

ABS-TD3: Efficient IoT Data Submission in DAG-based DLTs for Digital Circular Economy

Konstantinos Voulgaridis^a, Dimitrios Karampatzakis^a, Panagiotis Sarigiannidis^b, Thomas Lagkas^a

^a*Department of Informatics, Democritus University of Thrace, Kavala Campus, 65404, Greece*

^b*Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, 50100, Greece*

Abstract

Distributed Ledger Technologies (DLTs) underpin Digital Circular Economy (DCE) systems that rely on efficient IoT data flows. Shimmer, a DAG-based DLT optimized for IoT, enables feeless transactions with parallel validation through its tip-selection mechanism. On such ledgers, message fragmentation induces a latency-throughput tradeoff as per-block cost rises with parallel validation. Such efficiency lowers energy and congestion, supporting DCE objectives. Yet, end-users cannot control payload size or network load, leading to unpredictable latency and high CPU use on submitting devices, increasing energy consumption. Existing approaches mostly modify ledger internals, overlooking adaptivity or end-user policies. We introduce ABS-TD3, an offline-to-online TD3 agent that receives the total message size and outputs the optimal per-block size for balancing latency and energy-efficient CPU utilization. The agent is pre-trained offline on real data with Retrieval Augmentation and adaptive weights for improved decision making, then transitioned online with prioritized replay and a novelty bonus, balancing exploitation-exploration, yielding stable adaptivity compared to standard RL approaches. ABS-TD3 is implemented on Shimmer and can integrate with future Tangle-based forks of pre-IOTA-Rebased frameworks, exposing the same client-side controls. ABS-TD3 is evaluated on Shimmer by submitting 8 message sizes ranging from 5KB to 100KB, under the 32 KB block-size limit, with 250 iterations per size via IOTA-SDK. Against max, min, random, and fixed-weight baselines, it reduces median latency by about 9% to 12% and median CPU utilization by about 12% to 17% versus max and

This work appears in the Internet of Things journal of Elsevier

DOI: <https://doi.org/10.1016/j.iot.2025.101814>

URL: <https://www.sciencedirect.com/science/article/pii/S2542660525003282>

random policies, enabling efficient IoT data submission for DCE platforms without altering DLT infrastructure.

Keywords: Distributed Ledger Technologies, Internet of Things, Reinforcement Learning, Digital Circular Economy, resource optimization

1. Introduction

Beyond finance, blockchain and the wider category of Distributed Ledger Technologies (DLTs) are recognized as efficient data mechanisms for enabling Digital Circular Economy (DCE) [1]. The corresponding approach has resulted in the development of alternative DLTs, different from the classic chained architecture, tailored for secure and distributed data sharing and storage [2].

Among them, Directed Acyclic Graph (DAG) architectures are capable of validating transactions concurrently, with new transactions validating subgroups of previously established transactions, enhancing throughput and scalability [3]. Such architectures and systems have been proven to be effective for vital domains, namely DCE, and related applications like Digital Product Passports (DPP), requiring efficient data exchange, management, and storage through technologies dedicated to the Internet of Things (IoT), and correlated fields like Industry 4.0 (I4.0) and Industry 5.0 (I5.0) [4]. Also, security-focused IoT research on emerging threats and Deep Learning (DL) methodologies [5][6], reinforces the need for robust and secure data submission paths in both DCE and IoT networks. Significant examples that utilize such functionalities and are defined for their practicality were the IOTA Tangle, and currently Shimmer, optimized for their implementation within IoT infrastructures [7].

Though, as IOTA has escaped the scope of an optimized DLT for IoT applications with the introduction of IOTA Rebased, Shimmer continues as a feeless and community-maintained DAG-based DLT. Before the corresponding shift, different optimization frameworks were developed by the research community, such as a stochastic generative model for the production of synthetic Tangle data for exploring the effects of arrival rates [8], tip selection advancements aiming towards faster confirmations [9], as well as domain-specific applications for feeless data distribution [10].

Moreover, the interaction with such DLTs can introduce impactful challenges, affecting the process of block submission and, as a result, the per-

formance of the submitted transaction. As seen in [11], one of the factors affecting the performance of a submission is the size of the data to be submitted within the ledger in terms of transaction throughput, leading operational DLTs and blockchains to define block size limits to ensure stability, while attempting to provide sufficient block confirmation latency. Additionally, the nodes utilized for the validation and confirmation of transactions can heavily impact the performance of a DLT, especially in relation to the task load of each device and the data traffic within the network [12]. Furthermore, DAG-based protocols present known issues of increased confirmation latency, as well as high computational costs for the preparation and submission of the transaction within the DLT’s network [13]. Such cases can be proven detrimental to the high-quality performance of DCE frameworks, with deterministic latency and efficiency in computational resources being essential for the fulfillment of essential circular processes [14]. In particular, Industrial IoT (IIoT) evidence signifies that reliability of implemented Wireless Sensor Networks (WSNs) can be heavily affected by latency variability and high resource requirements, which further motivates the need for optimized processes in DCE settings [15].

