{old}}(\cdot|\boldsymbol{s}_t)) \Big]. \quad (18)$$

In practice, smaller KL-Divergence indicates that the two policies are very similar, and *vice versa*. The cost function $J_\pi(\phi)$ is minimized to train the policy, therefore, the KL-divergence term has to be given a negative sign to maximize the distance between the new policy and the old policy sampled from the replay memory.

D. ALGORITHM IMPLEMENTATION

The proposed technique is mainly used with the SACv2 algorithm. It can also be implemented on DDPG and TD3, without utilizing D_{KL} calculation, since the D_{KL} value for the Gaussian noise of TD3 will have the same value for all old and current policies. The combined algorithm for the proposed technique applied to SACv2 [9] is described in Algorithm 1.

Algorithm 1 SACv2 with FER

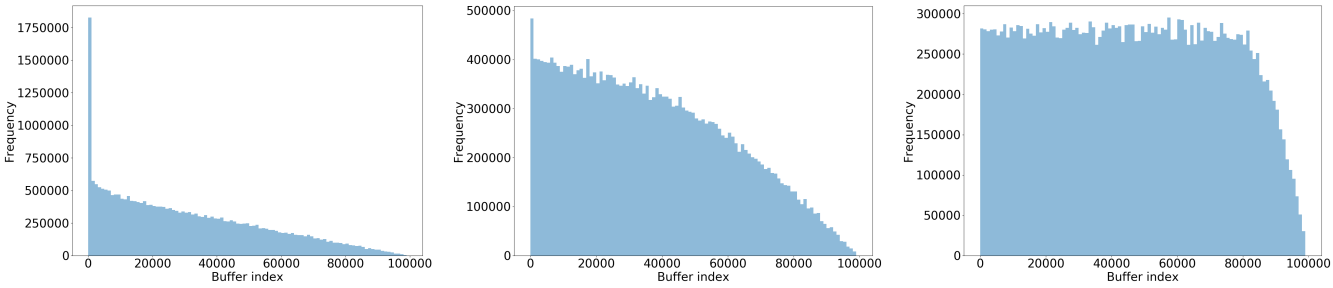
Initialize network parameters θ_1, θ_2 for critic networks Q_{θ_1},Q_{θ_2} and ϕ for policy network π_ϕ
Initialize learning rate values for the critic networks λ_Q, policy network λ_π, and entropy temperature λ_α
Initialize the soft-update constant τ
Initialize target critic networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2$
Initialize replay memory $\mathcal{D}$
Initialize FER window size σ
for each environment step **do**
  $\boldsymbol{a}_t \sim \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)$
  $\boldsymbol{s}_{t+1} \sim p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1}, \boldsymbol{\mu}_{\boldsymbol{a},t}, \boldsymbol{\sigma}_{\boldsymbol{a},t}\}$
end for
for each gradient steps **do**
  Sample batches of experiences using the FER window
  $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in 1, 2$
  $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\pi J_\pi(\phi)$
  $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
  $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in 1, 2$
  $\sigma \leftarrow \sigma = \beta_1 N + \beta_2$
end for

V. EXPERIMENTS

A. EXPERIMENT SETTINGS

The purpose of our experiment is to evaluate the practical effect of our proposed experience replay technique, FER, on the off-policy reinforcement learning algorithms. We show the learning performance improvements when FER is applied to the state-of-the-art algorithms such as SACv2, TD3, and DDPG for solving continuous control tasks in MuJoCo Gym

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access



(a) $\sigma = N/2$ **(b)** $\sigma = N/3$ **(c)** $\sigma = N/10$

FIGURE 4: Overall (accumulated) sampling frequency using the half-normal sampling probability. New experiences are located at large buffer indices.

[10], [13]. Furthermore, we compare our technique FER with the conventional and widely used sampling technique PER in the off-policy RL algorithms. In addition, we include a discussion on the result shown in ERO [30]. For a fair comparison with the prior works, we used the similar architectures, hyperparameters, activation functions, and optimizer as in [9], [7], [6] for the SACv2, TD3, and DDPG, respectively. We include the detailed experiment settings in Appendix A.

The MuJoCo [13] environment shown in Figure 5 has continuous control task simulations, where the goal of each task is to control every joint of a robot and make the robot move in a way to maximize the total accumulated reward. In the MuJoCo environments, we conduct simulations for Hopper, Walker, HalfCheetah, and Ant by using TD3 and SACv2 as a choice of base algorithms. However, we only simulate Humanoid using SACv2, since TD3 fails to make a learning progress in the Humanoid task, based on the results in [8], [9]. Therefore, there is no merit of applying FER to TD3 for the Humanoid task. And additionally, we run simulations for Hopper and Walker using DDPG as the third base algorithm and show the results of DDPG with and without the proposed technique, which is to improve the clarity of the simulation results shown in this section. Moreover, we also conduct experiments for RL-based autonomous vehicle control on CARLA simulator [32], shown in Figure 6, with a gym wrapper used in [33].

