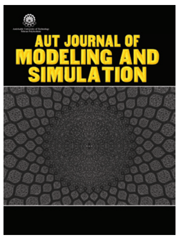
AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 57(1) (2025) 53-62
DOI: 10.22060/miscj.2025.23426.5373



Double Deep Q Network with Adaptive Prioritized Experience Replay

Majid Adibian, Mohammad Mehdi Ebadzadeh*

Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

ABSTRACT: In some deep reinforcement learning models, an experience replay buffer is utilized to address the issue of sequential data dependencies and leverage useful data generated in the past. Then, the prioritized experience replay (PER) method modified sampling from this buffer for training the DQN model, moving away from random selection to choosing each transition in proportion to its temporal difference (TD) error. This method does not use the importance of transitions and the number of times that each transition contributes to model training. We proposed a new prioritization method for calculating the probability of selecting each transition adopted to the importance of that transition and the contribution counter. So in this method, instead of relying solely on the TD error, three additional values, including transition reward, transition counter, and policy probability (RCP values) are incorporated. These three values are obtained for each transition, and after normalization, they are used to calculate the probability of that transition in the sampling from the replay buffer. Experiments conducted on some Atari environments demonstrate that each of these values can significantly improve the episode return compared to the simple prioritization method. Furthermore, three aggregation functions, including min, max, and mean, are proposed to utilize all three RCP values in data prioritization. The results of the experiments indicate that the aggregation function should be determined based on each environment, but the 'mean' aggregation function can be a preferred choice due to its acceptable performance across different environments and the incorporation of all three RCP values.

Review History:

Received: Aug. 08, 2024
Revised: Jul. 08, 2025
Accepted: Aug. 03, 2025
Available Online: Aug. 05, 2025

Keywords:

Deep Reinforcement Learning
Prioritized Experience Replay
Deep Q-Network

1- Introduction

In recent years, deep reinforcement learning (DRL) has garnered significant attention as a powerful framework for training agents to make sequential decisions in complex and dynamic environments. By integrating deep neural networks with reinforcement learning algorithms, DRL has achieved remarkable success across a wide range of applications, including game playing, robotics, and autonomous systems.

One of the foundational models in this field is the Deep Q-Network (DQN) [1], which introduced the use of neural networks to approximate the action-value function. DQN enables agents to estimate the optimal action in a given state by learning a value function Q(s,a), thereby playing a pivotal role in the advancement of deep reinforcement learning.

In online reinforcement learning, agents interact with the environment incrementally, generating a continuous stream of experience data. A key challenge in this setting is the strong temporal correlation between successive data samples, which violates the assumption of independent and identically distributed (i.i.d.) inputs typically required for stable neural network training. Moreover, important past experiences may be rapidly overwritten, limiting opportunities for efficient reuse.

*Corresponding author's email: ebadzadeh@aut.ac.ir

To mitigate these issues, the technique of experience replay was introduced [2], in which transitions collected during interactions are stored in a replay buffer and later sampled randomly for training. This approach decorrelates the data, enhances stability, and improves sample efficiency, particularly in environments where data acquisition is expensive or limited.

Traditional experience replay employs uniform random sampling, treating all stored transitions equally. However, this assumption neglects the fact that not all experiences are equally informative. Some transitions carry richer learning signals than others and can be more beneficial if sampled more frequently.

To address this, the Prioritized Experience Replay (PER) method was proposed as an improvement over uniform sampling [3]. PER assigns higher sampling probabilities to transitions with larger temporal-difference (TD) errors, which reflect a greater discrepancy between predicted and actual values. Transitions with higher TD-errors are considered more surprising and informative, suggesting that revisiting them can yield larger updates and accelerate the learning process.

Although Prioritized Experience Replay (PER) has improved learning by prioritizing transitions with large TD errors, it suffers from several limitations. PER solely relies on the TD error for prioritization, which may overlook other meaningful signals, such as the actual reward received, the frequency with which a transition is replayed, or the significance of an action as indicated by the policy. Moreover, it uses a fixed exponent for prioritization throughout training, which may not be optimal as the learning progresses.

To address these limitations, we propose an Adaptive Prioritized Experience Replay method that augments the traditional PER framework by introducing three additional features for each transition: reward, transition usage counter, and policy probability—collectively termed RCP values. These values are normalized and used to dynamically adjust the prioritization of transitions, leading to more informative sampling and potentially improved learning stability and efficiency. Our method also allows for various aggregation strategies over RCP values, providing flexibility to adapt across different environments.

The next sections first provide a review of related research in the field of experience replay and deep reinforcement learning. Based on insights from previous studies, we then introduce our proposed method. Finally, the effectiveness of the approach is evaluated through experiments on several Atari environments.

2- Related Works

In both human cognition and real-world decision-making, experiences that involve significant outcomes—such as large rewards or noticeable errors—tend to be recalled and processed more frequently [4-7] This observation has inspired the development of mechanisms in reinforcement learning that aim to mimic such prioritization by replaying past transitions based on their potential contribution to learning [8]. Rather than treating all experiences equally, these methods emphasize the most informative ones, thereby enabling agents to focus computational resources on transitions that are expected to yield greater improvements in policy or value estimation.

