IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 14 November 2023, accepted 30 November 2023, date of publication 4 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3339248

RESEARCH ARTICLE

Enhanced Experience Prioritization: A Novel Upper Confidence Bound Approach

BÁLINT KŐVÁRI[1], BÁLINT PELENCZEI[2], AND TAMÁS BÉCSI[1], (Member, IEEE)
[1]Department of Control for Transportation and Vehicle Systems, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, 1111 Budapest, Hungary
[2]Systems and Control Laboratory, Institute for Computer Science and Control, 1111 Budapest, Hungary

Corresponding author: Tamás Bécsi (becsi.tamas@kjk.bme.hu)

This work was supported in part by the European Union within the framework of the National Laboratory for Autonomous Systems under Grant RRF-2.3.1-21-2022-00002; and in part by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, through the TKP2021 Funding Scheme, under Project BME-NVA-02. The work of Bálint Pelenczei was supported by the New National Excellence Program of the Ministry for Culture and Innovation under Grant ÚNKP-23-1-I-BME-19. The work of Tamás Bécsi was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences under Grant BO/00233/21/6.

ABSTRACT Value-based Reinforcement Learning algorithms achieve superior performance by utilizing experiences gathered in the past to update their so-called value-function. In most cases, it is accomplished by applying a sampling strategy to an experience buffer, in which state transitions are stored during the training process. However, the design of such methods is not so intuitive. General theoretic approaches tend to determine the expected learning progress from each experience, based on which the update of neural networks can be carried out efficiently. Proper choice of these methods can not only accelerate, but also stabilize the training significantly by increasing sampling efficiency, which indirectly leads to a reduction in time and computing capacity requirements. As one of the most critical aspects of using Machine Learning (ML) based techniques originates from the lack of decent computing power, thus endeavour to find optimal solutions has long been a researched topic in the field of Reinforcement Learning. Therefore the main focus of this research has been to develop an experience prioritization method acquiring competitive performance, besides having the overall cost of training considerably lowered. In this paper, we propose a novel priority value assignment concept for experience prioritization in Reinforcement Learning, based on the Upper Confidence Bound algorithm. Furthermore, we present empirical findings of our solution, that it outperforms current state-of-the-art in terms of sampling efficiency, while enabling faster and more cost-efficient training processes.

INDEX TERMS Deep learning, experience prioritization, experience replay, machine learning, Q-learning, reinforcement learning, sampling.

I. INTRODUCTION

In recent times, numerous researches in fields, including autonomous vehicle control [1], robotics [2], manufacturing [3], forecasting [4] and resource management [5], have pointed out several advantages provided by the utilization of Artificial Intelligence. Primarily, due to the complexity of newly arising optimization and control tasks, usage of traditional approaches, especially in real-time applications, is immensely limited or not even feasible by virtue of their significant computational demand throughout the online decision making process.

On the other hand, in case of Deep Learning-based algorithms, prediction time of a pre-trained neural network is in the order of milliseconds, hence real-time applicability is ensured. However, these methods also have a crucial disadvantage being, that during the offline training process, enormous time and computational capacity requirements emerge.

Firstly, according to the database being analyzed in [6], the average computational cost of Deep Learning models in Large-Scale Era, for which extended training data is available, is about $2.44 \cdot 10^{24}$ FLOPs. However, even for the

The associate editor coordinating the review of this manuscript and approving it for publication was Maria Chiara Caschera.

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

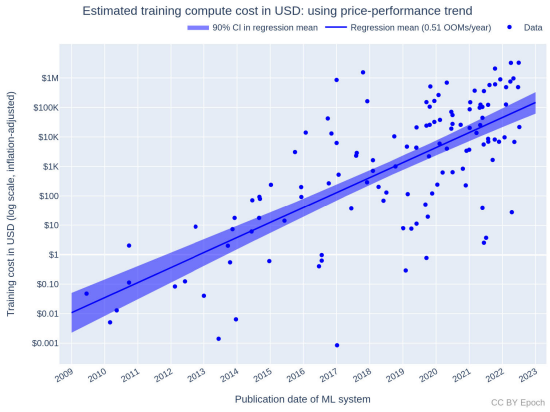

FIGURE 1. Inflation-adjusted training cost of Machine Learning systems published in recent years showing an increasing trend expressed in *USD* [7].

most powerful AI supercomputing platform (being NVIDIA HGX with 16x NVIDIA A100 GPUs), it takes nearly a year to train such an average model utilizing speed of 10 petaFLOPS. Since training time hampers the efficiency of Deep Learning research and development by making it more challenging to explore different architectures, hyperparameters, or data augmentation techniques, it is considered a key limitation.

Secondly, the true cost of training Deep Neural Networks extends beyond computational requirements and into economic considerations. Understanding and quantifying involved costs is vital for researchers, practitioners, and stakeholders to make informed decisions and optimize resource allocation in Deep Learning projects. As shown in Fig. 1, trends of recent years indicate highly increasing cost of trainings, thus efficiency of training algorithms evolves into a more and more critical objective to deal with. Therefore, we consider this task, to optimize the efficiency of such algorithms, as the main goal of this research.

In summary, our focus is on addressing the inefficiency inherent in training deep neural networks, a challenge marked by its considerable computational and time requirements. This bottleneck not only hinders progress of research and development in the field, but also imposes substantial additional costs in industrial applications, where deep neural networks have emerged as a preferred tool, given their superior performance in various problem domains. Industries heavily reliant on real-time applications, such as autonomous vehicles or live-streaming analytics, face potential performance setbacks due to these inefficiencies. Additionally, the economic implications are significant, encompassing increased computational costs and also a potential reduction in competitiveness. Hence, addressing this challenge both serves as a direct response to the economic viability and success of real-world applications, particularly in time-sensitive environments, and also ensures academic advancement.

Undoubtedly, the inefficiency of training methods can be derived from the way training data is treated across the update process of Neural Networks. Early solutions began to address this phenomenon, on the analogy of neurobiology, by utilizing so-called replay methods, hence facilitating memory formation and retrieval by reactivating neural patterns, as elucidated in [8]. Although experience replay has been adopted to all three major categories of Machine Learning, the relevance of such seems greatest in Reinforcement Learning, where enhancements may substantially boost performance. It can be achieved by dealing with sampling inefficiency, which is considered a bottleneck, that hinders the scalability and real-world applicability of RL algorithms. In contrast to other Machine Learning areas of concern, such as supervised learning, where sampling inefficiency is not a prominent concern, the challenge lies prominently within the realm of Reinforcement Learning. Furthermore, unlike supervised learning, where the average number of epochs for training a model typically ranges in the order of hundreds, it is well-acknowledged in RL, that attaining the desired behavior often necessitates orders of magnitude more episodes.

Lately, many researchers strive to proceed in this topic by uncovering more and more concepts with the same focus to resolve the issue of inefficient sampling. In the following section, we are going to provide a brief overview of advancements in this field with particular attention to experience prioritization techniques.

A. RELATED WORK

Firstly, in order to demonstrate the relevance and importance of Reinforcement Learning methods in practical applications, especially in real-time domains, a few promising areas are introduced. In [9], stochastic game theory has been utilized in order to model the nature of attacker-defender roles present in a potential smart electrical power grid cyberattack. The solution providing the best achievable defense strategies, according to their experiments both on the benchmark W&W 6-bus and a relatively large IEEE 30-bus system, was deep Q-learning, achieving superior performance compared to 3 other state-of-the-art Reinforcement Learning methods being PPO, AC and PG. In the domain of control systems, [10] introduced an adaptive approach with the use of Policy Iteration algorithm for the optimal regulation problem in discrete-time linear time-invariant systems. Notably, this algorithm ensures convergence, leveraging only state measurements without requiring prior knowledge of the system dynamics. In [11], a Reinforcement Learning framework has been proposed for trajectory planning of an aerial drone, integrating it to a pre-existent system of moving elements by resolving potential conflicts in advance. Despite the fact, that RL-based solutions have long been successfully deployed in such complex practical tasks, the problem of inefficient training scenarios yet remained unchanged, for which experience prioritization may provide a solution.