Considering the aforementioned relationship between DLT conditions and performance factors, it has been identified that, from the aspect of the end-user interacting with a DAG-based DLT, direct or indirect control of the network conditions is unrealistic. Through interaction and experimentation with the Shimmer DLT, it has been observed that the size of the data within a transaction provides an on-average tradeoff between confirmation latency and computational costs, and the associated energy consumption, considering the additional uncontrollable network conditions. Specifically, for the total size of data in bytes to be submitted, smaller sizes per transaction lead to lower computational costs, and consequently reduced power use, and larger confirmation latencies. Simultaneously, larger sizes per transaction lead to larger computational costs, meaning higher power consumption as well, and lower confirmation latencies. Consequently, the capability of providing adaptive block sizing will be able to shield DCE systems from native DLT congestion during submission of circular data, while providing the end-user interactions with sufficient throughput.

In this application paper, we present Adaptive Block Sizing TD3 (ABS-TD3), an implementation of Reinforcement Learning (RL), and specifically of an offline-to-online Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm, for the resolution of the corresponding performance issues for

end-use block sizing. RL is a well-known Machine Learning (ML) paradigm, broadly integrated in dynamic environments for sequential decisions, while enabling adaptive policies to be learned through interactions with the respective environment and acquiring feedback in the form of rewards [16]. Such technologies are also consistent with recent studies indicating the effectiveness of autonomous data-driven control in interconnected systems [17], with metaheuristic search being a potential alternative for cases related to network optimization [18]. Moreover, TD3 is an off-policy RL algorithm for continuous action spaces, making it ideal for precise control tasks in dynamic environments and leading to stable and reliable learning [19].

Our contribution is dedicated on the proposal of ABS-TD3 that receives the total size of data to be submitted within the feeless Layer 1 (L1) of the Shimmer DLT, and selects the optimal size in bytes that the data need to be split into per transaction, in an attempt to balance, or even minimize, the aforementioned tradeoff of confirmation latencies and computational costs that potentially support power efficiency, while adapting to the unpredictable conditions of the Shimmer network. Additionally, combinations of novel RL mechanisms were utilized in both the offline and online aspects of the TD3 model to facilitate the learning of the offline basis and ensure adaptability of its online transition. Specifically, in the offline pre-training, we have included Retrieval Augmentation for enhanced learning through real static network data, and adaptive weights for latency and CPU for improved decision-making. For the online transition, we have incorporated prioritized replay with exponential decay and a count-based novelty bonus for balancing exploration and exploitation during adaptation in live network conditions.

The structure of our paper is as follows: in Section 2 we present related findings of the research community based on proposed approaches on efficient transaction-block submission, as well as RL-based resource-aware frameworks. In Section 3, we proceed by formulating the problem we attempt to resolve, we analyze the Shimmer DLT and its respective network conditions, as well as we present the architecture of our ABS-TD3. In Section 4, we showcase the implementation and experimental specification of our proposed system and present a detailed evaluation of its respective results. Finally, we close our work with the corresponding Discussion and Conclusion, in Sections 5 and 6, respectively.

2. Related Work

In this section, we discuss previous research conducted on the topic of efficient block submission within blockchain technologies or DLTs in general, organized in 3 different pillars: a) Non-RL block submission solutions, b) RL-based solutions for efficient block submissions, and c) Offline-to-Online TD3 solutions for resource-aware systems.

2.1. Non-RL block submission solutions

With the rising research interest in blockchain technologies, several schemes have been proposed as an attempt to provide efficient solutions for latency minimization while improving resource utilization. Starting with [20], the researchers propose an end-to-end latency model for blockchains supported by Proof of Work consensus mechanisms, enhanced with queuing delays, mining durations, and block propagation effects, including fork probabilities, resulting in the derivation of an optimal block size that balances the tradeoff between smaller and larger block sizes.

Moreover, the authors of [21] introduce Blockmess, an architecture focused on a parallel chain DLT that dynamically adjusts the number of active chains in order to adapt to current throughput needs, as a means of avoiding excessive latencies and heavy node load. Specifically, Blockmess proceeds by adapting workload fluctuations, and resulting in significant mitigation of latency degradation compared to static scalability means, effectively optimizing the observed latency-throughput tradeoff.

Also, in the work of [22], the authors apply two approaches, the multi-objective particle swarm optimization and the strength Pareto evolutionary algorithm, as a means of detecting optimal block sizing and balancing block transmission and composition latencies. The respective work demonstrates optimal maintenance throughput without causing congestion to participating miners.

2.2. RL-based solutions for efficient block submission

Moving on to RL-based solutions, focused on efficient processing during block submission, the researchers of [23] have presented a model called ROBB, a recurrent proximal policy optimization model for optimal block processing in Bitcoin, that was formulated in a simulated environment of the blockchain and trained in selecting the optimal time windows for block submission, considering network conditions and pending transactions.