Building on this intuition, experience replay has become a foundational technique in deep reinforcement learning (DRL), particularly in off-policy settings where data efficiency is critical. By allowing agents to store and reuse past experiences, replay buffers have significantly improved training stability and sample efficiency. One of the most influential advancements in this area is the introduction of prioritized experience replay (PER) [3], which proposes that not all experiences are equally valuable for learning. Instead of sampling transitions uniformly, PER assigns higher sampling probabilities to transitions with larger temporal-difference (TD) errors, under the hypothesis that such transitions contribute more to updating the value function effectively.

This notion has not only improved the performance of standard Deep Q-Networks (DQN) but has also proven effective when scaled to distributed settings, where multiple actors generate experiences concurrently and a centralized learner selectively samples from a shared prioritized buffer [9]. Such distributed architectures have highlighted the scalability and adaptability of PER in high-throughput reinforcement learning environments.

Despite the success of PER in value-based methods like DQN, its direct application to actor-critic architectures—particularly in continuous control tasks—has shown limitations. Empirical and theoretical investigations suggest that high TD-error transitions, which are typically favored by PER, may not always align with the learning needs of the actor component. Specifically, such transitions may introduce instability in policy gradients, leading to divergence from the optimal policy direction [10]. To address this, various adaptations of PER have been proposed for actor-critic settings, including mechanisms that balance the prioritization between actor and critic objectives to ensure more stable and effective policy updates.

Further enhancements to PER have explored augmenting the priority computation with additional features beyond TD-error alone. For instance, several studies propose combining model-free and model-based elements, using predictive signals from learned environment models to enrich the prioritization criteria [11]. This hybrid approach introduces auxiliary indicators that can act as proxies for sample informativeness, leading to more balanced and curriculum-like sampling strategies. By leveraging both learned dynamics and value approximations, such model-augmented techniques aim to improve generalization and accelerate convergence without significantly increasing computational overhead.

To overcome the limitations of relying solely on TD-error, some approaches have proposed correcting or complementing the priority scores using adaptive mechanisms. These include self-adjusting schemes that estimate the discrepancy between actual and stored priorities over time, thereby reducing bias introduced by stale or outdated samples [12]. In parallel, other methods have introduced value-theoretic perspectives, interpreting TD-error as a proxy for the marginal value of each experience. Such interpretations have led to theoretical bounds that relate the priority of a transition to its potential contribution to the cumulative reward, thereby offering more principled ways to guide replay sampling [13]. Together, these efforts address the challenge of maintaining meaningful and up-to-date prioritization throughout training.

Another line of research has focused on incorporating uncertainty into the prioritization process. Traditional PER may inadvertently prioritize transitions with high TD-errors arising from noise rather than from actual learning potential. To mitigate this, uncertainty-aware variants of PER have been proposed, which leverage epistemic uncertainty estimates to distinguish informative transitions from stochastic outliers [14]. These approaches aim to reduce the impact of noisy experiences that could mislead learning. Similarly, uncertainty-based filtering has been employed in continual learning settings to retain only experiences that are both uncertain and diverse, thereby optimizing memory usage and preventing catastrophic forgetting over task sequences [15].

Beyond uncertainty, reward-based signals have also been explored as alternative or complementary cues for prioritization. Drawing inspiration from cognitive neuroscience, some methods incorporate reward prediction error (RPE) as a prioritization metric, arguing that discrepancies between predicted and actual rewards reflect transitions with high learning potential [16]. By integrating RPE into the prioritization scheme—alongside or instead of TD-error—these methods aim to better align sampling with policy improvement objectives, particularly in environments where value estimation may be unreliable or delayed.

The role of PER has also been re-examined in multi-task and multi-agent reinforcement learning contexts, where naive sampling strategies often lead to performance imbalances across tasks or agents. In such settings, task-aware prioritization mechanisms have been developed to dynamically allocate sampling attention to underperforming tasks or agents, thereby promoting balanced learning [17]. Moreover, integrating shared and task-specific feature representations within the replay mechanism further enhances the effectiveness of such prioritization by enabling better generalization across related tasks. These approaches highlight the growing need for context-sensitive prioritization strategies in complex RL environments.

In parallel with algorithmic innovations, several studies have explored the interplay between experience prioritization and broader training paradigms such as curriculum learning. By viewing the replay buffer as a mechanism for structuring the order of experience exposure, some approaches implicitly adopt curriculum-like strategies, where simpler or more informative samples are presented earlier in training [11, 18]. This perspective aligns with findings that structured sampling—not just random replay—can accelerate convergence and improve final policy quality. The integration of curriculum principles into experience replay thus opens avenues for more adaptive and staged learning strategies.

The effectiveness of PER also depends heavily on the operational characteristics of the replay buffer itself, such as its capacity, update frequency, and the ratio of learning to data collection. Recent empirical studies have shown that increasing buffer size or adjusting the replay ratio can have non-trivial and sometimes counterintuitive effects on learning dynamics [19]. For instance, uncorrected multi-step returns—though theoretically inconsistent—may yield better empirical performance when combined with large replay memories. These insights suggest that prioritized sampling must be considered in conjunction with the broader design of the experience replay mechanism to achieve optimal results.