Shortly after the emergence of Reinforcement Learning, the need for prioritizing important state transitions became apparent. In [12], the prioritized sweeping algorithm has been introduced, which utilizes the change in absorption probabilities (Δ parameters) in order to determine importance

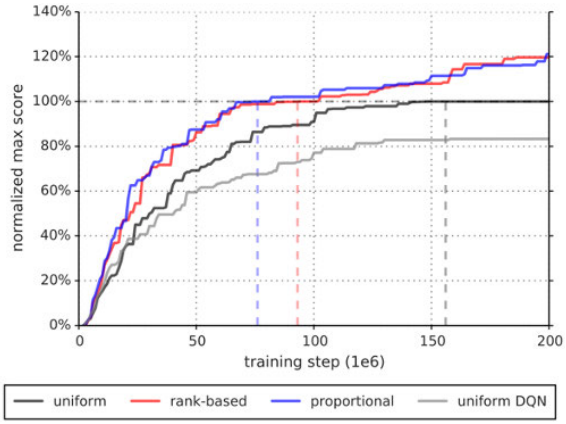

FIGURE 2. Results of Prioritized Experience Replay [14] showing an increase in training efficiency.

of a transition. Higher Δ values are considered more interesting from the update's point of view, as they indicate a potential change in absorption probabilities of the state's predecessors. Between two consecutive real-world observations, transitions are being sorted based on their priority until a processing step limit is reached, resulting in more important states being positioned at the top of the priority queue.

The concept of storing state transitions have not been implemented until the idea has been published much later, in [13], where the so-called experience replay memory has been proposed. This memory enabled agents to exploit experiences of the past by saving the elements that clearly define a transition. Afterwards, during the network updates, stochastic sampling is carried out on the memory using uniform distribution.

However, as mentioned earlier, this method treats all experiences as equal, being a limitation, that needed to be addressed in order to develop an optimal solution. A combination of the above mentioned theories has been demonstrated in [14], where authors suggested an approach, namely Prioritized Experience Replay (PER), to handle experiences of varying importance throughout the training. By assigning priority values to each state transition, thus formulating the probability distribution over the entire replay memory, they empirically managed to demonstrate increased training efficiency, as shown in Fig. 2. As PER quickly gained more and more popularity among researchers, it soon became the dominant method in the field, serving as the basis for many subsequent techniques, described later on. However, these techniques have still not been able to replace PER, being the most commonly used method in literature to date, hence it remained to be current state-of-the-art. Since this algorithm serves as the basis of comparison in this paper as well, for further details on methodology, refer to II-B1.

The authors of [16] adapted Prioritized Experience Replay for Deep Deterministic Policy Gradient agents, while [15] proposed Ape-X, an extension for PER to a distributed setting. In Ape-X, the agent is split into multiple entities, as shown on the schematic in Fig. 3, including several

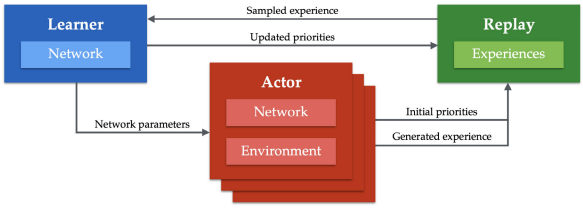

FIGURE 3. Schematic of the distributed agent concept in Distributed Prioritized Experience Replay [15].

actors and a single learner. Each actor has its own separate environment, but their replay memory is shared among the others. The role of the actors in the training is to generate training data, while the learner is responsible for the network updates and priority value assignment in the buffer. Authors of [17] developed Prioritized Sequence Experience Replay, which can also be considered as an expansion of the original PER idea. The authors have theoretically proven, that by prioritizing experience sequences, this method is guaranteed to converge faster compared to PER given, that the learning rate of the asynchronous Q-learning algorithm equals to 1 and all possible action sequences are executed exhaustively. Even though the proposed method demonstrated superior performance compared to the original PER algorithm across a substantial majority of Atari benchmark tasks, experimental findings reveal an increase in absolute training time and, consequently, resource utilization in case of expanding the decay window, thus managing longer experience sequences.

Various other approaches have been proposed tackling the problem of inefficient sampling, such as [18], in which two specialised actor-critic algorithms have been introduced, called TRACER and eNACER. In TRACER, an augmentation of the Advantage Actor Critic (A2C) algorithm has been carried out with the adaption of a Trust Region Policy Optimisation method, utilizing experience replay. eNACER is a modified version of the so-called Natural Actor-Critic method, where gradients are estimated with the procedure of least squares, instead of being calculated explicitly from the Fisher Matrix. In case of both methods, sampling efficiency has shown to be increased relative to other RL-based solutions in the comparison. Reference [19] presented another method, being STochastic Ensemble Value Expansion (STEVE), that combines superior performance of model-free approaches with advantageous sampling complexity of model-based algorithms. Based on a statistical analysis, they showed a massive increase in sampling efficiency compared to model-free solutions on a variety of complex tasks. For a comprehensive overview of other potential solutions to inefficient sampling, see [20].

In addition to the methods described so far, there are a handful of entirely different experience prioritization techniques, including the one developed in [21], where a so-called Energy-Based Prioritization(EBP) framework has been utilized in a robotic arm manipulation environment. The core idea is to determine total energy of a trajectory by computing cumulative transition energy based on principles

of physics according to 1 as:

$$E_{traj}(\tau) = E_{traj}(s_0, s_1, \dots s_T) = \sum_{t=1}^{T} E_{tran}(s_{t-1}, s_t) \quad (1)$$

where τ is the trajectory and s_i are the states of time step i. The agent then stores these energy values for the corresponding state transitions and prioritizes trajectories with higher energy values, which eventually serve directly as a probability distribution during sampling. Reference [22] introduced the concept of Curiosity-Driven Prioritization (CDP), that mainly focuses on infrequent experiences. By estimating density ρ of each transition via a Variational Gaussian Mixture Model (V-GMM), the agent aims to prioritize experiences with lower densities. In order to avoid noises caused by outliers of high deviation in density values and therefore ensure robustness, a rank-based prioritization is used after sorting the replay memory based on complementary densities ($\bar{\rho} = 1 - \rho$). Reference [23] proposed Topological Experience Replay (TER), which prioritizes state transitions by constructing a graph from gathered training data and applying a reverse sweep algorithm to update a Q-function. The graph is structured in a way, that vertices represent environment states, while edges are equvivalent of state transitions aka experiences. The algorithm constantly keeps track of terminal and root nodes, hence the task is to follow a sampling strategy by determining a sequence of worthwhile transitions. In this case, it is achieved by applying reverse Breadth-First search (BFS) starting from terminal states, followed by its predecessors. Lastly, [24] defined learn-ability of a training sample and introduced an algorithm that allows the agent to leverage information content on outstanding efficiency by distinguishing unlearnable and noisy samples. Learn-ability, according to their hypothesis, is calculated from so-called reducible TD-error, which theoretically measures how much the TD-error can be expectedly be decreased over training. For the sake of formalizing Reducible Loss (ReLo), which serves as a priority metric throughout the update process, two separate losses needs to be determined: one with respect to the action network L_θ, and the other one with respect to the target network $L_{\hat{\theta}}$, from which ReLo is computed according to 2 as:

$$ReLo = L_\theta - L_{\hat{\theta}} \quad (2)$$

In summary, various approaches, including PER, have been proposed to address the problem of inefficient sampling in Reinforcement Learning. These methods have demonstrated improved training efficiency and sampling performance, leading to advancements in the field, although prevailing experience prioritization method has remained to be the approach of Prioritized Experience Replay, which therefore is still considered state-of-the-art.

B. CONTRIBUTION

Regardless of the specific training algorithm used, one cannot argue that sampling strategy has a crucial role in training efficiency. Moreover, recent developments in the field of experience prioritization have seen a limited number of fundamentally new ideas, with many existing solutions relying heavily on the well-established concept of Prioritized Experience Replay. This claim can be attributed to the fact, that although bountiful approaches have been proposed over time, as detailed in I-A, such methods focusing on increasing sampling efficiency have been improving at a relatively slow pace compared to other elements critical in Reinforcement Learning, where breakthroughs have been accomplished in various subfields, including algorithms [25], [26], neural network architectures [27], [28] and also state space representations [29], [30]. Consequently, the predominant technique for prioritizing experiences in RL has persisted since the aforementioned method gained recognition. This enduring popularity can be attributed to several factors, including its inherent simplicity, suitability for a diverse set of algorithms and domains, and its potential for integration with various algorithms to improve their performance and stability. In summary, considering the above mentioned justifications, it can be concluded that Prioritized Experience Replay continues to be one of the prevailing state-of-the-art techniques.

