Reducing Experience-Level Non-Stationarity in Multi-Agent Reinforcement Learning via Policy Constrained Replay

Manas Shil
Indian Institute of Technology Roorkee
Roorkee, India
manas_s@mfs.iitr.ac.in

G N Pillai
Indian Institute of Technology Roorkee
Roorkee, India
gn.pillai@mfs.iitr.ac.in

Manu Kumar Gupta
Indian Institute of Technology Roorkee
Roorkee, India
manu.gupta@mfs.iitr.ac.in

***Abstract*—Replay buffer sampling in multi-agent off-policy learning suffers from experience-level non-stationarity, where outdated transitions no longer align with the current joint policy, leading to biased and unstable updates. Existing approaches address this issue through importance sampling or recency-based replay; however, importance sampling corrects gradient bias without restricting replay selection, whereas recency-driven methods prioritize freshness over relevance and may discard older transitions that remain informative. To address this gap, we propose policy-consistent prioritized experience replay (PC-PER), a trust-region experience replay framework that constrains replay based on policy consistency. PC-PER assigns each replay transition a policy-drift score measuring the mismatch between the stored behavior policy and the current policy. A PPO-style clipping mechanism then adapts the trust-region radius online, resulting in a policy-consistency window of transitions eligible for replay. Within this window, transitions are prioritized based on their temporal-difference error to emphasize samples with higher learning value. Across a broad and diverse suite of multi-agent continuous-control benchmarks from MuJoCo and PettingZoo, PC-PER consistently outperforms all existing baselines, achieving an average performance improvement of 14.6%.**

***Index Terms*—Multi-agent deep reinforcement learning, experience replay, non-stationarity, off-policy learning.**

I. Introduction

Deep reinforcement learning (DRL) has become a strong approach for sequential decision-making, showing good results across many complex tasks [1]. By combining reinforcement learning with deep networks, it can handle high-dimensional states and continuous control problems, which makes it useful in real-world settings [2]–[7]. Most DRL methods fall into value-based or actor–critic frameworks, and are further divided into on-policy and off-policy learning. Among them, off-policy methods such as DQN [2], DDPG [8], TD3 [9], and soft actor-critic [10] are widely used because they learn more efficiently from data. They reuse past experiences through a replay buffer, which improves sample efficiency and stabilizes training by reducing correlations in the data.

These off-policy frameworks naturally extend to multi-agent settings, where multiple learners interact and adapt concurrently within a shared environment. Many real life problems like network packet routing [11], traffic control [12], smart grid control [13] are required to be modeled as multi-agent systems. When off-policy reinforcement learning is extended to multi-agent settings, the role of experience replay becomes significantly more complex. In multi-agent deep reinforcement learning, each agent learns while interacting with other agents that are also adapting their behaviors over time. As a result, the environment perceived by any individual agent is no longer stationary, in contrast to single-agent learning scenarios. As agents continue to update their policies, experiences collected in earlier stages of training may fail to represent the current joint behavior of all agents. This creates a misalignment between the data stored in the replay buffer and the policies being optimized. Reusing such outdated experiences can bias value estimates and disrupt the learning process, making experience replay a prominent source of non-stationarity in multi-agent systems.

Several methods have been proposed to handle the non-stationarity caused by experience replay in multi-agent learning. One simple approach is to use smaller or more recent replay buffers so that outdated data has less effect [14]. Another direction uses importance sampling to adjust for the mismatch between the behavior policy and the current policy [15]. ReF-ER [16] tackles this issue by controlling how replayed samples align with the current policy, using importance weights along with gradient suppression and KL-based regularization. This idea is later extended to cooperative multi-agent settings under centralized training with decentralized execution [17]. Other works also study replay strategies to improve coordination and sample efficiency [18]–[22]. However, most of them focus on interaction or sharing of experiences, and do not directly handle the non-stationarity at the level of replayed data itself.

