Experience Replay Optimization via ESMM for Stable Deep Reinforcement Learning

Richard Sakyi Osei, Daphne Lopez
School of Computer Science Engineering and Information Systems
Vellore Institute of Technology, Vellore, India

***Abstract*—The memorization and reuse of experience, popularly known as experience replay (ER), has improved the performance of off-policy deep reinforcement learning (DRL) algorithms such as deep Q-networks (DQN) and deep deterministic policy gradients (DDPG). Despite its success, ER faces the challenges of noisy transitions, large memory sizes, and unstable returns. Researchers have introduced replay mechanisms focusing on experience selection strategies to address these issues. However, the choice of experience retention strategy has a significant influence on the selection strategy. Experience Replay Optimization (ERO) is a novel reinforcement learning algorithm that uses a deep replay policy for experience selection. However, ERO relies on the naïve first-in-first-out (FIFO) retention strategy, which seeks to manage replay memory by constantly retaining recent experiences irrespective of their relevance to the agent's learning. FIFO sequentially overwrites the oldest experience with a new one when the replay memory is full. To improve the retention strategy of ERO, we propose an experience replay optimization with enhanced sequential memory management (ERO-ESMM). ERO-ESMM uses an improved sequential retention strategy to manage the replay memory efficiently and stabilize the performance of the DRL agent. The efficacy of the ESMM strategy is evaluated together with five additional retention strategies across four distinct OpenAI environments. The experimental results indicate that ESMM performs better than the other five fundamental retention strategies.**

Keywords*—*Experience replay; experience replay optimization; experience retention strategy; experience selection strategy; replay memory management

I. Introduction

Deep reinforcement learning (DRL) has emerged as a robust framework for training agents to make intelligent decisions in complex environments [1] in the area of health [2], [3], [4], automobile [5], [6], robotics [7], [8], energy [9], [10], [11] and others. RL algorithms aim to maximize cumulative rewards by allowing an agent to interact with an environment, learning from trial and error. One crucial component of RL is experience replay (ER), which involves reusing past experiences to enhance the learning process. ER has been successful in the RL field since its implementation in the deep Q-network (DQN) algorithm [12] and has undergone numerous improvements by researchers. It has proven to be a valuable technique, facilitating improved sample efficiency, breaking transition correlations, stabilizing learning dynamics, and reducing the cost of training [12], [13], [14], [15]. However, the effectiveness of ER strongly relies on selecting and retaining relevant experiences [16]. Experience sampling involves the sequential or stochastic (random) selection of an experience index (the array index of the stored transitions) to determine the experience to use for training the RL agent. In contrast, experience retention focuses on strategies that can be adopted to determine the experiences to be stored and how these experiences can be managed in the replay buffer to optimize the learning process.

In recent years, various ER strategies have been proposed to address the challenges associated with experience retention. Isele and Cosgan [17] suggests that strategies for sampling experiences can be based on surprise, reward, state-space coverage, global training distribution, and state-action similarities. Similarly, de Bruin et al. [16] identified the full database (Full DB), first-in-first-out (FIFO), temporal difference error (TDE), and exploration as strategies for experience retention. They assert that while the FIFO strategy sequentially replaces old experiences irrespective of their relevance to learning, the exploration strategy stochastically overwrites the least-explored experience. The TDE strategy stochastically overwrites the least surprising experience, while the Resv approach ensures that "observed" experiences have equal retention rights. Moreover, the Full DB strategy uses a large amount of memory to store all experiences and does not require the removal of experiences from the replay memory.

Even with the improvements in ER strategies, they have limitations. The TDE is susceptible to noise, differences in function approximation accuracy, and randomness in the environment [15], [16], [18], [19], [20]. The FIFO strategy is affected by rapid changes in the state distribution, and Resv has premature data distribution convergence and poor coverage of state action over the optimal policy. Exploration takes longer to learn the correct value function, and Full DB promotes the rise of irrelevant experiences in the replay buffer [16]. Hence, there is still ample room for improvement in designing more efficient and effective experience retention strategies.

The proposed experience replay optimization with enhanced sequential memory management (ERO-ESMM) is a novel reinforcement learning algorithm that aims to improve the learning stability of DRL agents. The ERO-ESMM algorithm uses an enhanced sequential memory management (ESMM) strategy to manage the replay memory efficiently and stabilize the agent's performance. Compared to five existing experience retention strategies, the experimental results indicate that ERO-ESMM exhibits superior performance.

In this study, we present an enhanced experience retention strategy for DRL. Our proposed strategy aims to improve the

efficiency and effectiveness of experience replay by carefully managing experiences within the replay memory. The primary advancements presented in this study can be outlined as follows:

1) Firstly, we develop three new retention strategies to improve the efficiency and effectiveness of experience replay.

2) Secondly, we investigate the effects of six retention strategies, including the enhanced FIFO, on the ERO-enhanced DDPG algorithm.

3) Finally, we propose an enhanced framework incorporating the highest-performing retention strategy into the ERO framework.

We review existing experience retention strategies in Section III to achieve these objectives. We then present our enhanced retention strategy, outlining its core principles and rationale in Section III. Subsequently, we describe the experimental setup used to evaluate the performance of our approach in Section IV and provide detailed results and analysis. Finally, in Section V, we conclude our study and recommend future work in the field.

II. Related Work

We recognize that sampling and retention strategies are essential to experience replay in reinforcement-learning algorithms. This section briefly overviews the experience replay mechanism and sampling strategy. However, the section focuses on retention strategies, explicitly identifying those that effectively improve the performance of RL algorithms that use experience replays. The section explains the actor-critic method since the study seeks to improve an actor-critic algorithm (DDPG).

A. Experience Replay

With ER, an agent generates experiences using an exploration-exploitation method at specific intervals and stores them in fixed-size replay memory. The agent then samples these experiences uniformly and randomly from the replay buffer into a mini-batch and repeatedly uses them to train the RL algorithm. This random selection prevents high correlation among the sampled experiences.