As a result, the primary contribution of this research is to demonstrate the superiority of the proposed experience prioritization approach over the existing state-of-the-art method documented in literature. This is achieved through a comprehensive evaluation that exhaustively analyzes performance indicators of sampling processes. Therefore, contributions of this paper can be summarized as follows:

- This paper proposes a novel priority value assignment concept, that incorporates a new approach for the exploration-exploitation trade-off. The results show, that by leveraging limitations of the aforementioned state-of-the-art technique, sampling efficiency can be further enhanced using the proposed method.
- The Deep-Q Network algorithm has been trained along with the newly introduced experience prioritization technique and current state-of-the-art method to compare their performance on four commonly used Reinforcement Learning tasks present in literature.
- Strictly under identical conditions, a meticulous evaluation of these agents has been conducted, with specific focus on assessing the effectiveness of experience sampling strategies applied during the training process.

II. METHODOLOGY

A. REINFORCEMENT LEARNING

As a result of rapid increase in available computing capacity, Reinforcement Learning has garnered significant attention [31] from many researchers having interests in Machine Learning due to its promising properties and satisfactory results provided in various sequential decision-making problems.

Reinforcement Learning stands as a special branch within the field of Machine Learning. Unlike Supervised Learning, RL does not necessitate an immense, labeled dataset, acquisition of which is often limited or not even feasible in certain scenarios. Instead, the required training data is generated through a sequence of interactions between the agent and an environment object.

A key differentiating factor of RL compared to other Machine Learning methods originates from the above mentioned interaction iterations. From each iteration a state transition tuple needs to be composed, in which a so-called reward function is applied in order to quantify the goodness of actions, subsequently being utilized during network updates. This characteristic provides the advantage of not being restricted by constraints imposed by an initial dataset, but rather influenced by the designer's choices in defining crucial elements, such as state and action spaces, as well as the mentioned reward function. Thoughtful selection of abstractions in a given task plays a vital role in the credit assignment process, ultimately shaping the attainable performance of agents.

Since online prediction time of a trained neural network is in the order of milliseconds, one out of several advantages of RL is its limitless real-time applicability, that enables agents to quickly react and adapt to changing conditions or events, being a useful property in dynamic environments such as robotics or traffic control tasks [32], [33]. Furthermore, an additional benefit of applying neural networks as non-linear function approximators is the attribute of scalability, which proves particularly advantageous in complex structures composed of multiple interconnected components. This characteristic empowers RL agents to expand their capabilities to handle large-scale systems, effectively coordinating actions and making decisions across multiple levels of abstraction. This is especially relevant in domains such as network management, supply chain optimization, and smart grids [34], [35], [36]. Lastly, a virtue of RL, long-studied by many, is generalization capability, that facilitates handling variations, uncertainties, unforeseen circumstances and edge cases emerging within the environment. It ensures, that decisions remain reliable and robust, even in situations that deviate from the training distribution. Moreover, this property forms the foundation of transfer learning techniques [37], enabling RL agents to transfer their acquired policies from one task or environment to another. Hence it minimizes the need for extensive retraining, reducing costs in terms of time and computational resources, while expediting learning in additional scenarios.

The mathematical framework of Reinforcement Learning is determined by the Markov Decision Processes (MDP) [38], often referred to as $< S, A, R, P >$, where the object responsible for the actions is called an agent. A Markov Decision Process is clearly defined by four elements, which are respectively as follows:

- $s_t \in S$ is the state at time step t, where S is the state space
- $a_t \in A$ is the action chosen at time step t from state s_t, where A is the action space
- r_{t+1} is the reward quantifying the quality of state transition from state s_t due to the selection of action a_t
- $P(s_{t+1}|s_t, a_t)$ is the probability that choosing action a_t from state s_t at time step t will lead to state s_{t+1} at time step $t+1$

As per described above, the training involves a sequence of interactions between the agent and the environment, leading to generation of state transition tuples, hence forming the required training samples. In this context, the environment serves a dual purpose in this process: on the one hand, it provides essential information about the state of the environment, enabling the agent to make decisions based on the observed data. On the other hand, it quantifies the effectiveness of chosen actions by returning a scalar feedback value, known as the reward. An iteration of the training loop, depicting the aforementioned methodology, is as follows:

1) The agent uses the observation received from the environment at the previous time step to make a decision.
2) This decision is then conveyed to the environment as an action, leading to a change of the environment's current state.
3) The change in the environment's state causes a state transition, which encompasses two important outcomes: the subsequent state representation and a corresponding reward value.
4) These metrics are then relayed back to the agent.
5) The new state representation is utilized by the agent for subsequent decision-making, while the reward value serves as a feedback signal, based on which the update of neural network can be carried out.

Throughout the training, weight and bias parameters of the neural network are iteratively updated using two additional key components, being the loss function and the backpropagation method. From a mathematical perspective, the agent's objective during training is to maximize the cumulative reward, which is a weighted sum of rewards obtained over an episode. This mathematical goal is formally represented in 3 as:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} \cdot R_t \tag{3}$$

where t is the time step, T is the time step at which termination of current episode occurs, R_t is the value of reward received at time step t and γ is the so-called discount factor used to balance importance of immediate rewards over future rewards.

1) VALUE-BASED METHODS

Value-based methods represent a category of algorithms in Reinforcement Learning with the aim of giving an accurate estimation of the so-called value function associated with states, action advantages and state-action pairs. The value

function characterizes the expected return or cumulative reward achievable by the agent, while following a specific policy throughout the episode. Hence, the primary objective of the agent is to acquire an optimal value function, that maximizes expected cumulative reward. By approximating values of states or state-action pairs, the agent gains the ability to make well-informed decisions regarding which actions to execute for the purpose of maximizing its long-term reward. Value of a state can be calculated according to 4 as:

$$V_\pi(s) = \mathbf{E}_\pi \left[G_t | S_t = s\right] \tag{4}$$

where $\mathbf{E}$ is the expected value operator, G_t is the cumulative reward, s is the state for which the value estimation is carried out and π is the policy the agent needs to follow after time step t to realize the estimated value.

It is essential to acknowledge, that value-based methods primarily focus on estimating the value function, rather than directly acquiring a policy. However, a policy can be derived from the estimated values by selecting the action with the highest value within each state.

2) Q-LEARNING ALGORITHMS

Among various approaches within the value-based algorithm family, an outstanding example is provided by Q-learning methods [39]. Q-learning, being an off-policy and model-free algorithm class, meaning that it does not require prior knowledge of the environment's inner dynamics or transition probabilities. During the training process, estimation of the action-value function, commonly referred to as the Q-function, is continuously updated. Similarly to general value-based solutions, the Q-value can be acquired according to 5 as:

$$Q_\pi(s, a) = \mathbf{E}_\pi \left[G_t | S_t = s, A_t = a\right] \tag{5}$$

where an additional a parameter is included in comparison with 4, which is the action chosen from state s.

The iterative update of Q-values is based on the observed rewards and state transitions experienced during the interaction sequence, starting from an initial Q-table or Q-function, and choosing actions based on an exploration-exploitation strategy, such as ϵ-greedy. Using these observations, Q-values are updated according to the Bellman equation, which expresses the relationship between Q-values of successive states according to 6 as:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)] \tag{6}$$

where α is the learning rate, r is the reward value, γ is the discount factor and $\max_{a'} Q(s', a')$ is the highest Q-value obtainable from subsequent state s'.

3) DEEP Q-NETWORK

Deep Q-Network, often abbreviated as DQN, is a Reinforcement Learning algorithm, that combines principles of Q-learning with powerful capabilities of deep neural networks. Originally introduced in [13] demonstrating the method on Atari games, DQN has garnered significant attention from researchers due to its impressive performance not only in the gaming domain, but also in a diverse set of fields including robotics, recommendation systems and autonomous driving.