The versatility of prioritized experience replay has also been demonstrated through its application in domain-specific problems beyond standard benchmark environments. For instance, in financial decision-making tasks such as optimal trade execution, PER has been employed alongside heuristic policies to enhance learning efficiency in noisy market conditions [20]. In robotics and communication-constrained multi-agent systems, PER-inspired scheduling strategies have been used to prioritize information that maximizes utility under bandwidth limitations [21]. These applications confirm that the principles of PER can be effectively extended to structured real-world problems where sample efficiency and relevance are critical.

Taken together, these advancements reveal a clear trajectory in the evolution of experience replay—from uniform random sampling toward more intelligent, context-aware, and adaptive prioritization schemes. Whether through theoretical formulations, hybrid value-model integrations, uncertainty-driven sampling, or task-specific strategies, the field continues to refine the criteria by which past experiences are deemed valuable. These developments underscore the central role of experience replay in modern reinforcement learning and motivate continued exploration into how the structure and content of the replay buffer can be optimized for different learning objectives and domains.

Building on these prior advancements, we propose an adaptive experience replay mechanism that enhances the original PER framework. Our method introduces three transition-specific signals—reward, usage counter, and policy probability—and combines them to dynamically determine the importance of each transition. This approach aims to provide a more flexible and informative sampling strategy for training deep RL models.

3- Adaptive Prioritized Replay Buffer

3- 1- Prioritized Experience Replay

If data (s,a,s',r) exists for each transition in the replay buffer, the DQN method attempts to predict the Q-value for state s_1 and action a using a neural network θ. The target Q-value is obtained using a target model θ', which is a copied version of the model θ.

$$Q^*(s,a) = r + \gamma Q_{\theta'}\left(s', argmax_a\left(Q_\theta(s',a)\right)\right) \tag{1}$$

$$|\delta| = \left|Q^*(s,a) - Q_\theta(s,a)\right| \tag{2}$$

In this equation, $|\delta|$ is the magnitude of the temporal-difference (TD) error, and γ is a discount factor.

The prioritized experience replay (PER) method utilizes the TD error and assigns a probability to each data in the buffer so that data with higher errors are more likely to be selected.

$$P(j) = \frac{|\delta_j|^\alpha}{\sum_i |\delta_i|^\alpha} \tag{3}$$

Based on the PER paper [3], the exponent α is set to a fixed value of 0.6. To address bias in high-error data, importance sampling weight is applied to reduce the impact of model updates on data with higher priority.

$$\omega_j = \left(\frac{1}{N}\cdot\frac{1}{P(j)}\right)^\beta \tag{4}$$

$$\Delta = \Delta + \omega_j.\delta_j.\nabla_\theta Q(s,a) \tag{5}$$

$$\theta = \theta + \eta.\Delta \tag{6}$$

Which N is the replay buffer size and exponent β is scheduled from the initial value $\beta_0 = 0.4$ to 1 linearly [3]. By employing this method, higher-error data can contribute more to the training, ensuring their valuable information is effectively learned. This method can improve learning, increasing the return value in many environments compared to the DQN method with uniform random sampling from the replay buffer.

3- 2- Prioritization with RCP Values

In this research, we consider the PER method as the baseline and propose a novel method for calculating the priority of each data. In this method, in addition to the TD error, we incorporate three additional values, including the reward of the transition, the number of times that each transition has been used in training the model, and the probability of the action in the current state $(\pi(a \mid s))$, in the calculation of the priority of that data. We will refer to these three values as RCP (Reward, Counter, Policy).

Reward: Assume two transitions have been used in training the model, both having the same TD error. However, if the reward obtained from the first transition is significantly higher than that of the second, learning valuable information in the first transition will be more important. This is in contrast to the PER method, which considers equal probabilities for these two data due to their identical TD errors.

Because of the variety of reward values defined in different environments and transitions, the reward value in each transition is normalized to have a number between 0 to 1 as the reward of any transition (see Algorithm 1).

Counter: In cases where two data points have the same TD error but one has been used 10 times in training while the other only once, it is better to assign higher priority to the less frequently used data. Additionally, if a data point is noisy and consistently has a high error, without considering the frequency of its contribution to the training, it can be consistently utilized, posing challenges to the training process. We know that the PER method does not account for these factors.

To use this value in calculating the priority of the transitions, the sigmoid function is used to obtain the normalized counter, which is a number between 0 to 1 (see Algorithm 1).

In this sigmoid-based normalization, we use a scaling factor of -4 in the exponent to control the steepness of the curve. This value ensures that transitions which have been used frequently in training quickly receive lower normalized scores, thereby reducing their sampling probability and helping the model focus on less-reused but potentially informative transitions. (Fig. 1)

Policy $(\pi(\boldsymbol{a} \mid \boldsymbol{s}))$: A higher probability of selecting action a in state s, denoted as π(s, a), indicates that the Q-value for this state-action pair is relatively higher than for other actions in the same state. This reflects the agent's confidence in choosing this action under its current policy. While this aspect is not considered in the original PER method, incorporating π(s, a) into the prioritization process helps emphasize transitions that are more aligned with the agent's learned behavior.