ER was first introduced by Lin [21] in the early 1990s, but it gained widespread attention when Mnih et al. [12] combined it with a deep convolutional neural network to create a groundbreaking deep Q-network (DQN) algorithm. Since then, ER has become a critical component of RL algorithms, allowing them to use experience effectively and reduce the interactions required with the environment. Before the introduction of ER, algorithms such as Q-learning [20], which relied on a tabular data storage mechanism, could not retain previous state-action values because the current ones of the same state-action pair would overwrite them. This "catastrophic forgetting' [22], [23], [24], [25], [26] behavior leads to slower learning and poor algorithm convergence.

Although many algorithms that implement diverse sampling strategies have been developed, the size and data structure of the replay buffer, mini-batch size, experience retention rate, experience sampling, and retention techniques significantly influence the performance of these algorithms [40]. The selection of the experience index for experience sampling or retention can be sequential or random (uniform or prioritized probability). Sequential index selection is not appropriate for experience sampling because it creates a high correlation among the selected experiences, which subsequently slows down the agent's learning [12], [13], [34], [41]. Table I shows that most cited RL algorithms use a sequential index selection approach to remove experiences from the replay memory.

TABLE I. Some RL Algorithms and their Experience Index Selection Approach for Experience Retention

Algorithm	Experience Index Selection Approach		
	Sequential	*Random*	
		Uniform	*Priority*
Deep Q-Network (DQN) [12]	✓		
Double DQN[27]	✓		
Dueling DQN[28]	✓		
Prioritized Experience Replay (PER) [29]	✓		
Deep Deterministic Policy Gradient (DDPG) [30]	✓		
Twin Delayed Deep Deterministic Policy Gradient (TD3) [18]	✓		
Trust Region Policy Optimisation (TRPO) [31]	✓		
Proximal Policy Optimisation (PPO) [32]	✓		
Episodic Memory Deep Q-Network (EMDQN) [33]	✓		
Advantage Actor-Critic (A2C) + Prioritized Stochastic Memory Management (PSMM) [20], [34]			✓
DQN + Dual Memory Structure (DMS) [12], [35]	✓		✓
DDPG + Experience Replay Optimisation (ERO) [30], [36]	✓		
Combined Experience Replay (CER) [37]	✓		
Attentive Experience Replay (AER) [38]	✓		
Selective Experience Replay (SER) [17]		✓	✓
Prioritized Sequence Experience Replay (PSER) [39]	✓		

The advancements in ER are incorporated in many popular RL algorithms, such as DQN [12], dueling DQN [28], double DQN [27], twin delayed deep deterministic policy gradient (TD3)[18], deep deterministic policy gradient (DDPG) [30], proximal policy optimization (PPO)[32], episodic memory deep Q-Network (EMDQN) [33], and trust region policy optimization (TRPO) [31], still use the naïve ER uniform random sampling strategy. Other algorithms, such as prioritized experience replay (PER) [36], prioritized sequence experience replay (PSER) [37], experience replay optimization (ERO) [38], and attentive experience replay (AER) [39], implement prioritized strategies. Equally, prioritized stochastic memory management (PSMM) [20], combined experience replay (CER) [37], selective experience replay (SER) [17], and episodic memory control (EMC)[40] use experience retention

strategies (memory management strategies). In contrast, some replay strategies focus on the structure of the replay memory instead of the content[35], [42], [43]. ERO has proven superior among prioritized selection algorithms, owing to its easy adaptation and generalization to multiple environments [23].

B. Experience Retention Strategies and Algorithms

Experience retention plays a critical role in the success of ER algorithms. We can only select the experiences available in the replay buffer for training. If valuable experiences are maintained in the buffer, there will be a higher probability of sampling a mini-batch full of relevant experience to train the RL agent. In contrast, the worst training could be given to the agent. Therefore, it is imperative to investigate and unearth innovative ways to improve existing retention strategies or, better still, develop new ones.

The naïve approach of randomly selecting buffered experiences uniformly or managing the replay memory with a simple FIFO strategy is simple but less successful than the prioritized approach for managing the replay buffer [16], [29], [44]. Recent enhanced works on experience replay have relied on rule-based strategies that directly prioritize transitions through sampling strategies or indirectly through retention strategies [36]. However, some prioritized strategies incorporate a certain degree of randomness during implementation, using hyperparameters to regulate prioritization. Prioritization relies on features such as the temporal difference error (TDE), reward signal, similarities or diversity of states[18], or a combination of any of these features [20], [36]. A comprehensive study by de Bruin et al. [16] outlines age, exploration, and surprise as the criteria for retaining experiences in the buffer.

Retention depends on the duration for which an experience remains in the buffer. FIFO, Full DB, and Reservoir are strategies that rely on age. Although FIFO uses sequential indexing to remove old experiences without regard for their contribution to learning, Reservoir overwrites experiences in a uniformly random fashion and has limited state-action coverage. The Full DB method accommodates all experiences until the end of the training but may retain irrelevant experiences.

It is worth noting that there are better choices than the exploration criteria when dealing with problems that require minimal interaction with the agent's environment [16]. TDE is an expression of the surprise between the targeted and predicted q-values. Overfitting can occur if not parameterized and regulated [29], [40]. Actor-Critic Method Two major approaches in RL, value-based and policy-based methods, have been widely explored. Value-based methods estimate the value function, while policy-based methods directly optimize the agent's policy [1]. However, each approach has its limitations. Value-based methods tend to suffer from overestimation, the curse of dimensionality, and are often computationally expensive. On the other hand, policy-based methods can be inefficient in exploring the environment and may need help with convergence [13], [18], [45]. To address these challenges, the actor-critic algorithm, a hybrid approach, combines the strengths of value-based and policy-based methods [13], [18], [46]. It consists of two key components: the actor and the critic. The actor represents the policy and selects actions based on the observed states. The critic estimates the value function, providing feedback to the actor by evaluating the chosen actions.

