Sample Efficient Experience Replay in Non-stationary Environments

Anonymous submission

Abstract

Reinforcement learning often face non-stationary environments in real-world applications, which requires agents to adapt to changing dynamics by correcting outdated experiences during training. Experience replay is a technique used in reinforcement learning to store and reuse past experiences, which can improve the efficiency of agent training. However, existing methods assign sample priorities based on TD-error, which may not effectively support rapid adaptation in non-stationary environments. This is because samples that have not been sufficiently explored tend to exhibit high TD-error, potentially leading to biased prioritization. In this paper, we introduce a new metric called Environmental Difference Error(ED-error), which quantifies the sample value in non-stationary environments by measuring the magnitude of state transition function changes. Based on ED-error, we propose the Environmental Prioritized Experience Replay(EPER), a prioritized experience replay method to address the challenges posed by non-stationary environments. Experimental results show that EPER outperformed all baselines, requiring the fewest steps for performance recovery when the environment changed, resulting in a 42.45% improvement compared to the best baseline.

Introduction

Reinforcement learning (RL) finds wide-ranging applications in real-world scenarios(Lillicrap et al. 2015; Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018), with many of these environments being characterized by non-stationarity. In the context of RL, a non-stationary environment refers to a dynamic system where the underlying properties relevant to RL (e.g. environment dynamics, state transition probabilities) change over time(Padakandla 2021). Unlike stationary environments, where these properties remain constant, non-stationary environments introduce additional complexity as the environment's behavior evolves. The changes in a non-stationary environment is unpredictable, implying the absence of universally applicable patterns to follow. This requires RL agents to continuously adjust their policy to accommodate the evolving environment(Ditzler et al. 2015).

Efficient sampling, which maximizes learning efficiency with a limited number of samples, is particularly important in non-stationary environments for RL. The goal of RL is to maximize the cumulative reward by continuously interacting with the environment and learning from experiences to estimate the accumulated reward by Q-function. In non-stationary environments, the state transition function changes over time. Inefficient sampling decrease the learning efficiency of the agent, potentially leading to suboptimal policies and even reward collapse(Ditzler et al. 2015). Additionally, in real-world applications, inefficient sampling increases the demand for sample quantity, resulting in additional time and financial costs(Yu 2018).

Experience Replay (ER) is a technique used in RL, which improving sample efficiency and learning stability by storing transactions in a buffer and randomly samples them for training. Traditional ER uses random sampling, while many recent works adopt temporal difference error (TD-error) as a criterion for prioritized sampling to further improve sampling efficiency[???]. TD-error measures the discrepancy between the predicted Q-value and the actual observed reward, which reflects the impact of a certain transaction on the Q-function. Therefore, prioritizing transactions with higher TD-error leads to more informative updates of the Q-function and accelerates learning.

However, using TD error as a priority in non-stationary environments fails to accurately reflect the importance of transitions for improving sample efficiency. This is because transitions with high TD error in non-stationary environments can be attributed to the following reasons:

(a) The transition exists in a state-action space that has undergone changes due to the non-stationarity of the environment.

(b) The transitions correspond to unexplored state-action spaces, which are low-probability events, or are generated by a random policy (e.g., ε-greedy policy).

In non-stationary environments, (a) is considered crucial for adapting quickly to changing environments as it reflects changes in the state transition function, while (b) is beneficial for exploration purposes but hinders rapid adaptation. Using TD error as the metric for prioritized sampling can lead to both (a) and (b) having high priority, thus hindering the agent's ability to quickly adjust its policy in non-stationary environments.

Different from stationary environments, using ER in non-stationary environments faces several unique challenges:

Firstly, there is a lack of a metric that accurately measures the change in the value of transactions for Q-function estimation due to non-stationarity, to distinguish (a) from (b). To overcome this challenge, we introduce Environment Difference Error (ED-error) to measure the impact of state transition function changes on Q-functions. The ED-error quantifies the difference between Q-function estimates before and after the change in the state transition function, excluding differences caused by temporal variations. This is achieved by computing the difference between the current Q-function and the Q-function prior to the change in the state transition function. The difference between the two for the same state-action pair arises due to variations in the environment.

The application of ED-error in ER introduces additional challenges: Firstly, transactions before and after environmental changes require different methods of priority calculation, making traditional unified sampling ineffective. Secondly, after a change is detected in the state transition function, extensively training on a few high-priority post-change transactions potentially lead to overfitting. A selective approach is needed to stabilize the training by reusing pre-change transactions. It is necessary to determine which pre-change transactions remain applicable in the new environment, and which ones have become outdated or even harmful[??]. To address this issue, we propose Environment Priority Experience Replay (EPER), which samples separately from pre- and post-change transition for training. For post-change transactions, the larger the ED-error, indicating a greater deviation from the expected returns of the old policy, the higher the priority. For pre-change transactions, the opposite holds true.

