Sample Efficient Experience Replay in Non-stationary Environments

Paper #964

Abstract. In non-stationary environments, off-policy reinforcement learning (RL) agents require effective sampling to quickly adapt to changing environmental dynamics. Most existing prioritized replay mechanisms operate under the assumption of a stationary environment, overlooking the negative interference of outdated samples in non-stationary settings. In this paper, we introduce a new metric Environment Discrepancy (DoE), which quantifies the value of transitions in non-stationary environments by measuring changes environmental dynamics. Based on DoE, we propose Discrepancy of Environment Prioritized Experience Replay (DEER), a sample efficient experience replay in non-stationary environments. It employs binary classifier-based density estimation to monitor environmental dynamics in real time and adaptively adjust sampling strategy. DEER prioritizes transitions beneficial to the latest policy, thus enabling swiftly correcting policy biases induced by environmental dynamics. Experimental results across four non-stationary Mujoco tasks with two off-policy algorithms demonstrate that DEER significantly improves sample efficiency compared to baselines.

1 Introduction

Reinforcement Learning (RL) finds wide-ranging applications in real-world scenarios, many of which are characterized by non-stationarity [21, 25, 26, 9, 18, 45, 12]. Non-stationary environments refer to systems where environmental dynamics (i.e. state-transition probability function and reward) evolve over time[27, 19]. The complexity of real-world environments and the unpredictability of changes implies the absence of a universal pattern. This requires agents to efficiently utilize new environmental information to achieve rapid adaptation[4].

Off-policy RL, which allows agents to learn from past experiences, has been widely employed to handle the problem of sparse and expensive sampling in high-dimensional continuous action-state spaces[37, 41]. This is achieved through the use of Experience Replay (ER)[16], which stores and reuses past experiences (known as transitions) for training. Many works have explored the use of non-uniform sampling to improve sample efficiency. The mainstream direction involves using temporal difference error (TD-error) as a prioritized metric, which has demonstrated effectiveness in stationary environments[31, 34, 14, 7, 3]. TD-error measures the disparity between the estimated return and the actual return. By prioritizing transitions with higher TD-error, informative experiences are frequently reused, thus accelerating the training.

In non-stationary environments, past experiences are no longer representative of the current environment, which poses a challenge to effective sampling[28]. Unfortunately, existing methods overlook the negative interference caused by outdated samples in non-stationary settings. TD-error is jointly determined by the environmental dynamics and policy improvement. When the value function has been well-trained to adapt to the new environment, the transition collected before environmental changes has a higher TD-error than those collected in the new environment. Prioritizing these outdated samples actually emphasizes irrelevant experiences to the current environment and hinder learning. Similar issues also exist with other prioritization metrics, such as reward-based[3] or access frequency-based[34]. The crux of the issue lies in the fact that prioritizing metrics solely determined by policy improvement overlooks the changing dynamics of the environment, thereby failing to assess the importance of transitions in the current environment.

To address this challenge, we introduce Discrepancy of Environment Dynamics (DoE) to quantify the impact of environmental dynamics on a transition. For the same state-action pair, the difference in the action value functions serves as an intuitive representation of the dynamics disparity within the environment. DoE is calculated as the difference between the value of the current action value function and the value prior to the evolution, excluding the difference caused by policy improvements.

Based on DoE, we propose Discrepancy of Environment Prioritized Experience Replay (DEER), which adaptively prioritizes experiences that facilitate both policy improvement and environmental adaptation in non-stationary environments. Given the complexity of modeling dynamic patterns in such environments, we employ a binary classifier to evaluate the probability density of reward sequences within adjacent time windows to indirectly observe changes in environmental dynamics. We employ Jensen-Shannon divergence as a metric to quantify the extent of change in the current environment. Additionally, the intrinsic characteristics of non-stationary environments necessitate the use of different priority metrics for transitions before and after the change. For pre-change transitions, a lower DoE indicates their effectiveness in the new environment, enabling the exclusion of outdated samples. For post-change transitions, we use a hybrid metric priority sampling based on real-time density differences, incorporating both TD-error and DoE. This design enables adaptive prioritization between policy improvement and environmental adaptation, while maintaining balance and diversity in sampling.

Our contributions are summarized as follows:

(1) We propose DoE, a priority metric that quantifies the importance of samples in non-stationary environments, by calculating the impact of environmental dynamics changes in the environment on the value of the transition.
(2) We propose DEER, a novel hybrid metric prioritized experience replay method that adaptively adjusts sampling weights in response to changes in environmental dynamics. DEER enhances sample efficiency in non-stationary environments by prioritizing

valuable samples within the current environment.

(3) We combined two popular off-policy RL algorithms to validate the effectiveness of the DEER across different non-stationary continuous control tasks. The results demonstrate that the proposed method outperforms all baselines.

2 Methodology

We propose Discrepancy of Environment Prioritized Experience Replay (DEER), a sample efficient experience replay in non-stationary environments. Figure 1 shows the workflow of DEER.