The key idea behind DQN is to use a deep neural network to estimate the Q-function, hence the handling of high-dimensional, complex input spaces is easily feasible. Approximation of the optimal Q-function is resolved by minimizing the mean squared error loss between predicted and target Q-values, as can be seen formally described in 7:

$$\mathfrak{L}_{MSE} = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2 \tag{7}$$

where N is the number of training samples, y_i is the target value and $\hat{y}_i$ is the predicted value for sample i.

Introduction of the DQN algorithm incorporates two main novelties, one of which involves the application of two identically structured neural networks. The primary network, known as the action network, is responsible for decision-making, while the secondary network, referred to as the target network, is used to compute target Q-values. Update of the neural network is performed on the action network, while parameters are periodically copied to the target network at a fix rate of time steps. This approach helps stabilize the learning process by providing more reliable target values during training achieved by reducing correlation between consecutive state transitions. Target Q-values are calculated using the Bellman equation, similarly to traditional Q-learning, according to 8:

$$Q\left(s_t, a_t; \theta_t\right) = r_{t+1} + \gamma \cdot \max_{a} Q\left(s_{t+1}, a_{t+1}; \theta_t^*\right) \tag{8}$$

where θ_t is the parameter set of the action network, while θ_t^* is the parameter set of the target network.

Furthermore, DQN uses a technique known as experience replay, which enhances learning efficiency. This involves storing the agent's experiences in a tuple, encompassing the corresponding state, action, reward and next state, in a designated buffer called the experience replay memory. During training, a batch of experiences is stochastically sampled from the replay buffer according to a uniform distribution, aiding in decorrelation of successive training samples and reducing bias in the parameter updates.

In summary, the modified training process, as illustrated in Fig. 4, showcases the two key innovations of DQN: the utilization of an experience replay memory and the presence of a separate target neural network both contributing to a decrease in correlation between experiences resulting in a more efficient training method.

B. EXPERIENCE PRIORITIZATION

DQN algorithm, as described above, introduced the concept of experience replay memory to facilitate the reuse of

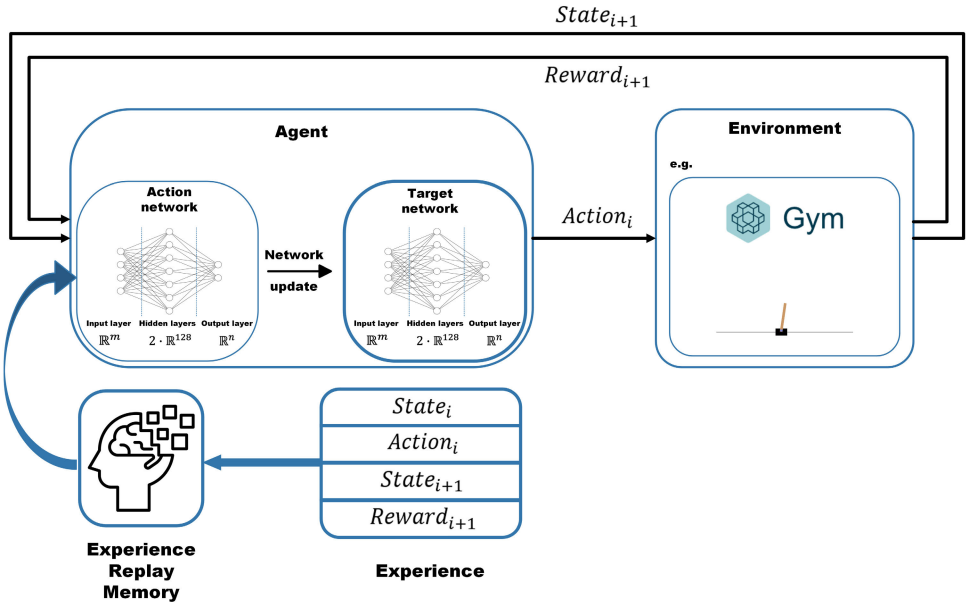

FIGURE 4. Training loop of Deep-Q Network algorithm emphasising two proposed additional features: the separate target network and the experience replay memory.

training samples gathered in the past, thereby accelerating convergence and enhancing training efficiency. However, experiences are handled quite suboptimally by employing uniform stochastic sampling during training, resulting in equal probability for all experiences. Consequently, this raises the question, whether information density of experiences is truly equivalent. While state transitions occuring more frequently in the environment may contain redundant information, there may also be occasional cases from which the constructed experiences are crucial from the agent's perspective.

For this reason, it can be stated that importance of training samples, in a given state of the agent, may differ from one another. The following sections provide comprehensive insights into both current state-of-the-art method and the newly proposed solution, which aim to enhance sampling efficiency by prioritizing certain experiences during the replay of state transitions.

1) PRIORITIZED EXPERIENCE REPLAY

Prioritized Experience Replay (PER), proposed in [14], is still the most commonly utilized experience prioritization technique in literature, focusing on addressing the aforementioned issue of difference between importance of training samples.

The underlying idea behind experience prioritization is straightforward: set up an order from available state transitions or assign a metric to each experience, thus constructing a list, ranking training samples based on their importance at a given state of the agent's knowledge. In case of value assignment-based methods, the task is to assign a so-called priority value such, that the higher the importance of experience is, the greater its priority value. The challenge lies in determining the appropriate metric or criterion by which the priority of individual experiences can be computed.

In the context of Prioritized Experience Replay, the prioritization relies on the Temporal Difference (TD) error, which serves as a measure of discrepancy between the observed reward during the state transition and the predicted value for the initial state. Essentially, higher TD-error indicates a greater expected learning progress, indicating that the specific sample contains valuable information, that can contribute significantly to the agent's training. Probability distribution for stochastic sampling over the entire experience replay memory can be calculated according to 9 as:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{9}$$

where $P(i)$ is the probability of choosing training sample i, α is the exponent determining the scale of prioritization and p_i is the priority value of transition i.

The original paper presents two separate approaches for calculating priority values, one of which is an indirect, rank-based prioritization method formalizing priorities as shown in 10:

$$p_i = \frac{1}{rank(i)} \tag{10}$$

where $rank(i)$ means the numeric rank of sample i in the ordered list constructed based on absolute TD-errors of experiences $|\delta_i|$.

The other design, utilized more frequently in literature, introduces a direct proportional prioritization, where priority values are calculated according to 11 as:

$$p_i = |\delta_i| + \epsilon \tag{11}$$

where $|\delta_i|$ is the absolute TD-error of sample i and ϵ is a small constant preventing edge cases of having zero TD-error.

In order to address bias caused by the utilization of non-uniform sampling, importance sampling weights have been proposed to adjust the updates of neural networks, ensuring a balance in training processes. Mathematically, importance sampling weights are formulated as shown in 12:

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta} \tag{12}$$

where β is the exponent determining the scale of bias compensation, N is the number of samples present in the replay buffer and $P(i)$ is the probability of choosing transition i.

Substantial benefits of Prioritized Experience Replay in terms of training efficiency and final performance have been demonstrated by numerous empirical experiments across diverse domains, ranging from games to more complex continuous control tasks [40], [41].

2) ENHANCED REWARD PRIORITIZATION METHOD

So-called exploration-exploitation dilemma is a fundamental challenge in Reinforcement Learning, that arises from the need to balance between exploring unknown situations of the environment and exploiting existing knowledge to maximize cumulative reward.

It is considered a trade-off, in which exploration term refers to the process of gathering new training data from the environment, that have not been explored extensively. This enables the agent to discover potentially superior actions or states, that could lead to higher long-term rewards. On the other hand, exploitation involves leveraging the already acquired knowledge to choose actions, that are known to yield higher immediate rewards based on past experiences. Achieving an optimal balance between the two terms is crucial for training and sampling efficiency. Various exploration strategies and algorithms have been developed, according to [42], in order to address the trade-off, such as ϵ-greedy, softmax exploration, Thompson sampling, and Upper Confidence Bound (UCB), originally proposed in [43], described in detail later on.

Maintaining balance between exploration and exploitation is crucial in case of Prioritized Experience Replay as well. Algorithm of PER may inadvertently prioritize certain transitions excessively due to stochastic sampling heavily relying on TD-error metrics of experiences, being an exclusively exploit term. Therefore it can potentially hinder exploration, as per the above mentioned theory, for which this paper provides a solution by resolving the dilemma through a novel approach.