In methods like DQN, where the policy is not explicitly modeled, π(s, a) can be estimated using a softmax function over Q-values. Unlike raw Q-values, which are unbounded and often unstable early in training, the policy probability is normalized and bounded between 0 and 1, making it a more stable signal for prioritization. Moreover, it introduces a probabilistic perspective on policy alignment, which complements the TD error and reward signals by highlighting transitions that reinforce the agent's consistent decision-making patterns. This can improve learning stability and policy convergence.

$$\pi(a_i \mid s) = \frac{e^{Q(s,a_i)}}{\sum_j e^{Q(s,a_j)}} \tag{7}$$

In the proposed method, $\pi(a \mid s)$ is first calculated using this relationship, and then it is used in a new prioritization process (see Algorithm 1).

3- 3- Adaptive Prioritization

In this research, we have utilized RCP values to enhance the prioritization of each transition in the buffer more effectively. As seen in equation (3), the parameter α is the exponent of TD error and determines how much prioritization is used. If we set this parameter to zero, the probabilities of all transitions become equal, and sampling will be uniformly at random. However, as this parameter increases, we move further away from random sampling.

Despite the importance of the value of this parameter, the PER method sets a constant value for it and does not change it during training. In the proposed prioritization method, this parameter is defined based on the information associated with each data. So the priority value for each data is determined using TD error and data-dependent parameter α.

$$P(j) = \frac{|\delta_j|^{\alpha_j}}{\sum_i |\delta_i|^{\alpha_i}} \tag{8}$$

Where the parameter α is dependent on the values of the normalized reward, normalized counter, probability of action in the current state $(\pi(a \mid s))$ (RCP values), or a combination of them.

ALGORITHM 1: CALCULATE RCP VALUES

```
1   $r_{mean} \leftarrow 0$; $r_{std} \leftarrow 1$
2   Procedure Normalize Reward ( $r_{batch}$ )
3       $r_{mean} \leftarrow \lambda.mean(r_{batch}).(1-\lambda).r_{mean}$
4       $r_{std} \leftarrow \lambda.std(r_{batch}).(1-\lambda).r_{std}$
5       $upper \leftarrow \lambda.r_{mean} - 2r_{std}; lower \leftarrow \lambda.r_{mean} + 2r_{std}$
6       $r_{batch} \leftarrow max(max(r_{batch}, lower), upper)$
7       $Return\ (R_{batch} - lower)/(upper - lower)$
8   Procedure Get Policy ( $Q(s,a)$ , $a_i$ )
9       $\pi(s,a_i) \leftarrow e^{Q(s,a_i)} / \sum_j e^{Q(s,a_j)}$
10      $Return\ \pi(s,a_i)$
11  Procedure Normalize Counter ( $s_1, a.s_2$ )
12      $c \leftarrow 1/(1+e^{counter(s_1,a,s_2)-4})$
13      $Return\ c$
```

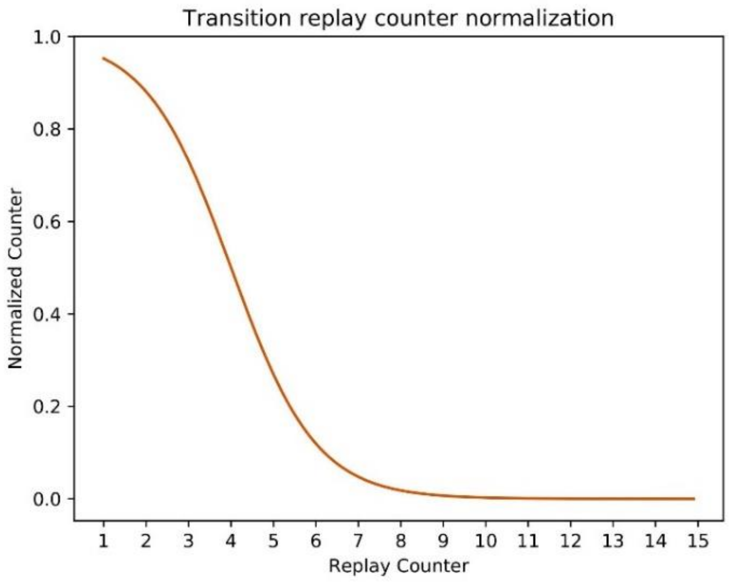


Fig. 1. Normalizing transition replay counter using Sigmoid function.

$$\alpha_j = f\left(r_j, \pi\left(a_j \mid s_j\right), counter\left(s_j, a_j, s'_j\right)\right) \tag{9}$$

So the training algorithm for DQN using the proposed prioritized replay buffer will be like the PER, except that the parameter α is based on RCP values (Algorithm 2).

In the next section, the impact of utilizing each of the three values of RCP and various combinations of them will be examined.

