# Revisiting Experience Replay in Non-Stationary Environments

Derek Li
University of Alberta
Edmonton, Canada
xzli@ualberta.ca

Andrew Jacobsen
University of Alberta
Edmonton, Canada
ajjacobs@ualberta.ca

Adam White
University of Alberta
Edmonton, Canada
amw8@ualberta.ca

## Abstract

Experience replay (ER) has been widely used to improve the sample efficiency of deep Reinforcement Learning (RL) agents. In this paper, we investigate its effectiveness in non-stationary domains under the online RL setting, where an agent must balance exploration and exploitation and perform updates on every timestep where reward maximization is measured over the a lifelong learning process. We compare several variants of ER with an online agent using the same function approximation and found that existing ER algorithms on top of an ANN-based learner struggle in simple non-stationary domains. We show that a recency-biased correction in minibatch sampling could greatly mitigate the negative impact from non-stationary data in ER.

***Keywords:*** reinforcement learning, experience replay, non-uniform sampling, distribution shift, non-stationary markov decision process

## 1 INTRODUCTION

Reinforcement Learning (RL) applications in real life such as robotics, traffic control, and autonomous driving frequently encounter non-stationary sensory data. Non-stationary domains arise often in practice because we typically assume the agent has fewer parameters than the number of states, and thus an agent cannot perfectly represent the underlying MDP, requiring that the agent must generalize and balance how accurate its estimates are across states. As a result, the world may appear non-stationary to the agent [1].

Experience replay (ER) is becoming widely adopted in deep RL systems [2]. Sampling real transitions from an experience replay avoids model errors that could potentially bias agent updates. In the meantime, an agent also needs to be robust to changes that are not captured by the agent's state representation. Directly applying an optimized policy learned from offline data could very likely lead to failure eventually. Therefore, online learning is important for an agent to adapt to non-stationarity due to a changing environment, rewards, and policies that can result in a faulty model. Pure online learning does not take full advantage of past experiences [3] especially when the underlying process is generated from a highly temporally dependent distribution, like most settings in RL. Thus, learning from experience replay is crucial in creating sample efficient RL agents in generalized environments [4].

Even though ER helps break the correlation in time series data and makes the learning process more sample efficient [5], it does not necessarily hold true when the underlying Markovian process is non-stationary, since an experience replay would allow the problematic non-stationary data to affect the learning process much longer through being repeatedly sampled from the buffer. Mitigating the interference caused by non-stationary data becomes a bigger problem in replay memories. This work attempts to answer the following questions. Are there situations where uniform sampled ER underperforms conventional online learning in a non-stationary setting? How can we improve an RL agent using experience replay to better adapt to non-stationary data? To address these questions,

- We provide an example of the detrimental effect a large experience replay buffer could have on an agent's learning process in a non-stationary Blocking Maze environment, compared to an online SARSA agent baseline, in adapting to environmental changes.
- We propose a simple and effective recency-biased correction scheme that improves the performance and robustness in learning from non-stationary data while using experience replay.
- We show empirically that applying this correction to the state-of-the-art non-uniform sampling method can significantly improve performance against non-stationarities caused by the environment and bad exploration.
- We provide insights in understanding replay capacity and reduce the need for its careful tuning in practice.

Non-uniform sampling has been studied in supervised and active learning settings [6]. It is proposed to actively sample based on the uncertainty of data in [7–11]. Other works suggest to sample based on factors such as informativeness and diversity [12, 13]. In addition, Zhao et al. (2015) studied stochastic optimization with importance sampling and has shown that a sampling distribution proportional to the gradient norm could reduce the variance and improve the convergence rate under suitable conditions [14]. Katharopoulos et al. (2018) proved that the gradient norm of parameters in a DNN is found to be upper bounded by the gradient norm

with respect to the pre-activation output of the last layer, and can be approximated using the online loss [15].

## 2 PROBLEM SETTING

We consider the online control problem under a batch reinforcement learning setting in a non-stationary Markov decision process (MDP). In this setting, an RL agent interacts with a non-stationary MDP while maintaining an experience replay buffer $\mathcal{H}$ that contains a sliding window of state action transition tuples $O := (S, A, R, S', A')$. The behavior of the agent is determined by its policy $\pi : S \rightarrow A$. The value function $Q^\pi$ is used to evaluate the goodness of a policy, which is defined as the expected discounted return starting in $s$ and following $\pi$ thereafter. The value function is learned incrementally through semi-gradient TD learning updates [16]. The goal of the agent is to maximize the cumulative reward during its lifetime, measured by discounted return. $Q^\pi$ is approximated using a neural network $Q_\theta$ parameterized by $\theta$, similar to the DQN[17]. State observations $s \in S \subset \mathbb{R}^d$ are used as input of the value function network.

Two key design decisions when working with an ER buffer include which experience to add to the memory, and which are sampled at each iteration. This work only considers the latter and assumes the lifecycle of an experience follows the first in first out (FIFO) principle in the replay memory. Note that experience replay does not constitute a complete RL algorithm without combining with a learning agent. In our work, we use the combination of an experience replay and a Sarsa base agent to focus on a simple online TD control setting with neural network as function approximation [16].