Upper Confidence Bound is a popular algorithm used in Reinforcement Learning to tackle the exploration-exploitation problem. It is based on the principle of balancing by so-called confidence bounds to estimate the value of different actions or states. In UCB, each action or state is associated with an upper confidence bound value, that quantifies the uncertainty or confidence. The UCB value is calculated as shown in 13:

$$UCB = \bar{x_j} + \sqrt{\frac{2 \cdot ln\,(n)}{n_j}} \tag{13}$$

where $\bar{x_j}$ realizes the exploitation term, meaning the average observed reward obtained from state or action j. The idea behind the exploration component is to prioritize actions or states with higher uncertainty, as they may have the potential for higher rewards. Thus n_j is the number of times action or state has been selected, while n is the overall number of selections.

In this paper, we propose a novel approach for reward prioritization based on an alternative priority value assignment concept incorporating the mathematical formalization of Upper Confidence Bound into the context of experience replay. Our method calculates priority values assigned to training samples according to 14 as:

$$P(i) = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}} + c_p \cdot \sqrt{\frac{2 \cdot ln(\max\limits_k n_k)}{n_i + \epsilon}} \tag{14}$$

where the exploration component is represented by the relation between the n_i, which is the fit count indicating the number of times experience i has been utilized for network updates, and n_k, which is the fit count of the experience that has the highest count in the memory buffer. ϵ is a small constant, that prevents zero division and c_p is another constant within the interval of [0; 1] regulating balance between exploration and exploitation.

Our method combines the advancement from resolving deficiencies of the PER algorithm and the formation of balance between exploration and exploitation, resulting in higher sampling efficiency and an accelerated converge compared to current state-of-the-art method.

Like all algorithms, the method presented herein possesses inherent limitations. Notably, certain RL algorithms, such as TRPO, SARSA, PPO, traditional Q-learning and Monte Carlo methods, deviate from the common practice of incorporating experience replay methods. In these cases, training samples generated during the learning process are immediately utilized to tune the neural network. Consequently, the utilization of UCB-based experience prioritization, a mechanism employed to enhance performance by addressing the exploration-exploitation dilemma, is unfeasible for such algorithms. It is noteworthy, however, that these alternative algorithms inherently incorporate distinct solutions to overcome the same challenges associated with the exploration-exploitation dilemma within the realm of Reinforcement Learning.

III. ENVIRONMENT

The environment, as mentioned above, plays a vital role in the Reinforcement Learning training loop, as it realizes an external system or simulated world, within which state transitions are carried out based on the agent's

decisions. By offering essential feedback values, namely state observations and rewards, the agent is able to derive information from its actions' consequences to generate required training data.

Typically, the environment comprises of several components, including a state space, an action space, transition dynamics, and a reward function. The state space encompasses a set of possible states to which transitions can lead to, while the action space defines available legal actions, that the agent can execute from a given initial state. Transition dynamics elucidate the process of evolving from one state of the environment to another based on chosen actions. Furthermore, the environment incorporates a so-called reward function, through which a scalar feedback value is determined, which conveys the immediate efficiency of the agent's actions.

The form of the environment can manifest in several ways, ranging from straightforward grid worlds and board games to sophisticated simulations mirroring real-world systems, such as robotics, traffic networks, or financial markets. In this study, we utilize sandbox-like, widely used environments provided by the Gym package in order to highlight the impact of different algorithms on sampling strategies.

A. GYM ENVIRONMENTS

Gym is a widely used toolkit in literature for developing, evaluating and comparing Reinforcement Learning-based methods. Originally proposed in [44], this framework has since gained significant attention due to its numerous benefits, some of which are being the following:

- *Standardization*: Offering a standardized interface, Gym environments enable researchers to effectively compare and evaluate their Reinforcement Learning algorithms in a fair and reproducible way. This consistent interface allows algorithms to be seamlessly applied to various tasks, while ensuring accurate comparisons between different approaches.
- *Accessibility*: Being an openly accessible, free resource and having a comprehensively documented API both contribute to the level of accessibility, allowing researchers of all levels of expertise and experience to utilize the features provided.
- *Flexibility*: Structure of the Gym framework is designed to support future customizations and extensions according to specific needs and ideas in a relatively simple manner. This adaptability facilitates modifications of existing pre-defined environments as well as implementation of new ones, hence boosting research and development in emerging domains and exploration of innovative problem formulations.

Apart from the aforementioned advantages, the toolkit offers a vast collection of tasks and problems suitable for experiments to be carried out on different solutions, including classic control tasks, robotics simulations, and Atari games. Each environment is encapsulated in a Gym wrapper and comes with a well-defined interface, enabling agents to interact with the environment through their chosen actions and receive state representations and rewards in return. In Fig. 5, a schematic illustration of the Gym interface is shown, where the central component is responsible for both issuing control commands, conveying them to the actual environment and realizing data transfer between the communicating parties.

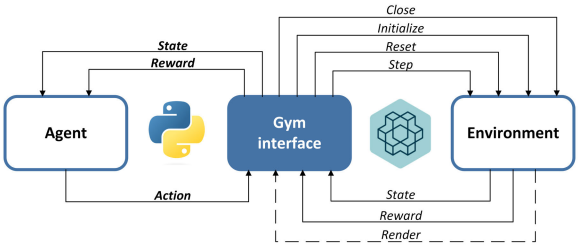

FIGURE 5. Schematic of the Gym interface.

For the sake of diversity, four completely different problems have been chosen for the thorough investigation of our method, thus formulating a varied set of challenges. As the complexity of each environment can vary significantly, it is crucial to define the relevant criteria for assessing their relative difficulty via objective metrics. One approach is to compare the complexity of state and action spaces, hence higher-dimensional state spaces may pose greater challenges for exploration, while larger action spaces may increase the complexity of decision-making.

In addition, the resolution to the exploration- exploitation dilemma, namely through exploration efficiency, measures the agents' effectiveness in navigating the state space. Moreover, environments featuring sparse rewards, characterized by delayed and infrequent feedback, introduce additional challenges to the aforementioned trade-off.

Post-analysis of training runs is instrumental in assessing task difficulty. Learning curve analysis on the one hand examines convergence rates, where slower rates may indicate more challenging tasks, on the other hand it also measures stability of the learning process across different runs. Reward distribution analysis, which involves investigating reward entropy, offers insights into the level of challenge, with higher entropy suggesting increased uncertainty.

Quantitatively measuring the difficulty of a Reinforcement Learning problem is a challenging yet essential aspect of comprehending algorithm performance. In order to facilitate an accessible framework for benchmarking, the environments presented in this paper are all categorized as relatively easy on an absolute scale. However, given the diversity of problems, a rank based on difficulty can be constructed, in which Cliff Walking emerges as the easiest task for the agent to completely learn the desired behavior, followed by CartPole, Acrobot, and finally, the Taxi environment.

1) CARTPOLE

The CartPole environment is a classic control task included in the toolkit with the aim to simulate a simplified scenario of an unstable pole balanced on a cart. Initially proposed in [45],

this problem formulation is based on the inverted pendulum, which serves as a benchmark in control theory for testing different approaches. The goal of the agent within the training is to learn maintaining the pole in an upright position as long as possible.

The state vector of the environment consists of four physical parameters being the cart's position along the track x, the cart's velocity v, the angle of the pole relative to the vertical θ and the angular velocity of the pole ω. The action space defines two distinct actions {0; 1} for each state, allowing only the specification of a fixed-magnitude force's direction applied to the cart.

Episodes are deemed successful in case the termination occurs after 475 time steps. The agent receives a reward of $+1$ for a successful episode, while failing beforehand results in a penalty of -1. Each time step not leading to immediate termination of the game yields a reward value of 0.

2) ACROBOT

Introduced in [46], Acrobot is another a physics-based simulation, modelling a two-link pendulum system with an actuated joint between the links. The objective in this environment is to swing the free end of the outer link up to a predefined target height as quickly as possible by applying appropriate torque through the actuator in each time step.