4- Experiments and Results

In this study, the Double DQN (D2QN) model has been employed using Prioritized Experience Replay (PER) as our baseline, referred to as PER-DQN. The neural network architecture follows the design introduced by Mnih et al. [1], consisting of three convolutional layers with ReLU activation functions, followed by two fully connected layers. The input to the network is a stack of four consecutive grayscale frames of size 84×84 pixels. The final layer outputs Q-values for each possible action in the environment.

ALGORITHM 2: DOUBLE DQN WITH ADAPTIVE PRIORITIZED EXPERIENCE REPLAY

```
    Input: minibatch $k$, step-size $\eta$, replay period $K$ and size $N$, exponents $\alpha$ and $\beta$, budget $T$
1   Initialize replay memory $H = \emptyset$, $\Delta = 0$, $p_1 = 1$
2   Observe $S_0$ and choose $a_0 \sim \pi_\theta(s_0)$
3   for $t = 1$ to $T$ do
4       Observe $s_t$, $r_t$, $\gamma_t$
5       Store transition $(s_{t-1}, a_{t-1}, r_t, \gamma_t, s_t)$ in $H$ with maximum priority $p_t = max_{i<t}(p_i)$
6       if $t \equiv 0$ mod $K$ then
7           for $j=1$ to $k$ do:
8               $\alpha_j = f(R_{j-1}, \pi(a_{j-1} | s_{j-1}), counter(s_j, a_{j-1}, s_j))$
9               Sample transition $j \sim P(j) = p_j^{\alpha_j} / \sum_i p_i^{\alpha_i}$
10              Compute importance-sampling weight $\omega_j = \left(N \cdot P(j)\right)^{-\beta} / max_i(\omega_i)$
11              Compute TD-error $\delta_j = r_j + \gamma_j Q_{target}\left(s_j, argmax_a Q(s_j, a)\right) - Q\left(s_{j-1}, a_{j-1}\right)$
12              Update transition priority $p_j \leftarrow |\delta_j|$
13              Accumulate weight-change $\Delta \leftarrow \Delta + \omega_j . \delta_j . \nabla_\theta Q(s_{j-1}, a_{j-1})$
14          Update weights $\theta \leftarrow \theta + \eta . \Delta$, reset $\Delta = 0$
15          From time to time copy weights into target network $\theta_{target} \leftarrow \theta$
16      Choose action $a_t \sim \pi_\theta(s_t)$
```

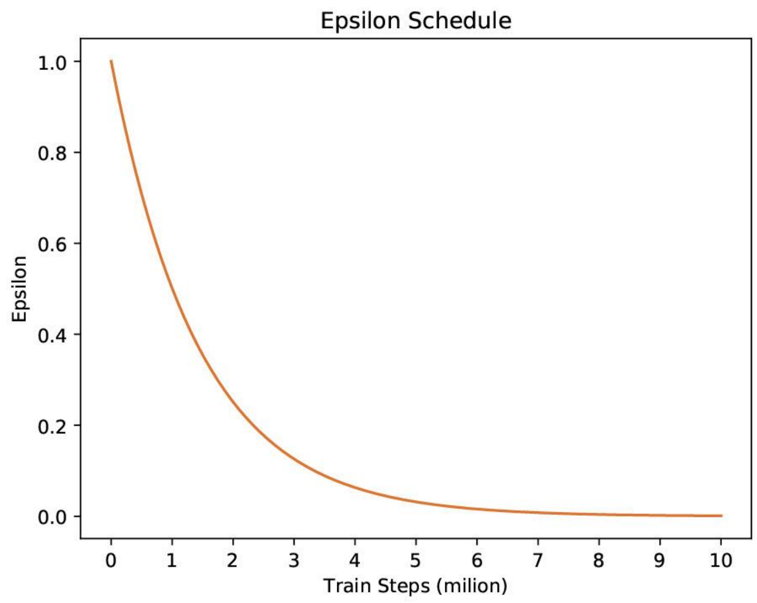
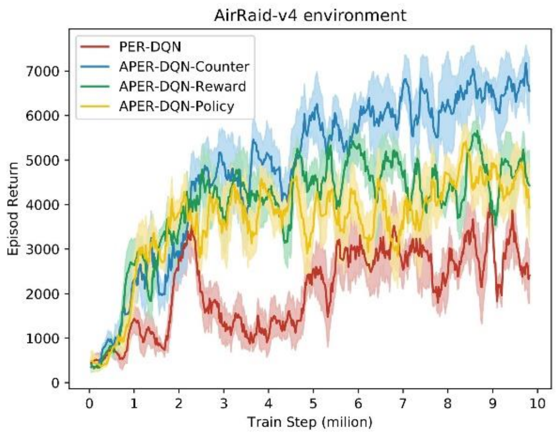


Fig. 2. Scheduling epsilon from 1 to 0.001 exponentially.