## 3 PRIORITISED REPLAY

The idea of prioritizing experiences in RL using sample gradients was originally proposed in prioritized sweeping [18]. It was later adopted and extended in model-based planning and the Dyna architecture [16, 19, 20]. In addition, TD-errors have also been used to drive exploration [21]. The current state-of-the-art Prioritized Experience Replay algorithm (PER) uses an experience's TD error to approximate an agent's surprise and reweigh experiences for sampling [2]. This design is shown to contribute to the most improvement in Rainbow DQN [22]. The intuition is that learning from transitions with value estimates most different from the current belief will lead to the most performance gain. The sampling probability of an experience $i$ is

$$P(i) = \frac{\delta(i)^\alpha}{\sum_k \delta(k)^\alpha} \tag{1}$$

for some $\alpha > 0$. When $\alpha$ is 0, it is equivalent to uniform sampling. Previous works suggest that both surprise and on-policyness play an important factor in developing a sample efficient replay strategy. Combined Experience Replay (CER) was proposed to mitigate the performance degradation from having a large replay buffer and help the learning process to stay close to being on-policy, by appending the online example to each sampled batch [23]. Fedus et al. (2020) showed from experiments that increasing replay size and reducing the oldest policy improved agent performance [24]. However, it comes at the cost of a more restricted training regime. When a larger replay is used, it is required to train less frequently in proportion to maintain the "freshness" of data in the buffer, in order to be close to learning on-policy. This in effect reduces policy changes and diversity of stored data, which slows down the learning process and prevents the agent from taking full advantage of a larger buffer in complex and high-dimensional environments where large replays are required. Though it was effective in stationary settings such as the Atari Arcade learning Environment [25], it might hurt the learning and adaptability of an agent in non-stationary domains.

In principle, we would like the sampling distribution over replay to match the expected future state occupancy of the agent. But in reality, making a direct prediction is computationally expensive and sometimes intractable. The idea of CER stands because the future state occupancy can be roughly approximated by the most recent trajectories in expectation.

However, CER has some limitations. First, its effect on the gradient deteriorates as the batch size increases, and thus is not scalable with compute. Second, it does not explicitly consider non-stationarity in the data and thereby it's not practical in many real-world scenarios. Our recency-biased sampling strategy can be broadly viewed as an extension to CER. Similarly, we also study the large replay capacity issues in this work, but under a non-stationary setting.

To solve the aforementioned problems, we inject a recency bias into sampling weights as demonstrated in Figure 1. Our method extends the idea behind CER and discounts the priorities of state transition tuples based on their recency adjusted by how frequent they are added to the replay buffer. First, prioritizing more recent experiences in training helps the agent to learn from on-policy data. Second, adapting the priorities of experiences based on their underlying states rather than individual transition tuples helps maintain a good sample coverage over the underlying state space. In comparison, uniform sampling naturally favors transitions that are more abundant in the buffer. We define $N_{(s,a)}$ to be the number of times an agent state $(s, a)$ has been sampled and a scaling factor $\tau$. Then, the sampling probability of a state-action pair is given by:

$$P(s, a) \propto exp(-\tau \cdot N(s, a)) \tag{2}$$

As we can see in Equation 2, this approach unifies and generalizes uniform sampling and online methods. When $\tau = 0$, our approach samples uniformly; and when $\tau \rightarrow \infty$ it effectively trains the agent with online data only. This allows our agent to take the best of both worlds and be able to adapt

to non-stationary data in the buffer in a principled way. In addition, our recency-biased sampling strategy decouples the inherent timescales of the underlying data generating process from an agent's learning process by building the bias towards less frequently sampled agent states rather than temporally more recent transitions naively. This helps to ensure enough diversity in sampled batches and enables stable and efficient learning.

Ideally, we would like to discount sampling weights of a state instead of an individual experience, especially when one is very common. Only discounting weights of sampled transitions stored in a buffer doesn't account for the frequency of experience tuples generated by the same states. However, it is difficult to accurately represent this probability distribution in practice, since experiences stored in the replay are not ordered by their underlying states and neither are the states directly accessible except in a tabular setting. Therefore, we approximate through memoizing a cosine similarity matrix among all experiences in the buffer. Upon a sampled batch, we perform the following steps: 1) calculate and update the similarity scores among states in the batch; 2) update $N$ by adding the sum of similarities per experience as a result of the current batch. In essence, we discount the sampling weights of experience tuples in the buffer based on how similar they are to the sampled transitions to approximate a sampling distribution biased towards more recent experiences with consideration of their rareness.