2.1 Preliminary

We formalize the problem of RL in non-stationary environment as a family of Markov decision processes (MDPs)[27], which is defined as $\{\langle \mathcal{S}, \mathcal{A}, P_i, R_i, T_i, \gamma \rangle\}_{i=0}^{\infty}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space and R_i denotes the reward function. At each time step t, the agent following the policy $\pi(a_t \mid s_t)$ chooses an action $a_t \in \mathcal{A}$ and interacts with the environment. Then, the environment transits to the next state $s_{t+1} \in \mathcal{S}$ following the state-transition probability function $P(s_{t+1} \mid s_t, a_t)$ and gives a reward $r_t = R_i(s_t, a_t)$ to the agent. Ti denotes the time steps when environmental dynamics evolve, which is referred to as change point. Environmental dynamics are jointly determined by the state-transition probability function and the reward function, which denotes as $\langle P_i, R_i \rangle$. The state-transition probability function of non-stationary environments is defined as:

$$P(s_{t+1} \mid s_t, a_t) = \begin{cases} P_0(s_{t+1} \mid s_t, a_t), \ 0 \le t \le T_0 \\ P_1(s_{t+1} \mid s_t, a_t), \ T_0 \le t \le T_1 \\ \dots \\ P_i(s_{t+1} \mid s_t, a_t), \ T_{i-1} < t \le T_i \\ \dots \end{cases} \tag{1}$$

The reward function of non-stationary environments is defined as:

$$R(s_t, a_t) = \begin{cases} R_0(s_t, a_t), \ 0 \le t \le T_0 \\ R_1(s_t, a_t), \ T_0 \le t \le T_1 \\ \dots \\ R_i(s_t, a_t), \ T_{i-1} < t \le T_i \\ \dots \end{cases} \tag{2}$$

The objective of an agent is to learn a policy π to maximize the expected discounted return $\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right]$,where γ denotes the discount factor. Q-function (namely, action value function) is proposed[35, 22] to estimate expected discounted return. Given the state-action pair (s_t, a_t), the Q-function is defined as the expected discounted return following the policy and environmental dynamics:

$$Q(s_t, a_t) = \mathbb{E}_{s_j \sim P, a_j \sim \pi(a|s_j)}\left[\sum_{j=t}^{\infty} \gamma^{j-t} R(s_j, a_j)\right] \tag{3}$$

The policy in Q-learning derives from maximizing the expected discounted return, i.e., $\pi = \arg\max_a Q(s, a)$. The Q-function is updated by temporal difference learning:

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

where η is the learning rate and $r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$ is called the temporal difference error (TD-error).

Soft actor-critic (SAC)[8] incorporates entropy maximization to encourage exploration, while maximizing the expected return:

$$J(\pi_\phi) = \mathbb{E}_{s_j \sim P, a_j \sim \pi_\phi}\left[\sum_{j=t}^{\infty} R(s_j, a_j) + \lambda \mathcal{H}(\pi_\phi(a \mid s_j))\right] \tag{5}$$

where $\pi_\phi(\cdot)$ denotes the policy parameterized by neural networks ϕ, $\mathcal{H}(\cdot)$ denotes the entropy measure, and λ denotes the coefficient that determines the importance of entropy relative to reward.

Twin delayed deep deterministic policy gradient (TD3)[6] is an RL algorithm based on Deep deterministic policy gradient (DDPG)[15], which uses the deterministic policy gradient[32] to maximize the expected return:

$$J(\pi_\phi) = \mathbb{E}_{s_t \sim P, a_t = \pi_\phi(s_t)}\left[Q_\varphi(s_t, a_t) \mid_{a_t = \pi_\phi(s_t)}\right] \tag{6}$$

Moreover, TD3 improves DDPG using the Double Q-function[10] and delaying policy updates to alleviate the overestimation of expected returns:

$$Q_\varphi(s_t, \pi_\phi(s_t)) = \min_{i=1,2} Q_{\varphi_i}(s_t, \pi_\phi(s_t)) \tag{7}$$

where φ_1 and φ_2 denote two different Q-functions parameterized by neural networks, respectively. Delaying policy updates is to reduce the frequency of policy updates to improve training stability.

2.2 Discrepancy of Environment (DoE)

Off-policy RL uses temporal difference learning to update the policy and the Q-function, by comparing the estimation error (i.e., TD-error) in the value of the current action-state pair with that of the action-state pair at the next time step. For a certain transition (s_k, a_k, r_k, s_{k+1}), the TD-error is defined as follows:

$$TD(s_k, a_k, r_k, s_{k+1}) = r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k) \tag{8}$$

In stationary environments, the absolute value of TD-error can be used as the priority metric. Higher absolute value of TD-error indicates a larger value estimation error for the action-state pair in the transition, indicating more opportunities for policy improvement. However, after a change in environmental dynamics, transitions collected before the change may have higher priority than those obtained recently. To better illustrate this problem, we first derive the priority of transition (s_k, a_k, r_k, s_{k+1}) in the following form:

$$\begin{aligned} & |TD(s_k, a_k, r_k, s_{k+1})| \\ & = |r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k)| \\ & = \Big| r_k + \gamma \mathbb{E}_{s_j \sim P_i, a_j \sim \pi(a|s_j)}\left[\sum_{j=k+1}^{\infty} \gamma^{j-t-1} R_i(s_j, a_j)\right] \\ & \quad - \mathbb{E}_{s_j \sim P_i, a_j \sim \pi(a|s_j)}\left[\sum_{j=k}^{\infty} \gamma^{j-t} R_i(s_j, a_j)\right] \Big| \\ & = \left| R(s_k, a_k) - \mathbb{E}_{s_k \sim P_i, a_k \sim \pi(a|s_k)}\left[R_i(s_k, a_k)\right] \right| \end{aligned} \tag{9}$$

Note that the Q-function estimates expected returns following the current environmental dynamics $\langle P_i, R_i \rangle$ and the reward function $R(s_k, a_k)$ is given by Eq. 2. Eq. 9 indicates that priority is determined by the reward function and the agent's estimation of rewards following the current state-transition probability function. With the