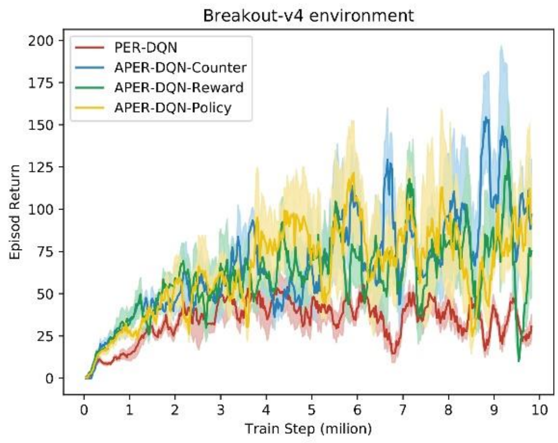
Multiple Atari environments are utilized for training the models. Each model is trained for up to 10 million steps using a mini-batch size of 32 and a replay buffer with a capacity of one million transitions. We use the Adam optimizer with a learning rate of 0.00025. The target network is updated every 10,000 steps. The epsilon-greedy strategy is employed for exploration, with epsilon decaying exponentially from 1.0 to 0.001 (Fig. 2).

To evaluate the models during training, the agent undergoes five evaluation episodes every 100,000 training steps using the current policy, and the average and standard deviation of the return values across these 5 episodes are recorded and plotted to visualize performance trends over time.

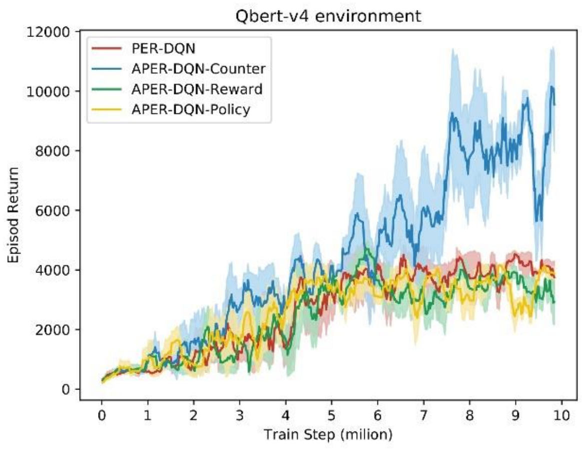
All implementations were carried out in Python 3.8 using PyTorch 1.12.1. The Atari environments were obtained using OpenAI Gym along with the Arcade Learning Environment (ALE).



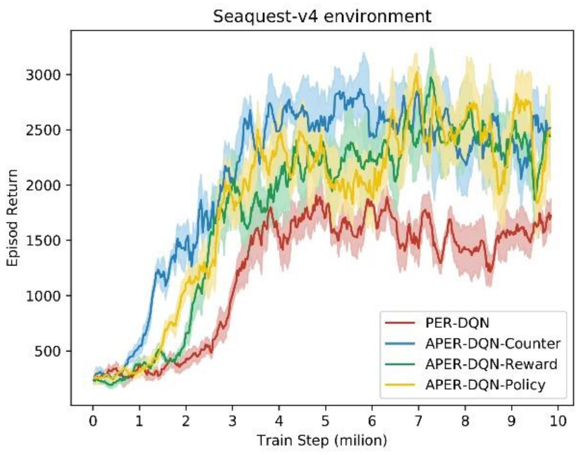
(a)



(b)



(c)



(d)

Fig. 3. Comparison of episodes returns in DQN with prioritized experience replay (PER-DQN) and Adaptive prioritized experience replay (APER-DQN) using RCP values in prioritization.

4- 1- Impact of RCP Values

Initially, we investigate the influence of each of the three RCP values on the episodic returns. To achieve this, we employ four Atari environments: Seaquest, AirRoad, Qbert and Breakout. PER-DQN is trained on these environments as the baseline model.

For each of the three RCP values, the training of DQN with adaptive prioritized experience replay (APER) is performed, incorporating only one of these values at a time. The variations in the returns of evaluation episodes for each model and environment are observed in the plots depicted in Fig. 3.

Based on these plots, it is evident that using each of the RCP values in the calculation of the priority of each data has significantly enhanced the model compared to the PER method. After a detailed examination, we can find out that the impact of using any value of RCP values varies across different environments. However, in general, it can be observed that the effects of the two methods, policy and reward, are closer to each other, while the counter method has generally achieved a greater improvement.

This improvement is particularly noticeable in environments with high reward variance, such as Seaquest and AirRaid, where our method exhibited smoother and more stable learning curves. This indicates that incorporating RCP values can enhance robustness against stochasticity in the reward signal by diversifying the prioritization mechanism beyond noisy TD errors.

After identifying the beneficial effects of these values on improving model training, we will explore different aggregation functions in the next experiment to achieve the optimal configuration.