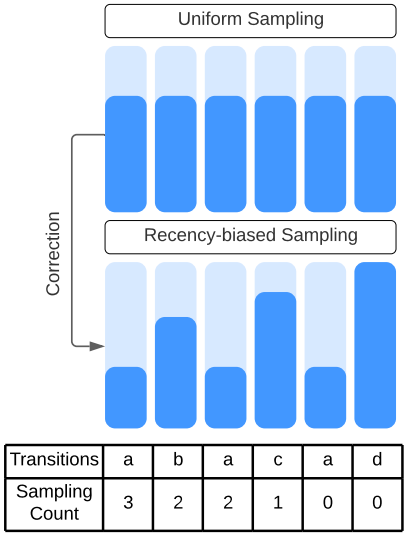



**Figure 1.** Recency-biased Sampling

As far as we are aware, our work is the first to advocate prioritizing over state-action pairs than individual transitions when sampling from an ER. But it comes at the cost of down-weighing the samples based on their similarities to the state transitions in the sampled batch of an update, and memoizing a similarity matrix among state representations across the replay. This costs us an additional memory of $O(|\mathcal{H}|^2)$. We also make the assumption that state representations change slowly during the lifetime of an agent [26], and only updating the similarities among sampled data in an ad-hoc fashion is sufficient.

Memoization is used to make the computation iterative and cheap, $O(n^2 d^2)$ for a batch of size $n$, while approximating the ground truth similarities among states as accurately as possible. In practice, memoization and incremental batch updates gives our approach a nice synergy when combined with other non-uniform sampling such as PER, since the similarities among more frequently sampled transitions, which are usually more important ones to learn from, are quickly propagated whenever a new sample is added. This in turn helps us to approximate the desired state distribution more accurately when sampling for the next batch. Moreover, since PER could only update priorities for the sampled batch, our method also downweighs similar state transitions in the buffer, it serves as a partial remedy to reduce wasteful sampling as a result of out-of-date priorities in PER.

Additionally, unlike CER, the effectiveness of our method does not degrade as the batch size grows with more computation. In fact, our method merely tries to approximate the on-policy distribution using experiences sampled from a replay buffer with enough diversity and support. Therefore, it can be used to correct any off-policy sampling strategies, such as PER. In the experiment section, we will compare a PER variant with recency-biased sampling against other popular ER strategies.

## 4 EXPERIMENTS

We use experiments in two simulation domains to demonstrate the pitfall of Experience Replay methods and the improvement from applying a recency-biased adjustment to the sampling probabilities. In the first experiment, we showcase that an online Sarsa agent using function approximation is able to achieve more cumulative rewards in a non-stationary environment than state-of-the-art experience replay methods given a large replay capacity. We then go on to show that recency-biased correction can help mitigate the negative impact of non-stationarity in the experience replay and adapt more successfully to a sudden change in the environment — a common scenario in real-world problems. In the second experiment, we show that non-stationarity could still be introduced into the buffer even if the environment is fully deterministic and rewards are stationary. Policy changes can

[1]There are two ways to derive the TD-error in this step, using $A_t$ directly from the sampled transition tuple v.s. evaluating $A_t \sim \pi_\theta$ at the training step. We chose the first approach for stability and to most clearly highlight the impact of non-stationary data.

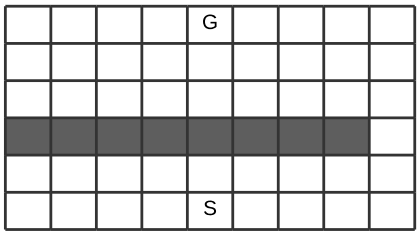
**Algorithm 1** Sarsa-NN with recency-adjusted PER

**Input:** minibatch $k$, step-size $\eta$, replay period K and size N, exponents $\alpha$ and $\beta$, scaling factor $\tau$, budget $T$.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe $S_0$ and choose $A_0 \sim \pi_\theta(S_0)$
**for** $t = 1$ to $T$ **do**
    Observe $S_t, R_t, \gamma_t$
    Choose action $A_t \sim \pi_\theta(S_t)$
    Store experience $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t, A_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$, and recency-biased weight $g_t = 0$
    **for** $j = 1$ *to* $k$ **do**
        Sample an experience, $j \sim p(j) = e^{-\tau \cdot g_j} \cdot p_j^\alpha / \sum_i e^{-\tau \cdot g_j} \cdot p_i^\alpha$
        Compute importance-sampling weight, $w_j = (N \cdot P(j))^{-\beta}) / \max_i w_i$
        Compute Sarsa TD-error[1], $\delta_j = R_j + \gamma_j Q_{target}(S_j, Q(s_j, a_j)) - Q(S_{j-1}, A_{j-1})$
        Update transition priority $p_j \leftarrow |\delta_j|$
        **for** transition $m \in (S_j, A_j)$ **do**
            Update recency-biased weights $g_m \leftarrow g_m + 1$
        **end for**
        Accumulate weight change, $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \Delta_\theta Q(S_{j-1}, A_{j-1})$
    **end for**
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
**end for**

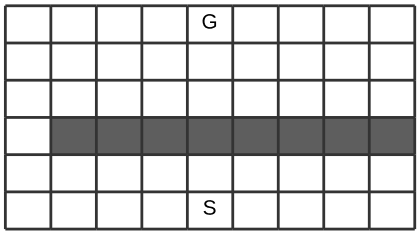
also negatively impact the stationarity of the data and severely hurt the performance of all ER-based methods, and recency-biased sampling can be used to improve the performance significantly.