B. EVALUATION ON THE MUJOCO GYM ENVIRONMENT

Figure 7 and 8 show the average return of multiple evaluations during training of SAC, TD3, and DDPG for various ER sampling techniques such as the proposed FER, PER, and the conventional uniform sampling. We perform simulations for the vanilla SAC, TD3, and DDPG (with uniform sampling) to provide baseline performance, followed by the simulations with PER and FER. For the experiments, we train five different agents with different random seeds for the network initialization and ten evaluations are conducted every 1000 (training) environment steps. The solid curves represent the average of 10 evaluations from 5 different training instances, while the shaded regions indicate the 1-standard deviation. For this experiment, we choose $\beta_1 = 1/3$ and $D_{\text{KL,max}} = 5$.

First of all, we observe, in Figure 7, that there are often enhancements in the learning speed and/or in the final performance when the proposed FER is applied. The results show that SAC-FER consistently outperforms SAC both in terms of learning speed and the final performance. We also confirm that SAC-FER outperforms SAC-PER for all simulated environments in terms of learning speed, which is because the proposed FER improves the severely unbalanced sampling frequency in the early stage of learning. For the final performance against SAC-PER, SAC-FER excels in Hopper, HalfCheetah, and Ant by a noticeable distinction, but performs slightly less for the Walker and Humanoid environments. The slight deficiency in the final performance of the total return for the Walker and Humanoid environments is caused by the brittleness of FER to the size of the sampling window, on which we give a further discussion on the effect of the sampling window size of FER in Section V-D.

On the other hand for TD3, the results in Figure 8a-8d show that TD3-FER consistently outperforms the learning speed for all of the MuJoCo tasks. However, we notice a deteriorated result in the final performance of TD3-FER for the Ant, as shown in Figure 8d. It is observed that the TD3-FER performance significantly dropped after around 1 million environment steps, which is caused by the insufficient exploration of the TD3 algorithm; as observed from the simulation results, TD3-FER reaches the maximum performance of vanilla TD3 before 1 million environment steps, whereas the vanilla TD3 and TD3-PER reach the maximum performance around 2.5 and 3.0 million environment steps, respectively. This result indicates that the TD3-FER has a remarkable learning speed improvement over the vanilla TD3 by 2.5 times.

For DDPG algorithm, we only conduct the experiments using Hopper-v2 and Walker2d-v2 tasks as shown in Figure 8e-8f, since the purpose of this experiment is to show the ability of FER to enhance the DDPG algorithm. Therefore, we choose simple tasks, where DDPG works well in general, and it is difficult to learn complex tasks such as HalfCheetah-v2 and Ant-v2 using DDPG. We observe the benefits of applying FER to the DDPG algorithm as shown. The vanilla DDPG, DDPG -PER, and DDPG-FER performs similarly for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access

