AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 57(1) (2025) 53-60
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 has garnered significant attention as a powerful framework for training agents to make sequential decisions in complex environments. This approach combines the power of deep neural networks with reinforcement learning algorithms to achieve remarkable performance in various tasks, ranging from game playing to robotics and autonomous driving.

One of the most well-known models in deep reinforcement learning is the Deep Q-Network (DQN) proposed by [1]. DQN introduced a neural network architecture that learns to approximate the state-action value, Q(s, a), from the input transitions, enabling agents to find the best action in any state.

In online reinforcement learning, agents sequentially interact with the environment, generating a stream of data samples, and these data are used in model training. One of the challenges of this approach is the sequential dependency of data, which contradicts the i.i.d. assumption data used in neural networks. Additionally, valuable data generated in the past is lost in this process, reducing the potential for reuse.

To address these issues, the concept of experience replay has been introduced [2], utilizing a buffer to store previously generated data and sampling from this buffer for training. Employing this method in online reinforcement learning improves the results and can also be beneficial when training data are limited.

In conventional deep reinforcement learning approaches, sampling from the replay buffer is typically done uniformly at random. However, it is well-known that not all data in the buffer contribute equally to the learning process. Some transitions are more informative than others, making it beneficial to prioritize their selection. The Prioritized experience replay (PER) [3] method introduces a more intelligent approach to sampling data from the buffer instead of uniform random.

The primary motivation behind PER stems from the variation of transitions in terms of their significance; they may be more or less surprising, redundant, or directly relevant to the learning task. The key concept driving this method is the use of the temporal-difference (TD) error, which serves as a proxy for the informativeness of each data transition. Transitions with higher TD errors are considered to contain valuable information that can enhance the learning process and are therefore prioritized for selection with higher probability.

In this study, we proposed Adaptive Prioritized Experience

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

Replay that considers not only the TD error but also other features of the data, including the probability of an action in a given state $\pi(a|s)$, the reward obtained from the transition, and the frequency of using the transition in training. In this method, unlike the previous approach, instead of using a constant value for the exponent of the TD error (like PER), this exponent is calculated based on three proposed values related to each transition.

In the next section, relevant research studies are reviewed, and after getting familiar with the past replated works, the proposed method is introduced. Finally, the results of the proposed method are presented in some Atari environments.

2- Related Works

In the real world and within human cognition, recent events associated with significant rewards or errors tend to be more frequently replayed [4-7]. This observation has been leveraged in various algorithms, where data are sampled and utilized with different priorities [8].

In deep reinforcement learning, several methods utilize the temporal-difference (TD) error to prioritize data. One notable approach is the Prioritized Experience Replay (PER) method [3], which calculates the probability of selecting each transition based on this error. To mitigate bias towards high-error data and to maintain diversity, importance sampling weights are employed, adjusting the magnitude of weight update during network training.

Numerous enhancements have been proposed to improve the PER method, enhancing its efficiency and applicability. For example, to handle larger datasets effectively Distributed Prioritized Experience Replay method [9] is introduced that uses a distributed architecture.

PER was originally introduced for the DQN model and showed promising results. However, recent research tried to extend its application to other deep reinforcement learning models. In the model proposed by [10], the PER method is integrated into the DDPG model, yielding enhanced results. Also, while empirical studies have demonstrated limitations of the PER in off-policy actor-critic models, [11] proposed some modifications to this method to enable its use in actor-critic models. Additionally, in research conducted by [12], the concept of PER was extended to model-based approaches.

Recent studies have proposed refinements to the PER method and addressed identified limitations. For instance, one approach aims to incorporate the value of each transition in prioritization [13], while another seeks to prevent priority values from becoming outdated during training [14]. Similar to these studies, we proposed adaptive experience replay, which is a new method to enhance the original PER method by introducing three values for each transition to prioritize the importance of data in the replay buffer.

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.

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

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

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.

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

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.