In both experiments, we compare an online Sarsa agent with function approximation against four agents using experience replay with different sampling strategies. The value functions of all five agents are parameterized using the same neural network architecture (2-layer FCN with ReLU activations). It is trained by minimizing the $L_2$ norm of the TD error $\delta(o)$ using the Adam optimizer [27] and a constant step size, where $o \in O$ is sampled from the buffer. Target network is removed to reduce confounding factors and for simplicity. It is noted in our early exploratory study that using a target network doesn't improve the performance of the agents, possibly due to little interference effect is involved as the representation is one-hot and environments are deterministic.

**Blocking Maze.** The first environment is a variant of the Blocking Maze [16], which is a $6 \times 9$ gridworld with obstacles in Figure 2. The starting state is [0,4] and the goal state is [5,4], which are separated by a wall in the 3rd row. The non-stationarity is introduced by an environment change during the middle of a run. Initially, the right path to the goal is open. Once the agent is able to achieve a stable policy, at the




**(a)** Episode 1~150

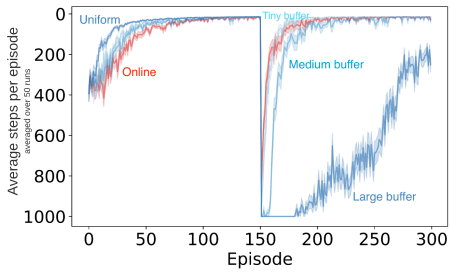



**(b)** Episode 151~300

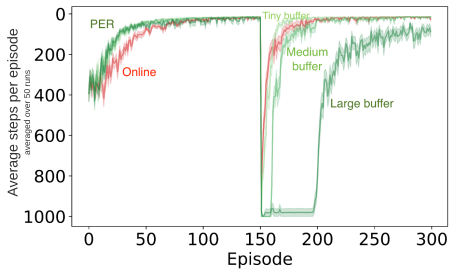
**Figure 2.** Blocking Maze

end of the 150th episode, the obstacles are shifted to the right to block the right path and leave a left path open instead. The agent needs to adapt to this change in the environment and learn to reach the same goal through a new path. The discount rate of the environment $\gamma$ is 0.9. The agent is able to move one step towards any one of the 4 directions per time step and receives a reward of 1 at the goal state and 0 everywhere else. An episode would be terminated if the agent hasn't reached the goal in 1000 time steps. For exploration, the agents use an epsilon-greedy policy. To improve online behavior and reduce variance across different runs, we apply simulated annealing to $\epsilon$. At the start of a run, we set the $\epsilon$ to 1 and is geometrically reduced by a factor of 0.99 after each episode. We averaged the steps taken to reach the goal over an agent's lifetime. The experiment contains 50 independent runs of 300 episodes for each agent.

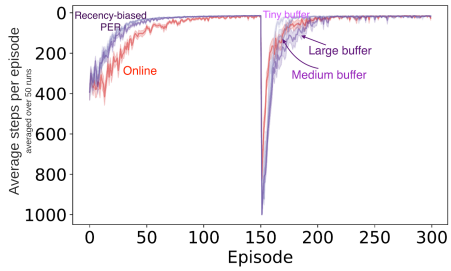
Results indicate that recency-biased PER can be just as competitive as an online agent in adapting to non-stationary environments such as the Blocking Maze and is more robust to larger replays and distributional shifts in the data, compared to other ER methods in Figure 3. Figure 4 contains step size sensitivity plots of agents under different replay capacities. The performance of state-of-the-art experience replay methods including Uniform, CER, and PER degrades as replay capacity increases. The online agent doesn't suffer from this problem but its sample efficiency can be improved in a non-stationary problem. In comparison, by applying the recency-biased correction to a PER agent, we are able to mitigate the curse of replay capacity significantly while




**(a)** Online v.s. Uniform



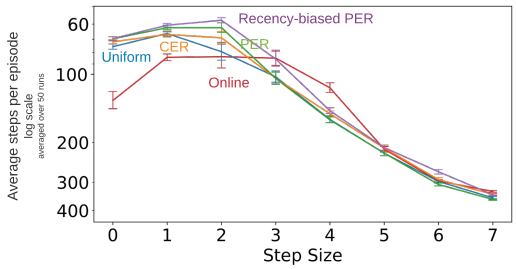

**(b)** Online v.s. PER



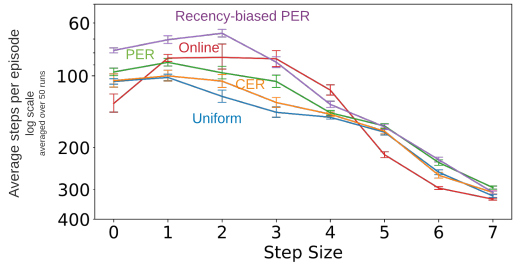

**(c)** Online v.s. PER w/ recency bias

**Figure 3.** Comparison of learning curves under tiny, medium, and large buffer sizes (50, 10k, 50k) in Blocking Maze. Each curve corresponds to the number of steps it takes for an agent to reach the goal at each episode. A time-out termination of 1000 steps is in effect. When the original path is blocked after the 150th episode, it takes online Sarsa much less time to adapt to the new path, compared to all ER-based agents, except for the recency-biased PER variant. Among experience replay methods, both Uniform and PER suffer from significant performance degradation when replay capacity increases. In contrast, the recency-biased PER agent is impacted minimally.