The actor is typically implemented as a parametric model, like a neural network, which maps states to a probability distribution over actions. It explores the environment, collects experiences, and adjusts its policy based on the rewards it receives. The critic, represented by a parametric model, estimates the value function by approximating the expected cumulative reward associated with states or actions. By combining the strengths of both approaches, actor-critic algorithms can achieve faster convergence, more stable learning, and better performance in a wide range of RL problems. This hybrid approach has found successful applications in various domains, including robotics control, game playing, and natural language processing [15], [34].

III. METHODOLOGY

This section briefly introduces ERO and PSMM while paying particular attention to the selection and retention strategies used. It further presents the proposed framework and implemented algorithms.

A. Experience Replay Optimization

ERO is an experience selection method that relies on Reward and TDE for prioritization [36]. Unlike other TDE prioritization sampling strategies that favor experiences with higher TD errors, ERO selects less surprised TDE experiences and uses a novel replay policy network for the prioritization process. A mini-batch of high-priority transitions (transitions with vector 1) was created, and its elements were uniformly sampled to train the agent [23]. After the agent interacts with the environment, the transitions are stored in the replay buffer and subsequently prioritized through a Boolean (0, 1) vectorization process using the replay policy. During training, the replay policy receives feedback from the environment for policy evaluation.

Since the performance of a sampling strategy is highly dependent on the implementing algorithm and the benchmark environment [16], there is the need for a sampling method that can learn and adapt to different algorithms and environments - ERO does rightly so. ERO still uses the FIFO retention strategy despite its novel adapting strategy and superior performance over prioritized sampling methods such as PER [36], [44].

Hence, when the replay memory exceeds its capacity, the oldest transition is sequentially replaced with a new transition, irrespective of its importance in learning. Nonetheless, when relevant transitions are retained and frequently sampled using an intelligent index selection strategy, we are optimistic that the agent's performance and convergence rate will improve [14], [45], [47]. Therefore, there is a need to augment ERO with a memory management mechanism that is better than FIFO[16]. The beauty and novelty of the ERO algorithm depend on its replay policy network, which relies on Eq. (1) to Eq. (4). Table II explains the notations used in the equations, and the replay policy update is presented in Algorithm 1 [36].

$$\lambda = \left\{ \phi\left(f_{B_i} \middle| \theta^{\phi}\right) \middle|\, B_i \in B \right\} \in \mathbb{R}^N \quad (1)$$

$$B^s = \{B_i | B_i \in B \wedge I_i = 1\} \quad (2)$$

$$r^r = r_\pi^c - r_{\pi'}^c \quad (3)$$

$$P(I|\phi) = \sum_{j:B_j \in B^{batch}} r^r \nabla\theta^\phi [I_i \log\phi + (1 - I_i)\log(1 - \phi)] \quad (4)$$

where, ϕ denotes the function approximator, B_i is a transition in the replay buffer B. θ^ϕ denotes the parameters of ϕ, f_{B_i} is a feature vector, and N is the number of transitions in a mini-batch. The priority score function is expressed as $\phi(f_{B_i}|\theta^\phi) \in (0,1)$, where the priority score is represented by Lambda (λ). $I_i \in \{0,1\}$ is the Bernoulli distribution of sample B^s. The replay reward, cumulative reward of the current policy, and cumulative reward of the previous policy are denoted by r^r, r_π^c, and $r_{\pi'}^c$ respectively.

Algorithm 1: PolicyUpdate

Input:
  Cumulative reward of current policy r_π^c
  Cumulative reward of previous policy $r_{\pi'}^c$
Output:
  Sample subset B^s
Calculate replay reward based on (3)
For (each replay updating step) do
  Randomly sample batch $\{B_i\}$ form B
  Update replay policy based on (4)
End
Sample subset B^s from B using (2)

B. Prioritized Stochastic Memory Management

Experience replay selection strategies are often the focus of researchers. However, it is essential to note that a poorly designed retention technique can negatively impact the performance of the learning agent [16], [17], [20]. One proposed method for effective replay memory management is prioritized stochastic memory management (PSMM), which was introduced by Ko and Chang [35]. The PSMM employs a stochastic approach to remove the history with the least TDE or return when the replay memory is full, using the probability computed in Eq. (5).

$$P_i = \frac{exp\left(-\rho(\alpha_{actor} R_{norm}^{(i)} + \alpha_{critic} TDE_{norm}^{(i)})\right)}{\sum_{i=1}^{c} exp\left(-\rho(\alpha_{actor} R_{norm}^{(i)} + \alpha_{critic} TDE_{norm}^{(i)})\right)} \quad (5)$$

The computation of the probability for elimination (P_i) in Kwon and Chang's method involves the utilization of historical information through return and temporal difference error (TDE) metrics [25]. These metrics were normalized to restrict their values from 0 to 1, facilitating unbiased evaluations and promoting stable memory management. The method employs several hyperparameters, such as α_{actor}, α_{critic}, and ρ, which are optimized for improved performance. The α_{actor} and α_{critic} determine the relative weights assigned to the actor and critic components, respectively, whereas ρ represents a probability tuning parameter. This approach ensures the effective utilization of historical data and facilitates the optimization of the method's parameters.