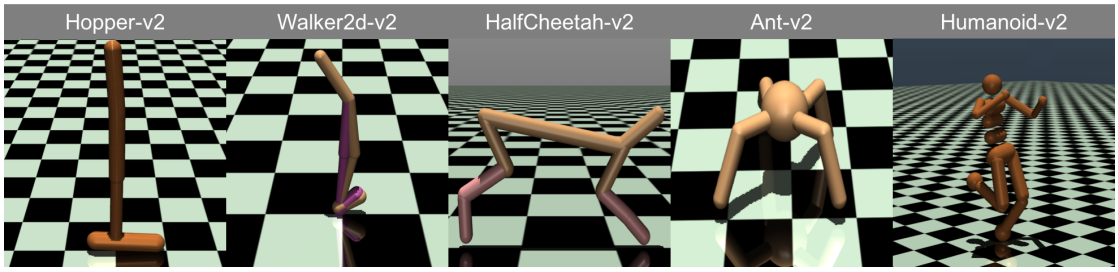


FIGURE 5: MuJoCo Gym environment [13] locomotion tasks used for the experiments

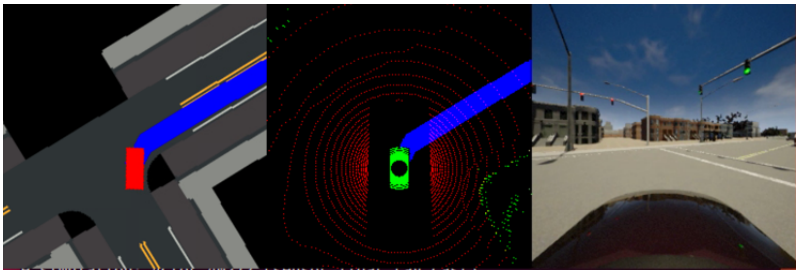
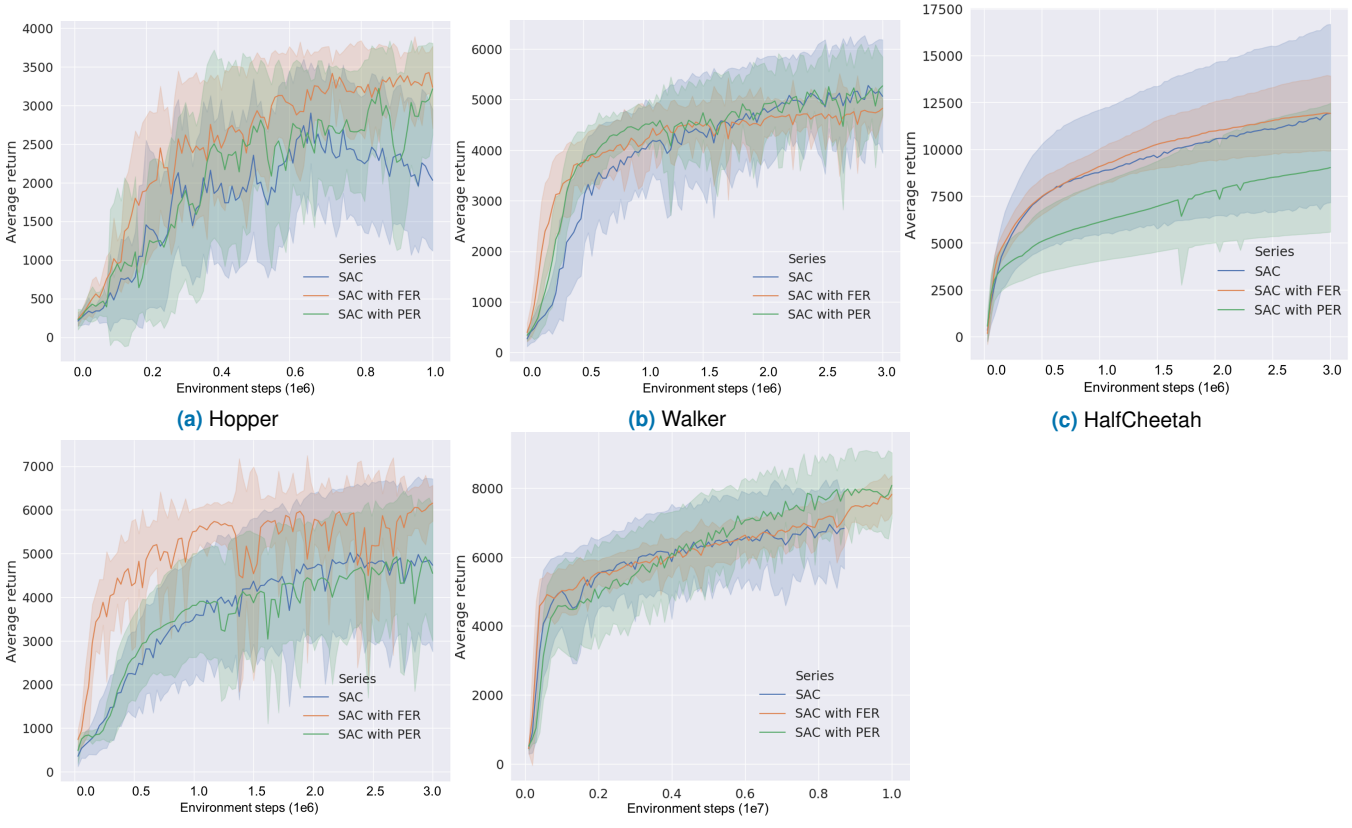


FIGURE 6: CARLA Gym environment using CARLA simulator and OpenAI Gym wrapper provided by and used in [33], vehicle states are utilized as the input for the policy