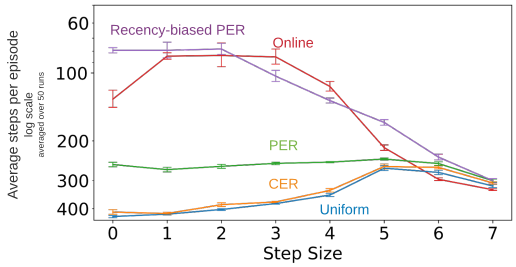
retaining the most sample efficiency we could enjoy from




**(a)** Tiny buffer




**(b)** Medium buffer




**(c)** Large buffer

**Figure 4.** Learning rate sensitivity plots under the tiny buffer (a), medium buffer (b), and large buffer (c) setting. The step sizes on the x-axis are ordered in {0.000078125, 0.00015625, 0.0003125, 0.000625, 0.00125, 0.0025, 0.005, 0.01}. The y-axis displays the average steps per episode in log scale and standard errors estimated from 50 independent runs.

using an experience replay. In particular, the Recency-biased PER outperforms other ER-based methods and is generally more robust to changes in step size under all 3 scenarios. It also achieves better sample efficiency in Figure 4a and 4b and stays competitive in 4c against the online baseline.

**Random Walk.** Our second experiment is conducted in a 100-state Random Walk environment shown in Figure 5. An agent starts in the 50st state and its goal is the 100th state. The agent is given the option to move left or right at each time step. The environment is deterministic and if an agent chooses to step left at the leftmost state, it will remain in the same state. Similar to our first experiment, we

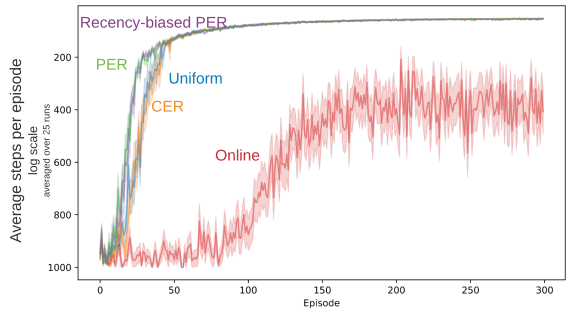
use a sparse rewards setting, and the agent is only given a reward of 1 once it reaches the goal state. The difficulty of the environment is the classic exploration-exploitation trade-off. The agent needs to learn the optimal policy that always moves right; but in order to do that, it needs to first fully explore the environment under a predefined exploration policy, where each state further away from the starting point is exponentially harder to reach, and at the same time avoid getting trapped in the left half of the state space where no reward will be given. The non-stationarity in this experiment is induced by policy changes. We compare the agents under two epsilon-greedy policies, a constant one, versus one using simulated annealing. The discount rate $\gamma$ is 0.99. A timeout termination of 1000 steps is also applied. The experiment consists of 25 runs. Each run uses a different random seed and contains 300 episodes. The agents' performance is compared using the time steps taken to reach the goal averaged across episodes.


**Figure 5.** 100-State Random Walk

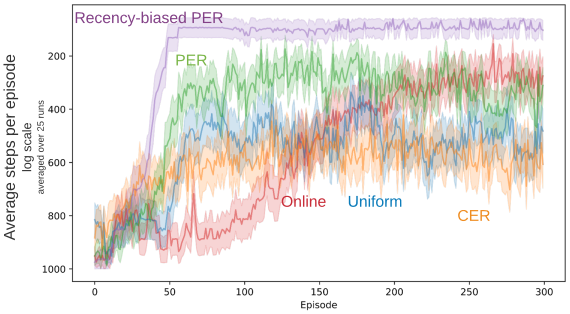
In the first experiment shown in Figure 6a and 7a, where the agent starts out with an exploratory policy ($\epsilon = 0$) at the beginning of a run and gradually decayed $\epsilon$ over time, we expect its behavior policy to change slowly and the resulting distribution shift among collected data in the buffer stays benign. In this scenario, we observed that ER methods achieve superior sample efficiency compared to the online agent just as expected. Among different sampling strategies, PER and its recency-biased variant have better performance in the initial phase but all four ER methods are able to attain the same convergence. in comparison, the second experiment in Figure 6b and 7b tests the scenario where a constant $\epsilon$ is used. The consequence is that an agent would under-explore the environment and its learned policy would change in a non-stationary fashion. As a result, the distribution of states collected in the buffer would shift rather dramatically at the beginning of the learning process, so that we can observe if recency-biased correction could help combat this problem. The results suggest that the Recency-biased PER agent is indeed able to mitigate the impact from such type of non-stationarity since it outperforms both other ER-based and online agents both in sample efficiency and convergent performance by a large margin.