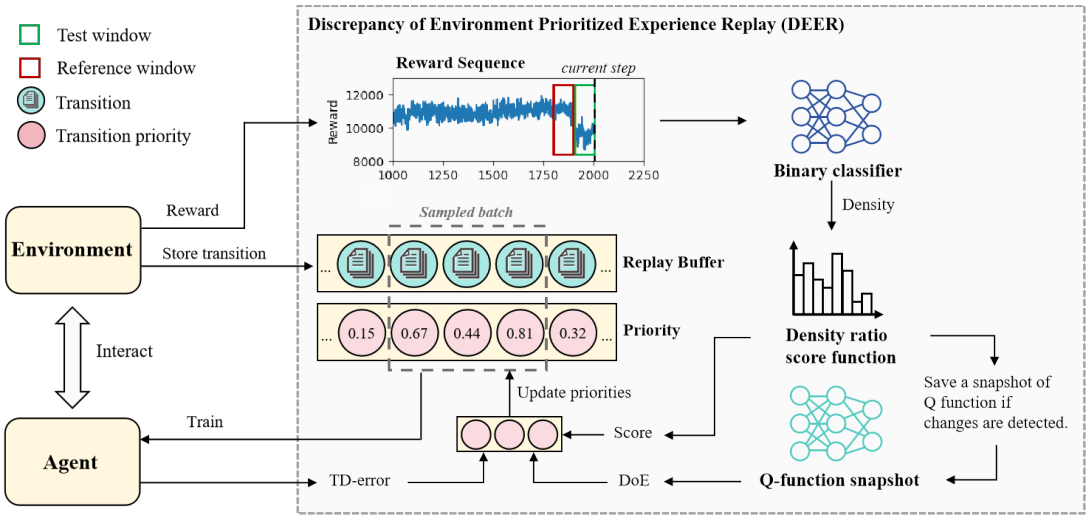


Figure 1. The workflow of DEER. For each reference-test time window pair, density scores are calculated in real-time using a binary classifier and a density score function. Upon detection of a change point, a snapshot of the Q-function is saved to compute the DoE of the transitions. Transitions are prioritized based on a hybrid metric of TD-error and DoE.

Q-function fine-tuned for the new environmental dynamics, transitions collected in the previous environmental dynamics are given higher priority than those in the current dynamics. This is because of the greater disparity between the previous reward function and the current reward function. This results in higher-priority transitions representing outdated experiences, which in turn prevents the agent from adapting to the new environmental dynamics.

To address the problem, we propose a metric called "Discrepancy of Environment (DoE)", which quantifies the impact of environmental dynamics on the value estimation of action-state pair in the transition. We define the DoE for transition (s_k, a_k, r_k, s_{k+1}) as the difference in the Q-function between the pre- and post-change environmental dynamics for the state-action pair (s_k, a_k):

$$\begin{aligned} &DoE\left(s_k, a_k\right) \\ &= \left|Q_i\left(s_k, a_k\right) - Q_{i-1}\left(s_k, a_k\right)\right| \\ &= \left|\mathbb{E}_{s_j \sim P_i, a_j \sim \pi\left(a \mid s_j\right)}\left[\sum_{j=k}^{\infty} \gamma^{k-t} R_i\left(s_j, a_j\right)\right]\right. \\ &\left.-\mathbb{E}_{s_j \sim P_{i-1}, a_j \sim \pi'\left(a \mid s_j\right)}\left[\sum_{j=k}^{\infty} \gamma^{j-t} R_{i-1}\left(s_j, a_j\right)\right]\right| \end{aligned} \tag{10}$$

where $Q_{i-1}(\cdot)$ denotes the Q-function at time step T_{i-1}, i.e., the latest Q-function learned by the agent in the pre-change environmental dynamics $\langle P_{i-1}, R_{i-1}\rangle$ and $Q_i(\cdot)$ denotes the current Q-function in the post-change environmental dynamics $\langle P_i, R_i\rangle$. As shown in Figure 2, DoE measures the bias of the Q-function caused by the environmental dynamics change. High DoE transitions in the new environmental dynamics can guide the agent to fine-tune the Q-function, thereby improving the policy to adapt to the changes. Low DoE transitions in the old environmental dynamics can help the agent retain old information which is still applicable in the new environment.

2.3 Monitoring Changes in Environmental Dynamics

The calculation of DoE relies on the Q-function of the change point. Since the environment dynamics are implicitly modeled in the state-transition probability function, we monitor the environmental dynamics through environment-agent interaction trajectories. Let $\tau = [s_0, a_0, s_1, a_1, \ldots, s_i, a_i, \ldots]$ be the trajectory generated by the agent following the policy $\pi(a \mid s)$. The probability density of trajectory is jointly determined by the policy and the environment dynamics:

$$P(\tau) = P(s_0) \prod_{t=0}^{\infty} \pi(a_t \mid s_t) P(s_{t+1} \mid s_t, a_t) \tag{11}$$

where $P(s_0)$ denotes the initial state distribution, i.e., the probability density that trajectory τ starts from state s_0. We note that when the initial state density and the policy remain constant, a change in the environment dynamics implies a change in the trajectory distribution. Let us define $\mathbf{r}_\tau = [r_0, r_1, \ldots, r_i, \ldots]$ as the reward sequence obtained by the agent following the trajectory τ, where $r_i = R(s_i, a_i)$. The probability density of the reward sequence $P(\mathbf{r}_\tau)$ is also determined by Eq. 11. This is because the reward is generated by the reward function and is independent of the state-transition probability function. Based on the above, we approximate the change of environmental dynamics by analyzing changes in the reward sequence.