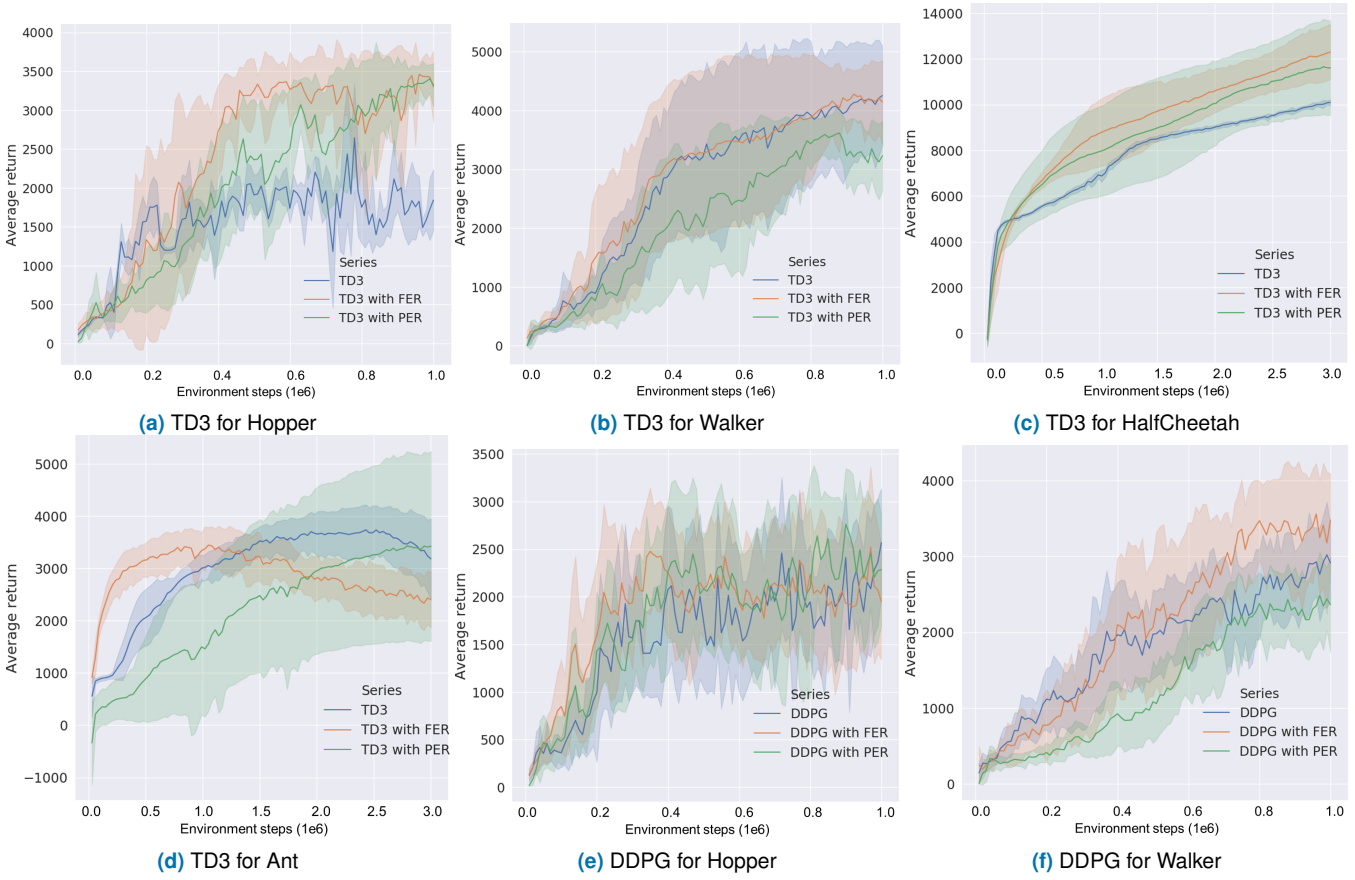


(a) Hopper **(b)** Walker **(c)** HalfCheetah

(d) Ant **(e)** Humanoid

FIGURE 7: Training curves on MuJoCo simulation benchmarks using SAC algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access



(a) TD3 for Hopper **(b)** TD3 for Walker **(c)** TD3 for HalfCheetah

(d) TD3 for Ant **(e)** DDPG for Hopper **(f)** DDPG for Walker

FIGURE 8: Training curves on MuJoCo simulation benchmarks using TD3 and DDPG algorithms.

the Hopper-v2 task, since the the training has a big variance. Therefore, it is difficult to justify the winner for this task. However, the observation is clear with the Walker2d-v2 task, where DDPG-FER overcomes the vanilla DDPG and DDPG-FER with a remarkable results. In fact, DDPG-FER shows an improved learning speed for the Hopper-v2 task. These additional results prove further the ability of FER to enhance performance of an off-policy RL algorithm.