## 5 DISCUSSION

In this work, we aimed to demonstrate that increasing replay capacity can be detrimental in the non-stationary setting. In Figure 3, we've shown the comparisons of an online and



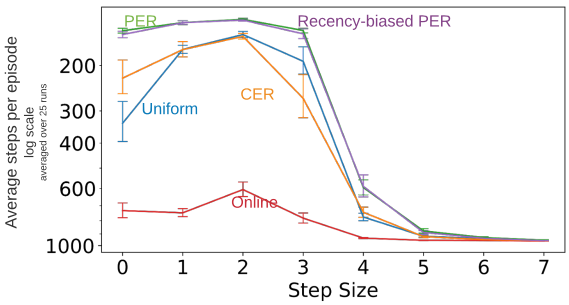

**(a)** Exponentially-decayed $\epsilon$



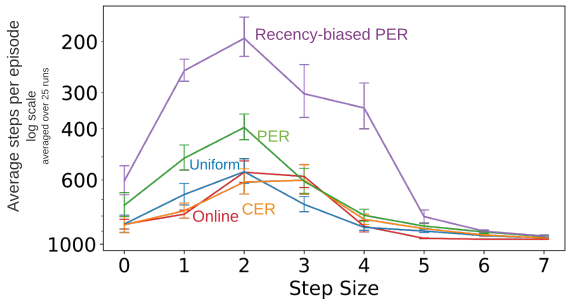

**(b)** $\epsilon = 0.1$

**Figure 6.** Learning curves of agents under two exploration settings in the 100-state Random Walk. Each curve corresponds to the average number of steps for an agent to reach the goal per episode. A time-out termination of 1000 steps is in effect.

several replay-based Sarsa agents in a Blocking Maze experiment. We observed that when the environment changes, where the original path to the goal is closed and a new one is revealed, conventional sampling strategies without a recency-biased correction all suffer significantly from having a large buffer and underperform an online agent with the same model architecture. The highly non-stationary data causes a larger distributional shift in the buffer, which would make past experiences quickly becoming less relevant. However, a smaller ER buffer would also defeat its whole purpose, which is to reuse as much past experience as we have for better sample efficiency [3]. Furthermore, an ER buffer too small could also potentially break the i.i.d assumption of the training data and lose the convergence guarantee. In practice, when RL practitioners design an experience replay, the common practice is to choose the appropriate replay size based on memory constraints and their understanding of the complexity of the problem being solved. The relationship between replay size and the non-stationarity of the underlying data generating process is often overlooked. Moreover,




**(a)** Exponentially-decayed $\epsilon$




**(b)** $\epsilon = 0.1$

**Figure 7.** Learning rate sensitivity plot for two exploration settings of the 100-state random walk problem. The indexes on the x-axis correspond to step sizes same as in Figure 3 and the replay capacity is 1000. The first plot compares the agents' performance under a nice exploration policy, which gradually decays $\epsilon$ in order to keep smooth changes of policy over time. In this case, both PER and its recency-biased variant are able to outperform other ER-based and online agents. In the second plot, where a constant $\epsilon$ is applied, both CER and Uniform agents are just doing as bad as the online Sarsa agent. PER is doing considerably better, but nowhere near its performance in the benign policy change setting. In contrast, recency-biased adjustment, outperforming other methods by a large margin and is overall more robust to different learning rates, is able to achieve comparable performance to the benign setting.

the degree of non-stationarity is a measure that's hard to quantify and estimate and often influenced by a combination of factors, including the environment, the choice of function approximation, and the exploration-exploitation tradeoff. Therefore, choosing the right replay capacity when designing an RL agent with a neural network function approximation can become tricky, and luckily recency-biased sampling helps alleviate this problem.

Recency-biased sampling could help experience replay adapt to environmental and policy changes. In the experiments, we have shown that adding a recency bias to the sampling weights could notably improve the performance of PER in non-stationary tasks. We have demonstrated its effectiveness in two small domains that simulate two different sources of non-stationarity that could exist in an ER buffer. In both Blocking Maze and the constant epsilon setting of the Random Walk, PER suffers less from non-stationarity compared to uniform sampling and CER. When combined with the recency-biased correction PER is able to minimize the degradation and stay the most competitive in all scenarios.