While there has been good progress on experience replay in multi-agent RL, most methods rely on simple heuristics like recent sampling, importance weighting, or priority-based selection. These either prefer newer data without checking if it matches the current joint policy, or just reduce the effect of mismatched samples instead of stopping their reuse. As a result, useful experiences may be ignored, while outdated ones can still influence learning, especially when policies change quickly. This shows a clear gap—there is no replay method

that enforces policy consistency while still keeping informative samples, which is important in off-policy multi-agent settings.

Motivated by this gap, we propose PC-PER (Policy-Consistent Prioritized Experience Replay). The key contributions are:

- We propose a *policy-consistency experience replay* framework for off-policy multi-agent reinforcement learning that explicitly regulates the reuse of past transitions based on their compatibility with the current joint policy, addressing experience-level non-stationarity.
- We introduce a trust-region mechanism in policy space to restrict replayed experiences to a consistency window, and adapt the trust-region radius online using a PPO-style controller, eliminating the need for manual threshold tuning.
- Within the policy-consistent replay set, we incorporate temporal-difference–based prioritization to preserve informative transitions with high learning value while maintaining policy validity.
- We empirically demonstrate that the proposed replay strategy improves training stability and sample efficiency across standard multi-agent benchmarks from multi-agent MuJoCo and PettingZoo, achieving an average performance improvement of 14.6% over existing experience replay methods.

To the best of our knowledge, this is the first work to introduce a trust-region–inspired policy consistency window for addressing experience-level non-stationarity and to combine it with temporal-difference–based prioritized experience replay in multi-agent reinforcement learning.

II. Preliminaries

In this section, we briefly discuss the preliminaries that are important for our study.

A. Markov Games

We model multi-agent interaction as a Markov game, which extends MDPs to multiple agents. It consists of a state space $\mathcal{S}$, action spaces $\mathcal{A}_1, \ldots, \mathcal{A}_N$, and a transition function $\mathcal{T}$ that maps the current state and joint actions to a distribution over next states. Each agent i receives a reward r_i that depends on the state and actions of all agents.

In decentralized settings, agents do not see the full state. Instead, each agent observes a local view $\mathcal{O}i$ and follows a stochastic policy π_i to choose actions. The goal of each agent is to maximize its expected discounted return $R_i = \sum t = 0^T \gamma^t r_i^t$.

B. Centralized Training with Decentralized Execution

We adopt the centralized training with decentralized execution (CTDE) paradigm where agents are allowed to leverage centralized information during training, while their policies remain fully decentralized at execution time.

Let $\mathcal{S}$ denote the global state space, $\mathcal{A}_i$ the action space of agent i, and $\mathcal{O}_i$ its local observation space. Each agent selects actions using a stochastic policy $\pi_i : \mathcal{O}_i \rightarrow \mathrm{Dist}(\mathcal{A}_i), \quad a_i \sim \pi_i(a_i \mid o_i)$, which depends only on its private observation $o_i \in \mathcal{O}_i$ when interacting with the environment. During training, however, the value function of agent i is learned using centralized information. Specifically, the centralized action-value function is defined as

$$Q_i^{\pi}(s, a_1, \ldots, a_N) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i^t \,\middle|\, s_0 = s,\ a_j \sim \pi_j(o_j)\ \forall j\right],$$

where $s \in \mathcal{S}$ represents the full system state, $(a_1, \ldots, a_N)$ denotes the joint action, and $\gamma \in (0, 1]$ is the discount factor. The objective of agent i is to maximize its expected return, $J_i(\pi_i) = \mathbb{E}_{s \sim \rho^{\pi},\, a \sim \pi}\left[Q_i^{\pi}(s, a_1, \ldots, a_N)\right]$, where $\rho^{\pi}(s)$ denotes the discounted state visitation distribution induced by the joint policy π.

C. Experience Replay in CTDE

In the centralized training with decentralized execution (CTDE) framework, experience replay enables sample-efficient off-policy learning by reusing past interactions collected by multiple agents. At each time step, the replay buffer stores transitions of the form $(s_t, a_1^t, \ldots, a_N^t, r_1^t, \ldots, r_N^t, s_{t+1})$. Minibatches sampled from this buffer are used to update centralized value functions, whereas policy updates remain fully decentralized. While experience replay improves data efficiency and reduces sample correlation, its use in CTDE-based multi-agent learning introduces additional challenges. As agents update their policies concurrently, stored transitions may no longer be compatible with the current joint policy, leading to policy mismatch during replay. Without explicitly accounting for this effect, replayed experience can bias value estimation and destabilize training.