From the results of state-of-the-art off-policy RL algorithms such as SAC, TD3, and DDPG on various Mujoco tasks, we observe that the major benefit of applying FER for ER is either the strong improvement in the learning speed at the early training stage or final performance improvement. FER mostly achieves faster learning speed, comparing to the vanilla and PER-applied off-policy RL algorithms. We summarize the experiment results for sampling techniques and off-policy RL algorithms in Table 1. The performance of ERO is not compared nor shown in the table, since ERO requires huge amount of computing resource (e.g., 4×10^9 weights) for each experience sampling network. For example, while other studies such as [6] achieve a final average return of about 3300 using a batch size of 100, ERO achieves a final average return of about 1750 using a batch size of 64 for Hopper task in its original study [30], which is because the overall computation increases with the batch size in general. Therefore, in Table 1, we compare FER to PER and uniform sampling technique. Note also that in many cases, FER also improves the final performance of the base algorithm and overcomes the final performance of the vanilla and PER-applied algorithms. This is because the FER-applied algorithms have a more uniform distribution of the overall experience sampling frequency as shown in Figure 4, whereas the vanilla algorithms have a severely unbalanced overall experience sampling frequency as shown in Figure 2. Lastly, it should be noted that the exploration ability still depends on the vanilla algorithm, but the experience utilization ability is enhanced by the ER sampling technique such as the proposed FER. When sufficient exploration is not available from the vanilla algorithm such as TD3 shown in Figure 8a-8d, FER can still improve the learning speed, but not the final performance.

C. EVALUATION ON THE CARLA GYM ENVIRONMENT

In the CARLA simulator, the input to the RL agent is the ego vehicle's states such as heading and offset from the center of the lane and the goal for the task is to learn how to drive, i.e., how to follow the lane by controlling the acceleration and steering angle of the ego-vehicle.

To further demonstrate the improvement with the proposed FER in complicated tasks, we conduct experiments on the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access

TABLE 1: Maximum average return over 10 different trials on the last (training) environment steps. The orange colored values indicate the winning technique on the tested environment in terms of the mean value and standard deviation and the teal colored values are for the second best performance within 10% under the best record.

Environment	DDPG-FER	DDPG-PER	DDPG
Hopper-v2	2004.32±734.35	2322.35±1137.11	2307.41±964.16
Walker2d-v2	3351.89±756.33	2431.28±639.23	2981.31±661.59
Environment	**TD3-FER**	**TD3-PER**	**TD3**
Hopper-v2	3423.82±854.25	3319.54±327.22	2307.41 ± 964.16
Walker2d-v2	4610.36±646.45	4545.88±729.07	4537.47±661.59
HalfCheetah-v2	12379.75±1157.66	11834.28±2431.5	10086.00 ± 2239.99
Ant-v2	2286.91±1148.82	3499.59±1849.74	3535.57±1194.68
Environment	**SAC-FER**	**SAC-PER**	**SAC**
Hopper-v2	3349.75±602.68	3245.26±702.29	2100.67 ± 1175.73
Walker2d-v2	4754.63 ± 527.11	5430.44±229.2	4775.61 ± 901.32
HalfCheetah-v2	11926.61±2031.61	8978.87 ± 3419.21	11997.41±5343.63.12
Ant-v2	6310.1±421.75	4889.96 ± 1353.32	4243 ± 2818.10.78
Humanoid-v2	7973.82± 466.31	7420.95±2861.25	6953.37 ± 1203.83
CARLA	1105.71± 148.93	473.32 ± 351.77	403.72 ± 213.17

CARLA, an autonomous vehicle simulator, with an OpenAI Gym wrapper from [33]. The task is to learn how to drive along a lane, with ego-vehicle states as an input and the acceleration and steering angle for the vehicle as an action output. The vehicle states consist of heading error, lateral error relative to the center of the lane, vehicle speed, and indication whether there is an obstacle within some radius. The main difference between the CARLA gym and MuJoCo gym in terms of complexity is in the radomization of the initial state. In the MuJoCo gym locomotion tasks, initial states are randomized around a particular point, e.g., the Humanoid robot will always start at the standing-up state with a slight difference in the joint state values. However, the CARLA gym simulator starts in a variety of initial states, as it can starts at the middle of a straight lane, roundabout, or near an intersection. In the CARLA environments, we could only apply FER to the SAC algorithm, because SAC is more capable of solving complex tasks, where wider exploration is possible by utilizing stochastic policy. This is different than DDPG and TD3 that use deterministic policies and are not capable of solving complex tasks such as CARLA. The last row of Table 1 shows the final performance for the CARLA, and Figure 9 shows the experiment results for the vanilla SAC with uniform sampling, SAC-PER, and SAC-FER. Evaluations are conducted every 1000 environment steps, where the solid curves represent the average from 10 different evaluations and the shaded region depicts the 1-standard deviation of the evaluation results. For this experiment, we choose $\beta_1 = 1/3$ and $D_{\text{KL,max}} = 5$.