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

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

$$\theta = \theta + \eta.\Delta \quad (6)$$

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 $(\pi(a|s))$, 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 the counter normalize function, normalization initially results in slight changes near 1 with an increasing number of uses, but as the count increases further, the value rapidly approaches 0.1 (Fig. *1*)

Policy $\left(\pi(\boldsymbol{a} \mid \boldsymbol{s})\right)$: A higher probability of action in the current state indicates that the Q-value for that state-action pair is higher than the Q-values for the same state and other actions, implying the greater importance of this transition, but this aspect is not considered in the PER method. In a method like DQN where $\pi(a \mid s)$ is not directly available, this value can be calculated using the state-action values.

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

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 .

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

ALGORITHM 1: CALCULATE RCP VALUES

$r_{mean} \leftarrow 0$; $r_{std} \leftarrow 1$

Procedure Normalize Reward (r_{batch})

  $r_{mean} \leftarrow \lambda.mean(r_{batch}).(1-\lambda).r_{mean}$

  $r_{std} \leftarrow \lambda.std(r_{batch}).(1-\lambda).r_{std}$

  $upper \leftarrow \lambda.r_{mean} - 2r_{std}$; $lower \leftarrow \lambda.r_{mean} + 2r_{std}$

  $r_{batch} \leftarrow max(max(r_{batch}, lower), upper)$

  Return $(R_{batch} - lower)/(upper - lower)$

Procedure Get Policy ($Q(s,a)$, a_i)

  $\pi(s,a_i) \leftarrow e^{Q(s,a_i)} / \sum_j e^{Q(s,a_j)}$

  Return $\pi(s,a_i)$

Procedure Normalize Counter ($s_1, a.s_2$)

  $c \leftarrow 1/(1+e^{counter(s_1,a,s_2)-4})$

  Return c

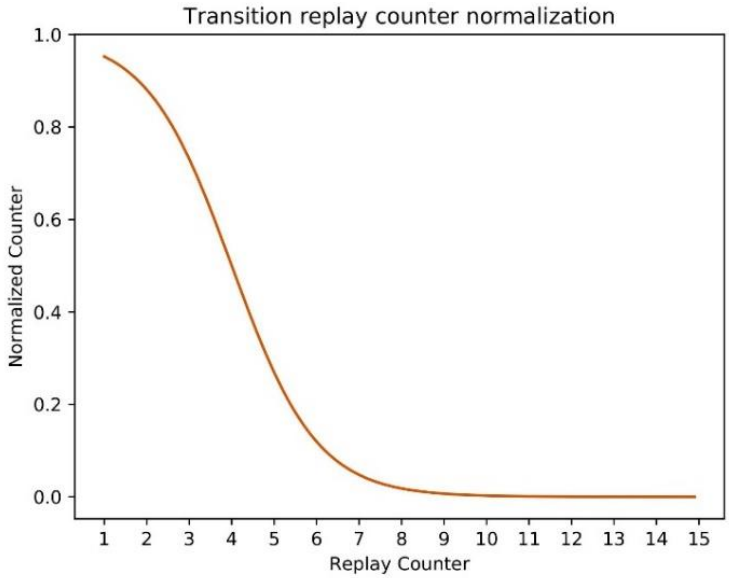

Fig. 1. Scheduling epsilon from 1 to 0.001 exponentially

ALGORITHM 2: DOUBLE DQN WITH ADAPTIVE PRIORITIZED EXPERIENCE REPLAY

Input: *minibatch* k, *step-size* η, *replay period* K *and size* N, *exponents* α *and* β, *budget* T

Initialize replay memory $H=\emptyset$, $\Delta=0$, $p_1=1$

Observe S_0 *and choose* $a_0 \sim \pi_\theta(s_0)$

for $t=1$ *to* T *do*

  Observe s_t, r_t, γ_t

  Store transition $(s_{t-1}, a_{t-1}, r_t, \gamma_t, s_t)$ *in* H *with maximum priority* $p_t = max_{i<t}(p_i)$

  if $t \equiv 0 \bmod K$ ***then***

    for *j=1* ***to*** k ***do***:

      $\alpha_j = f(R_{j-1}, \pi(a_{j-1} \mid s_{j-1}), counter(s_j, a_{j-1}, s_j))$

      Sample transition $j \sim P(j) = p_j^{\alpha_j} / \sum_i p_i^{\alpha_i}$

      Compute importance-sampling weight $\omega_j = \left(N \cdot P(j)\right)^{-\beta} / \ max_i(\omega_i)$

      Compute TD-error $\delta_j = r_j + \gamma_j Q_{target}\left(s_j, argmax_a Q(s_j, a)\right) - Q(s_{j-1}, a_{j-1})$

      Update transition priority $p_j \leftarrow \mid \delta_j \mid$

      Accumulate weight-change $\Delta \leftarrow \Delta + \omega_j . \delta_j . \nabla_\theta Q(s_{j-1}, a_{j-1})$

    Update weights $\theta \leftarrow \theta + \eta . \Delta$, *reset* $\Delta = 0$

    From time to time copy weights into target network $\theta_{target} \leftarrow \theta$

  Choose action $a_t \sim \pi_\theta(s_t)$

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

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

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.

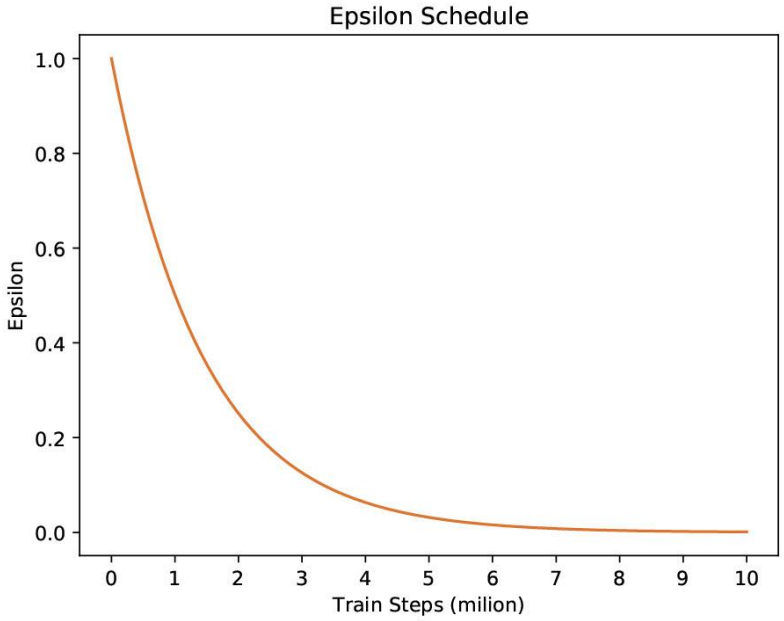

Fig. 2. Normalizing transition replay counter using the Sigmoid function.

4- Experiments and Results

In this study, the DQN model has been employed using PER as our baseline. We will refer to this model PER-DQN. Multiple Atari environments are utilized for training the models, and for each experiment model is trained for up to 10 million steps using a batch size of 32. Additionally, the epsilon-greedy method is employed for generating transitions and storing them in the buffer, with epsilon exponentially decreasing from 1 to 0.001 (Fig. *2*).

To evaluate the models during the training process, the agent undergoes five episodes using the trained model every 100 training steps, and the average and standard deviation of the return values for these 5 episodes are calculated, providing a graphical representation of the changes over time.

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.

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.

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 Algorithm 2, the function f 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 the episode return values in various environments. Additionally, it is observed that the impact of each aggregation method varies across different environments, and an appropriate aggregation function must be selected based on the target environment. Despite this variability, considering the significant improvement achieved in many environments with the *mean*