III. Policy-Consistent Prioritized Experience Replay (PC-PER)

We propose a generalized replay sampling framework that reduces experience-level non-stationarity in off-policy multi-agent reinforcement learning (MARL). The method operates entirely at the level of experience replay and is independent of the underlying actor–critic algorithm. It can therefore be directly integrated with commonly used CTDE-based MARL methods such as MASAC (multi agent version of [10]), MADDPG [23], and MATD3 [24] without modifying their learning objectives or update rules.

A. Replay Buffer with Policy Snapshot

Each stored transition additionally records the policy outputs used at the time of data collection. Specifically, a transition in the replay buffer is represented as

$$\tau_t = \left(s_t, \mathbf{o}_t, \mathbf{a}_t, \mathbf{r}_t, s_{t+1}, \boldsymbol{\psi}_t^{\text{old}}\right),$$

where $\mathbf{o}_t = (o_t^1, \ldots, o_t^N)$ denotes the joint observations of all agents, $\mathbf{a}_t = (a_t^1, \ldots, a_t^N)$ the joint action, and $\mathbf{r}_t = (r_t^1, \ldots, r_t^N)$ the corresponding reward vector. The term $\boldsymbol{\psi}_t^{\text{old}} = (\psi_1^{\text{old}}, \ldots, \psi_N^{\text{old}})$ stores the policy outputs of all agents at the time the transition was generated. For stochastic policies, ψ_i^{old} corresponds to the action distribution parameters $\pi_{\theta_i^t}$ (e.g.,

logits or mean and variance), while for deterministic policies it corresponds to the executed action μ_{θ_i}. Retaining these quantities enables explicit and efficient measurement of policy drift between the behavior policies used during data collection and the current policies at replay time.

B. Unified Policy Drift Metric

For a given transition τ, the per-agent policy drift is defined as

a) Stochastic policies:

$$d_i(\tau) = D_{\mathrm{KL}}\Big(\pi_{\theta_i^{\text{old}}}(\cdot \mid o^i) \,\Big\|\, \pi_{\theta_i^t}(\cdot \mid o^i)\Big).$$

b) Deterministic policies:

$$d_i(\tau) = \Big\|\mu_{\theta_i^{\text{old}}}(o^i) - \mu_{\theta_i^t}(o^i)\Big\|_2 .$$

where μ_{θ_i} denotes a deterministic actor. This definition accommodates both entropy-regularized and deterministic policy gradient methods. We aggregate drift across agents to obtain a joint consistency score: $C(\tau) = \frac{1}{N}\sum_{i=1}^{N} d_i(\tau)$. This scalar quantity measures the degree of policy mismatch associated with replaying transition τ.

Definition 1 (ζ-Consistent Experience Replay)**.** *Consider a cooperative multi-agent reinforcement learning system with N agents trained under the CTDE paradigm. Let $C(\tau)$ denote the aggregated policy drift associated with a replay transition τ.*

A transition τ is said to be ζ-consistent *at training iteration t if $C(\tau) \le \zeta_t$.*

A replay sampling procedure is called **ζ-consistent** *at iteration t if all transitions replayed at that iteration satisfy the above condition.*

Definition 2 (ζ-Stationary Experience Replay)**.** *If the replay sampling procedure is ζ-consistent with respect to ζ_t, then the learning updates induced by replay are said to be* ζ-stationary. *In this case, the distribution of replayed experience deviates from the current joint policy by at most ζ_t in policy space.*

Furthermore, if the per-agent policy drifts satisfy $d_i(\tau) \le \zeta_i, \quad \forall i \in \{1, \dots, N\}$, then the induced joint replay process is ζ-stationary with $\zeta = \frac{1}{N}\sum_{i=1}^{N} \zeta_i$.