The experiment result in Figure 9 shows that SAC-FER outperforms the vanilla SAC and SAC with PER, in both the learning speed and the final performance. SAC-FER achieves more than 1.5 and 2.3 times higher average return than SAC-PER and vanilla SAC, respectively, after 80k environment steps with a relatively small variance over 10 evaluations. This result shows similar observation with the results in MuJoCo locomotion tasks, empirically proving that the off-policy RL algorithm can be substantially enhanced by incorporating FER as an ER sampling technique.

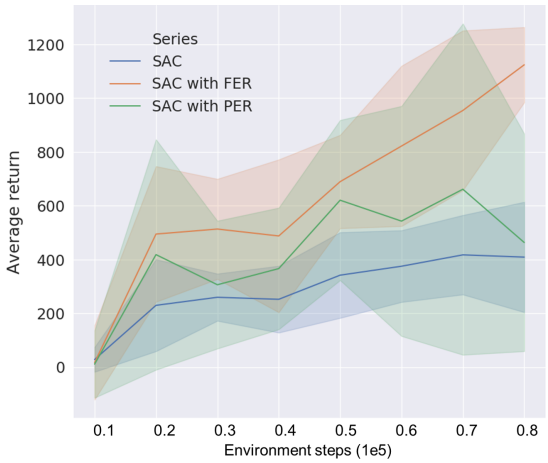


FIGURE 9: Learning curves for CARLA simulation using SAC algorithm

D. ABLATION STUDIES

In this subsection, we examine how the performance of the off-policy RL algorithms with FER affected by different hyperparameter settings, i.e., the general sizing factor β_1 and maximum distance between policies $D_{\text{KL,max}}$ defined in Section 3.2. Our ablation studies are based on the performance evaluation through hyperparameter fine-tuning over multiple simulations as optimum hyperparameters should be determined to achieve the best performance [34], [35].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3085142, IEEE Access

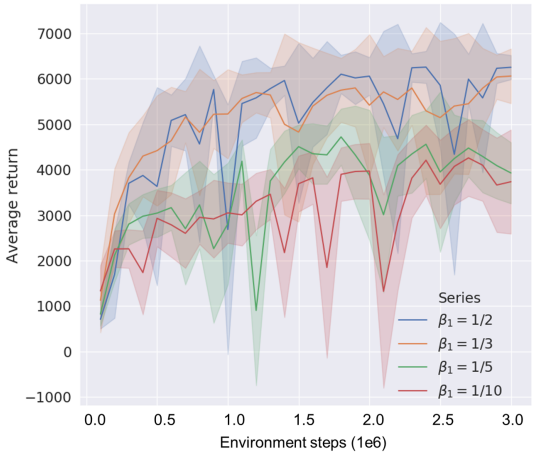


FIGURE 10: Learning curves for Ant-v2 environment using SAC-FER algorithm with different β_1 values

FIGURE 11: Learning curves for Ant-v2 environment using SAC-FER algorithm with different c $= D_{\text{KL,max}}$ values

1) General sizing factor of the sampling window.

The quantity β_1 is important in FER, since it determines the sampling probability distribution at every sampling instance. In Figure 10, we show the performance of SAC-FER for the Ant environment with different β_1 values. When β_1 is set to a small value, for example, $1/5$ and $1/10$, the σ of the half-normal sampling probability becomes smaller and, thus, relatively much higher sampling probabilities for newer experiences. Therefore, the agent is not sampling enough old experiences that are required for generalization and results in deterioration of the training performance. On the other hand, using a big β_1 results in a larger σ, which produces a uniform-like sampling probability causing an unbalanced overall sampling frequency similar to the graph in Figure 4a. From the results shown in Figure 10, setting $\beta_1 = 1/2$ results in a slower learning speed, comparing to $\beta_1 = 1/3$,

As observed from the results in Figure 7, 8, and 9, $\beta_1 = 1/3$ that has an overall sampling frequency as in Figure 4b produces good training performance for multiple MuJoCo tasks. However, the optimal value of β_1 could be slightly different for different environments, so it is useful to tune β_1 properly for different environments.

2) Maximum distance between policies.