Using recency-biased correction provides more insights into the effective horizon of the experience replay and hence reduces the need for careful tuning of replay capacity. Zhang & Sutton (2017) suggest that an agent's learning process is heavily influenced by the replay capacity [23]. Either too small or too big could severely hurt its online performance. As we have seen in the previous experiments, recency-biased correction works well both in the small and large buffer settings. More interestingly, the normalized $\tau$ value and min weight parameters of the ER could help us understand if we are using a buffer too small or too big for a task by giving us an estimate of the expected sampling probabilities of transitions in the buffer that decays over time. For example, in the last Random Walk experiment, the optimal $\tau$ value is 0.03125 and the corresponding min weight is 0. Since we sample once per new online experience, and the recency-biased weights follow an exponential distribution, we could derive that the sampling probability relative to the most recent transition would become smaller than 1% after the first 150 transitions. In other words, the most recent 150 experiences take up most of the probability mass of the sampling distribution and are the most useful for the agent to achieve its best performance in this task. This information can help us better choose the appropriate replay capacity without sweeping across a large range of values as sometimes it could be costly or even impossible under a strict memory constraint.

In essence, our method serves as a bridge between learning from all past experiences available v.s. learning in a purely online fashion. This strategy helps an agent adapt to various degrees of non-stationarity by trading off sample efficiency. On one hand, similar to the eligibility trace (ET) in spirit, it weighs experiences with an exponential decay for the learning process and uses a temperature parameter to determine an effective lookback horizon. On the other hand, the differences include, 1) ET considers returns from trajectories and here we only consider individual transitions, 2) our method extends over episodic boundaries to all experiences available even for a distributed replay setting while ET cannot, and more fundamentally, 3) ET directly builds the bias in aggregating the returns and our method embeds the bias indirectly through non-uniform sampling [28]. Nonetheless, they both

serve the purpose of connecting two effective learning methods (e.g. TD(0) and Monte Carlo in ET) and take advantage of the best of both worlds.

## 6 CONCLUSION

In this work, we've shown that conventional experience replay methods suffer from non-stationary data. This results in slower convergence and higher variance compared to an online Sarsa baseline. We demonstrated this problem in a Blocking Maze experiment and proposed a recency-biased approach to re-weigh experiences in a replay buffer. It is shown to significantly improve the performance of a popular ER method, PER, against the detrimental effect of non-stationary data introduced through environmental changes and bad exploration in two simulated environments, especially in a large replay setting. The limitation of our method includes the introduction of an additional hyperparameter, $\tau$, which is used to adapt to different degrees of non-stationarity. Also, we resort to a state similarity matrix to prioritize experiences based on their underlying states, the effectiveness of which is limited by the goodness of this approximation. This suggests a number of exciting extensions for future work. First, it would be very beneficial to further investigate challenges that arise from non-stationary data in larger domains since non-stationarity is present in many applications in the real world. Second, the online adaptation of $\tau$ could lead to a more efficient and robust ER sampling algorithm with recency bias. Third, our experiments hint at the benefit of rethinking experience replay in terms of designing a memory, versus simply a sliding window of recent experience. To this end, better state aggregation, abstraction, and inference techniques are highly desirable so that we could better store, sample, and amortize experiences based on their underlying states, which is similar to how human brains process, learn and forget in changing environments [29, 30].

## References


[1] Richard S Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pages 871–878, 2007.

[2] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[3] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[4] Shimon Whiteson, Brian Tanner, Matthew E Taylor, and Peter Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 120–127. IEEE, 2011.

[5] Guy Bresler, Prateek Jain, Dheeraj Nagaraj, Praneeth Netrapalli, and Xian Wu. Least squares regression with markovian data: Fundamental limits and algorithms. *arXiv preprint arXiv:2006.08916*, 2020.

[6] Burr Settles. Active learning literature survey. 2009.

[7] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[8] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[9] Maya Kabkab, Azadeh Alavi, and Rama Chellappa. Dcnns on a diet: Sampling strategies for reducing the training set size. *arXiv preprint arXiv:1606.04232*, 2016.

[10] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. *arXiv preprint arXiv:1704.07433*, 2017.

[11] Hwanjun Song, Sundong Kim, Minseok Kim, and Jae-Gil Lee. Ada-boundary: accelerating dnn training via adaptive boundary batch selection. *Machine Learning*, 109(9):1837–1853, 2020.

[12] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1137–1144, 2008.

[13] Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

[14] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9. PMLR, 2015.

[15] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[18] David Andre, Nir Friedman, and Ronald Parr. Generalized prioritized sweeping. *Advances in Neural Information Processing Systems*, 10:1001–1007, 1997.

[19] Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*, 2012.

[20] Yangchen Pan, Muhammad Zaheer, Adam White, Andrew Patterson, and Martha White. Organizing experience: a deeper look at replay mechanisms for sample-based planning in continuous state domains. *arXiv preprint arXiv:1806.04624*, 2018.

[21] Adam White, Joseph Modayil, and Richard S Sutton. Surprise and curiosity for big data robotics. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[22] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

[24] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[25] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[26] Wesley Chung, Somjit Nath, Ajin Joseph, and Martha White. Two-timescale networks for nonlinear value function approximation. In *International conference on learning representations*, 2018.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *arXiv preprint arXiv:2007.06049*, 2020.