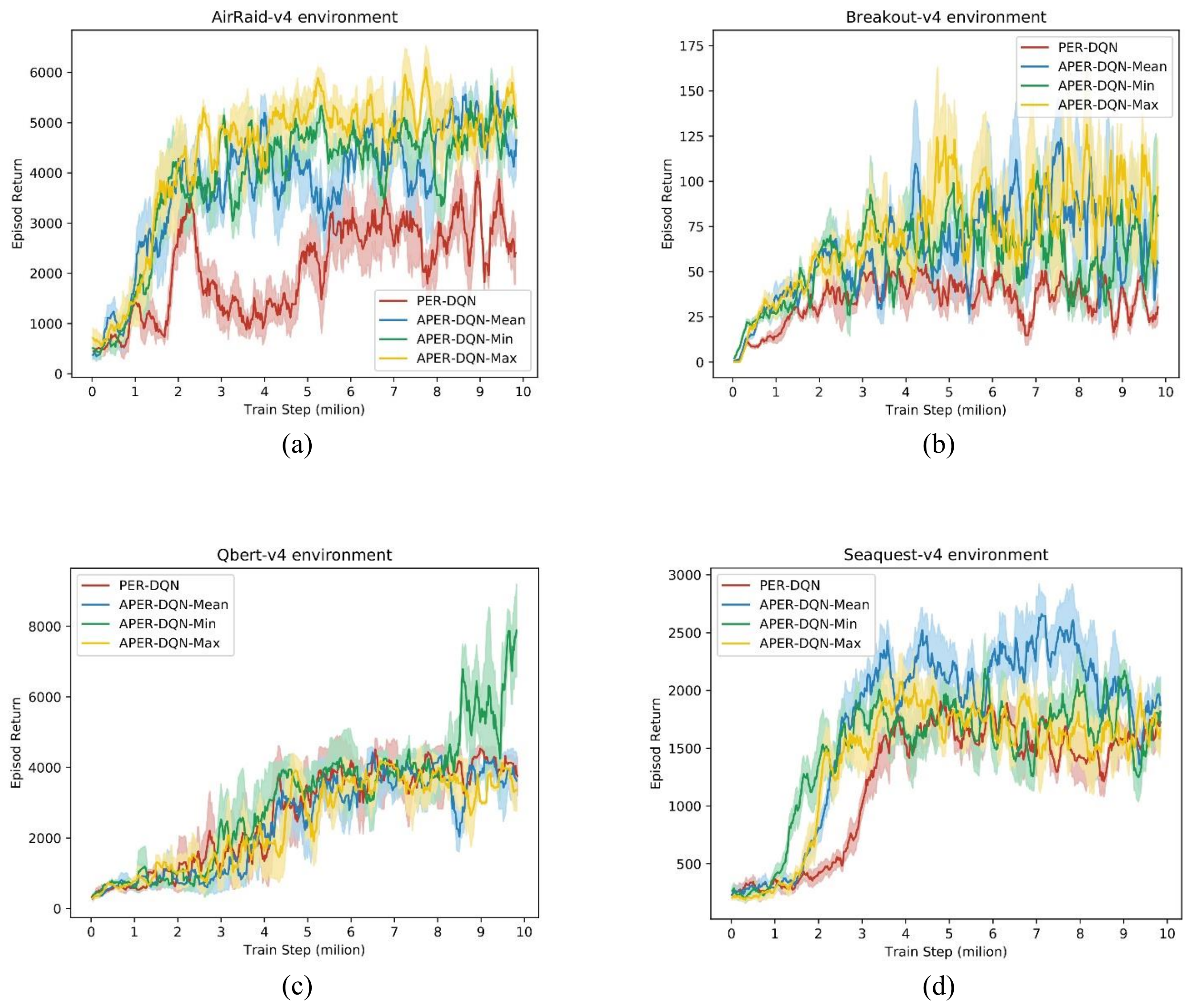


Fig. 4. Comparison of episodes returns in DQN with prioritized experience replay (PER-DQN) and Adaptive prioritized experience replay (APER-DQN) using different aggregation functions.

4- 2- RCP Values Aggregation

To utilize all three RCP values, three functions have been considered for their aggregation. In each of these three methods, the RCP values are combined, resulting in a single value that is the exponent of TD error (equation (9)) To combine these three values, minimum, maximum, and average, have been employed to achieve the optimal aggregation function. Practically, in the Algorithm 2, the function is defined to minimize, maximize, or average three RCP values.

To evaluate the three mentioned combination methods, the same four Atari environments are used. The proposed model (APER-DQN) is trained with each aggregation function on these environments and is compared with the results of PER-DQN. The results of this experiment are illustrated in the plots of Fig. 4.

According to the obtained results, it is evident that the use of each aggregation method can increase episode return values in various environments. Additionally, it is observed that the impact of each aggregation method varies depending on the environment, and an appropriate aggregation function must be selected based on the specific characteristics of the task.

Despite this variability, the mean aggregation method demonstrated consistently stable performance across all tested environments. Unlike min or max, which can overemphasize a single component of the RCP values, the mean method ensures a balanced contribution from all three signals—reward, counter, and policy probability—reducing the risk of bias toward any individual factor.

Considering both its stable empirical performance and its balanced nature, the mean function can be regarded as a strong default choice for aggregation when environment-specific tuning is not available.

While the proposed aggregation functions (min, max, mean) offer effective ways to integrate RCP values, their performance may vary across different types of environments. In particular, environments with sparse rewards pose unique challenges, as the reward component of RCP may lack sufficient informative value. In such cases, direct reliance on immediate rewards may lead to unstable prioritization or even ineffective sampling.