Our contributions can be summarized as follows:

(1) We propose ED-error, a novel metric to quantifies the importance of transaction in a non-stationary environment by measuring the magnitude of the state transition function change.

(2) We propose EPER, an experience replay method specifically designed for non-stationary environments. It allows for separate evaluation of sample importance from pre- and post-change transactions, and prioritizes transactions based on their ED errors.

(3) We evaluate EPER in the MiniGrid environment, including two non-stationary settings(noisy observation and obstacle) out of the four tasks. EPER outperformed 3 baselines, achieving the highest average cumulative reward. EPER required the fewest steps for performance recovery when the environment changed, resulting in a 42.45% improvement compared to the best baseline. Specifically, in the task with added obstacles, EPER reduced the average time steps for recovery by 52.94%, and in the task with noisy observations, it reduced it by 28.86%.

Related Work

Random uniform sampling ignores the importance of each transition, so a series of sampling methods have been proposed to improve sampling efficiency. Schaul et al. (2015) proposed the Prioritized Experience Replay(PER), a sampling method that uses TD as a priority. PER uses TD to sample proportionally in the replay buffer, achieving better performance than random uniform sampling. Zhang and Sutton (2017) proposed the Combined Experience Replay(CER), which can be seen as an extreme case of PER. PER makes the latest transition have a higher probability of being sampled, while in CER the latest transition will definitely be sampled. Brittain et al. (2019) proposed the Prioritized Sequence Experience Replay(PSER). This sampling method assigns priority to the latest transition while updating the priority of the remaining transitions in the replay buffer.

In other works, other metrics are used as sampling priorities. Sun, Zhou, and Li (2020) propose the Attentive Experience Replay(AER), a sampling method that prioritizes transitions in the replay buffer based on their similarity to the current state. The Remember and Forget Experience Replay(ReF-ER) classifies transitions as ”near-policy” or ”far-policy” by the ratio of the current policy to the past policy and samples only the transitions of near-policy(Novati and Koumoutsakos 2019). Fujimoto, Meger, and Precup (2020) equivalently represented the sample priority in PER as a loss function and proposed the Loss-Adjusted Prioritized (LAP) experience replay, a sampling method that optimizes the sample priority through a loss function.

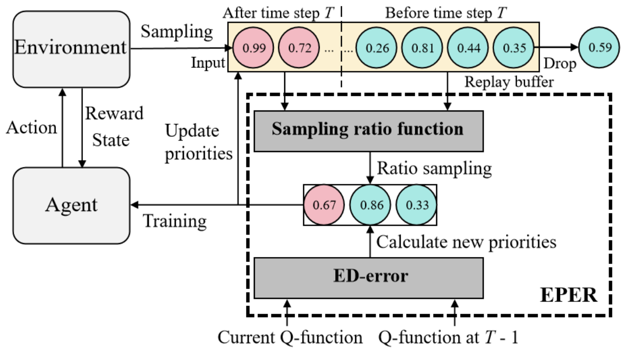


Figure 1: The workflow of EPER. T is the time step in which the environment changes.

Methodology

We propose a new experience replay method called EPER, which selects transitions from the replay buffer that reflect environmental non-stationarity to correct the agent's estimates of expected returns. For this, we introduce a novel metric called ED-error in EPER. The ED-error measures the disparity between the Q-function estimation of a transition before the change of the state transfer function and the current Q-function estimation of the same transition. Samples are assigned higher priority if the ED-error indicates a larger difference between the two Q-function estimates. This prioritization mechanism enables EPER to select transitions that reflect significant variations in the environment, ensuring that the agent's estimates of expected returns are appropriately adjusted.

Figure 1 shows the workflow of EPER. When the state transition function changes at time step T, EPER stores the Q-function at time step T-1. The stored Q-function is subsequently used to compute the ED-error for newly acquired transitions and transitions in the replay buffer. Then, EPER selects a proportional subset of transitions from both pre- and post-time step T to compose a minibatch designated for agent training. In our approach, we use ED-error as a metric for priority sampling. EPER sample two parts of a transaction (pre-change and post-change) separately, as opposed to previous methods that sampled from the entire replay buffer.