We use density-ratio estimation (DRE)[17, 39] based on binary classifier[11] on reward sequence to detect change points. DRE allows for online computation, which is suitable for off-policy RL with online transition collection[17, 13]. We use two adjacent time windows to generate reference and test sets from the reward sequence and set a binary label $l \in \{0, 1\}$ for them. Specifically, let $\widetilde{\mathbf{r}}_{rf} = \{(\mathbf{r}_{i:j}, l) \mid t - 2m + 1 \leq i < j \leq t - m, l = 0\}$ and $\widetilde{\mathbf{r}}_{te} = \{(\mathbf{r}_{i:j}, l) \mid t - m < i < j \leq t, l = 1\}$ be the reference and test sets at time step t, respectively, where $\mathbf{r}_{i:j} = [r_i, r_{i+1}, \ldots, r_j]$ and m denotes the window size. We use a binary classifier to estimate the probability density of the reference and test sets:

$$P(\mathbf{r} \mid f(\mathbf{r}) = l) = \begin{cases} P_{rf}(\mathbf{r}) & \text{if } l = 0 \\ P_{te}(\mathbf{r}) & \text{if } l = 1 \end{cases} \tag{12}$$

where $P_{rf}(\cdot)$ and $P_{te}(\cdot)$ denote the density functions of the reference and test sets, respectively, and $f(\cdot)$ denotes the binary classifier. We use a neural network as a binary classifier and calculate the loss using the following cross-entropy function:

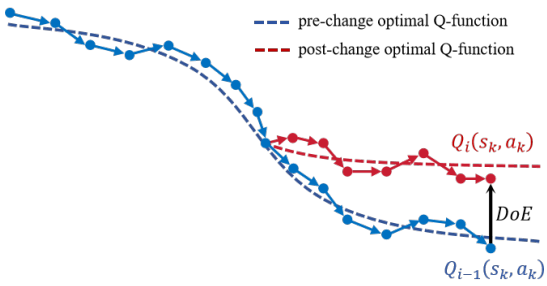


Figure 2. Illustration of fine-tuning the Q-function with the high DoE transition. Training on high DoE transitions enable the agent to fine-tune the Q-function towards to the post-change optimal Q-function, rather than the pre-change one.

$$\mathcal{L}(f) = -\frac{1}{n_{rf}} \sum_{\mathbf{r} \in \widetilde{\mathbf{r}}_{rf}} \log\left(1 - f(\mathbf{r})\right) - \frac{1}{n_{te}} \sum_{\mathbf{r} \in \widetilde{\mathbf{r}}_{te}} \log f(\mathbf{r}) \quad (13)$$

where n_{rf} and n_{te} denote the number of samples in the reference and test sets, respectively. In practice, we partition the reward sequence within the window into equidistant segments based on the number of samples to generate reference or test sets.

The change point is detected by the density ratio score function. When the density ratio score function $S(\widetilde{\mathbf{r}}_{te}, \widetilde{\mathbf{r}}_{rf}) \geq \mu$, where μ is the detection threshold, we consider that the change point occurs at time step $t - m$. We use the Jensen-Shannon divergence[36] as the density ratio score function:

$$\begin{aligned} S(\widetilde{\mathbf{r}}_{te}, \widetilde{\mathbf{r}}_{rf}) = &\log 2 + \frac{1}{2n_{te}} \sum_{\mathbf{r} \in \widetilde{\mathbf{r}}_{te}} \log P(\mathbf{r} \mid f(\mathbf{r}) = 1) \\ &+ \frac{1}{2n_{rf}} \sum_{\mathbf{r} \in \widetilde{\mathbf{r}}_{rf}} \log P(\mathbf{r} \mid f(\mathbf{r}) = 0) \end{aligned} \quad (14)$$

The derivation of Eq. 14 is detailed in Appendix A.

2.4 Discrepancy of Environment Prioritized Experience Replay (DEER)

The inherent characteristics of non-stationary environments necessitate distinct approaches to calculating the importance of a sample before and after a change. When no change point is detected, DEER uses the absolute value of the TD error as a priority. In the following, we will specifically analyze how DEER calculates the priority for pre- and post-change transitions after a change point is detected at time step T_{i-1}, respectively.

For pre-change transitions, we prioritize those with small DoE. This is because a low DoE indicates that the transition is less affected by changes in environmental dynamics, implying its continued relevance in the new environment. The priority of a pre-change transition is calculated as:

$$p_k = 2sig\left(-\left|DoE\left(s_k, a_k\right)\right|\right) \quad (15)$$

where p_k denotes the priority of the transition (s_k, a_k, r_k, s_{k+1}) collected at time step $k\,(k \leq T_{i-1})$ and $sig(\cdot)$ denotes the sigmoid function to normalize the priority.

For post-change transitions, DEER uses hybrid metric priority sampling based on real-time density scores that integrates both TD-error and DoE. The density ratio score, as defined in Eq.14, measures the magnitude of fluctuations in the reward sequence. A high score indicates that the agent has not yet adapted to new environment. In such cases, we prioritize post-change transitions with high DoE to facilitate rapid adaptation to the new environment; conversely, when the score is low, we prioritize post-change transitions with high TD errors to expedite policy improvement. We use dynamic weighting prioritized sampling, defined as follows:

$$\begin{aligned} p_k = &S(\widetilde{\mathbf{r}}_{te}, \widetilde{\mathbf{r}}_{rf})\left(2sig\left(DoE\left(s_k, a_k\right)\right) - 1\right) + \\ &\left(1 - S(\widetilde{\mathbf{r}}_{te}, \widetilde{\mathbf{r}}_{rf})\right)\left(2sig\left(\left|TD\left(s_k, a_k, r_k, s_{k+1}\right)\right|\right) - 1\right) \end{aligned} \quad (16)$$

where p_k denotes the priority of the transition (s_k, a_k, r_k, s_{k+1}) collected at time step $k\,(k > T_{i-1})$ and $S(\widetilde{\mathbf{r}}_{te}, \widetilde{\mathbf{r}}_{rf})$ denotes the density ratio score at the current time step t. To avoid overfitting caused by frequent replays of high-priority transitions, the probabilities of sampling pre- and post-transitions are normalized as follows:

$$P(k) = \frac{p_k^{\alpha}}{\sum_i p_i^{\alpha}} \quad (17)$$

where $\alpha \in (0, 1]$ is the degree of priority used, and a smaller α indicates that DEER is closer to random uniform sampling. Due to the prioritized sampling changing the distribution on which the Q-function relies for estimating the expected return, we employ importance sampling to correct bias. The weight for updating the Q-function using the transition is as follows:

$$w_k = \frac{1}{\left(N \cdot P(k)\right)^{\beta} \cdot \max_i w_i} \quad (18)$$

where N denotes replay buffer size and $\beta \in (0, 1]$ denotes the bias correction degree. A larger beta indicates more full correction for prioritized sampling, requiring modifications to the Q-function updates in RL. For example, the corresponding Q-function update of the transition in Q-Learning is as follows:

$$\begin{aligned} Q(s_k, a_k) \leftarrow &Q(s_k, a_k) + \\ &w_k \cdot \eta\left[r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k)\right] \end{aligned} \quad (19)$$

The pseudocode for DEER is provided in Appendix B.