To address this, the adaptive prioritization mechanism can be adjusted to reduce the influence of the reward component and instead place more emphasis on the counter and policy components, which remain informative regardless of reward sparsity. Additionally, smoothing techniques such as moving averages or simple reward shaping can be applied to create a more useful reward signal. This flexibility allows the proposed method to maintain robust performance even in environments with highly sparse or delayed rewards.

One potential concern with prioritization strategies is the risk of overfitting due to repeatedly sampling a narrow subset of transitions. Our method mitigates this by integrating multiple signals—reward, usage frequency, and policy probability—each of which is normalized and aggregated in a way that smooths prioritization. The use of the counter component, in particular, penalizes overused transitions, promoting diversity in sampling. Furthermore, the flexibility in aggregation allows tuning the prioritization aggressiveness depending on the environment. These mechanisms collectively help prevent overfitting to noisy or sparse reward patterns.

In addition to mitigating overfitting, our adaptive prioritization approach also encourages exploration throughout training. The counter component reduces the likelihood of repeatedly selecting the same transitions, thus promoting exposure to underutilized experiences. This allows the agent to explore a wider variety of state-action pairs, particularly those that may have been initially overlooked. Furthermore, the inclusion of policy probability in the prioritization scheme promotes sampling transitions where the agent has low confidence in its actions, further supporting exploration of less certain regions of the state space. While regret-based analysis is beyond the scope of this study, qualitative results in stochastic environments like Qbert and Breakout suggest that the proposed method enhances both exploration and learning stability.

5- Conclusion

In the DQN method, to address the issue of temporal data dependencies and reusing previously generated data, an experience replay buffer is employed that stores the constructed transitions, and training samples are randomly selected from this buffer.

The prioritized experience replay method introduces an approach where sampling from this buffer is no longer uniformly random, and each data has a probability of being selected based on its TD error.

In this paper, we tried to enhance the prioritized experience replay method by modifying the prioritization process. In the proposed approach, the calculation of the probability for selecting data is not solely based on the TD error but also incorporates three additional values: reward, the number of times that each transition has been selected, and $\pi\left(a\,|\,s\right)$.

Through designed experiments, the impact of using each of these three values in prioritizing data was examined, revealing that all three values can bring significant improvements. Subsequently, three aggregation methods, including min, max, and mean, were suggested for these three values, allowing the utilization of all three values in calculating the probability of selecting each data. It was found that the aggregation function should be determined based on each environment, but the 'mean' aggregation function can be a preferred choice as it exhibits acceptable performance across various environments and also incorporates all three RCP values.

Although our experiments focused on the Double DQN architecture, the proposed adaptive prioritization strategy is model-agnostic and can be applied to other Q-learning-based algorithms such as Dueling DQN. We leave the evaluation of our method in these frameworks as part of future research.

References

[1] Mnih, V., et al., Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

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

[3] Schaul, T., et al., Prioritized Experience Replay. CoRR, 2015. abs/1511.05952.

[4] Atherton, L.A., D. Dupret, and J.R. Mellor, Memory trace replay: the shaping of memory consolidation by neuromodulation. Trends in Neurosciences, 2015. 38(9): p. 560-570.

[5] Ólafsdóttir, H.F., et al., Hippocampal place cells construct reward-related sequences through unexplored space. Elife, 2015. 4: p. e06063.

[6] Foster, D.J. and M.A. Wilson, Reverse replay of behavioural sequences in hippocampal place cells during the awake state. Nature, 2006. 440(7084): p. 680-683.

[7] McNamara, C.G., et al., Dopaminergic neurons promote hippocampal reactivation and spatial memory persistence. Nature Neuroscience, 2014. 17(12): p. 1658-1660.

[8] Van Seijen, H. and R. Sutton. Planning by prioritized sweeping with small backups. in the International Conference on Machine Learning. 2013. PMLR.

[9] Horgan, D., et al. Distributed Prioritized Experience Replay. in International Conference on Learning Representations (ICLR). 2018.

[10] Hou, Y., et al. A novel DDPG method with prioritized experience replay. in IEEE international conference on systems, man, and cybernetics (SMC). 2017. IEEE.

[11] Saglam, B., et al., Actor prioritized experience replay. Journal of Artificial Intelligence Research, 2023. 78: p. 639-672.

[12] Oh, Y., et al. Model-augmented prioritized experience replay. in International Conference on Learning Representations (ICLR). 2021.

[13] Li, A.A., Z. Lu, and C. Miao, Revisiting prioritized experience replay: A value perspective. arXiv preprint arXiv:2102.03261, 2021.

[14] Zhang, H., et al., Self-adaptive priority correction for prioritized experience replay. Applied sciences, 2020. 10(19): p. 6925.

HOW TO CITE THIS ARTICLE
M. Adibian, M. M. Ebadzadeh, Double Deep Q Network with Adaptive Prioritized Experience Replay, AUT J. Model. Simul., 57(1) (2025) 53-62.

DOI: 10.22060/miscj.2025.23426.5373