C. Trust-Region Consistency Window

To reduce experience-level non-stationarity in off-policy multi-agent learning, we restrict replay to transitions whose associated policy drift lies within a bounded region in policy space. Concretely, we define a consistency window

$$\mathcal{W}_t = \{\tau \in \mathcal{B} \,:\, C(\tau) \le \zeta_t\},$$

where ζ_t denotes the trust-region radius at training iteration t. This window contains all replay transitions that are ζ_t-consistent with respect to the current joint policy. Only transitions within $\mathcal{W}_t$ are considered eligible for replay, ensuring that learning updates operate within an approximately ζ_t-stationary regime.

1) Adaptive Trust-Region Radius: Rather than selecting the trust-region radius ζ_t manually, we adjust it online using a simple controller inspired by trust-region methods such as proximal policy optimization [25]. Let

$$\bar{C}_t = \mathrm{median}\{C(\tau) : \tau \in \mathcal{B}\},$$

denote the median policy drift measured across the replay buffer (or a uniformly sampled subset for efficiency). The median is chosen because of its robustness towards outliers [26], [27].

Given a target drift scale ζ_{target}, the trust-region radius is updated according to

$$\zeta_{t+1} = \begin{cases} \zeta_t \cdot \kappa, & \bar{C}_t > 1.5\,\zeta_{\text{target}}, \\ \zeta_t / \kappa, & \bar{C}_t < 0.5\,\zeta_{\text{target}}, \\ \zeta_t, & \text{otherwise}, \end{cases}$$

where $\kappa > 1$ is a small multiplicative factor. This adaptive mechanism expands the consistency window when policy drift is large and contracts it as policies stabilize, allowing the replay process to balance data availability and policy consistency without requiring hand-tuned thresholds.

2) Sensitivity and sample availability discussion.: A key concern with trust-region–based replay is that if the consistency range is too tight, very few samples get reused, which can hurt diversity and slow learning. To handle this, the framework adjusts the trust-region radius based on how the policy is changing. The points below explain how this controller adapts under different policy dynamics.

- **Large policy drift.** When policies change quickly, most stored samples show high drift. The controller increases ζ_t, which widens the window $\mathcal{W}_t$ so that enough transitions remain usable and training does not stall.
- **Stable policies.** As updates become smaller, drift reduces. The controller then decreases ζ_t, tightening the window and focusing replay on samples that better match the current policy.
- **Adaptive replay.** The window is not fixed. It adjusts automatically with policy behavior—expanding when more data is needed and shrinking when consistency becomes more important.

Overall, the adaptive trust-region mechanism balances replay diversity and policy consistency throughout training, making the replay process robust to the choice of drift scale and eliminating the need for manual threshold tuning or additional safeguards.

D. TD-Error–Based Prioritization Within the Trust Region

While the trust-region window ensures policy consistency, it does not distinguish between informative and uninformative transitions. To address this, we prioritize transitions within $\mathcal{W}_t$ based on their temporal-difference (TD) error.

For each agent i, the TD error associated with transition τ is $\delta_i(\tau) = \left|r^i + \gamma Q_i^{\text{targ}}(s', \mathbf{a}') - Q_i(s, \mathbf{a})\right|$. We define the joint TD error as $\delta(\tau) = \frac{1}{N}\sum_{i=1}^{N} \delta_i(\tau)$.

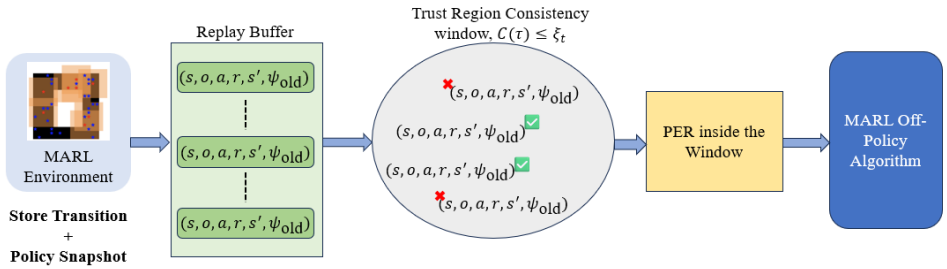

Fig. 1: Overview of the proposed Policy-Consistent Prioritized Experience Replay (PC-PER) framework, illustrating policy-consistent replay with TD-error-based prioritization.