Preliminary

The non-stationary reinforcement learning can be formalized as a Markov decision process(MDP). Specifically, it is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the state transition function, r is the reward function, and γ is the discount factor. At each time step t , the agent chooses an action $a_t \in \mathcal{A}$ to interact with the environment. Then, the environment transits to the next state $s_{t+1} \in \mathcal{S}$ according to the conditional state transfer function$P\left(s_{t+1}, r_t \mid s_t, a_t\right)$ and and gives a reward r_t to the agent. The state transition function is defined as:

$$P\left(s_{t+1}, r_t \mid s_t, a_t\right)=\begin{cases} P_0\left(s_{t+1}, r_t \mid s_t, a_t\right), \; 0 \leq t < T_0 \\ \ldots \\ P_i\left(s_{t+1}, r_t \mid s_t, a_t\right), \; T_{i-1} \leq t < T_i \\ \ldots \\ P_n\left(s_{t+1}, r_t \mid s_t, a_t\right), \; T_{n-1} \leq t \leq T \end{cases} \tag{1}$$

Where $P_i\left(s_{t+1}, r_t \mid s_t, a_t\right)$ denotes the i-th joint probability distribution of the environment transitioning to state s_{t+1} and reward r_t given the state-action pair s_t, a_t. For brevity, we denote $P_i\left(s_{t+1}, r_t \mid s_t, a_t\right)$ as P_i. An non-stationary environment can be regarded as a dynamically changing environment, where the state transition function P changes with time step t. When $n \rightarrow \infty$, (1) represents scenarios where the state transition function changes at each time step t, such as fluctuations in natural lighting or temperature.

The goal of the agent is to learn a policy π to maximize the expected discounted return $\mathbb{E}\left[\sum_{t'=t}^{T} \gamma^{t'-t} r_t\right]$. The value function and Q-function (namely, the action-value function) [???] are proposed to help the agent learn better policies. Determining the state s_t and the policy π, the value function $v\left(s_t\right)$ is defined as follows:

$$v\left(s_t\right)=\mathbb{E}_\pi\left[\sum_{t'=t}^{T} \gamma^{t'-t} r_t\right] \tag{2}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expectation of a random variable that the agent follows the policy π. Given the state s_t and action a_t, the Q-function $Q\left(s_t, a_t\right)$ is defined as the expectation of the value function:

$$\begin{aligned} & Q\left(s_t, a_t\right)=\mathbb{E}_\pi\left[r_t+\gamma v\left(s_{t+1}\right)\right]= \\ & \sum_{s_{t+1}, r_t} P\left(s_{t+1}, r_t \mid s_t, a_t\right)\left[r_t+\gamma \mathbb{E}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'}\right]\right] \end{aligned} \tag{3}$$

The optimal policy usually derives from maximizing the expectation of the value function, i.e. $\pi = \arg\max_{a_t} Q\left(s_t, a_t\right)$. The Q-function $Q\left(s_t, a_t\right)$ is updated by temporal difference learning:

$$\begin{aligned} Q\left(s_t, a_t\right) \leftarrow & Q\left(s_t, a_t\right)+ \\ & \alpha\left[r_t+\gamma Q\left(s_{t+1}, a_{t+1}\right)-Q\left(s_t, a_t\right)\right] \end{aligned} \tag{4}$$

where α is the learning rate. $r_t+\gamma Q\left(s_{t+1}, a_{t+1}\right)-Q\left(s_t, a_t\right)$ is called the temporal difference error(TD-error) and is denoted as δ^{TD} in the paper.

Sample prioritization based on environmental difference

In previous methods, the priority of a transition is measured by the absolute value of the TD-error. A higher TD-error value implies a greater impact on the agent's Q-function estimation, thus these transitions are considered to have higher priority. During training, transitions with higher priority are selected with a higher probability, which increases the number of training iterations on those important samples and accelerates the convergence of policy. To understand the difference between using TD-error as priority in stationary and non-stationary environments, we first consider a simple and stationary environment, such as the Atari 2600[???] or Grid World[???]. In the environment, given the state s, action a, the next state s' and reward r are uniquely determined, which is denoted by the state transition function:

$$\begin{aligned} P\left(s_{t+1}, r_t \mid s_t, a_t\right) & =\operatorname{Pr}\left\{s_{t+1}=s', r_t=r \mid s_t=s, a_t=a\right\} \\ & =1 \end{aligned} \tag{5}$$

where $\operatorname{Pr}\{\cdot\}$ denotes the probability. In this case, the environment is stationary, and the agent easily learns each possible pair of next states s' and rewards r. Given a transition $\left(s_{t-1}, a_{t-1}, r_{t-1}, s_t\right)$, the TD-error is denoted as:

$$\begin{aligned} & \delta^{TD}=\sum_{s_{t+1}, r_t} P\left(s_{t+1}, r_t \mid s_t, a_t\right)\left[\gamma r_t+\gamma^{t'-t+1}\right. \\ & \left.\mathbb{E}\left[\sum_{t'=t+1}^{T} r_{t'}\right]\right]-\gamma^{t'-t+1} \mathbb{E}\left[\sum_{t'=t}^{T} r_{t'} \mid s_{t-1}, a_{t-1}\right] \end{aligned} \tag{6}$$