The computations for $R_{norm}^{(i)}$ and $TDE_{norm}^{(i)}$ are shown in Eq. (6) and Eq. (7), respectively.

$$R_{norm}^{(i)} = \begin{cases} \frac{R^i - R_{min}}{R_{max} - R_{min}}, & if\ R_{max} \neq R_{min} \\ 0 & otherwise \end{cases} \quad (6)$$

$$TDE_{norm}^{(i)} = \begin{cases} \frac{TDE^i - TDE_{min}}{TDE_{max} - TDE_{min}}, & if\ TDE_{max} \neq TDE_{min} \\ 0 & otherwise \end{cases} \quad (7)$$

$$P_i = \frac{\left(R_{norm}^{(i)}\right)^\alpha}{\sum_{i=1}^{c}\left(R_{norm}^{(i)}\right)^\alpha} \quad (8)$$

$$P_i = \frac{exp\left(-\rho(\alpha_{actor} R_{norm}^{(i)})\right)}{\sum_{i=1}^{c} exp\left(-\rho(\alpha_{actor} R_{norm}^{(i)})\right)} \quad (9)$$

TABLE II. SYMBOLS AND NOTATIONS USED IN THIS SECTION

Notation	Explanation
ϕ	Function approximator
B	Replay buffer
B_i	A transition in the replay buffer B
θ^ϕ	Parameters of the function approximator
f_{B_i}	Feature vector
λ	Priority score
r^r	Cumulative reward
r_π^c	Cumulative reward of current policy
$r_{\pi'}^c$	Cumulative reward of previous policy
B^s	A specified batch size of sampled transitions
ρ	Probability tuning parameter
α	Parameter for tuning the probability of elimination

C. Proposed Framework

Researchers have recently harnessed and combined various algorithms' strengths to create resilient, stable, and generalized hybrid algorithms. Our proposed framework amalgamates the ERO framework and enhances the FIFO retention strategy.

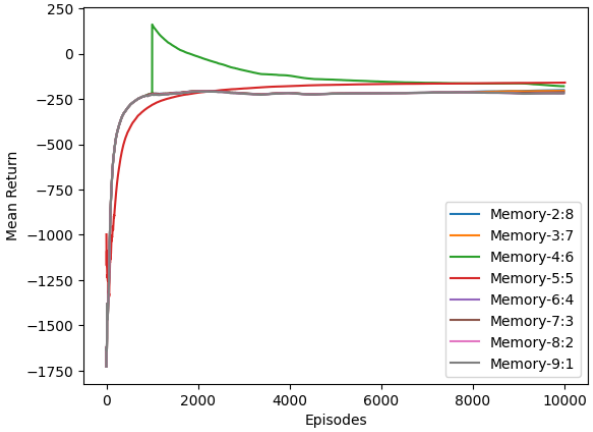

Fig. 1. A preliminary experiment was conducted to identify the optimum experience retention ratio. When the replay memory capacity is reached, the buffered experiences are overwritten using a ratio. A ratio of 2:8 means 20% of the old experiences are sequentially overwritten with new experiences, and 80% are retained. However, for a ratio of 8:2, only 20% of the buffered experiences are retained.

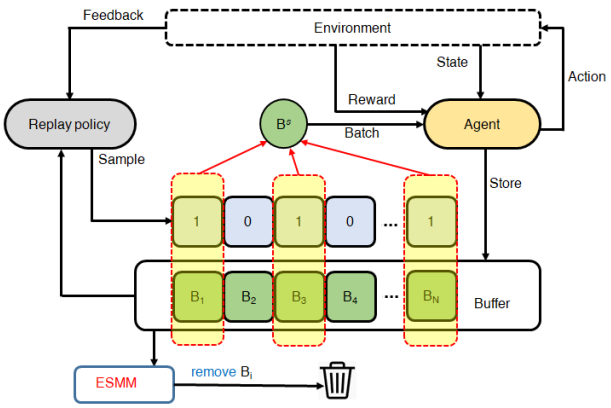

Fig. 2. Proposed Framework: experience replay optimization with enhanced sequential memory management (ERO-ESMM). Transitions from the environment are stored in the replay buffer. Mini-batches from the transitions are vectorized, prioritized by the replay policy, and sampled uniformly at random to train the agent. After training, the replay policy receives feedback for policy evaluation. When the memory is full, ESMM ensures that transitions in the first half of the replay memory are sequentially overwritten with new ones.

The ESMM, PSMM(α), and PSMM(ρ) retention strategies were developed to create the new framework. While PSMM(α) and PSMM(ρ) use Eq. (8) and Eq. (9), respectively, ESMM extends the FIFO retention strategy by sequentially overwriting older transitions in the first half of the memory when the memory is full. The one-half was arrived at after preliminary experiments were conducted using different ratios of the replay buffer for experience retention. Fig. 1, which shows the preliminary experiment results, confirms that an even distribution of old and new experiences in the replay buffer enhances the performance of the RL agent.

The ESMM, PSMM(α), and PSMM(ρ) strategies and FIFO, Full DB, and Resv were further investigated to ascertain their effects on the ERO-enhanced DDPG algorithm. The strategy with the highest mean return, ESMM, was incorporated into the ERO framework to propose an improved framework, an experience replay optimization with enhanced sequential memory management (ERO-ESMM). The proposed framework and its algorithm are shown in Fig. 2 and Algorithm 2, respectively.

Algorithm 2: ERO-ESMM Enhanced DDPG

Initialize policy π, replay policy ϕ and buffer B
For (each iteration) do
  For (each time-step t) do
    Select action a_t according to π and state s_t
    Execute action a_t and observe s_{t+1} and r_t
    If (B is full) then
      If (index i+1 == ½ len(B)) then
        i=0
      Else
        i =(i+1) mod len(B)
      End
      Store transition(s_t, a_t, r_t, s_{t+1}) at B_i
    End
    If (episode is complete) then
      Calculate the cumulative reward r_π^c
      If ($r_{\pi'}^c \neq null$) then
        B^s = PolicyUpdate($r_{\pi'}^c$,r_π^c , B)
      End
      Set $r_{\pi'}^c \leftarrow r_\pi^c$
    End
  End
  For (each training step) do
    Uniformly sample a batch $\{B_i\}$ from B^s
    Update the critic of π
    Update the actor of π
    Update the target networks
    Update the λ for each transition in $\{B_i\}$
  End
End

D. Setup of RL Environment

To ascertain the efficiency of the proposed framework, we conducted a series of experiments in the Pendulum-v0, MountainCarContinuous-v0, LunarLandarContinuous-v2, and the BipedalWalker-v3 environments [48] of the OpenAI Gym as a platform for the evaluation and analysis of results. Screenshots of these environments are shown in Fig. 3.