Algorithm 1 Policy-Consistent Prioritized Experience Replay (PC-PER)

Require: Replay buffer $\mathcal{B}$ storing policy snapshots
Require: Target drift ζ_{target}, scaling factor κ
while training **do**
Compute policy drift:
$C(\tau) = \frac{1}{N} \sum_{i=1}^{N} d_i(\tau)$
Adapt trust-region radius (PPO-style):

$$\zeta_{t+1} = \begin{cases} \kappa\, \zeta_t, & \bar{C}_t > 1.5\, \zeta_{\text{target}}, \\ \zeta_t / \kappa, & \bar{C}_t < 0.5\, \zeta_{\text{target}}, \\ \zeta_t, & \text{otherwise} \end{cases}$$

Select policy-consistent replay window:
$\mathcal{W}_t = \{\tau \in \mathcal{B} : C(\tau) \leq \zeta_t\}$
Compute TD-error–based priority:
$p(\tau) = (\delta(\tau) + \varepsilon)^{\alpha}$
Sample minibatch using:
$P(\tau) = \frac{p(\tau)}{\sum_{\tau' \in \mathcal{W}_t} p(\tau')}$
Update the underlying off-policy MARL learner
end while

The priority assigned to transition τ is $p(\tau) = (\delta(\tau) + \varepsilon)^{\alpha}$, where $\alpha \geq 0$ controls the strength of prioritization and $\varepsilon > 0$ ensures numerical stability. Transitions are sampled according to

$$P(\tau) = \frac{p(\tau)}{\sum_{\tau' \in \mathcal{W}_t} p(\tau')}.$$

The proposed framework separates experience usefulness from simple recency by enforcing a policy-space constraint on replayed transitions and applying prioritization only within this consistent set. By selecting transitions that remain aligned with the current joint policy and emphasizing those with strong learning signals, PC-PER reduces experience-level non-stationarity without discarding informative past interactions.

PC-PER differs from importance sampling and recency-based methods by filtering data before replay instead of adjusting it afterward. It selects only those transitions that lie within a bounded region of the current policy, separating policy consistency from transition importance. This way, older but still relevant samples can be reused, without favoring only recent data or relying on corrective weights. In this setup, each transition stores the policy used during collection. During replay, the policy drift $C(\tau)$ is computed, and only samples within the trust-region threshold ($C(\tau) \leq \zeta_t$) are considered. These selected transitions are then prioritized using TD error. Fig. 1 shows the overall idea, and Algorithm 1 provides the pseudo-code.

IV. Experiment Evaluations

We evaluate PC-PER on cooperative multi-agent continuous-control tasks from MuJoCo and PettingZoo. It is compared with Importance Sampling and ReF-ER under the same architectures and hyperparameters for fairness. Performance is measured using mean episode return and standard deviation over multiple seeds, capturing both performance and stability. We use MASAC as the base framework under the CTDE setting. Both actor and critic have two fully connected layers of 256 units each, initialized with Xavier initialization. Target networks are copied from the main networks. Training uses the Adam optimizer with a learning rate of 0.001. The replay buffer size is 5×10^6, with a batch size of 128. Target critics are updated using soft updates with coefficient 0.005. All hyperparameters are fixed across experiments. For PC-PER, we set $\zeta_{\text{target}} = 0.04$ and $\kappa = 3$, which worked well in practice. An initial exploration of 5000 steps is used to fill the buffer before training begins.

A. Environments

To test how general the method is, we run experiments on five cooperative multi-agent continuous-control tasks with different dynamics and coordination needs. This includes three MuJoCo environments—6-Agent HalfCheetah, 3-Agent Hopper, and 3-Agent Pusher—covering different levels of locomotion difficulty and interaction. We also use two PettingZoo tasks: Simple Spread with three agents for basic coordination, and Multiwalker with four agents, which is more challenging and requires tighter coordination and stability. Details of observation and action spaces are omitted due to space limits and can be found in [28], [29].