The authors of [24] attempt to tackle the issue of throughput bottleneck in blockchain-based mobile edge computing systems. Specifically, in order to achieve scalability, they proceed by addressing the problem as a Markov Decision Process while utilizing Deep RL models for the respective decision. Their corresponding results demonstrate a rise in throughput while outperforming static approaches.

A more ledger-based application was implemented in [25]. Specifically, the authors proceed with configuring the parameters of a consortium blockchain through solving a multi-objective optimization problem with explainable Deep RL approaches. The results indicate that the proposed solution provided an improved trade-off compared to static configurations.

Beyond submission policies, the authors of [26] implement a combination of blockchain-based mechanisms with Deep RL algorithms for task offloading in IoT-fog-cloud systems. Specifically, the researchers utilize a double-dueling DQN with a PDS-based online update for offloading processes, while also integrating smart contracts for resource efficiency.

2.3. Offline-to-Online TD3 solutions for resource-aware systems

Although the literature still lacks implementations related to offline-to-online TD3 systems within the domain of DLTs, the corresponding approach has proven to be sufficient in dynamic environments with requirements on resource optimization.

Specifically, the authors of [27] attempted to handle resource allocation in mobile edge computing systems by utilizing Federated Learning (FL). However, they observed a trade-off occurring between the agent's accuracy and edge device energy requirements, formulating the core problem of their research. The researchers proposed the combination of FL with TD3 due to the continuous action requirements, with the presented solution achieving maximum accuracy ratio.

Additionally, in the work presented in [28], the researchers attempt to tackle computation offloading in dynamic cache-assisted vehicular NOMA-MEC systems. The respective problem is resolved with a TD3-based algorithm, enhanced with an action transformation mechanism for the processing of discrete decisions and a heuristic baseline. The results of the corresponding work indicate efficiency in energy consumption compared to defined benchmarks and other RL algorithms like DDPG.

Taken together, the studies presented in the corresponding section confirm the correlation of block sizing with confirmation latencies and CPU

utilizations in DLTs, as well as the capability of integrating RL principles for automated parameter configurations.

Generally, non-RL strategies, such as fixed heuristics and schedulers, are focused on stable traffic and DLT parameters, with the possibility of manual tuning for additional adaptation. On the other hand, RL mechanisms are structured on policy updating based on environment feedback, with cases like [29], where offline-to-online RL agents are pre-trained models with real logged data and yield significant performance during online adaptation, or RL solutions based on episodic memory that retrieve past training episodes for improving sampling efficiency under dynamic conditions [30]. Even though RL solutions provide significant capabilities, they can be constrained by continuous alternating conditions and rewards, computationally expensive explorations, including low-quality training data, leading to unstable performance [31].

However, the aforementioned studies lack key insights regarding the potential combination of offline-to-online RL-based solutions and block sizing. Specifically, none of them exploits the capabilities of offline-to-online RL principles for resolving the issue of block sizing in DLTs, while simultaneously minimizing latency and CPU utilization. Consequently, they leave unresolved the challenge of enabling end-users to interact with DLTs like Shimmer efficiently, without the need to control network conditions. To close this gap, we introduce ABS-TD3 to provide adaptive block sizing in real-time and balance the tradeoff between confirmation latency and CPU utilization.

3. Problem Formulation and Analysis

The following section is focused on the presentation of the methodology followed for the development and evaluation of an offline-to-online TD3 system for optimized block submission in DAG-based DLTs. We begin by defining the problem to be solved and its respective objectives that guided our proposed solution. Subsequently, we describe the network environment, namely the Shimmer DLT, that was utilized for the problem definition, as well as its respective solution, and we analyze the development process of the RL agent. Both the offline pre-training and online transition are detailed, highlighting their integrated enhancements, such as: a) Retrieval Augmentation (RA), b) adaptive weights within the reward function, and c) Prioritized Experienced Replay (PER) with exponential decay and novelty bonus. Finally, implementation specifics and evaluation approaches are discussed to

ensure precise performance assessment and justification of the solution’s effectiveness.

3.1. Problem Definition and Objectives

In recent years, DAG-based DLTs have emerged as promising and innovative alternatives to traditional chain-based blockchain networks with improved performance. Unlike traditional blockchains, DAG architectures enable the creation, submission, and verification of multiple blocks, or transactions, simultaneously, with a significant impact on scalability. However, this concurrent mechanism introduces new challenges, directly affecting the end-users in terms of block creation processes with unpredictable latencies or requirements on computational resources during data block submissions. Consequently, balancing these factors is critical for the maintenance of high throughput in such DLTs [32].

As mentioned in the introduction, the problem of minimization regarding latency and computational resources for the data block submission process, from the end-user’s perspective, is correlated with the dynamic selection of block parameters, under network conditions that are unable to be controlled. Blocks with larger message sizes are susceptible to increased CPU loads, potentially causing delays or leading to resource exhaustion on the user’s device. Conversely, the reduction of data block sizes, until the total message size is submitted, risks increasing transaction preparation times, while under-utilizing the full capacity of the DLT. This creates a nuanced trade-off space where end-users must proceed with proper configuration regarding block parameters to balance preparation latency, CPU usage, and network dynamics to optimize the respective data submission process.