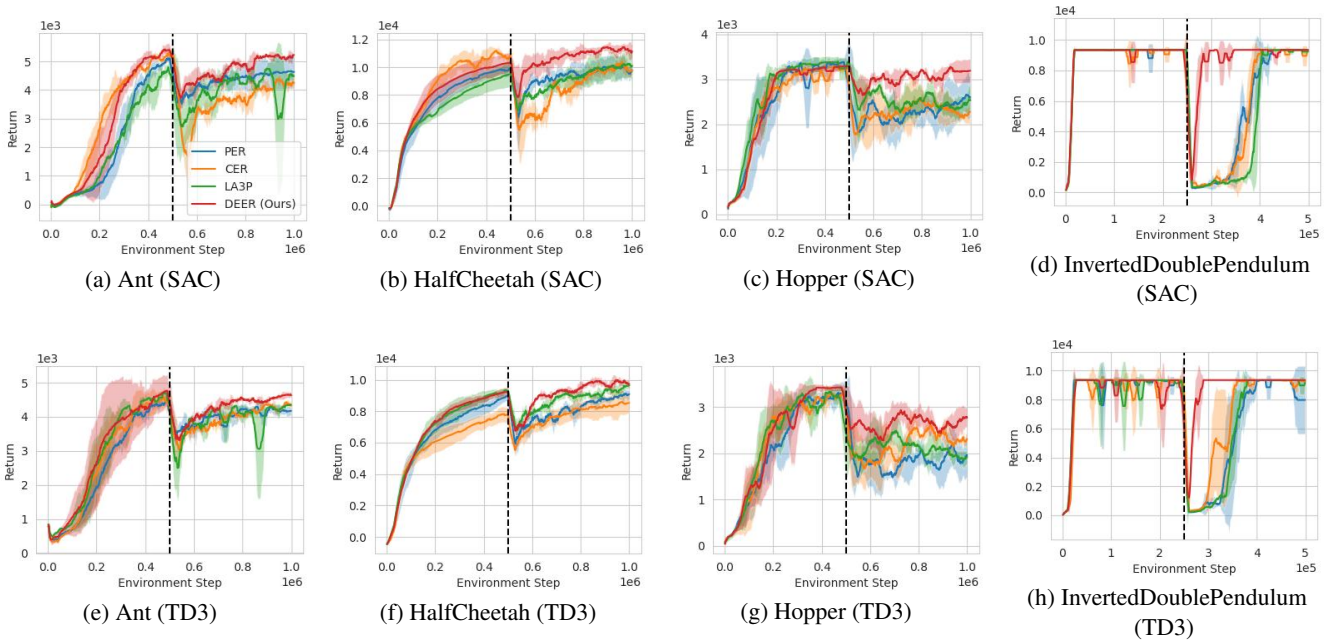
3 Evaluation

3.1 Non-stationary Environment Setups

We evaluated the proposed integration with two popular off-policy algorithms, SAC[8]and TD3[6] , in four extensively studied robotic continuous control tasks provided by the MuJoCo gymnasium[**?**]: Ant-v4, HalfCheetah-v4, Hopper-v4 and InvertedDoublePendulum-v4.

To introduce non-stationarity, we incorporated a series of offsets for friction coefficients and joint damping coefficients. This aims to emulate realistic scenarios where objects interact with varying environmental conditions, such as different surfaces and mediums. The offsets are determined in preliminary experiments and are set to the largest feasible values while ensuring convergence of SAC with uniform sampling. For further details, refer to Appendix C.

For all experiments, the total training step is 5×10^5 (for InvertedDoublePendulum task) or 1×10^6 (for other tasks). Non-stationary factors is introduced when training reaches half of the total steps. All experiments are run with three random seeds, with standard deviation represented as shaded regions. All RL-related networks have



(a) Ant (SAC) (b) HalfCheetah (SAC) (c) Hopper (SAC) (d) InvertedDoublePendulum (SAC)

(e) Ant (TD3) (f) HalfCheetah (TD3) (g) Hopper (TD3) (h) InvertedDoublePendulum (TD3)

Figure 3. Sample efficiency comparison results of DEER and baselines on two algorithms (SAC and TD3) and four tasks. Solid lines represent the average of all three experimental runs, shaded areas indicate standard deviations. The black dashed line indicates the time step at which environmental dynamics change.

	Environments	DEER (Ours)	PER	CER	LA3P
SAC	HalfCheetah	**11217.36 ± 1632.96**	9888.86 ± 1612.75	9989.40 ± 2503.86	10044.00 ± 1648.40
	Ant	**5152.87 ± 763.89**	4655.31 ± 805.98	4208.40 ± 1094.69	4009.42 ± 1692.77
	Hopper	**3186.27 ± 492.45**	2464.25 ± 825.44	2297.34 ± 815.68	2433.01 ± 768.01
	InvertedDoublePendulum	**9295.86 ± 580.11**	9078.97 ± 1241.33	9279.46 ± 710.52	9281.74 ± 577.53
TD3	HalfCheetah	**9769.36 ± 1489.35**	8966.81 ± 999.40	8470.38 ± 980.40	9417.82±1317.21
	Ant	**4553.80 ± 411.47**	4135.54 ± 708.44	4299.45 ± 601.03	4315.81 ± 339.06
	Hopper	**2647.03 ± 762.97**	1967.12 ± 816.78	2306.03 ± 812.98	2056.73 ± 758.13
	InvertedDoublePendulum	**9295.86 ± 580.11**	9078.97 ± 1241.33	9279.46 ± 710.52	9281.74 ± 577.53

Table 1. Convergence performance comparison results of DEER and baselines, presented as the average return ± standard deviation.