B. Results and Discussion

Fig. 2 illustrates the learning behavior of the proposed PC-PER method in comparison with several commonly used replay strategies, namely Importance Sampling, ReF-ER, and uniform sampling. Each curve represents the average episodic

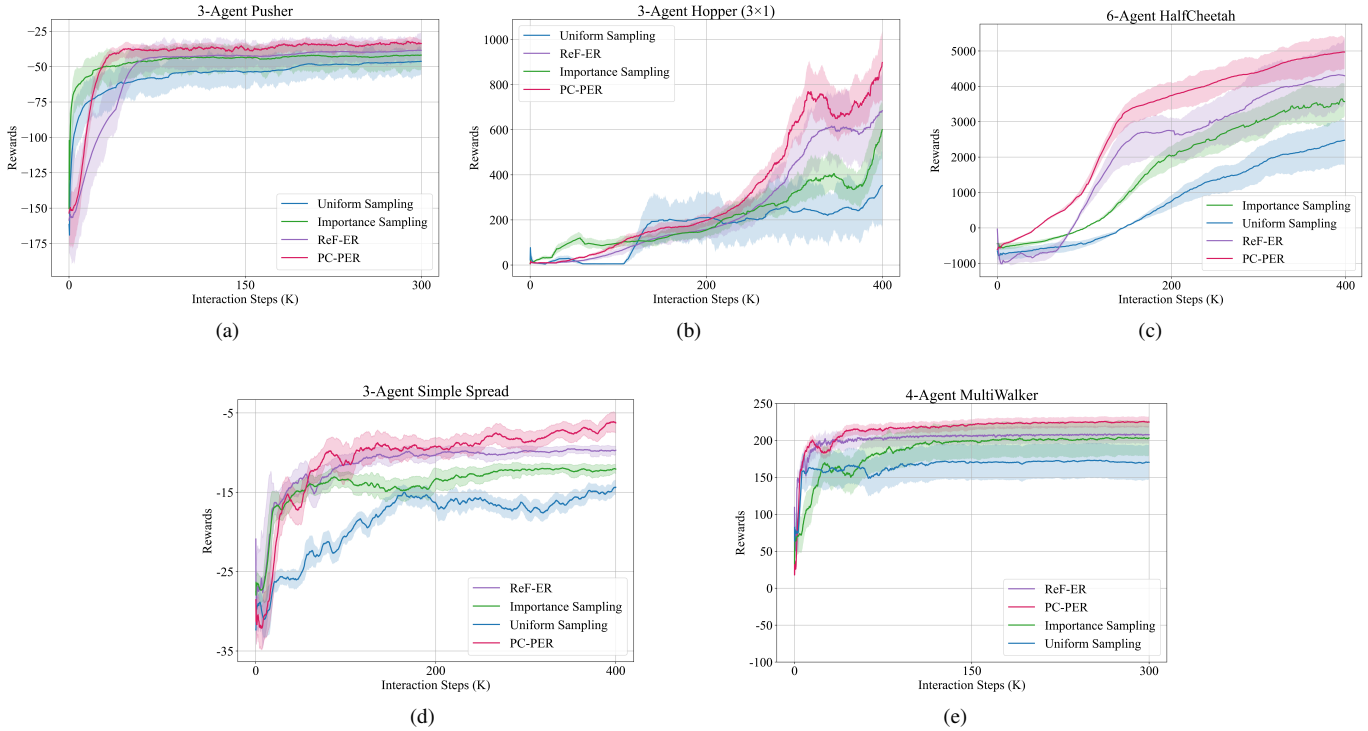

Fig. 2: Overall performance comparison across environments. (a) pusher, (b) hopper, (c) halfcheetah, (d)simple spread, and (e) multiwalker

TABLE I: Performance comparison of different replay strategies across multi-agent environments.