Observations comprise sine and cosine values of the two poles' angles denoted as $sin\theta_1, cos\theta_1$ and $sin\theta_2, cos\theta_2$, as well as the angular velocities of the poles ω_1 and ω_2 respectively. Actions correspond to the torque being applied by the actuator and are selected from the set $\{-1; 0; 1\}$ Nm.

The agent incurs a negative reward of -1 for each non-terminal step, thus encouraging agile completion of the task. However, when the free end reaches the target height, an exceptional reward value of 0 is received.

3) CLIFF WALKING

Adopted from [47], Cliff Walking is a grid-world setting, where an agent explores a grid map containing a cliff, aiming to navigate from a fixed initial position to a predetermined goal cell, while minimizing the number of movements and avoiding falling off the cliff.

The state representation is a vector of length 48, where each element represents the state of the rectangular 4 x 12 grid-world's cells. The agent starts at a designated starting position located in the bottom-left corner, with the goal cell situated in the bottom-right. The agent is able to take actions in four directions: up, right, down and left, within the grid. However, there is a cliff region along the bottom row of the grid, that immediately assigns a negative reward to the corresponding state transition as a penalty.

Generally, a reward value of -1 is collected in each time step, unless the agent falls into a cliff, resulting in a reward of -100 instead. An episode terminates if the agent reaches the goal cell on the map or a predefined limit of time steps.

4) TAXI

The Taxi task, originally presented in [48], features a simplified simulation of a taxi operation within a grid-world environment. The primary objective for the agent is to deliver passengers from designated pick-up to drop-off cells, while minimizing travel time.

The map is represented as a 5 x 5 grid, allowing the agent to occupy any of the 25 locations. There are four passengers and five possible allocations for them, including one scenario in the car. Additionally, there are four desired drop-off points, thereby constructing a vector of length 500, from which the scalar observation can be calculated as shown in 15:

$$s = ((t_r * 5 + t_c) * 5 + p) * 4 + d \tag{15}$$

where s is the scalar representation of the state, t_r and t_c denote the row and column of the agent's location respectively, p indicates the passenger's position and d signifies the destination's location. The action space consists of six elements: moving up, down, left or right, as well as performing pick-up and drop-off actions.

The agent is granted a reward of $+20$ for successfully delivering a passenger to the correct location, although incurs a penalty of -10 in case the pickup or drop-off actions are executed at an incorrect location. Throughout the episode, at each time step, a time penalty is given with a value of -1 reflecting the secondary goal of completing the task rapidly.

B. TRAINING

The training process holds principal importance in Deep Learning methods on account of the agent's performance is evolved throughout this progress. In order to generate a diverse set of training data, randomization is a key component. In each environment, except for Cliff Walking, the initial state is randomized over legal situations available, in a standardized manner, utilizing the Gym framework. This randomization of commencing conditions enables the environment to provide wide range of experiences, facilitating the development of a robust agent. Moreover, precise definition of termination conditions is also essential for the analyzed problems, although it is also provided by the Gym package.

Experiments have been conducted on both agents, utilizing Prioritized Experience Replay and a novel Upper Confidence Bound-based prioritization approach. To validate our hypothesis and to demonstrate the robustness of our solution, a total of 5 training runs per environment have been carried out for both algorithms under identical conditions. Since further analysis about the robustness of our approach would require transitioning to a sim2real scenario, for the sake of straightforward comparability, an easily accessible, yet quite pragmatic method has been adopted, namely diversifying the training schemes by seeded pseudorandomness.

In the first stage, general trainings have been accomplished with alike hyperparameter sets, and neural network

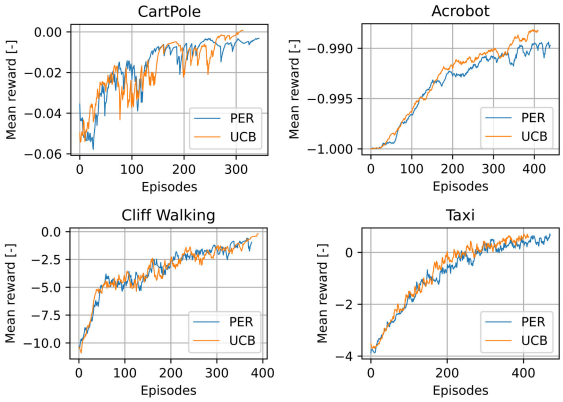

FIGURE 6. Mean reward of the evaluated solutions during 5 training runs.

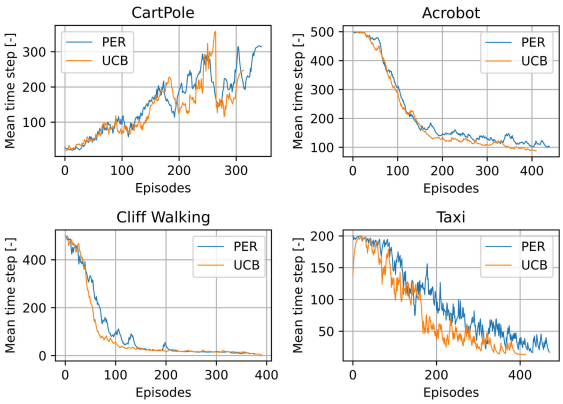

FIGURE 7. Time steps of the prioritization methods during 5 executions.

architectures on the four, previously described environments. Final hyperparameters and neural network structure utilized by both agents have been determined by grid search to obtain the highest achievable performance. This segment of the experimentation pointed out, that performance of such can be attained by the exact same values for PER and the UCB-based prioritization meaning, that sampling strategy does not influence the agents' sensitivity to these parameters. The only value that differs from, more specifically is additional to, the PER methodology is c_p introduced in Eq. 14, a constant regulating balance between exploration and exploitation. As the extent of exploration, from the training efficiency's point of view, is a highly task-dependent property, one has to experiment with its value in the interval of [0; 1] in order to fine-tune the exploration-exploitation trade-off, thus establishing the right balance.

The latter analysis primarily aimed to compare performance indicators of the proposed method relative to state-of-the-art algorithm. In the subsequent phase however, a unique warm-up stage has been implemented prior to updating any of the neural networks' parameters. During this stage, experience replay memories have been filled up with the same state transition tuples regarding each method, which remained unchanged throughout the training process, in order to highlight the difference in sampling efficiency between the two techniques, thereby further supporting our theoretical claims.

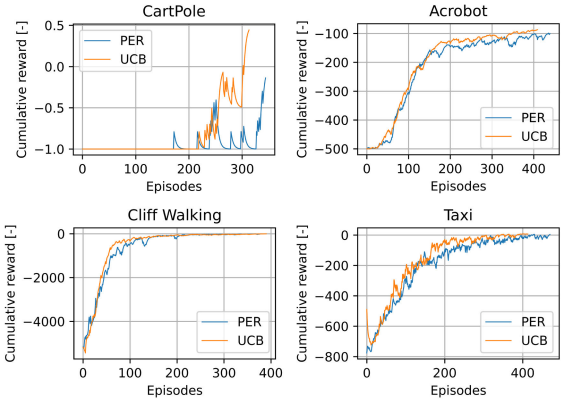

FIGURE 8. Cumulative reward of the applied techniques during 5 seeded scenarios.

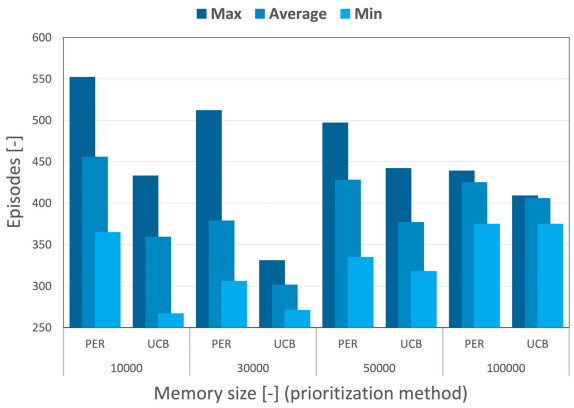

FIGURE 9. Statistical comparison of 5 training runs on the Acrobot environment utilizing identical, fixed-size replay memories.