Consequently, the core optimization challenge is the integration of suitable solutions for tackling multiple conflicting objectives, namely minimizing block preparation latency and CPU usage, while maximizing throughput in uncontrollable network conditions, and by taking into consideration the respective block sizing factors. As a result, end-users that interact with DLTs for data sharing face dynamic and even stochastic environments, where transaction rates and network congestion vary unpredictably. Static strategies would be inadequate, signaling a lack of adaptability, leading to suboptimal performance. Therefore, we are facing a form of multi-objective optimization problem under uncertainty, considering the requirement of end-users to dynamically configure parameters for efficient block preparation and responsiveness.

This work aims to resolve these challenges through the development of an adaptive block submission policy that focuses on the end-user’s perspective, for the dynamic selection of block size in bytes, in which the total message size to be submitted needs to be split, to minimize submission latency and CPU utilization without sacrificing performance. To successfully address the complex nature of the problem, we employ RL, and specifically an offline-to-online TD3 algorithm, suitable for continuous control tasks in dynamic environments, specifically for the Shimmer DLT network. Our objectives are defined as follows:

- a) Development of a robust RL-based learning framework for efficient block submission in the Shimmer DLT.
- b) Integration and combination of novel mechanisms on the offline basis and online transition of the TD3 model, to ensure effectiveness and adaptability in Shimmer’s real-time conditions.
- c) Demonstration of the system’s performance through comparison with different types of baseline policies.

3.2. *Shimmer DLT Analysis*

With the Shimmer DLT being the main focus of our solution, it is essential to provide information related to its principles and features to understand its core functionalities, including the presentation of data related to the interaction with the respective network in terms of latency and CPU utilization.

3.2.1. *Structure of Shimmer DLT*

The Shimmer DLT is a novel feeless L1 and scalable DLT, built upon the DAG-based architecture of the IOTA network called the Tangle, while being famous for its optimized structure for the support of IoT applications. Originally, the Shimmer DLT acted as the staging network for the IOTA Tangle, enabling developers and end-users to research, develop, and test innovative mechanisms before the official deployment within the IOTA Tangle. Just like the IOTA Tangle, the network is supported through Hornet nodes, which are extremely lightweight and can be installed by the end-users through their dedicated software [33]. As mentioned above, DAG-based architectures in DLTs are defined by their capability of enabling the creation and confirmation of multiple transaction blocks in parallel, surpassing traditional chain-based blockchains in terms of throughput. Specifically, each new transaction submitted and confirmed within the Tangle structure of Shimmer verifies two previous transactions [34].

Generally, a user is able to submit individual transactions, with a maximum permitted size up to 32KB (32768 Bytes) [35], through the utilization of IOTA Software Development Kit (IOTA SDK), with the requirement of breaking the message into smaller fragments if its total size exceeds the permitted limit. Once the message is prepared into its dedicated block, it is posted in Shimmer through a known Hornet node, with the node gossiping the related information to its peers for its confirmation and solidification. The respective process is shown in Fig. 1.

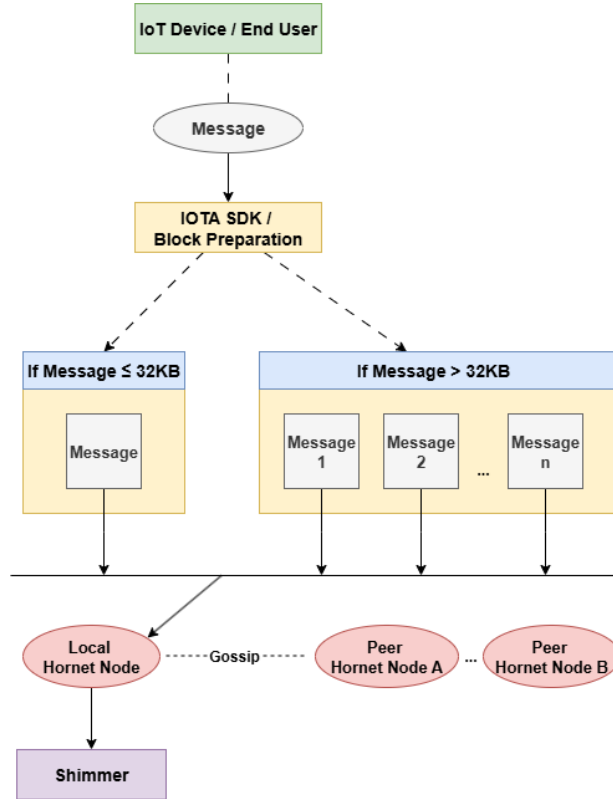


Figure 1: Operational Flow of Message Submission into Shimmer

At its core, Shimmer utilizes an Unspent Transaction Output (UTXO) model within the Tangle structure of the DLT, with submitted transactions defining the outputs of previous submissions as inputs, and consequently uses the respective inputs for the generation of new outputs [36]. Also, apart from its established feeless L1, it integrates an Ethereum Virtual Machine (EVM) compatible environment at its Layer 2 (L2), called ShimmerEVM, enabling

the development and deployment of smart contracts [37]. Moreover, Shimmer implements a dynamic tip selection algorithm that determines which unconfirmed transactions will be approved by new transactions to ensure stability and fairness through prioritization of the tips being published by trustworthy nodes [38].