Environment	PC-PER	ReF-ER	Importance Sampling	Uniform Sampling
Pusher (3 agents)	**-30.4** $\pm$ 5.5	-34.9 ± 6.1	-35.6 ± 6	43 ± 9.1
HalfCheetah (6 agents)	**4900** $\pm$ 711	4200 ± 809	3600 ± 721	2750 ± 783
Hopper (3 agents)	**882** $\pm$ 130	742 ± 300	695 ± 200	390 ± 250
Simple Spread (3 agents)	**-8.4** $\pm$ 2.1	-10.1 ± 2.4	-12.3 ± 2.3	-14.4 ± 4
Multiwalker (3 agents)	**221** $\pm$ 16	205 ± 40	201 ± 43	168 ± 51

return over multiple independent runs, while the shaded regions indicate variability across random seeds. Across all considered environments, PC-PER demonstrates a clear advantage over the baselines.

Overall, PC-PER achieves an average performance gain of 14.6% relative to the strongest competing method. This improvement highlights the importance of jointly considering policy consistency and learning signal strength when prioritizing replayed experience. By restricting replay to policy-consistent transitions and further emphasizing samples with high temporal-difference error, PC-PER is able to make more effective use of stored data without introducing additional heuristics or manual tuning. The final performance statistics reported in Table. 1 further support these observations and confirm the consistent benefits of the proposed replay strategy across different multi-agent scenarios. Training was run for 300K timesteps in environments that stabilized early (e.g., Pusher and Multiwalker) and for 400K timesteps in the remaining tasks to ensure clearer evaluation.

C. Computational Cost Analysis

PC-PER adds only a small overhead compared to standard off-policy MARL training. Computing policy drift for each transition scales linearly with the number of agents, i.e., $\mathcal{O}(N)$. The median drift used to update ζ_t can be estimated from a small random subset of the replay buffer, since it reflects the typical stored behavior. Building the consistency window involves linear filtering, $\mathcal{O}(|\mathcal{B}|)$ in the worst case, but this can be spread across training steps. Prioritization based on TD error is the same as standard PER and does not change the asymptotic cost of critic updates. Similar replay-related costs also appear in ReF-ER and importance-sampling methods, which require buffer-wide operations or per-sample weighting.

D. Ablation Study

To understand the role of each part of PC-PER, we compare it with simpler versions. We test PC-only (policy-consistent replay without TD prioritization), PER-only (TD prioritization without policy filtering), and uniform sampling as a baseline. As seen in Fig. 3, PC-only clearly outperforms PER-only,

showing that maintaining policy consistency during replay is important for handling non-stationarity in multi-agent settings. Adding TD-based prioritization on top of this (full PC-PER) gives further gains. In short, consistency helps stabilize learning, and prioritization improves efficiency.

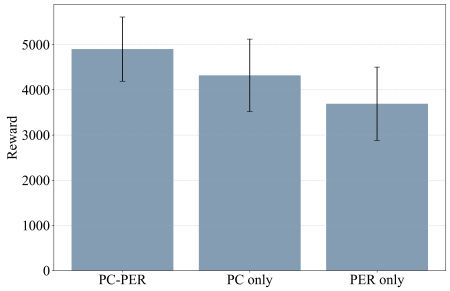

Fig. 3: Ablation study on HalfCheetah

V. Conclusion

In this paper, we introduced policy-constrained prioritized experience replay (PC-PER), a replay-based framework aimed at reducing experience-level non-stationarity in off-policy multi-agent reinforcement learning. PC-PER constrains replayed transitions using a policy-consistency measure and combines this constraint with temporal-difference–based prioritization. Unlike prior replay strategies that rely primarily on recency or age-based sampling, the proposed approach retains older transitions that remain compatible with the current policies and continue to provide useful learning signals. This design avoids bias toward either recent experience or purely correction-based replay. Experimental results on several multi-agent benchmarks demonstrate that PC-PER consistently outperforms existing replay methods,achieving an average performance improvement of 14.6%. Overall, PC-PER provides a simple and effective mechanism for managing non-stationary experience in multi-agent learning while introducing only minimal additional computational overhead. A promising direction for future work is to extend PC-PER with agent-specific or state-dependent consistency thresholds, allowing different agents or regions of the state space to tolerate different levels of policy drifts.