C. EVALUATION

The raw data, obtained from recorded measurements of the above mentioned training methods, has undergone a comprehensive evaluation procedure, while evolution of various training and performance metrics have been monitored with a specific emphasis on the efficiency of applied sampling strategies. These metrics include the number of time steps in the episode until termination, the mean reward per episode, the cumulative sum of rewards throughout an episode and also absolute time of training runs, to which computational resource requirements are similarly proportional as per expectations.

For the sake of establishing a thorough overview of the prioritization techniques and acquire an extensive insight into disparate solutions, the second training approach has been further analyzed using multiple fixed-length experience buffers employed. These memories have been configured with the following capacities of 10000, 30000, 50000 and 100000, respectively.

IV. RESULTS

A. FIRST PHASE—GENERAL EVALUATION

In this section, a detailed analysis of the processed training data is demonstrated with respect to Prioritized Experience Replay and the novel Upper Confidence Bound-based

TABLE 1. Statistical comparison of absolute mean training times [s] with corresponding standard deviations.

Prioritization method	CartPole	Acrobot	Cliff Walking	Taxi
PER	218.6(±66.14)	704.2(±116.21)	140.4(±49.25)	4866.4(±2077.81)
UCB	**154.4**(±30.96)	**493.6**(±55.79)	**129.4**(±38.67)	**4415.2**(±1441.34)
Difference of UCB relative to PER	−29.4%	−29.9%	−7.8%	−9.3%

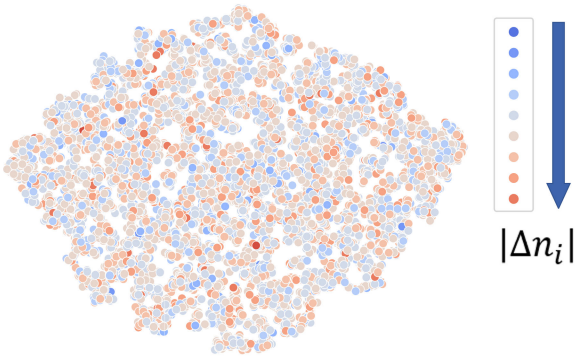

FIGURE 10. Two-dimensional t-SNE embedding of state transitions' initial states at the end of training in the Acrobot enviroment.

TABLE 2. Statistical comparison of mean number of episodes required to learn the task.

Prioritization method	CartPole	Acrobot	Cliff Walking	Taxi
PER	380.6	425.0	**352.0**	542.4
UCB	**305**	**405.8**	382.4	**418.6**
Difference of UCB relative to PER	−19.9%	−4.5%	+8.6%	−22.8%

prioritization approach, while monitoring and evaluating the evolution of commonly examined training metrics.

Results of mean reward values collected per episode are displayed in Fig. 6 for each of the four inspected environments, during 5 seeded training runs using a simple low-pass smoothing filter for the sake of transparency in the values. The agent using our enhanced prioritization strategy is shown in orange, while the one utilizing state-of-the-art experience prioritization method is shown in blue. These results, as well as the displayed converge plots of mean time steps per episode in Fig. 7 and cumulative reward values throughout episodes in Fig. 8, equivalently demonstrate the ultimate superiority of our approach over the other concept in terms of sampling efficiency, as enhanced attributes of the convergence curves are clearly visible. Not only do they show oscillations of lower magnitude and frequency during training, but the number of episodes and absolute time required for completely learning a task has also been decreased by a decent amount.

Mitigation of the required network updates on one hand leads to an increase in speed of convergence, but on the other hand results in a significant reduction in training time with the newly presented method compared to the PER algorithm. This phenomenon is shown numerically in Table 1, where a remarkable $10-30\%$ rate of improvement is observable, depending on the problem formulation. The evolution of the standard deviation values displayed within the brackets also indicate that our proposed method allows a much more stable learning process, with smaller deviations around the mean values. Similarly, Table 2 demonstrates the average number of episodes, that is required for the agent to learn completing a task. In most cases, our proposed algorithm achieves better performance looking at these values, except for the Cliff Walking environment, which might be because its straightforward solution cannot formulate any contrast between the two methods. However, regarding absolute time consumption, our method has achieved better results on this environment as well.

Consequently, since the absolute elapsed time for training is closely related to the need for computational resources, these findings point out an expected cutback of similar proportionality on computing capacity requirements. As detailed in the Introduction, the overall cost of training can be expressed as the sum of indirect costs arising from training time and computational demand of the process, hence a considerable advancement has been achieved on the basis of increased sampling efficiency.

B. SECOND PHASE–FIXED REPLAY MEMORY

In order to further validate our theory, that the enhanced experience prioritization approach obtained a higher level of sampling efficiency, an extensive supplementary experiment has been carried out, during which the two solutions are given identically filled experience replay memories with the same state transitions through a warm-up stage, thus the only factor influencing length of trainings is sampling efficiency. In case the agent makes better use of the same training data stored in each memory, in other words, samples from the experience pool on a higher efficiency, a fewer amount of network updates and therefore less training episodes are required for achieving the same level of performance.

In Fig. 9, average values of 5 seeded training runs have been displayed on the Acrobot task, while utilizing fixed replay memories of identical experiences. The involved metrics are minimum, maximum and average values of the number of episodes required to complete the training (in this case, moving average of last 10 episodes has to achieve a time step value below 100). It is clearly visible, that in each case of memory size, our proposed algorithm has managed to accomplish superior sampling strategy, as can be derived from all the statistical parameters.

Furthermore, Fig. 10 displays some insight of the higher performance gained by our proposed solution. The dots in the

figure represent single experiences, while coloring indicates the absolute difference in the number of updates, in which the given experience has been utilized, for each training sample, as can be seen in 16:

$$RGB = |\Delta n_i| = \left| n_{i,\,PER} - n_{i,\,UCT} \right| \tag{16}$$

where n_i is the fit count of transition i. Thus moving from blue to red colored locations highlights the increasing focus-shift regarding experiences between the two algorithms via the t-Distributed Stochastic Neighbor Embedding (t-SNE) method, applied to initial state representations of the final experience replay memory with size of 10000 after training in the Acrobot task.

V. CONCLUSION

Reinforcement Learning remains a uniquely prominent, yet quite intricate approach to several optimization and control tasks primarily concerning sequential decision making problems due to its potential to provide superior performance among other methods, such as Supervised Learning, where the agent is limited to the training dataset applied, and classical algorithms, that utilize tremendous volume of computational power in the online decision making process, thereby not allowing real-time applicability without restrictions.

However, a serious disadvantage of Deep Learning-based solutions, including RL algorithms, is that the training phase involves overwhelming resource allocation due to inadequate treatment of training data. Sampling efficiency is a crucial metric for assessing performance of such processes. Several concepts have been published to address this issue, one of which is experience prioritization, through which sampling can be improved by determining priority of each state transition stored in a memory.

In this paper, a novel approach has been proposed, that revisits the exploration-exploitation dilemma by formalizing a new priority value assignment concept. Drawing inspiration for mathematical formality from the Upper Confidence Bound algorithm, our approach incorporates the number of neural network updates, an experience has undergone as a basis of the exploration term, while the exploitation component remains to be the normalized Temporal Difference-error. Therefore, a new trade-off strategy has been presented in the field of Reinforcement Learning, addressing the problem of inefficient sampling.

Our results show, that utilizing our proposed method both improves sampling efficiency and enhances learning stability compared to current state-of-the-art technique, being Prioritized Experience Replay, as evidenced by statistical analysis of convergence curves. Furthermore, it can be stated, that by obtaining superior sampling strategy, absolute training time and, expectedly, computational resource requirements can be significantly reduced, leading to a cut down in costs of Reinforcement Learning trainings.

In our forthcoming pursuits, the primary objective is to apply the proposed experience prioritization method to much more complex tasks, in order to showcase its potential in leveraging information of state transitions with higher efficiency. Since our Upper Confidence Bound approach being a general methodology improvement, one only has to find suitable environments for further validation of the empirical findings. In this case, we have considered the task of lane keeping in the field of autonomous vehicle control, and the traffic signal control problem for multi-intersection transportation networks. Secondly, we aim to investigate further refinement of the trade-off components on attempt of finding an even more sophisticated abstraction of exploration or exploitation in the Reinforcement Learning framework. Eventually, we believe, that utilization of such trade-off could be intriguing in the context of Supervised Learning as well, since the problem of non-equivalent importance among training data is also relevant for this branch of Machine Learning, which can be resolved by establishing a proper balance using our proposed approach. In general, integration of the presented experience prioritization method into existing PER-based implementations consists only of two parts. Firstly, the utilized fit count metric of each experience, responsible for exploration, needs to be monitored for the whole duration of the training, and secondly, Eq. 14 has to be applied during the priority value assignment process, instead of the simple TD-error being used in PER.