(6) shows that when transition $\left(s_{t-1}, a_{t-1}, r_{t-1}, s_t\right)$ contains rare events or critical rewards, the transition has high TD-error. The former leads to biased estimation of the state transition function $P\left(s_{t+1}, r_t \mid s_t, a_t\right)$, while the latter generates larger disparities in expected returns. Therefore, by assigning higher priority to these rare transitions, the agent can learn crucial experiences that might be overlooked in a uniform random sampling. However, when introducing non-stationary factors into the environment, i.e., the state transition function$P$ is denoted by (1), the TD-error is denoted as:

$$\delta^{TD} = r_{t-1} + \sum_{s_{t+1}, r_t} P\left(s_{t+1}, r_t \mid s_t, a_t\right) \left[\gamma r_t + \gamma^{t'-t+1} \right.$$
$$\left. \mathbb{E}\left[\sum_{t'=t+1}^{T} r_{t'}\right]\right] - \sum_{s_t, r_{t-1}} P\left(s_t, r_{t-1} \mid s_{t-1}, a_{t-1}\right) \cdot$$
$$\left[r_{t-1} + \gamma^{t'-t+1} \mathbb{E}\left[\sum_{t'=t}^{T} r_{t'}\right]\right] \tag{7}$$

(7)shows that the situation is more complex, especially when the state transition function transitions from P_{i-1} to P_i. Because the agent's learned experiences are based on the old state transition function P_{i-1}, it introduces a bias in estimating the expected return, which causes substantial fluctuations in TD-error. Transitions with high TD-error may not represent the most valuable experiences, which can result in incorrect policy updates. Incorrect policy updates cause the agent to take sub-optimal or ineffective actions, which diminish the value of experiences in subsequent transitions. To address the problem, we propose a metric that reflects the difference in expected returns due to a change in the state transfer function. Ideally, if the agent already knows to the expected return of all state-action pairs (s, a), then the agent just need to learn the new state transfer function P_i. We define the metric of the current transition $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ as:

$$\delta^{ED} = \sum_{s_t, r_t} P_{i-1}\left(s_t, r_{t-1} \mid s_{t-1}, a_{t-1}\right) \mathbb{E}\left[\sum_{t'=t}^{T} r_{t'}\right]$$
$$- \sum_{s_t, r_t} P_i\left(s_t, r_{t-1} \mid s_{t-1}, a_{t-1}\right) \mathbb{E}\left[\sum_{t'=t}^{T} r_{t'}\right] \tag{8}$$

We call the new metric "Environmental Difference Error(ED-error)", denoted as δ^{ED}. The intuition behind this is to utilize the estimator $Q_{i-1}\left(s_{t-1}, a_{t-1}\right) - Q_i\left(s_{t-1}, a_{t-1}\right)$ to approximate δ^{ED}, where Q_{i-1} and Q_i denote the Q-function under the state transition functions P_{i-1} and P_i, respectively. The ED-error quantifies the disparity between Q_{i-1} and Q_i, enabling it to guide the agent in gradually transitioning from Q_{i-1} to Q_i, thereby adapting to the new state transition function P_i.

However, it is not feasible to rely on the assumption that the state transition function changes only after the agent has fully explored the entire state-action space in each scenario. Consequently, the agent's estimation of $\mathbb{E}\left[\sum_{t'=t}^{T} r_{t'}\right]$ is also inaccurate. Moreover, the state transition functions P_{i-1} and P_i are not explicitly represented in the agent's policy. Thus, we use Q-function approximations at different time steps to estimate the expected returns under each state transition function in ED-error. Specifically, assume that the non-stationary environment is denoted by (1). When time step $T_{i-1} \leq t \leq T_i$, we define the ED of the current transition $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ as:

$$\delta^{ED} = r_{t-1} + Q^*\left(s_t, a_t\right) - Q\left(s_{t-1}, a_{t-1}\right) \tag{9}$$

where Q^* denotes the Q-function at time step $T_{i-1} - 1$ and Q denotes the current Q-function. Figure 2 shows the difference between ED-error and TD-error. TD-error is the difference between the Q-function of the current action-state pair and the Q-function of the last time step, while ED-error is the difference with the Q-function of the state transition function before the change.

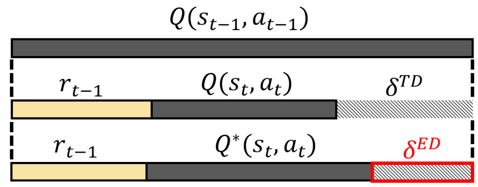


Figure 2: The ED-error and the TD-error of a transition $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$. Q denotes the current Q-function. Q^* denotes the Q-function before the state transition function change.