[256, 256] hidden nodes. The learning rates for both the actor and critic networks are set to 1×10^{-3}, with a discount factor of 0.99. The replay buffer has a capacity of 1×10^{6}. The training batch size is 256. For DEER, we employed a multi-layer perceptron-based binary classifier with hidden layers of [100, 100] and a maximum iteration of 50. The time window size for environment dynamics detection was set to 500, with 10 samples within the window, each consisting of 50 data points. A complete list of hyperparameters can be found in Appendix D.

3.2 Baselines

We compared DEER with the following three baselines:

(1) Prioritized Experience Replay (PER) [31]: A classic method first adopts prioritized sampling, which has been widely adopted and researched. It aims to enhance sample efficiency by allocating higher priority to transitions with high TD-error.

(2) Combined Experience Replay (CER) [43]: A replay method that emphasizes recent experiences to address the adverse effects of old data. In each sampling step, the most recently added transitions are always included in the batch.

(3) Loss Adjusted Approximate Actor Prioritized Experience Replay (LA3P) [29]: A method for improving PER to mitigate the detrimental impact of high TD-error samples on the policy network. It prioritizes transitions with reliable knowledge from critics (with low TD-error) to train the actor network.

3.3 Comparison Axes

Sample Efficient Sample efficiency refers to the level of performance an agent achieves when using a limited number of samples. It is essential in non-stationary environments as sparse new samples pose a significant obstacle to rapid adaptation. We measure the sample efficiency of algorithms by comparing the return (calculated as the sum of rewards within an episode) relative to the environment steps[40].

Convergence Performance To evaluate the impact of the proposed methods on long-term training, we tested the performance of the algorithms at convergence by calculating the average return of the last 100 training episodes. We omitted InvertedDoublePendulum task because all algorithms converged to optimal performance within 5×10^{5} steps (See Figure 3(d),(h)).

3.4 Comparative Experiments

Figures 3(a)-(d) and (e)-(h) show the episode return curves when various ER methods are integrated with SAC and TD3 algorithms, respectively. It's noticeable that in both scenarios, DEER (indicated by the red line) achieves higher overall returns compared to the baselines. Across various environmental changes, DEER exhibits less reduction in rewards and quicker recovery rates compared to other methods, indicating its efficiency in adapting to dynamic environments.

Table 1 presents the convergence performance. DEER demonstrates better convergence performance within the same number of environmental steps. Across all environments and integrated with the two algorithms, the average return of DEER over the last 100 episodes consistently exceeds those of other methods, with relatively lower standard deviations. This suggests that DEER facilitates agents in swiftly learning the characteristics of the post-change environment, achieving more optimized and stable performance in long-term training.

3.5 Ablation Study

Effectiveness of Each Component We conduct ablation experiments with the following setups in HalfCheetah environment with SAC, to evaluate the effectiveness of our two key designs of hybrid metric sampling (HM) and dynamic weighting (DW) .

- **DEER w/o DW** Density ratio score is solely used for detecting environmental changes, not for calculating dynamic sampling weights. The priority p_k of a post-change transition(s_k, a_k, r_k, s_{k+1}) is given by:

$$\begin{aligned} p_k =& 0.5 \cdot \left(2sig\left(DoE\left(s_k, a_k\right)\right) - 1\right) + \\ & 0.5 \cdot \left(2sig\left(\left|TD\left(s_k, a_k, r_k, s_{k+1}\right)\right|\right) - 1\right) \end{aligned}$$

- **DEER w/o HM**
  - **TD-Error**: Sampling priority solely based on TD-error. The priority of p_k a post-change transition (s_k, a_k, r_k, s_{k+1}) is given by:

$$p_k = 2sig\left(\left|TD\left(s_k, a_k, r_k, s_{k+1}\right)\right|\right) - 1$$

  - **DoE**: Sampling priority relies solely based on DoE. The priority p_k of a post-change transition (s_k, a_k, r_k, s_{k+1}) is given by:

$$p_k = 2sig\left(DoE\left(s_k, a_k\right)\right) - 1$$

The results in Figure 4 demonstrate that employing DW significantly enhances algorithm performance in response to the environmental dynamics change. The performance of HM without DW is secondary. This is attributed to its reliance solely on fixed density ratio scores, which failed to leverage environmental dynamics information. Both single metric methods exhibit weaker adaptability. This emphasizes that combining DoE and TD-error to calculate priority is helpful to improve sample efficiency, with further improvement achieved through the incorporation of dynamic weighting.

Effects of Time Window Size Figure 5 illustrates an example of reward sequence and real-time density ratio scores during SAC + DEER training in the HalfCheetah environment. When the environment changes (indicated by black dashed lines), the score exhibits a noticeable increase, indicating the detection of significant environmental dynamics.

The size of adjacent time windows is a critical hyperparameter when monitoring environmental dynamics, which refers to the length of the reference set and test set in the input binary classifier. As shown in Figure 6, various window sizes in 250, 500, 1000 show their effectiveness in detecting environmental changes. Smaller window size shows stronger responsiveness to changes, with more significant increases in score and lower detection latency. However, smaller window size may be sensitive to subtle environmental changes, leading to unstable weights. Additionally, window size affects computation time, as discussed in detail in Section 3.6.