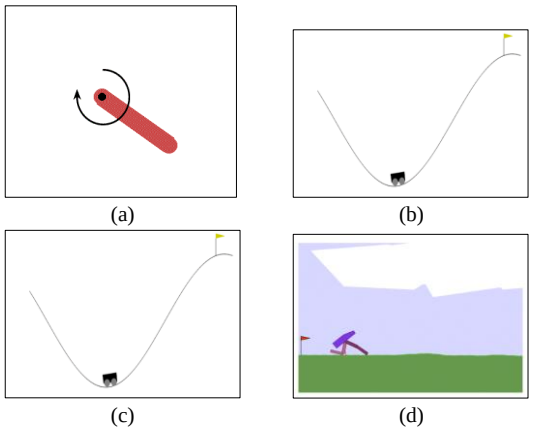

Fig. 3. Screenshots of two Classic Control (top row) and two Box2D (down row) environments from the OpenAI Gym. Fig. 3(a) and (b) represent the Pendulum-v0 and MountainCarContinuous-v0 environments, respectively. Fig. 3(c) and (d) represents the LunarLandarContinuous-v2 and the BipedalWalker-v3 environments respectively.

Pendulum-v0 presents a classical inverted pendulum swing-up problem, which demands that the agent persistently swing up the pendulum from an initial arbitrary position until it attains an upright position while its 3-dimensional observation space comprises angle, acceleration, and angular velocity, its action space is continuous, ranging between -2.0 (anti-clockwise torque) and 2.0 (clockwise torque). The agent's rewards depend on its actions and the associated state.

The MountainCarContinuous-v0 environment is another benchmark classical control environment that requires the RL agent to apply actions to a car to reach the top of a hill as quickly as possible. It is an extension of the classic "MountainCar-v0" environment but with continuous action space, making it more suitable for problems requiring continuous control. It consists of a 2-dimensional observation space of the car's position and velocity and a continuous action space between -1.0 and 1.0. The RL agent is negatively rewarded each time an action is taken until the car reaches the

top of the hill. Hence, the agent applies continuous efficient actions to overcome the car's inertia and climb the hill.

LunarLanderContinuous-v2 is an extension of the original Box2D discrete action space LunarLander-v2 environment, where the agent controls a lunar lander attempting to land on a designated landing pad on the moon's surface. In contrast to the discrete version, this environment allows the agent to apply a range of continuous actions (a throttle that varies between 0 and 1 and a rotation angle between -1 and 1) instead of selecting from a fixed set of discrete actions. It has an 8-dimensional observation space that includes information about the lander's position, velocity, orientation, angular velocity, whether the legs are touching the ground, and whether the lander has successfully landed. The RL agent receives positive rewards for moving closer to the landing pad and a significant positive reward for landing safely. Negative rewards are given for using fuel, and a slight negative reward is given for each time step.

BipedalWalker-v3 is similarly a Box2D environment where the agent controls a bipedal robot with four legs and must learn to make it walk and navigate through a complex terrain while avoiding obstacles. The challenge lies in learning a coordinated sequence of actions to control the robot's joints and achieve stable and efficient locomotion. This environment is represented by a 24-dimensional observation space, which contains information about the position, velocity, angle, angular velocity, and state of the joints, feet, and lidar sensors. The agent receives positive rewards for progressing forward and avoiding obstacles. However, negative rewards are given for using excessive torque or falling, encouraging the agent to discover stable and effective locomotion strategies.

The suitability of these environments for the DDPG algorithm is attributable to the availability of continuous tasks[30]. Thus, the selection of these environments was motivated by their compatibility with the requirements of the DDPG algorithm and their standard use in previous studies for evaluating RL algorithms [49].

E. Parameter Setting

Because the proposed framework extends the ERO-enhanced DDPG algorithm, the experimental configurations were based on the OpenAI DDPG stable baseline[50], and the hyperparameters for the sampling and retention methods were in line with ERO and PSMM, respectively. However, the memory size and number of time steps were adjusted during implementation.

In Table III, six notations and their explanations are clearly shown to facilitate comprehension of our visualizations. Aside from the Full DB, which has the memory size and number of time steps set to 2 × 106, the other five retention strategies were evaluated with a memory size of 1 × 106 and time steps of 2 × 106. Similar to PSMM[25], we set $\boldsymbol{\rho = 2.0}$, $\boldsymbol{\alpha_{actor} = 0.5}$ and $\boldsymbol{\alpha_{critic}}$ = 0.5 when implementing PSMM($\boldsymbol{\rho}$). In the PSMM(α), we rely on an α value 0.6[21]. Other parameters for the evaluation of experiments were done on an Intel(R) Xeon(R) CPU E3-1220 v6 @ 3.00GHz(4CPUs) with 32GB RAM and a Windows Server 2012R2 operating system.

TABLE III. EXPERIENCE RETENTION STRATEGIES EVALUATED IN THE STUDY

Notation	Explanation
FIFO	Sequentially overwrite old experiences with new ones.
ESMM	Replace experiences in the first half of the buffer with new ones.
Full DB	Experiences are not overwritten. The replay memory stores all experiences.
Resv	Experiences are uniformly, at random, overwritten with new ones.
PSMM(α)	Experiences are stochastically overwritten based on (8)
PSMM(ρ)	Experiences are stochastically overwritten based on (9)

IV. RESULTS AND DISCUSSION

This section examines the comparative effectiveness of the retention strategies within each environment. The metric for quantifying the performance of the evaluated strategies is the mean return. The return, also known as cumulative reward or cumulative return, is a measure of the overall success of the RL agent in achieving its goals. It is the summation of rewards received by the agent at each time step from the start of an episode until its termination[1], [51], [52]. The return is typically used to evaluate and compare the performance of different algorithms and policies in a specific reinforcement learning task. The mean return can be derived as follows:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \quad (10)$$

$$Mean\ return = \frac{G_t}{number\ of\ episodes} \quad (11)$$

where, G_t is the return, R_{t+1} is the reward at time step *t*, and *T* is the final time-step.