Distance maximization between old and new policies is applied to overcome the premature policy convergence when training using more samples of newer experiences than the samples of old ones in the RB, as discussed in Section IV-C. In Figure 11, we show that increasing the maximum allowable policy distance, $D_{\text{KL,max}}$, results in unstable training for SAC-FER in the Ant environment. This is because, the current policy, π_{current}, is trained to maximize the distance from the previous policies, in other words, it becomes a very different policy at every training step. In addition, the final performance also decreases as a result of the unstable policy development. The selected $D_{\text{KL,max}}$ should be higher within a reasonable magnitude than the average $D_{\text{KL,max}}$ between old policies, π_{old}. In the simulated MuJoCo tasks, the D_{KL} values between the old policies when the agent is developing are around 1, i.e., $D_{\text{KL}} \approx 1$. Therefore, we choose $D_{\text{KL,max}} = 5$ in our experiments in Section V-B, which performs well for multiple MuJoCo tasks. This $D_{\text{KL,max}}$ value is also treated as a hyperparameter that needs a fine-tuning and could be slightly different between environments.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have shown theoretical proof of unbalanced overall experience sampling frequency in the conventional ER sampling techniques for the off-policy RL algorithms and confirmed the mathematical analysis with preliminary experiments. Based on this observation, we propose a focused experience sampling technique on recent experiences, focused experience replay (FER), to mitigate the severely unbalanced overall sampling frequency caused by the conventional sampling techniques. Our experimental results indicate that the major advantage of applying FER to the off-policy RL algorithms is the improvement in the learning speed at the early stage of training. For instance, the learning speed of TD3-FER is 2.5 and 3.5 times faster compared to the vanilla TD3 and TD3-PER algorithms, respectively, for the Ant environment. Additionally, in the CARLA environment, the SAC-FER learns faster and has 2.3 and 1.5 times better final performance, compared to the vanilla SAC and SAC-PER, respectively. In addition, in more than half cases of our experiments, FER outperforms the final performance of the vanilla and PER-applied algorithms and, in other cases, FER maintains the similar performance to other algorithms.

Even if we have shown that FER can improve performance on some MuJoCo locomotion tasks and CARLA autonomous driving simulators, we believe that further study is required to address more on the unbalanced overall sampling frequency issue. One limitation of our technique is the adjustment of σ that relies on the general sizing factor, β_1, since inappropriate choice of β_1 could result in a lower final performance. The second limitation is that an experience sampling technique with focused sampling on the relatively newer experiences

TABLE 2: Network and hyperparameters for DDPG

Parameter	Value
First hidden layer	300
Second hidden layer	400
Learning rate	$3 \cdot 10^{-4}$
Optimizer	Adam [36]
Smoothing coefficient	$5 \cdot 10^{-3}$
Maximum RB size	1 million
Sampling batch size	256
Activation function	ReLU
Discount factor γ	0.99
Exploration noise	$\mathcal{N}(0, 0.2)$

TABLE 3: Network and hyperparameters for TD3

Parameter	Value
First hidden layer	300
Second hidden layer	400
Learning rate	$3 \cdot 10^{-4}$
Optimizer	Adam [36]
Smoothing coefficient	$5 \cdot 10^{-3}$
Maximum RB size	1 million
Sampling batch size	256
Activation function	ReLU
Discount factor γ	0.99
Exploration noise	$\mathcal{N}(0, 0.2)$

may lead to a premature convergence of the policy. Although we address this issue by using distance maximization in the policy update, there is still a hyperparameter $D_{\text{KL,max}}$ that needs to be tuned. Too large $D_{\text{KL,max}}$ may lead to a very unstable policy development, while too small $D_{\text{KL,max}}$ leads to premature policy convergence.

.

APPENDIX A NETWORK ARCHITECTURE AND HYPERPARAMETERS

We present the network architecture and hyperparameters for algorithms presented in the Section V. The hyperparameters are selected based on benchmark results in [6]- [9], [11].

For the implementation of DDPG, we use the network parameters as given in Table 2 that is based on the model of TD3 in [7], but instead of applying the twin-delayed version, we use the actor-critic agent as a single DDPG. The network parameters of TD3 and SAC are given in Table 3 and Table 4 that are based on the model proposed in [7] and [9], respectively.

TABLE 4: Network and hyperparameters for SAC

Parameter	Value
First hidden layer	256
Second hidden layer	256
Learning rate	$3 \cdot 10^{-4}$
Optimizer	Adam [36]
Smoothing coefficient	$5 \cdot 10^{-3}$
Maximum RB size	1 million
Sampling batch size	256
Activation function	ReLU
Discount factor γ	0.99

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

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

[3] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[4] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

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

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning." in *International Conference on Learning Representations*, 2016.