Environmental Prioritized Experience Replay

Non-stationary environments is caused by changes in the state transition function. Sudden changes in the state transition function result in inaccurate estimation of expected rewards by the Q-function. The ED-error measures the magnitude of changes in the state transition function of a transition and it indicates the degree to which it surprises the agent's estimate of the expected return. Thus high ED-error transitions can better guide the agent's correction of expected returns. However, using ED-error simply as a priority for sampling has several issues. Firstly, transitions under the current state transfer function usually have a higher ED-error because they directly reflect the degree of instability of the environment. However, there is a high degree of temporal correlation between these transitions, especially when the state transfer function has just changed. This frequent sampling can lead to agent overfitting [????] , which in turn reduces the agent's ability to generalize. Secondly, when the state-transfer function changes only locally, the high-ED-error transitions are concentrated in a subspace of the state-action space. This means that the agent will frequently explore in this subspace and ignore the rest.

To address these issues, we propose EPER (see Algorithm 1). EPER does not impose any requirements on the off-policy RL algorithm, which makes EPER easily applicable to many off-policy RL methods (e.g., dqn, ddpg, etc.). EPER saves the Q function for the current time step when it detects a change in the environment and uses ED-error as a priority, and this process continues until the transition after the change in the environment fills the entire replay buffer. Environmental changes are realized through the detection of rewards, which is in fact a change-point detection problem, a problem that has been solved in many works [???]. A simple approach is to infer a change in the state transfer function when a sustained sharp drop in reward is detected. Then, we define the sampling ratio function for the j-th batch of new samples as:

$$f(j) = \max\left(\epsilon^j, \frac{j}{L}\right) \tag{10}$$

Algorithm 1: Environmental prioritized experience replay

Input: Maximum time step T, batch size N, buffer capacity L, exponents α, β and ϵ, learning rate η.

Initialize state s_0 and replay buffer $\mathcal{D}$ of capacity L.
for $t = 1$ to T **do**
  Select and execute a_{t-1} according to the policy π, observe reward r_{t-1} and next state s_t.
  Store transition $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ in $\mathcal{D}$ with maximal priority $p_t = \max_{i<t} p_i$
  if *environment_change* **then**
    Store the Q-function as Q^*
    Calculate the ED-error for transitions in D by Eq.(9) and update the priorities: $p_k \leftarrow 1/\left|\delta_k^{ED}\right|$
    $j \leftarrow L$
    if $j > 0$ **then**
      $j \leftarrow j - 1$
      Set the sampling ratio function by Eq.(10)
      Sample a batch of $N * (1 - f(j))$ and $N * f(j)$ transitions in $\mathcal{D}$ by Eq.(11), both before and after the environment change, respectively.
      Compute the ED-error for the batch of transitions by Eq.(9) and update the priorities of the transitions after the environment change: $p_k \leftarrow \left|\delta_k^{ED}\right|$
      for $k = 1$ to N **do**
        Compute importance-sampling weight $w_k = (N * P(k))^{-\beta} / \max_i w_i$
        Accumulate weight-change $\Delta \leftarrow \Delta + w_k * \delta_k^{TD} * \nabla_\theta Q(s_{k-1}, a_{k-1})$
        Update weights $\theta \leftarrow \theta + \eta * \Delta$, reset $\Delta = 0$
      end for
    else
      Using the TD-error as the priority for experience replay.
    end if
  end if
end for

where L denotes the replay buffer capacity, and ϵ is an exponent. The sampling ratio function is used to ensure that transitions before and after a change in the state transfer function are selected, and that the ratio of pre-change transitions that are selected decreases as the number of them in the replay buffer decreases. The probability of sample k being selected is:

$$P(k) = \frac{p_k^\alpha}{\sum_i p_i^\alpha} \tag{11}$$

where α is an exponent. Importance-sampling(IS) weights are used to correct for bias, and the sampling weight of the final selected sample k is denoted as:

$$w_k = \left(\frac{1}{N \cdot P(k)}\right)^\beta \tag{12}$$

where N denotes minibatch size and β is an exponent. When the state transfer has changed, a high ED-error for new transitions means that it reflects the difference between the current environment and the environment before the change. However, for a sample that has been stored in the replay buffer, high ED-error means that it contains experiences that do not match the current environment. Therefore, for transitions that are newly sampled after a change of the state transition function, the priority is defined as $p_k = \left|\delta_k^{ED}\right|$; for transitions that are stored in the replay buffer before the change of state transition function, the priority is defined as $p_k = 1/\left|\delta_k^{ED}\right|$.