The performance of each retention strategy was evaluated based on the progressive average returns computed using Eq. (11). The results are illustrated in Fig. 4 to Fig. 7.

In the Pendulum environment, as shown in Fig. 4, the ESMM and PSMM(α) strategies exhibit superior performance compared to FIFO, while Full DB and PSMM(ρ) strategies perform worse. The Resv strategy exhibits limited effectiveness as it faces challenges in learning optimal policies to stabilize the pendulum. This struggle is reflected in a mean return of -1061.29.

Regarding the MountainCarContinuous-v0 environment, as indicated in Fig. 5, the ESMM strategy outperforms all other models, including FIFO. The Full DB and Resv strategies show some improvement over FIFO, while PSMM(α) and PSMM(ρ) perform worse with rewards of -6.05 and -6.46, respectively.

In the LunarLanderContinuous-v2 environment, as shown in Fig. 6, the ESMM, PSMM(α), and PSMM(ρ) strategies demonstrate superior performance compared to FIFO. The Full DB model performs worse than FIFO and PSMM(ρ), while the Resv model exhibits the poorest performance.

Likewise, in the BipedalWalker-v3 environment, the ESMM and PSMM(α) models perform better than the FIFO strategy. However, the Full DB and Resv models perform worse than ESMM and PSMM(α), with PSMM(ρ) showing the poorest performance among all models in this environment.

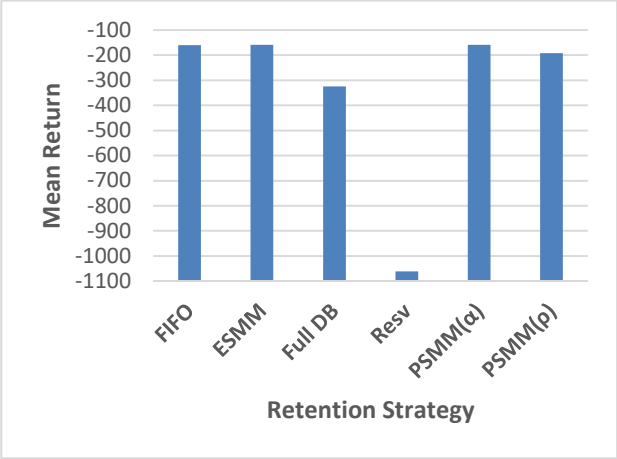

Fig. 4. Performance comparison of the six retention strategies evaluated on the Pendulum-v0 environment.

Fig. 5. Performance comparison of the six retention strategies evaluated on the MountainCarContinuous-v0 environment.

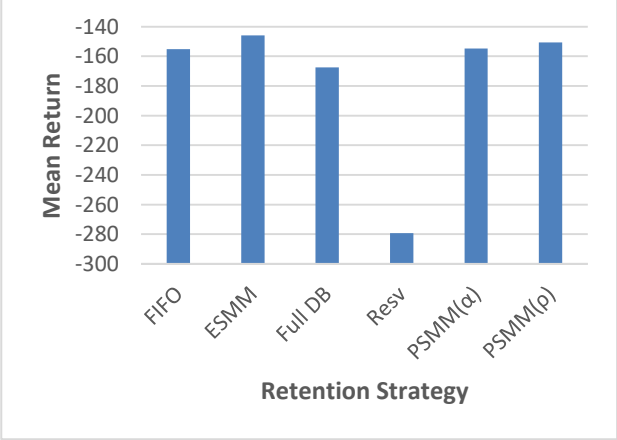

Fig. 6. Performance comparison of the six retention strategies evaluated on the LunarLandarContinuous-v2 environment.

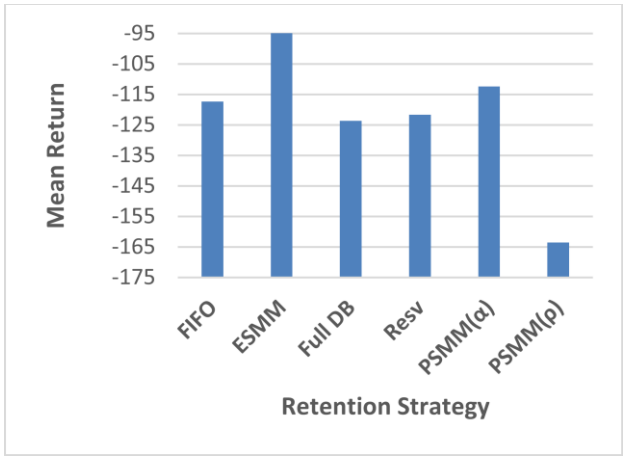

Fig. 7. Performance comparison of the six retention strategies evaluated on the BipedalWalker-v3 environment.

The results indicate significant model performance variation across different OpenAI Gym environments. The ESMM strategy generally exhibits better and more stable performance across the experimented environments, which can be attributed to its fair distribution of experiences in the replay buffer. While the PSMM(α) model performs better in the Pendulum environment, the PSMM(ρ) strategy performs better than other models, except ESMM, in the Lunar Lander Continuous-v2 environment.

Conversely, the Full DB model tends to perform worse in most environments because all experiences are stored, including the worst experiences obtained during the early stages of the RL agent's training. The Resv strategy's performance shows inconsistency across the environments due to its entirely random strategy for identifying the experience to remove when the replay buffer is full.