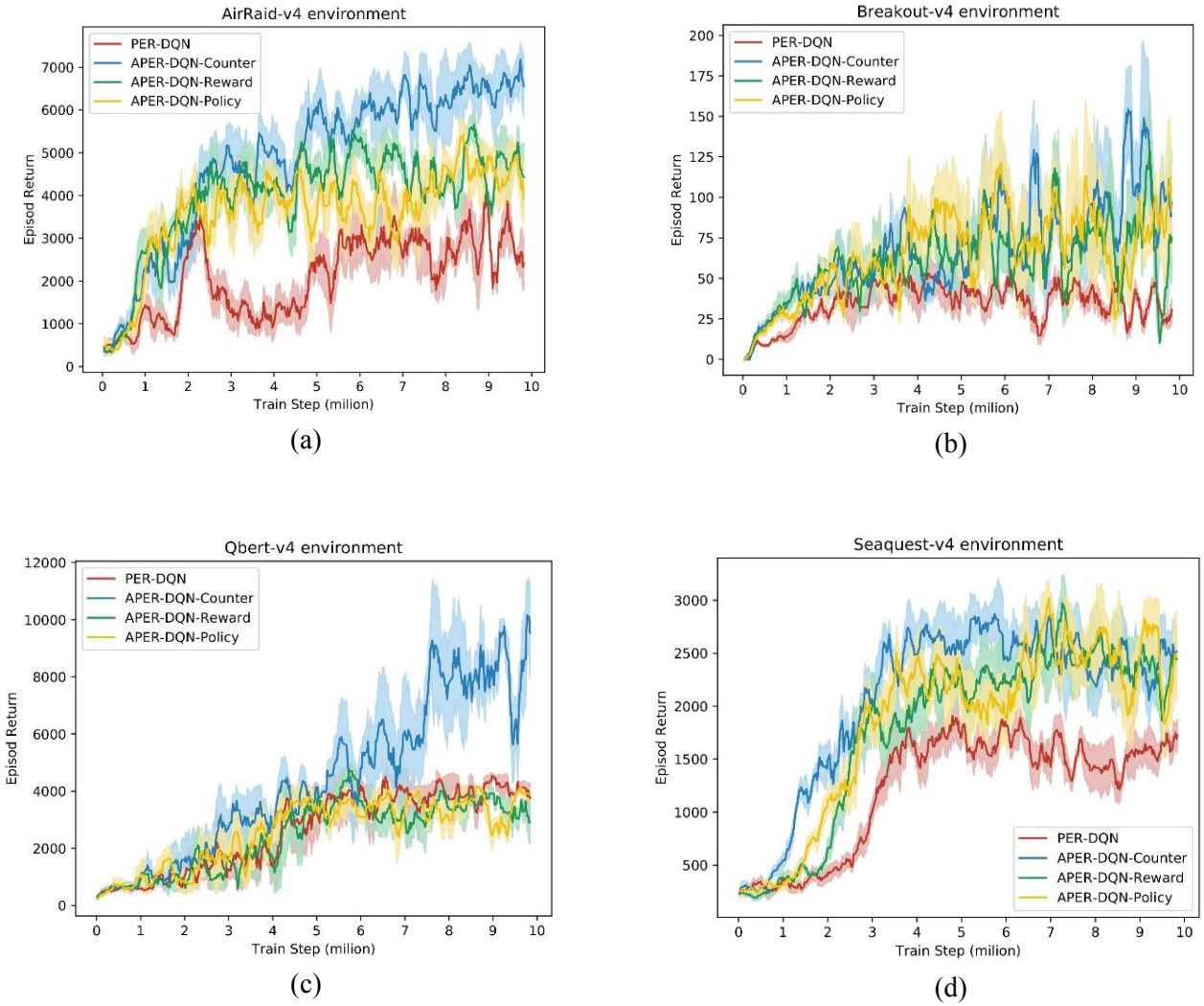

Fig. 3. Comparison of episodes returns in DQN with prioritized experience replay (PER-DQN) and Adaptive prioritized experience replay (APER-DQN) using RCP values in prioritization.

method and the utilization of all three RCP values in this approach, it can be generally stated that using *mean* as the aggregation function can be a preferred method.

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 $\pi(a|s)$.

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.