[7] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1587–1596.

[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." in *International Conference on Machine Learning*, 2018, pp. 1856–1865.

[9] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[11] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*, 2016, pp. 1329–1338.

[12] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, 2014, pp. 387–395.

[13] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[14] H. Dong, H. Dong, Z. Ding, S. Zhang, and Chang, *Deep Reinforcement Learning*. Springer, 2020.

[15] M. Yogeswaran and S. Ponnambalam, "Reinforcement learning: exploration–exploitation dilemma in multi-agent foraging task," *Opsearch*, vol. 49, no. 3, pp. 223–236, 2012.

[16] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[17] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *International Conference on Learning Representations*, 2016.

[18] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[19] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[20] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world

reinforcement learning," *ICML Workshop on Reinforcement Learning for Real Life*, 2019.

[21] L. F. Vecchietti, T. Kim, K. Choi, J. Hong, and D. Har, "Batch prioritization in multigoal reinforcement learning," *IEEE Access*, vol. 8, pp. 137 449–137 461, 2020.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[23] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] S. Grossberg, "How does a brain build a cognitive code?" *Studies of mind and brain*, pp. 1–52, 1982.

[25] A. R. Mahmood, H. Van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation." in *Advances in Neural Information Processing Systems*, 2014, pp. 3014–3022.

[26] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel ddpg method with prioritized experience replay," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, 2017, pp. 316–321.

[27] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. H. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017.

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

[29] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *International Conference on Learning Representations*, 2018.

[30] D. Zha, K.-H. Lai, K. Zhou, and X. Hu, "Experience replay optimization," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019, pp. 4243–4249.

[31] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee, "Diversity-driven exploration strategy for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[32] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, 2017, pp. 1–16.

[33] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021. [Online]. Available: https://github.com/cjy1992/gym-carla

[34] L. Fridman, J. Terwilliger, and B. Jenik, "Deeptraffic: Crowdsourced hyperparameter tuning of deep reinforcement learning systems for multi-agent dense traffic navigation," in *Neural Information Processing Systems (NIPS 2018) Deep Reinforcement Learning Workshop*, 2018.

[35] H. S. Jomaa, J. Grabocka, and L. Schmidt-Thieme, "Hyp-rl: Hyperparameter optimization by reinforcement learning," *arXiv preprint arXiv:1906.11527*, 2019.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

SEUNG-HYUN KONG (M'06–SM'16) is an Associate Professor in the CCS Graduate School of Green Transportation of Korea Advanced Institute of Science and Technology (KAIST), where he has been a faculty member since 2010. He received a B.S. degree in Electronics Engineering from Sogang University, Seoul, Korea, in 1992, an M.S. degree in Electrical and Computer Engineering from Polytechnic University (merged to NYU), New York, in 1994, a Ph.D. degree in Aeronautics and Astronautics from Stanford University, Palo Alto, in 2005. From 1997 to 2004 and from 2006 to 2010, he was with companies including Samsung Electronics (Telecommunication Research Center), Korea, and Qualcomm (Corporate R&D department), San Diego, USA for advanced technology R&D in mobile communication systems, wireless positioning, and assisted GNSS. Since he joined KAIST as a faculty member in 2010, he has been working on various R&D projects in advanced intelligent transportation systems, such as robust GNSS-based navigation for urban environment, deep learning and reinforcement learning algorithms for autonomous vehicles, sensor fusion, and vehicular communication systems (V2X). He has authored more than 100 papers in peer-reviewed journals and conference proceedings and 12 patents, and his research group won the President Award (of Korea) in the 2018 international student autonomous driving competition host by the Korean government. He has served as an associate editor of IEEE T-ITS and IEEE Access, an editor of IET-RSN and the lead guest editor of the IEEE T-ITS special issue on "ITS empowered by AI technologies" and the IEEE Access special section on "GNSS, Localization, and Navigation Technologies". He has served as the program chair of IPNT from 2017 to 2019 in Korea and as a program co-chair of IEEE ITSC2019, New Zealand.

I MADE ASWIN NAHRENDRA received the B.S. and M.S degree in Electrical Engineering from Bandung Institute of Technology, Indonesia, in 2018 and 2019, respectively. He is currently pursuing the Ph.D degree at the Robotics Program in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. His research interests include reinforcement learning, robotics, control theory, and autonomous vehicle.

DONG-HEE PAEK received the B.S. degree in Robotics from Kwang-Woon University, Korea, in 2019. He is currently pursuing the M.S. degree at the Robotics Program in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. His research interests include autonomous vehicle, reinforcement learning, and robotics.