3.6 Discussion

Sampling Behavior We analyzed the sampling behavior of DEER during training to explain why it outperforms in non-stationary environments. Figure 7 illustrates the priority distribution of samples in the replay buffer at different training stages (pre-change 10k, post-change 10k, post-change 100k, post-change 500k time steps). Overall, the priority of post-change transitions is higher than that of pre-change ones. This disparity is particularly noticeable during the early stages of environmental dynamics change. As training progresses, density ratio scores decrease. The overall sample distribution tends to revert to the distribution before the environmental change. This reflects DEER's strategy of balancing between pre- and post-change experience: leveraging new experience early on for rapid policy adjustment, while selectively reusing old experience to stabilize training.

Algorithm Agnosticism DEER is compatible with most off-policy RL frameworks, applicable to both discrete and continuous control tasks. Our evaluations focus on continuous control tasks due to their generally higher complexity. DEER's core design is DoE, which calculates as the expected return difference of pre- and post-change Q-functions. The algorithmic agnosticism of DEER stems from its reliance solely on the action value function, a fundamental component in nearly all popular off-policy algorithms. Experimental results in Figure 3 demonstrate DEER's enhancement over baselines when integrated with SAC (a-d) and TD3 (e-h).

Time Overhead The time-consuming aspects in DEER mainly involve (1) Sample priority updates: By utilizing SumTree, a binary tree structure storing priorities, we achieve sampling and weight updates within $\mathcal{O}(\log n)$ time complexity. (2) Density ratio scores Computation: The computation time of density ratio score strongly depends on the size of the adjacent time window. A smaller window size leads to more frequent score updates and longer computation time. In practice, we find a window size of $\sim$1000 is sufficient for most tested tasks.

Table 2 presents the runtime of each component, averaged over each training step. The experiments were run on an NVIDIA GeForce RTX 2080 Ti 11GB GPU. It can be observed that neither of the aforementioned components shows significant additional time overhead. Compared to the parameter update time of the RL networks, the additional time overhead is marginal ($\sim$3%).

4 Related Work

Experience Replay Methods Experience replay is a technique in off-policy RL where past experiences are stored and randomly sampled during training to improve sample efficiency and overall learning stability [16]. Random uniform sampling overlooks the significance of each transition[23, 42, 31]. The most common proportional sampling method involves prioritizing transitions based on

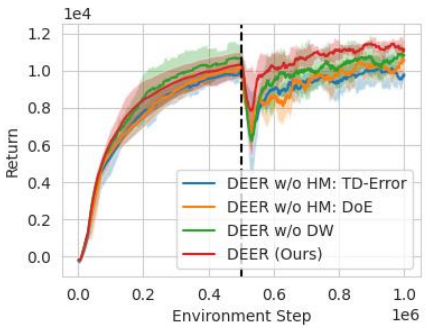


Figure 4. Ablation Result of Hybrid Metric (HM) priority and dynamic weight (DW)

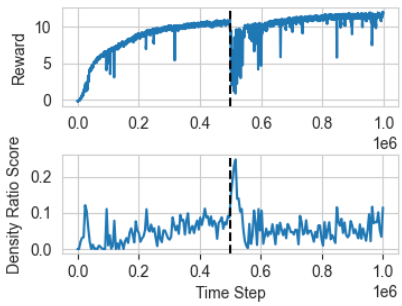


Figure 5. Reward sequence and corresponding real-time density ratio scores

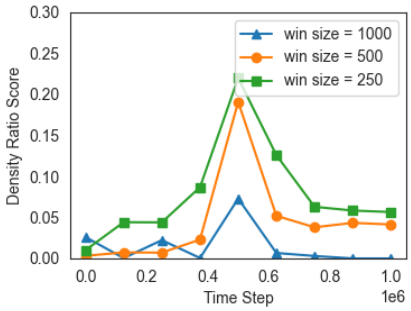
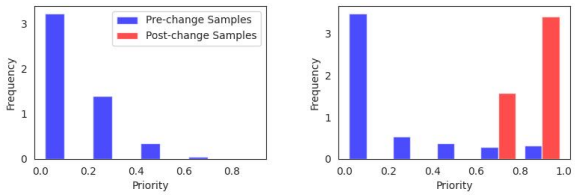


Figure 6. Effects of window sizes on density detection scores



(a) pre-change 10k (b) post-change 10k

(c) post-change 100k (d) post-change 500k

Figure 7. Priority distribution of pre- and post-change samples at different training stages in an experiment of SAC + DEER in HalfCheetah environment. The frequency is normalized as density.

Component		Average runtime (per step)
Network parameter updates		45.82ms
Priority updates		0.51ms
Score computation	window size = 100	0.91ms
	window size = 1000	0.54ms
	window size = 2000	0.42ms

Table 2. Runtime of each component of DEER, averaged over 10^6 steps of SAC + DEER running in HalfCheetah environment.

TD-error, which has demonstrated performance improvements in many RL tasks[34, 14, 7, 3]. Alternative metrics are used as sampling priorities in other studies. Prioritized Sequence Experience Replay (PSER) assigns high priority to significant transitions and increases the priority of previous transitions that resulted in significant transitions[2]. Remember and Forget Experience Replay (ReF-ER) classifies transitions as "near-policy" or "far-policy" based on the ratio of the current policy to the past policy, and only samples near-policy transitions[24]. Attentive Experience Replay (AER) is a sampling method that prioritizes transitions in the replay buffer based on their similarity to the current state[34]. Hindsight Experience Replay (HER)[1] and its variants[20, 44, 30] improve sample efficiency in sparse reward tasks by relabeling transitions to generate sufficient reward feedback.