However, significant changes have been established for the two DLTs, drastically shifting them from their original vision, as also seen in the work of [39]. Specifically, the initial scope of IOTA Tangle for providing an optimized space for IoT and data sharing applications through individual transactions and chainless mechanisms has shifted to the introduction of an object-focused IOTA Rebased with a smart contract-based L1 through the Move Virtual Machine (MoveVM). Similarly, Shimmer has shifted from being the staging network of IOTA Tangle to a community-maintained DLT, making it still suitable for applications requiring secure, efficient, and decentralized data sharing, while maintaining the original vision for an optimized distributed environment for IoT applications.

3.2.2. *Shimmer DLT Network Conditions*

An essential step towards the development of a solution that will focus on the optimization of the block submission process from the user’s perspective is the investigation and understanding of Shimmer’s internal network conditions. Considering the factors of data block sizes and their correlation with submission latency and computational resources utilized for the preparation of the respective blocks, a thorough investigation process was conducted through interaction with the Shimmer DLT.

To establish a realistic and comprehensive point of view of the network’s conditions, a series of data submissions were conducted. Specifically, a range of 8 different sizes in bytes was selected, namely [5000, 10000, 25000, 40000, 55000, 70000, 85000, 100000], reflecting the variation between small, medium, and larger message sizes that the end-user has to submit within Shimmer, with each size being submitted for 250 iterations in total. The number of blocks to be utilized was selected randomly, starting from the minimum number required to hold the message, while considering the limitation of 32KB per block for the cases where the sizes to be submitted exceed the max permitted size. Specifically, for sizes larger than 32KB, the total message size was first divided into equal-sized fragments to meet the specific requirement. Finally, a random upper bound was applied through the extension of up to 9 additional blocks. To simulate data context, and depending on the message

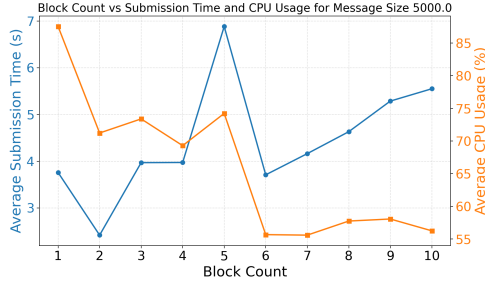
size, each block was filled with random characters of the English alphabet. Additionally, for message sizes that required more than one (1) block for their submission, the data were split into equal chunks based on the random selection of the block amount, with an additional block being added in case of a remainder.

Interaction with the feeless L1 of the Shimmer DLT was achieved through the utilization of the Python binding of the IOTA SDK, with the specific process being detailed in the following implementation sections. With each data submission, four key metrics were collected and stored within a dataset for further analysis, namely the size of the message, the number of blocks utilized for the submission, total submission latency in seconds (s), and the percentage (%) of average CPU utilization for the preparation of the respective data blocks.

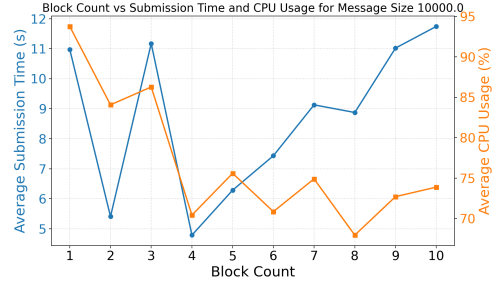
Based on the collected data, we begin by evaluating the correlation of the submission time and average CPU utilization with the selected number of blocks for each message size. In order to detect potential trends, for each message size, we implement an ascending order on the number of blocks utilized and calculate the average latencies and CPU utilization for each block count. The initial results of our investigation can be seen in Fig. 2, with the blue lines representing latencies and the orange lines representing CPU utilization.

Across all message sizes, it is observed that as the sizes grow larger or more blocks are utilized for their submission, there is a general rise in the related submission latency, though not monotonic, considering repeating fluctuations. For example, for 5000 Bytes an increase from 4 to 5 blocks presented a latency increase of $\sim 73\%$, while an increase of 5 to 6 blocks decreased average latency by $\sim 46\%$. At 40000 Bytes, from 5 blocks upward, there is a clear indication of stability regarding the increase of latency, such as an increase by $\sim 7.5\%$ from 5 to 6 blocks, and then an increase of $\sim 30\%$ when increasing to 7 blocks. Similarly, for higher loads such as 85000 Bytes, an increase of 3 to 4 utilized blocks, also increased average latency by $\sim 3\%$, and then an increase to 5 blocks, triggered an increase of $\sim 46\%$ latency. While still considering the case of fluctuations occurring due to network noise, congestion, or limited random selection of the respective block counts, it is safe to assume that with a clean division of the payload into data chunks, there is a minimization of the extra latency overhead.