VI. Acknowledgment

Manas Shil acknowledges the Ministry of Education, Government of India, for support through the Prime Minister's Research Fellowship (PMRF ID: 2803613), which contributed to the completion of this research.

References

[1] R. S. Sutton and A. G. Barto, An Reinforcement Learning: Introduction. *Mit Press*, 2012.

[2] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] Y. Huang, R. Cao, and A. Rahmani, "Reinforcement learning for sepsis treat ment: A continuous action space solution," in *Machine Learning for Healthcare Conference*, PMLR, 2022.

[5] Y.-R. Chen, A. Rezapour, W.-G. Tzeng, and S.-C. Tsai, "RL-routing: An SDN routing algorithm based on deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 3185–3199, 2020.

[6] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," *Neurocomputing*, vol. 497, pp. 64–75, 2022.

[7] K. Tan, Q. Wu, W. Wang, and J. Yu, "Reinforcement learning-based and energy efficiency-aware power management for RISC-V platforms," in *2025 International Joint Conference on Neural Networks (IJCNN)*, 2025, pp. 1–8.

[8] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *arXiv [cs.LG]*, 2015.

[9] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *arXiv [cs.AI]*, 2018.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv [cs.LG]*, 2018.

[11] G. Bono, J. S. Dibangoye, O. Simonin, L. Matignon, and F. Pereyron, "Solving multi-agent routing problems using deep attention mechanisms," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 12, pp. 7804–7813, 2021.

[12] T. Chu, J. Wang, L. Codeca, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, 2020.

[13] M. Shil, G. N. Pillai, and M. K. Gupta, "Multi-agent deep reinforcement learning based secondary voltage control of inverter-based AC microgrids," in *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society,* 2024, pp. 1–6.

[14] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," *arXiv [cs.MA]*, 2017.

[15] J. Foerster et al., "Stabilising experience replay for deep multi-agent reinforcement learning," in *International conference on machine learning*, pp. 1146-1155. PMLR, 2017.

[16] G. Novati and P. Koumoutsakos, "Remember and forget for experience replay," in *International Conference on Machine Learning*, pp. 4851-4860. PMLR, 2018.

[17] P. Weber, D. Wälchli, M. Zeqiri, and P. Koumoutsakos, "Remember and Forget Experience Replay for Multi-Agent Reinforcement Learning," *arXiv [cs.LG]*, 2022.

[18] Y. Wang and Z. Zhang, "Experience selection in multi-agent deep reinforcement learning," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019.

[19] Y. Mei, H. Zhou, T. Lan, G. Venkataramani, and P. Wei, "MAC-PO: Multi-agent experience replay via collective priority optimization," *arXiv [cs.LG]*, 2023.

[20] S. Ahilan and P. Dayan, "Correcting experience replay for multi-agent communication," *arXiv [cs.LG]*, 2020.

[21] X. Hu, J. Zhao, W. Zhou, R. Feng, and H. Li, "DIFFER: Decomposing individual reward for fair experience replay in multi-agent reinforcement learning," *arXiv [cs.MA]*, pp. 75048–75066, 2023.

[22] M. Gerstgrasser, T. Danino, and S. Keren, "Selectively sharing experiences improves multi-agent reinforcement learning," *arXiv [cs.LG]*, 2023.

[23] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*,30, 2017.

[24] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," *arXiv [cs.LG]*, 2019.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv [cs.LG]*, 2017.

[26] J. Rousseeuw Peter, Robust Regression and Outlier Detection. John Wiley & Sons, 2005.

[27] F. R. Hampel, "A General Qualitative Definition of Robustness," *Ann. Math. Stat.*, vol. 42, no. 6, pp. 1887–1896, 1971.

[28] "Gymnasium-robotics documentation," *Farama.org.* [Online]. Available: https://robotics.farama.org/envs/MaMuJoCo/index.html.

[29] "PettingZoo documentation," *Farama.org.* [Online]. Available: https://pettingzoo.farama.org/index.html.