RL in Non-stationary Environments In non-stationary environments, common reinforcement learning methods include the use of factored MDPs [25, 26, 9], which decompose the non-stationary action-state space into several relatively stationary subspaces. Working Memory Graph (WMG) combines Transformer-based architecture with the decomposition of the observation space to improve the sample efficiency of RL[18]. Cause-Effect Modeling Agent (CEMA) utilizes a structured world model to decompose the latent state space and modeling of sparse interactions[45]. AdaRL decomposes factor representations under the latent states, rewards, and action domains, and adjusts policies using transitions only within the target domain[12]. Some works focus on directly learning latent representations to model non-stationary environments. Lifelong Latent Actor-Critic (LILAC) achieves lifelong learning by jointly learning compact representations of MDPs and maximum entropy policies[38]. Zero-shot adaptation to Unknown Systems (ZeUS) uses the block contextual MDP architecture to directly model non-stationary features[33]. Combined Reinforcement via Abstract Representations (CRAR) builds a compact state abstraction to represent environmental features so that agents can adapt to different environments through transfer learning[5].

5 Conclusion

We propose DEER as a solution to the sample efficiency problem in non-stationary environments. DEER adopts a hybrid metric prioritization approach that dynamically adjusts sampling weights based on changes in environmental dynamics. DEER enhances sample efficiency in non-stationary environments by prioritizing valuable samples measured by DoE within the current environment. Experiment results validate the effectiveness of the DEER across different non-stationary tasks.

References

[1] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

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

[3] X. Cao, H. Wan, Y. Lin, and S. Han. High-value prioritized experience replay for off-policy reinforcement learning. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1510–1514. IEEE, 2019.

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

[5] V. François-Lavet, Y. Bengio, D. Precup, and J. Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3582–3589, 2019.

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

[7] Y. Gao, D. Zhou, Y. Shen, and X. Yang. Dual experience replay-based td3 for single intersection signal control. *The Journal of Supercomputing*, pages 1–22, 2024.

[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*, pages 1861–1870. PMLR, 2018.

[9] A. Hallak, F. Schnitzler, T. Mann, and S. Mannor. Off-policy model-based learning under unknown factored dynamics. In *International Conference on Machine Learning*, pages 711–719. PMLR, 2015.

[10] H. Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[11] J. Hillman and T. D. Hocking. Optimizing roc curves with a sort-based surrogate loss function for binary classification and changepoint detection. *arXiv preprint arXiv:2107.01285*, 2021.

[12] B. Huang, F. Feng, C. Lu, S. Magliacane, and K. Zhang. Adarl: What, where, and how to adapt in transfer reinforcement learning. *arXiv preprint arXiv:2107.02729*, 2021.

[13] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 389–400. SIAM, 2009.

[14] H. Li, X. Qian, and W. Song. Prioritized experience replay based on dynamics priority. *Scientific Reports*, 14(1):6014, 2024.

[15] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[16] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

[17] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[18] R. Loynd, R. Fernandez, A. Celikyilmaz, A. Swaminathan, and M. Hausknecht. Working memory graphs. In *International conference on machine learning*, pages 6404–6414. PMLR, 2020.

[19] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu. A survey on model-based reinforcement learning. *arXiv preprint arXiv:2206.09328*, 2022.

[20] Y. Luo, Y. Wang, K. Dong, Q. Zhang, E. Cheng, Z. Sun, and B. Song. Relay hindsight experience replay: Self-guided continual reinforcement learning for sequential object manipulation tasks with sparse rewards. *Neurocomputing*, 557:126620, 2023.

[21] D. J. Mankowitz, G. Dulac-Arnold, and T. Hester. Challenges of real-world reinforcement learning. 2019.

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

[23] 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*, 518 (7540):529–533, 2015.

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

[25] I. Osband and B. Van Roy. Near-optimal reinforcement learning in factored mdps. *Advances in Neural Information Processing Systems*, 27, 2014.

[26] R. Ouhamma, D. Basu, and O. Maillard. Bilinear exponential family of mdps: frequentist regret bound with tractable exploration & planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9336–9344, 2023.

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

[28] A. Rahimi-Kalahroudi, J. Rajendran, I. Momennejad, H. van Seijen, and S. Chandar. Replay buffer with local forgetting for adapting to local environment changes in deep model-based reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 21–42. PMLR, 2023.

[29] B. Saglam, F. B. Mutlu, D. C. Cicek, and S. S. Kozat. Actor prioritized experience replay. *Journal of Artificial Intelligence Research*, 78:639–672, 2023.

[30] E. Sayar, V. Vintaykin, G. Iacca, and A. Knoll. Hindsight experience replay with evolutionary decision trees for curriculum goal generation. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 3–18. Springer, 2024.

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

[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[33] S. Sodhani, F. Meier, J. Pineau, and A. Zhang. Block contextual mdps for continual learning. In *Learning for Dynamics and Control Conference*, pages 608–623. PMLR, 2022.

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

[35] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[36] D. Tu, Y. Li, and Y. Cai. A new perspective on detecting performance decline: A change-point analysis based on jensen-shannon divergence. *Behavior Research Methods*, 55(3):963–980, 2023.

[37] M. Uehara, C. Shi, and N. Kallus. A review of off-policy evaluation in reinforcement learning. *arXiv preprint arXiv:2212.06355*, 2022.

[38] A. Xie, J. Harrison, and C. Finn. Deep reinforcement learning amidst lifelong non-stationarity. *arXiv preprint arXiv:2006.10701*, 2020.

[39] M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, and M. Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370, 2013.

[40] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[41] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021.

[42] D. Zha, K.-H. Lai, K. Zhou, and X. Hu. Experience replay optimization. *arXiv preprint arXiv:1906.08387*, 2019.

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

[44] X. Zhao, J. Du, and Z. Wang. Hcs-r-her: Hierarchical reinforcement learning based on cross subtasks rainbow hindsight experience replay. *Journal of Computational Science*, 72:102113, 2023.

[45] A. Zholus, Y. Ivchenkov, and A. Panov. Factorized world models for learning causal relationships. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.