Evaluation

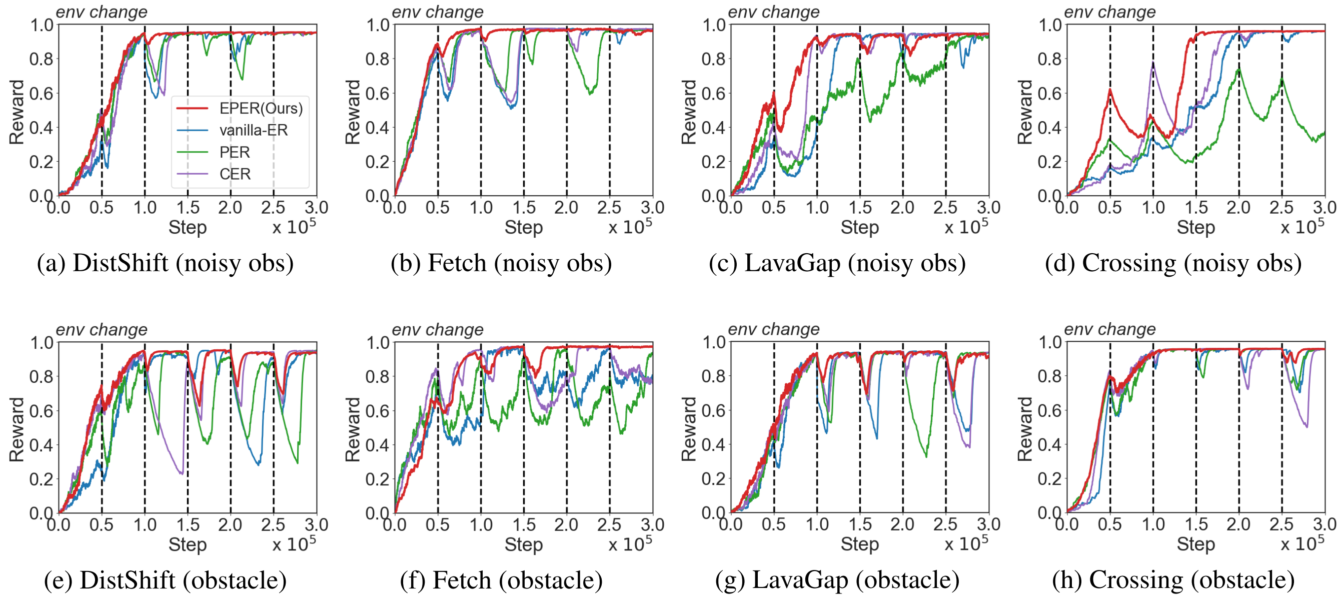
We examine the sample efficiency of various experience replay methods in reinforcement learning tasks. Our experiments employ a Deep Q-Network (DQN) as the reinforcement learning agent. For all tasks, our DQN model consists of a series of layers: three convolutional layers with 64, 128, and 256 nodes respectively, followed by three fully connected layers with 256, 128, and a number of nodes that match the number of actions available in the environment. We use the Adam algorithm (Kingma and Ba, 2014) with the learning rate 0.0001. We use a discount factor (gamma) of 0.99 and a batch size of 128. EPER has three hyperparameters, of which we use α for a range of $\{0.6, 0.7, 0.8\}$; β for annealing from 1 within the initial 50,000 time steps; and ϵ for a range of $\{0.98, 0.99\}$.

All experiments are performed on a server with 48 Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz and 4 NVIDIA GeForce RTX 2080 Ti 11GB GPU.

Nonstationary Environment Setups

We evaluated the performance of EPER in the discrete action space using Minigrid(Chevalier-Boisvert, Willems, and Pal 2018), a popular test environment for reinforcement learning research with a collection of 2D grid-world environments with goal-oriented tasks.

To evaluate the effectiveness of the EPER algorithm in non-stationary environments, we introduced specific non-stationary factors at certain time points during the training process. We evaluated the performance of the EPER algo-



(a) DistShift (noisy obs) (b) Fetch (noisy obs) (c) LavaGap (noisy obs) (d) Crossing (noisy obs)

(e) DistShift (obstacle) (f) Fetch (obstacle) (g) LavaGap (obstacle) (h) Crossing (obstacle)

Figure 3: Learning curves of DQN using different sampling methods in 4 non-stationary Minigrid environments. (a)-(d) depict the performance in environments with introduced observation noise. (e)-(h) depict the performance in environments with introduced obstacles. The dashed line indicates the timing of the non-stationarity change in the environment.

rithm and the baseline in non-stationary environments by observing the convergence speed and cumulative rewards of the models after the environment changed. We introduced 2 types of non-stationary factors from the perspectives of environment construction and agent observation during the agent-environment interaction process.

Noisy Observation: Introduce noise to the agent's environment perception to introduce non-stationarity in the observations. Many works have explored reinforcement learning models in noisy observations(Kilinc and Montana 2018; Shahryari and Doshi 2017; Wang, He, and Tan 2019), as in many real-world scenarios, observations from sensors such as cameras or microphones may contain inaccuracies due to noise or calibration issues. We follow the approach proposed in(Wang, He, and Tan 2019), where Gaussian noise is generated using a multivariate Gaussian distribution with a given mean and covariance matrix to add noise to the agent's observations, simulating the nonstationarity commonly encountered in real-world applications.