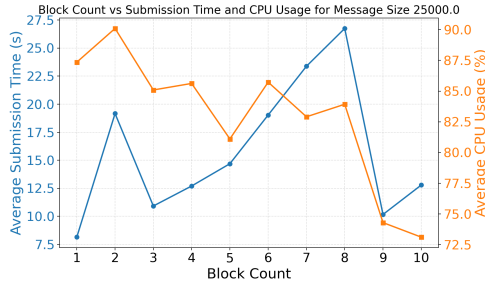
Regarding the CPU utilization, it is observed that the metrics follow a complementary pattern. For smaller sizes, such as 5000 Bytes although we



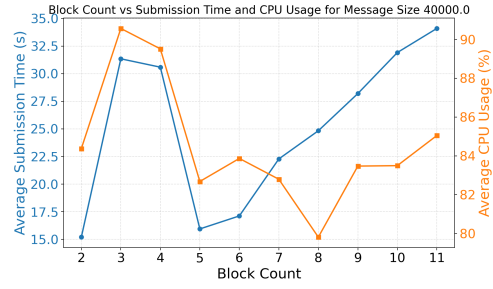
(a) Message Size: 5000 Bytes



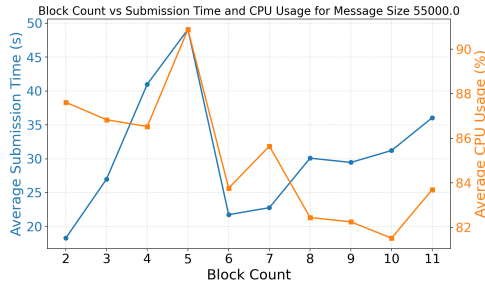
(b) Message Size: 10000 Bytes



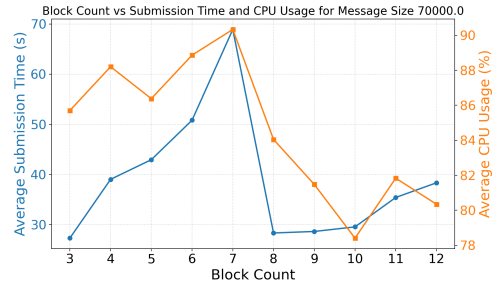
(c) Message Size: 25000 Bytes



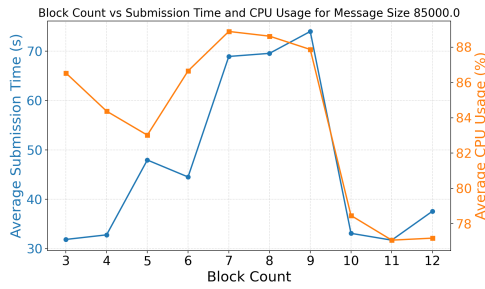
(d) Message Size: 40000 Bytes



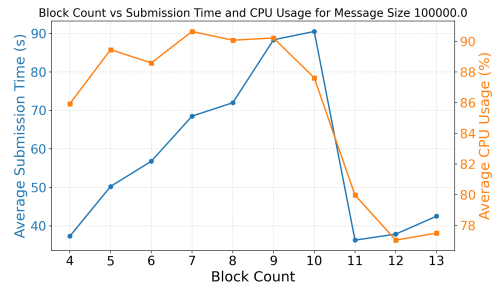
(e) Message Size: 55000 Bytes



(f) Message Size: 70000 Bytes



(g) Message Size: 85000 Bytes



(h) Message Size: 100000 Bytes

Figure 2: Correlation of average submission time and average CPU utilization with block count in the Shimmer DLT, for different message sizes.

observe a small increase in CPU by $\sim 7\%$ when increasing from 4 to 5 blocks, we also notice a significant decrease of $\sim 25\%$ when increasing to 6 blocks. At medium sizes, such as 70000 Bytes, we observe a steady decrease in utilized CPU from 7 to 10 blocks, ranging from $\sim 3\%$ to $\sim 7\%$. Finally, for larger sizes such as 100000 Bytes, we observe a steady decline in larger numbers of utilized blocks, specifically from 9 to 12 blocks, with a CPU decrease ranging from $\sim 3\%$ to $\sim 9\%$. However, just like in latencies, there are spots that signify fluctuations, and that proper division of the total message size may lead to both decreased latency as well as lower processing demands.