V. Conclusion

In this study, we developed three new retention strategies to improve the effectiveness of experience replay. These strategies include ESMM, PSMM(α), and PSMM(ρ). The development of these strategies is significant, as they provide new alternatives for managing the memory of an RL agent in a reinforcement learning setting. These strategies addressed specific challenges encountered in experience replay, such as the trade-off between memory usage and retention of relevant experiences.

This study also investigated the effects of six different retention strategies on the ERO-enhanced DDPG algorithm. These strategies include FIFO, Full DB, Resv, PSMM (α), PSMM(ρ), and our proposed method (ESMM). The results of this investigation are significant because they provide insights into the comparative performance of different retention strategies and help identify the most effective strategy. This information can guide the design of more efficient reinforcement learning algorithms.

Finally, we propose an enhanced framework incorporating the highest-performing retention strategy into the ERO framework. This enhanced framework, called experience-replay optimization with enhanced sequential memory management (ERO-ESMM), significantly contributes to reinforcement learning. By integrating the best-performing retention strategy, the framework offers a more optimized approach to experience replay, leading to improved performance in reinforcement-learning tasks.

Overall, the experimental results suggest that developing new retention strategies, combined with their investigation and incorporation into existing frameworks, can significantly improve the performance of reinforcement learning algorithms. These results have implications for future research and demonstrate the importance of exploring new techniques for optimizing reinforcement learning. In the future, we will use a separate neural network to predict the index of the experience to delete from the replay buffer when it exceeds its limit.

REFERENCES

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: an Introduction. MIT press, 2018.

[2] B. Varghese and S. Krishnakumar, 'Fast Fractal Coding of MRI Images using Deep Reinforcement Learning', IJACSA, vol. 12, no. 4, 2021, doi: 10.14569/IJACSA.2021.0120492.

[3] W. E. Fathy and A. S. Ghoneim, 'A Deep Learning Approach for Breast Cancer Mass Detection', International Journal of Advanced Computer Science and Applications (IJACSA), vol. 10, no. 1, Art. no. 1, 31 2019, doi: 10.14569/IJACSA.2019.0100123.

[4] S. Luo, 'Lung Cancer Classification using Reinforcement Learning-based Ensemble Learning', International Journal of Advanced Computer Science and Applications (IJACSA), vol. 14, no. 8, Art. no. 8, 54/30 2023, doi: 10.14569/IJACSA.2023.01408120.

[5] E. Jacinto, F. Martinez, and F. Martinez, 'Navigation of Autonomous Vehicles using Reinforcement Learning with Generalized Advantage Estimation', International Journal of Advanced Computer Science and Applications (IJACSA), vol. 14, no. 1, Art. no. 1, 31 2023, doi: 10.14569/IJACSA.2023.01401103.

[6] I. Rasheed, F. Hu, and L. Zhang, 'Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN', Vehicular Communications, vol. 26, p. 100266, 2020.

[7] Y. Han et al., 'Deep reinforcement learning for robot collision avoidance with self-state-attention and sensor fusion', IEEE Robotics and Automation Letters, vol. 7, no. 3, pp. 6886–6893, 2022.

[8] G. E. Setyawan, P. Hartono, and H. Sawada, 'Cooperative Multi-Robot Hierarchical Reinforcement Learning', International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 9, Art. no. 9, Dec. 2022, doi: 10.14569/IJACSA.2022.0130904.

[9] D. An, F. Cui, and X. Kang, 'Optimal scheduling for charging and discharging electric vehicles based on deep reinforcement learning', Frontiers in Energy Research, vol. 11, 2023, Accessed: Dec. 13, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fenrg.2023.1273820.

[10] H. Zhao et al., 'Combination optimization method of grid sections based on deep reinforcement learning with accelerated convergence speed', Frontiers in Energy Research, vol. 11, 2023, Accessed: Dec. 13, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fenrg.2023.1269854.

[11] V. L. Narayanan, 'Reinforcement learning in wind energy - a review', International Journal of Green Energy, vol. 0, no. 0, pp. 1–24, 2023, doi: 10.1080/15435075.2023.2281329.

[12] V. Mnih et al., 'Human-level control through deep reinforcement learning', Nature, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[13] Z. Wang et al., 'Sample efficient actor-critic with experience replay', in 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, 2017.

[14] D. Yang, X. Qin, X. Xu, C. Li, and G. Wei, 'Sample Efficient Reinforcement Learning Method via High Efficient Episodic Memory', IEEE Access, vol. 8, pp. 129274–129284, 2020, doi: 10.1109/ACCESS.2020.3009329.

[15] T. de Bruin, Sample efficient deep reinforcement learning for control. 2019. doi: 10.4233/uuid.

[16] T. De Bruin, J. Kober, K. Tuyls, and R. Babuška, 'Experience selection in deep reinforcement learning for control', Journal of Machine Learning Research, vol. 19, pp. 1–56, 2018.

[17] D. Isele and A. Cosgun, 'Selective experience replay for lifelong learning', in Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 3302–3309.

[18] S. Fujimoto, H. V. Hoof, and D. Meger, 'Addressing Function Approximation Error in Actor-Critic Methods', arXiv preprint arXiv:1802.09477, 2018.

[19] C. Kang, C. Rong, W. Ren, F. Huo, and P. Liu, 'Deep Deterministic Policy Gradient Based on Double Network Prioritized Experience Replay', IEEE Access, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3074535.

[20] T. Kwon and D. E. Chang, 'Prioritized Stochastic Memory Management for Enhanced Reinforcement Learning', in 2018 IEEE International Conference on Consumer Electronics - Asia, ICCE-Asia 2018, 2018, pp. 7–10. doi: 10.1109/ICCE-ASIA.2018.8552124.

[21] L. Lin, 'Self-improvement Based On Reinforcement Learning, Planning and Teaching', in Machine Learning Proceedings 1991, vol. 321, 1992, pp. 323–327. doi: 10.1016/b978-1-55860-200-7.50067-2.