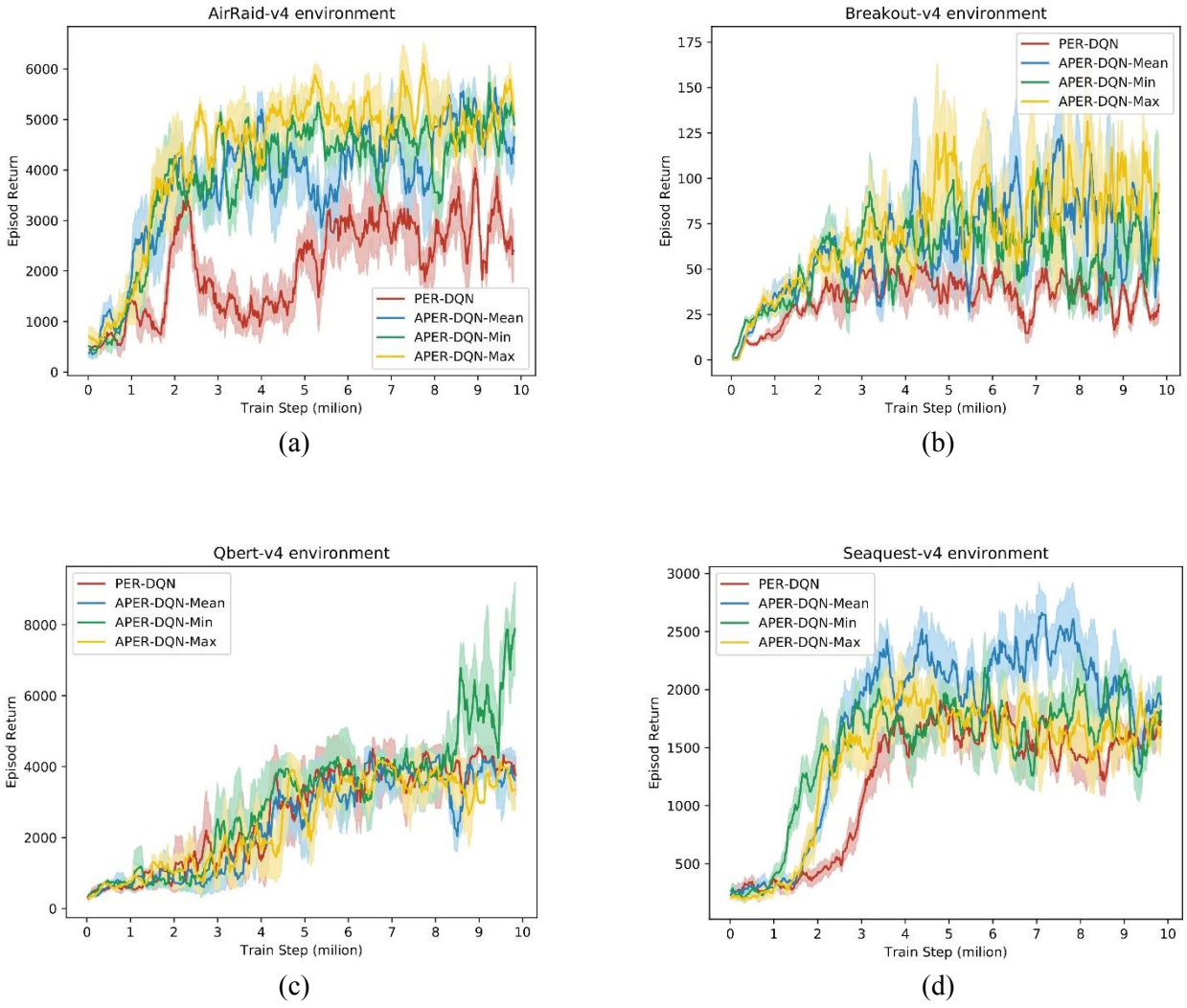

Fig. 4. Comparison of episodes returns in DQN with prioritized experience replay (PER-DQN) and Adaptive prioritized experience replay (APER-DQN) using different aggregation functions.

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) 53-60.

DOI: 10.22060/miscj.2025.23426.5373