To further examine the potential tradeoff between latency and CPU utilization, we proceed by calculating the corresponding metrics across all sizes simultaneously, once again in an ascending order of block count, as seen in Fig. 3. This overall representation signifies a clear positive correlation between the number of utilized blocks and the submission latency, with a steep increase in the beginning, for example a $\sim 57\%$ increase in average latency when rising from 1 to 2 utilized blocks, or a $\sim 22\%$ rise when increasing from 4 to 5 blocks, with small fluctuations such as a decrease of $\sim 7\%$ when increasing from 9 to 10 blocks that does not affect the overall trend. Simultaneously, CPU utilization exhibits a negative correlation with the respective block count, such as a decrease of $\sim 6\%$ when increasing from 1 to 2 blocks, or a drop of $\sim 3\%$ when increasing from 5 to 6 blocks, again with the occasional fluctuations, e.g., an increase of $\sim 2\%$ when increasing from 2 to 3 blocks. Once again, the particular analysis signifies an on-average efficiency in the computational resources required for the preparation of the blocks to be submitted.

To take it a step further and examine the respective correlations on a deeper level, we proceeded by calculating the respective latencies on a per-byte scale, through the following metrics and related equations (1)(2):

- Total submission time (T)
- Average CPU utilization (U)
- Message size (M)
- Submission time per byte (t_b)
- Average CPU utilization per byte (u_b)

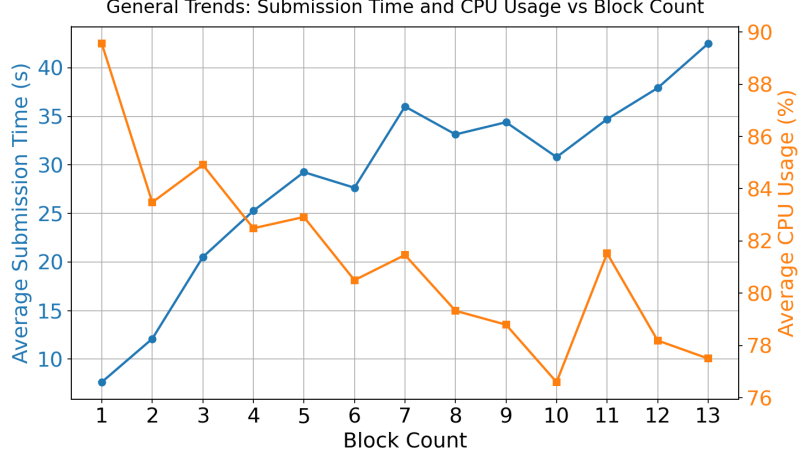


Figure 3: General trend of average latency and CPU utilization in the Shimmer DLT.

$$t_b = \frac{T}{M} \quad (1)$$

$$u_b = \frac{U}{M} \quad (2)$$

In Fig. 4, we present the related metrics of latency and average CPU utilization per byte, for each message size category, sorted in ascending order based on the number of utilized blocks for each submission, with applied regressions to depict visible trends considering the unpredictable noise of each submission.

As seen in the respective per-byte plots, the aforementioned trends are even clearer. Specifically, for mid-range message sizes (5000 Bytes - 40000 Bytes), the tradeoff between latency and CPU utilization becomes visible as the number of utilized blocks increases. Specifically, for the case of 5000 Bytes, as the block count increases, there is an average increase of $\sim 60\%$ for latency, with CPU decreasing by $\sim 34\%$. Similarly, at 40000 Bytes, there is an increase of $\sim 39\%$ for latency, with CPU decreasing by $\sim 7\%$. For medium sizes, such as 55000 Bytes, a similar correlation is observed, although less significant than smaller sizes, with an increase of $\sim 18\%$ for latency and a decrease of $\sim 7\%$ for CPU. Finally, for larger sizes, such as 100000 Bytes, the respective metrics either flatten or improve, showcasing an increase of $\sim 0.2\%$ for latency, with CPU decreasing by $\sim 12\%$. Consequently, both the



Figure 4: Correlation of submission time per byte and average CPU utilization per byte in ascending order of block count, for different message sizes

average metrics depicted in Fig. 2 and Fig. 3, as well as the per-byte metrics presented in Fig. 4, provide a solid foundation for developing a solution that will enable end-users to submit data in the Shimmer DLT efficiently, through their optimal chunking.

3.3. ABS-TD3 for efficient block submission

Considering the potential noise due to network conditions, as well as the on-average tradeoffs between submission latency and CPU utilization in relation to the number and size of utilized blocks, a suitable solution needs to be formulated to facilitate the process of block submission for the end-users.

For that purpose, we proceeded by leveraging the powerful capabilities of RL, by selecting the utilization of an offline-to-online TD3 model, trained with real network data, and enhanced with additional novel mechanisms to boost its learning capabilities and adaptability in real-time. The corresponding subsection will present in detail the process followed for the development of our proposed solution.

Regarding the interaction with Shimmer, we define a message as the full amount of data that a user intends to submit within the DLT. Before the submission, the max block size of $32KB$ is taken into consideration, and if necessary, the message is chunked through the slicing of the total amount into equal-sized fragments in bytes. Finally, each chunk is then inserted into a block, the on-chain container required for its submission within Shimmer.

The conceptual flow of the related interaction is as follows: a total amount of data is collected from IoT devices and DCE frameworks, which is then received from a data aggregator, responsible for their submission into Shimmer. Utilizing our solution, ABS-TD3, the total amount of data is partitioned in an optimized manner for its efficient submission into the Shimmer DLT. Each chunk is then sent to either a dedicated Hornet node or Shimmer’s public API, depending on the end-user’s available resources, for their submission into Shimmer. An overview of the respective topology and operational flow is presented in Fig. 5.