REFERENCES

[1] B. Kővári, F. Hegedüs, and T. Bécsi, "Design of a reinforcement learning-based lane keeping planning agent for automated vehicles," *Appl. Sci.*, vol. 10, no. 20, p. 7171, Oct. 2020.

[2] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, and S. Jo, "Review of machine learning methods in soft robotics," *PLoS ONE*, vol. 16, no. 2, Feb. 2021, Art. no. e0246102.

[3] B.-H. Li, B.-C. Hou, W.-T. Yu, X.-B. Lu, and C.-W. Yang, "Applications of artificial intelligence in intelligent manufacturing: A review," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 1, pp. 86–96, 2017.

[4] P. Gorzelanczyk, "Forecasting the number of road accidents in Poland depending on the day of the week using neural networks," *Periodica Polytechnica Transp. Eng.*, vol. 51, no. 4, pp. 431–435, 2023.

[5] S. Garg, S. Sinha, A. K. Kar, and M. Mani, "A review of machine learning applications in human resource management," *Int. J. Productiv. Perform. Manage.*, vol. 71, no. 5, pp. 1590–1610, May 2022.

[6] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute trends across three eras of machine learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.

[7] B. Cottier. (2023). *Trends in the Dollar Training Cost of Machine Learning Systems*. Accessed: Jun. 14, 2023. [Online]. Available: https://epochai.org/blog/trends-in-the-dollar-training-cost-of-machine-learning-systems

[8] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *Neural Comput.*, vol. 33, no. 11, pp. 2908–2950, 2021.

[9] M. Moradi, Y. Weng, and Y.-C. Lai, "Defending smart electrical power grids against cyberattacks with deep Q-learning," *PRX Energy*, vol. 1, no. 3, Nov. 2022, Art. no. 033005.

[10] S. E. Razavi, M. A. Moradi, S. Shamaghdari, and M. B. Menhaj, "Adaptive optimal control of unknown discrete-time linear systems with guaranteed prescribed degree of stability using reinforcement learning," *Int. J. Dyn. Control*, vol. 10, no. 3, pp. 870–878, Jun. 2022.

[11] D. Tompos and B. Németh, "Safe trajectory design for indoor drones using reinforcement-learning-based methods," in *Proc. IEEE 17th Int. Symp. Appl. Comput. Intell. Informat. (SACI)*, May 2023, pp. 27–32.

[12] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.*, vol. 13, no. 1, pp. 103–130, Oct. 1993.
[13] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[14] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.
[15] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018, *arXiv:1803.00933*.
[16] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 316–321.
[17] M. Brittain, J. Bertram, X. Yang, and P. Wei, "Prioritized sequence experience replay," 2019, *arXiv:1905.12726*.
[18] P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. Young, "Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management," 2017, *arXiv:1707.00130*.
[19] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 8224–8234.
[20] Y. Yu, "Towards sample efficient reinforcement learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 5739–5743.
[21] R. Zhao and V. Tresp, "Energy-based hindsight experience prioritization," in *Proc. Conf. Robot Learn.*, 2018, pp. 113–122.
[22] R. Zhao and V. Tresp, "Curiosity-driven experience prioritization via density estimation," 2019, *arXiv:1902.08039*.
[23] Z.-W. Hong, T. Chen, Y.-C. Lin, J. Pajarinen, and P. Agrawal, "Topological experience replay," 2022, *arXiv:2203.15845*.
[24] S. Sujit, S. Nath, P. H. M. Braga, and S. E. Kahou, "Prioritizing samples in reinforcement learning with reducible loss," 2022, *arXiv:2208.10483*.
[25] S. Dankwa and W. Zheng, "Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proc. 3rd Int. Conf. Vis., Image Signal Process.*, Aug. 2019, pp. 1–5.
[26] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double Q-learning: Learning fast without a model," 2021, *arXiv:2101.05982*.
[27] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active SLAM based on deep reinforcement learning under unknown environments," *Intell. Service Robot.*, vol. 13, no. 2, pp. 263–272, Apr. 2020.
[28] S. Sinha, H. Bharadhwaj, A. Srinivas, and A. Garg, "D2RL: Deep dense architectures in reinforcement learning," 2020, *arXiv:2010.09163*.
[29] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare, "Contrastive behavioral similarity embeddings for generalization in reinforcement learning," 2021, *arXiv:2101.05265*.
[30] K. Wang, K. Zhou, Q. Zhang, J. Shao, B. Hooi, and J. Feng, "Towards better Laplacian representation in reinforcement learning with generalized graph drawing," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11003–11012.
[31] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.
[32] S. Koh, B. Zhou, H. Fang, P. Yang, Z. Yang, Q. Yang, L. Guan, and Z. Ji, "Real-time deep reinforcement learning based vehicle navigation," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106694.
[33] C. Yan, X. Xiang, and C. Wang, "Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments," *J. Intell. Robotic Syst.*, vol. 98, no. 2, pp. 297–309, May 2020.
[34] J. Z. Leibo, E. A. Dueñez-Guzman, A. Vezhnevets, J. P. Agapiou, P. Sunehag, R. Koster, J. Matyas, C. Beattie, I. Mordatch, and T. Graepel, "Scalable evaluation of multi-agent reinforcement learning with melting pot," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6187–6199.
[35] M. Zhou, "SMARTS: Scalable multi-agent reinforcement learning training school for autonomous driving," 2020, *arXiv:2010.09776*.
[36] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.
[37] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13344–13362, Nov. 2023, doi: 10.1109/TPAMI.2023.3292075.
[38] M. Van Otterlo and M. Wiering, "Reinforcement learning and Markov decision processes," in *Reinforcement learning: State-of-the-Art*. Cham, Switzerland: Springer, 2012, pp. 3–42.
[39] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[40] P. Zhai, Y. Zhang, and W. Shaobo, "Intelligent ship collision avoidance algorithm based on DDQN with prioritized experience replay under COLREGs," *J. Mar. Sci. Eng.*, vol. 10, no. 5, p. 585, Apr. 2022.
[41] D. Fährmann, N. Jorek, N. Damer, F. Kirchbuchner, and A. Kuijper, "Double deep Q-learning with prioritized experience replay for anomaly detection in smart environments," *IEEE Access*, vol. 10, pp. 60836–60848, 2022.
[42] R. McFarlane, *A Survey of Exploration Strategies in Reinforcement Learning*, vol. 3. Montreal, QC, Canada: McGill Univ., 2018, pp. 17–18.
[43] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, Nov. 2002.
[44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," Tech. Rep., 2016. [Online]. Available: http://arxiv.org/abs/1606.01540
[45] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–846, Sep. 1983.
[46] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 8, 1995, pp. 1038–1044.
[47] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[48] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, Nov. 2000.

BÁLINT KŐVÁRI received the B.Sc. and M.Sc. degrees in vehicle engineering from the Faculty of Autonomous Vehicle Control Engineering, Budapest University of Technology and Economics, Budapest, Hungary, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Budapest University of Technology and Economics. His research interests include artificial intelligence, reinforcement learning, vehicle dynamics, and mechatronics.

BÁLINT PELENCZEI is currently pursuing the B.Sc. degree in vehicle engineering with the Budapest University of Technology and Economics, Budapest, Hungary. He is a Developer with the Systems Control Laboratory, Institute for Computer Science and Control. His research interests include artificial intelligence, mechatronics, machine learning, and intelligent transportation systems.

TAMÁS BÉCSI (Member, IEEE) received the M.Sc. and Ph.D. degrees from the Budapest University of Technology and Economics, Budapest, Hungary, in 2002 and 2008, respectively.

Since 2005, he has been an Assistant Lecturer with the Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, where he has been an Associate Professor, since 2014. His research interests include machine learning, embedded systems, traffic modeling, and vehicle control.