[29] Lauren Gravitz. The forgotten part of memory. *Nature*, 571(7766):S12–S12, 2019.
[30] Ronald L Davis and Yi Zhong. The biology of forgetting—a perspective. *Neuron*, 95(3):490–503, 2017.
[31] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

# Appendices

## A Parameter Settings

**Environment**. Both the Blocking Maze and the Random Walk are discrete environments that produce one-hot encoding state vectors as observations together with a sparse reward signal and a termination flag at the goal state. The blocking maze environment is a $6 \times 9$ girworld with obstacles as shown in Figure 2. The random walk environment consists of 100 states as in Figure 5, each connecting to its left and right neighbouring states, with the exception of the leftmost state. The start state is the 50th state and the goal state is the 100th state. Its starting state is [0,4] and the goal state is [5,4]. The discount rate of both environments $\gamma$ is 0.9.

**Experiment**. We performed 50 independent runs of experiments in the blocking maze environment and 25 runs in the random walk environment for each agent. Each run consists of 300 episodes and each episode is subject to a timeout termination of 1000 time steps.

**Agent**. The action value function network of the agents is a simple 2-layer fully-connected feedforward neural network (FFNN) architecture across the board. The forward pass of the network takes an input of dimension $d$ and passes through a linear transformation followed by a ReLU activation [31]. The same continues into the second layer with an activation size of $\frac{1}{2}d$ followed by the output layer with size equals to the number of available actions. We applied the same Sarsa base agent in both experiments and the learning algorithms only differ in the way they acquire training data. In particular, the Online agent only uses the current observation and immediate reward to perform the one-step semi-gradient TD update, and Uniform, CER, PER, and Recency-biased PER agents in comparison sample a mini-batch of size 10 from an experience replay buffer. In the blocking maze experiment, we examined 3 replay capacity settings, 50, 10k, and 50k. In the random walk experiment, the replay capacity is fixed at 1000. The agents use an $\epsilon$-greedy exploration policy with an episodic exponential decay of 0.99 except for the constant $\epsilon$ experiment where it was fixed at 0.1. The step size for the value function network is swept over {0.01, 0.005, 0.0025, 0.00125, 0.000625, 0.0003125, 0.00015625, 0.000078125}. For PER, $\alpha$ and $\beta$ are searched within the range of {1.0, 0.5, 0.25, 0.125} and its minimum weight is swept over {0.1, 0.05, 0.025, 0.0125, 0.00625, 0}. In addition, the temperature parameter $\tau$ in Recency-biased PER is searched under the range of {1.0, 0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0}.

## B Results

**Table 1.** Average steps per episode in Blocking Maze

| | Blocking Maze | | | Before environment change | | | After environment change | | |
|---|---|---|---|---|---|---|---|---|---|
| Replay Capacity | 50 | 10k | 50k | 50 | 10k | 50k | 50 | 10k | 50k |
| Online | 81.12 ± 2.84 | 83.61 ± 10.1 | **74.59 ± 2.51** | 102.33 ± 2.41 | 89.22 ± 2.04 | 87.48 ± 2.03 | **59.9 ± 5.32** | **78.0 ± 19.73** | **61.7 ± 4.87** |
| Uniform | 66.04 ± 1.56 | 101.58 ± 3.14 | 347.85 ± 4.95 | 73.2 ± 1.18 | 65.52 ± 1.23 | **53.65 ± 1.76** | **58.88 ± 3.01** | 137.65 ± 6.37 | 642.05 ± 9.95 |
| CER | 66.59 ± 1.7 | 99.89 ± 5.16 | 331.46 ± 5.74 | 74.08 ± 1.19 | 64.19 ± 1.11 | **53.41 ± 3.47** | **59.09 ± 3.3** | 135.58 ± 10.0 | 609.52 ± 13.2 |
| PER | 62.15 ± 1.42 | 87.69 ± 3.22 | 248.43 ± 1.56 | 68.9 ± 1.23 | **54.83 ± 1.04** | **49.29 ± 1.58** | **55.4 ± 3.09** | 120.56 ± 6.36 | 447.56 ± 3.38 |
| Recency-biased PER | **57.71 ± 1.51** | **66.24 ± 2.44** | **78.29 ± 4.5** | **58.95 ± 1.11** | 59.79 ± 1.02 | 65.16 ± 1.43 | **56.46 ± 3.05** | **72.68 ±4.79** | 91.42 ± 8.55 |

**Table 2.** Average steps per episode in Random Walk

| | Exponentially decayed $\epsilon$ | Constant $\epsilon$ |
|---|---|---|
| Online | 605.74 ± 39.51 | 564.21 ± 48.32 |
| Uniform | 151.67 ± 4.11 | 561.97 ± 54.73 |
| CER | 154.02 ± 3.97 | 601.51 ± 70.46 |
| PER | **132.31 ± 1.30** | 395.64 ± 40.38 |
| Recency-biased PER | **133.31 ± 1.33** | **194.93 ± 30.19** |