Obstacles: We introduced obstacles into the environment to create nonstationarity in its construction. The number of obstacles introduced depended on the number of empty squares in the environment. By incorporating obstacles, we aimed to simulate the nonstationary aspects often encountered in real-world scenarios. When the agent collided with an obstacle, a substantial penalty was subtracted from its reward, and the episode was terminated. This encourages the agent to change its existing strategy to adapt to the environment when it undergoes changes.

Baseline

We compared our proposed method EPER with several widely used baselines in the field of reinforcement learning that have been shown to improve the performance of deep reinforcement learning algorithms.

Vanilla-ER(Mnih et al. 2015): This is the standard experience replay method, where experiences are stored in a replay buffer and randomly sampled during training. The replay buffer is typically limited in size, and new experiences overwrite old ones when the buffer is full.

Prioritized Experience Replay (PER)(Schaul et al. 2015): This method assigns a priority to each experience based on its estimated error, with experiences that have higher errors given higher priority. During training, experiences are sampled based on their priority, with higher priority experiences sampled more frequently. This approach has been shown to improve the sample efficiency and overall performance of reinforcement learning algorithms.

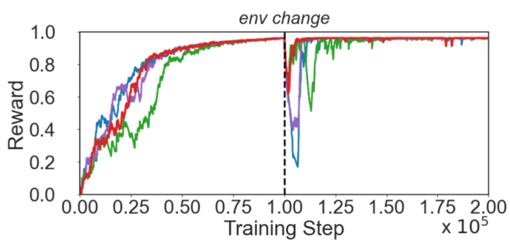
Combined Experience Replay (CER)(Zhang and Sutton 2017): This method is designed to address the negative influence of a large replay buffer on deep reinforcement learning algorithms. In CER, the latest transition is always added to the replay buffer, while the oldest transition is removed, so the size of the buffer remains constant. The corrected batch is then used to train the agent.

Comparative Evaluation

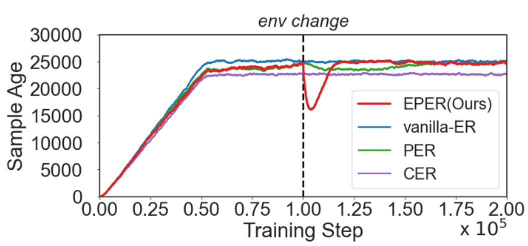
The experimental results demonstrate that EPER significantly outperforms other baselines in 4 kinds of Minigrid tasks with 2 kinds of non-stationarity. Specifically, Figure 3 shows the reward curves of DQN with different ER methods during training in all Minigrid tasks. For full evaluation, the non-stationarity parameter of the environments are changed every 50,000 time steps in all experiments (the environments is stationary within the first 50,000 time steps), and the training lasted for a total of 300,000 time steps. As

Environments		EPER(Ours)	vanilla-ER	PER	CER
Noisy Observation	MiniGrid-DistShift	**3946.40**	7572.20	8243.20	9026.20
	MiniGrid-Fetch	**15409.60**	18661.80	19464.20	15611.20
	MiniGrid-LavaGapS6	**7239.40**	12908.20	26516.00	8944.20
	MiniGrid-Crossing	**12223.40**	18265.00	39717.20	20987.60
Obstacle	MiniGrid-DistShift	**10594.60**	14352.80	24457.40	15785.20
	MiniGrid-Fetch	**9757.60**	34996.40	26055.00	26672.40
	MiniGrid-LavaGapS6	**8112.60**	15334.20	16878.40	14228.60
	MiniGrid-Crossing	**4756.40**	7013.20	10774.20	13919.80
	Average	**9005.00**	16137.98	21513.20	15646.90

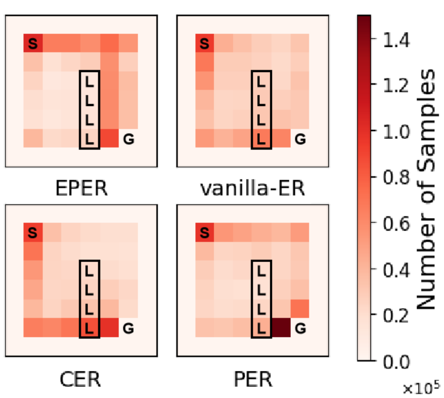
Table 1: Average number of steps required for different ER methods to restore the reward to its pre-change level after a change of environment.



(a) Reward curve



(b) Sample age curve



(c) The position of the agent in the samples after 10,000 training steps following a change in the environment.

Figure 4: Reward curve and sampling behavior by different ERs in the non-stationary Minigrid-Empty environment.