[22] Z. Li and D. Hoiem, 'Learning without forgetting', Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9908 LNCS, pp. 614–629, 2016, doi: 10.1007/978-3-319-46493-0_37.

[23] J. Kirkpatrick et al., 'Overcoming catastrophic forgetting in neural networks', Proceedings of the National Academy of Sciences of the United States of America, vol. 114, no. 13, pp. 3521–3526, Mar. 2017, doi: 10.1073/PNAS.1611835114.

[24] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, 'Measuring catastrophic forgetting in neural networks', in 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, 2018. doi: 10.1609/aaai.v32i1.11651.

[25] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, 'REMIND Your Neural Network to Prevent Catastrophic Forgetting', Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12353 LNCS, pp. 466–483, 2020, doi: 10.1007/978-3-030-58598-3_28.

[26] C. Greco, B. Plank, R. Fernández, and R. Bernardi, 'Measuring Catastrophic Forgetting in Visual Question Answering', in Lecture Notes in Electrical Engineering, vol. 714, 2021. doi: 10.1007/978-981-15-9323-9_35.

[27] H. V. Hasselt, A. Guez, and D. Silver, 'Deep Reinforcement Learning with Double Q-Learning', in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), 2016, pp. 2094–2100.

[28] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Frcitas, 'Dueling Network Architectures for Deep Reinforcement Learning', 33rd International Conference on Machine Learning, ICML 2016, vol. 4, no. 9, pp. 2939–2947, 2016.

[29] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, 'Prioritized experience replay', 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, pp. 1–21, 2016.

[30] Josh Achiam, 'Deep Deterministic Policy Gradient', OpenAI Spinning Up. [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/ddpg.t.

[31] P. Schulman, John and Levine, Sergey and Abbeel, Pieter and Jordan, Michael and Moritz, 'Trust region policy optimization', in International conference on machine, 2015, pp. 1889–1897. doi: 10.3917/rai.067.0031.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, 'Proximal Policy Optimization Algorithms', arXiv preprint arXiv:1707.06347, pp. 1–12, 2017.

[33] Z. Lin, T. Zhao, G. Yang, and L. Zhang, 'Episodic memory deep q-networks', in IJCAI International Joint Conference on Artificial Intelligence, 2018, pp. 2433–2439. doi: 10.24963/ijcai.2018/337.

[34] V. Mnih, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, and D. Silver, 'Asynchronous Methods for Deep Reinforcement Learning', in 33rd International Conference on Machine Learning, ICML 2016, 2016, pp. 1928--1937.

[35] W. Ko and D. E. Chang, 'A dual memory structure for efficient use of replay memory in deep reinforcement learning', in 2019 19th International Conference on Control, Automation and Systems (ICCAS), 2019, pp. 1483--1486.

[36] D. Zha, K. H. Lai, K. Zhou, and X. Hu, 'Experience replay optimization', IJCAI International Joint Conference on Artificial Intelligence, vol. 2019-Augus, pp. 4243–4249, 2019, doi: 10.24963/ijcai.2019/589.

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

[38] P. Sun, W. Zhou, and H. Li, 'Attentive Experience Replay', in Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 5900–5907. doi: 10.1609/aaai.v34i04.6049.

[39] M. Brittain, J. Bertram, X. Yang, and P. Wei, 'Prioritized sequence experience replay', arXiv, 2019.

[40] T. de Bruin, J. Kober, K. Tuyls, and R. Babuska, 'The importance of experience replay database composition in deep reinforcement learning', Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS), pp. 1–9, 2015.

[41] B. Mavrin, H. Yao, and L. Kong, 'Deep Reinforcement Learning with Decorrelation', Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.07765.

[42] W. Olin-Ammentorp, Y. Sokolov, and M. Bazhenov, 'A Dual-Memory Architecture for Reinforcement Learning on Neuromorphic Platforms', Neuromorphic Computing and Engineering, vol. 1, p. 024003, 2021.

[43] N. Kamra, U. Gupta, and Y. Liu, 'Deep Generative Dual Memory Network for Continual Learning'. 2017. [Online]. Available: http://arxiv.org/abs/1710.10368.

[44] R. Liu and J. Zou, 'The Effects of Memory Replay in Reinforcement Learning', 2018 56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018, pp. 478–485, 2019, doi: 10.1109/ALLERTON.2018.8636075.

[45] H. R. Maei, 'Convergent Actor-Critic Algorithms Under Off-Policy Training and Function Approximation'. arXiv, Feb. 21, 2018. Accessed: Nov. 24, 2023. [Online]. Available: http://arxiv.org/abs/1802.07842.

[46] V. R. Konda and J. N. Tsitsiklis, 'Actor-critic algorithms', Advances in Neural Information Processing Systems, pp. 1008–1014, 2000.

[47] J. Tarbouriech, M. Pirotta, M. Valko, and A. Lazaric, 'A Provably Efficient Sample Collection Strategy for Reinforcement Learning', pp. 1–33, 2020.

[48] G. Brockman et al., 'OpenAI Gym', arXiv preprint arXiv:1606.01540, pp. 1–4, 2016.

[49] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, 'Deep reinforcement learning that matters', in 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, 2018.

[50] N. Raffin, Antonin and Hill, Ashley and Ernestus, Maximilian and Gleave, Adam and Kanervisto, Anssi and Dormann, 'Stable Baselines', GitHub repository, 2019, [Online]. Available: https://github.com/hill-a/stable-baselines.

[51] E. F. Morales and H. J. Escalante, 'A brief introduction to supervised, unsupervised, and reinforcement learning', in Biosignal processing and classification using computational learning and intelligence, Elsevier, 2022, pp. 111–129. Accessed: Nov. 25, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128201251000178.

[52] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, An introduction to deep reinforcement learning, vol. 11. 2018. doi: 10.1561/2200000071.