	EPER	vanilla-ER	PER	CER
Average reward	**0.9357**	0.9062	0.8408	0.9158

Table 2: Average reward

mentioned before, figures 3(a)-(d) and (e)-(h) show the training curves for the Minigrid task with the introduction of observation noise and obstacles, respectively. Table 1 shows the average time step of reward required to restore the pre-change level after 5 changes in all Minigrid tasks. It is a performance indicator of the agent's ability to adapt to non-stationarity. The faster the reward restores after each change in the non-stationary environment means that the agent can adapt to it faster. Table 2 shows the average rewards of DQN with different ER methods for each episode in all Minigrid tasks.

We observe that EPER outperforms the other baselines in terms of average rewards in all the test tasks (average rewards are 2.17% higher than the optimal baseline) while the average time step for reward regain decreases by 42.45% compared to the optimal baseline. It demonstrates that EPER significantly improves its ability to adapt to environmental non-stationarity while maintaining DQN performance. In particular, the average number of time steps rewarded by EPER for restoring in the task of adding obstacles decreased by 52.94% compared to the optimal baseline (28.86% in the task of adding noisy observation). This is because obstacles only increase the non-stationarity of the part environment, whereas the noise we add in whole observations increased the non-stationarity of the overall environment.

Analysis on statistics of sampling behavior

We analyzed the sampling behavior of EPER during the training to determine the key reasons for its outperforms other baselines in non-stationary environments. **The experimental results demonstrate that EPER selects transitions reflecting changes in the environment for agent training, and tends to assign higher priority to transitions after**

changes in the environment.

Specifically, we chose one of the most basic scenarios, Minigrid-Empty, to show the sample distribution more clearly. As shown in Figure 4(c), the agent needs to reach the goal square (denoted as G) located at the bottom right corner from the initial square(denoted as S) at the top left corner. The reward is 1 after the agent reaches the goal square and a small penalty is subtracted based on the number of steps taken to reach the goal. There are no obstacles in the environment. To introduce non-stationarity, we add a 1*4 lava square (denoted as L) at 100,000 time steps. An episode is terminated with a zero reward when the agent touches a lava square.

Figure 4(a) shows the reward curve during training. Furthermore, Figure 4(b) shows the age of the transition used by the agent during training. The maximum age of the transition is 30,000, which is the same as the replay buffer size. A higher age means that the transition used by the agent for training is in the replay buffer for a longer time. Figure 4(c) shows the number of samples located in each square that are selected for agent training during the rest of the training process after the environment change. We can observe that the RL algorithm rewarded by using EPER after the environment change converges significantly better than the rest of the baseline. At the same time, the age at which EPER selects transitions after an environmental change decreases dramatically and transitions near lava squares are frequently selected for agent training. The rest of the baseline selects transitions with no significant change in age and tends to use routes that were already successful before the environmental change and does not consider the addition of lava. These results demonstrate that EPER prefers to select transitions near lava squares for agent training compared to the baseline. This is because of a significant increase in the non-stationarity of the environment when adding lava squares, resulting in a larger ED-error and no significant change in TD-error. The decrease in transition age in EPER after the environmental change demonstrates that EPER assigns a higher priority to transitions that are after the addition of lava. This is the same as intuition: after an environmental change, the newest transition is more reflective of the non-stationarity compared to the previous transition.

Conclusion

References

Brittain, M.; Bertram, J.; Yang, X.; and Wei, P. 2019. Prioritized sequence experience replay. *arXiv preprint arXiv:1905.12726*.

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for Gymnasium.

Ditzler, G.; Roveri, M.; Alippi, C.; and Polikar, R. 2015. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4): 12–25.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Fujimoto, S.; Meger, D.; and Precup, D. 2020. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in neural information processing systems*, 33: 14219–14230.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Kilinc, O.; and Montana, G. 2018. Multi-agent deep reinforcement learning with extremely noisy observations. *arXiv preprint arXiv:1812.00922*.

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

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Novati, G.; and Koumoutsakos, P. 2019. Remember and forget for experience replay. In *International Conference on Machine Learning*, 4851–4860. PMLR.

Padakandla, S. 2021. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6): 1–25.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Shahryari, S.; and Doshi, P. 2017. Inverse reinforcement learning under noisy observations. *arXiv preprint arXiv:1710.10116*.

Sun, P.; Zhou, W.; and Li, H. 2020. Attentive experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5900–5907.

Wang, Y.; He, H.; and Tan, X. 2019. Robust reinforcement learning in pomdps with incomplete and noisy observations. *arXiv preprint arXiv:1902.05795*.

Yu, Y. 2018. Towards Sample Efficient Reinforcement Learning. In *IJCAI*, 5739–5743.

Zhang, S.; and Sutton, R. S. 2017. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*.