3.3.1. Selection of TD3 as potential solution

The TD3 algorithm is a state-of-the-art Deep RL model, known for its capabilities in environments with continuous action space requirements, and the successor of the Deep Deterministic Policy Gradient (DDPG) algorithm. Specifically, key characteristics involve the utilization of twin Q-networks as a

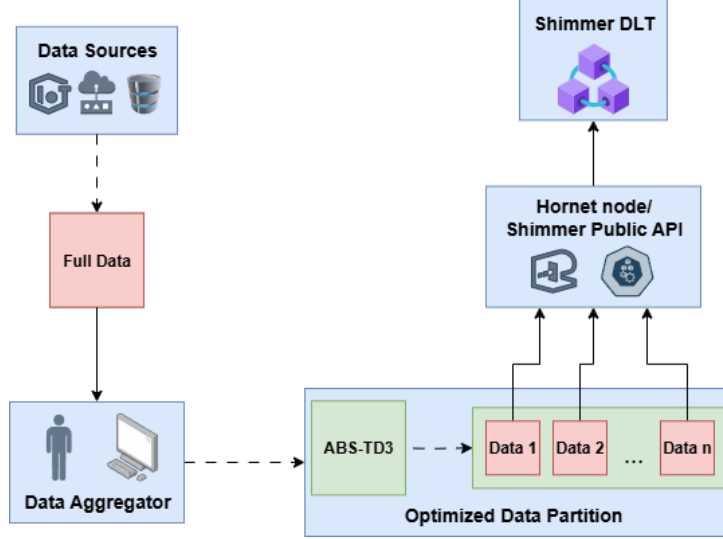


Figure 5: Operational Flow of IoT Data through ABS-TD3 for submission into Shimmer

solution to bias overestimation, target policy smoothing for improved robustness, and delayed policy updates for learning stabilization [40]. Additionally, with the increased demands for higher performance and adaptability, TD3 has progressed into an offline-to-online version, with the offline basis being focused on the pre-training of the agent through a static dataset, and then being transitioned into its online version, while being updated through live interactions with a real-world environment and real-time data [41][42].

To provide additional context regarding its principles, its key feature, as mentioned above, is the utilization of two Critic networks, with the model selecting the minimum value resulting from the two trained Q-networks that are based on the Bellman equation [43]. Moreover, its policy smoothing is achieved through the injection of noise during the target-action selection, as a means of balancing exploration and exploitation [40]. Also, the delayed policy update assists the agent in updating its Actor network less frequently than the twin Critic networks for stabilization purposes during training, proving that the TD3 algorithm matches or even outperforms other mainstream RL algorithms [44].

The selection of the TD3 algorithm over other competitive alternatives, such as the Soft Actor-Critic (SAC) algorithm, is motivated through comparative and empirical studies highlighting their similarities and differences in terms of performance. Generally, both TD3 and SAC are considered among

the leading algorithms for environments with continuous action space requirements [45]. However, the feature of policy entropy in SAC, which is related to its exploration sensitivity, may introduce additional complexities, no matter its stochastic nature. Specifically, the authors of [46] state that although SAC can provide effective exploration and exploitation, TD3 can exceptionally handle high-dimensional states and actions, especially in dynamic environments with a broad range of conditions, while also mentioning the inefficiencies that can occur from entropy tuning in SAC without adaptive mechanisms. The differences between the two algorithms regarding the performance of the training process are also shown in [47], where the authors developed a Deep RL agent for continuous docking control in underwater vehicles, demonstrating the efficiency of TD3 in timestep requirements during training, including the higher rewards it achieved compared to SAC. Finally, as a similar concept to our problem regarding decentralization and resource efficiency, the authors of [48] proceeded in the development of a Federated Learning (FL) agent for decentralized and adaptive optimization of energy resources, with TD3 outperforming SAC with higher rewards and a more stable performance.

These practical applications and empirical findings justify our selection for the offline-to-online TD3 framework, as its deterministic nature and additional enhancements provide greater stability, training performance, and adaptability in dynamic environments like the Shimmer DLT. In contrast, the stochastic behavior of SAC and its entropy-driven functionality, even though it outperforms or matches TD3’s performance in some cases, introduces sensitivity and finetuning complexities that can potentially hinder performance in the noisy and unpredictable conditions of the network.

3.3.2. *ABS-TD3 Offline Basis*

The first part of ABS-TD3 consists of the offline basis, enhanced with novel mechanisms that further empower the learning process and functionality of the agent, namely Retrieval Augmentation (RA), and an adaptive weighting mechanism within the parameters of the reward function of the model. These techniques are utilized for training the model to split the total message size to be submitted into optimal-sized chunks for efficient submission. An overview of the operational flow of the offline TD3 training can be seen in Fig. 6.

For our offline pre-training, we utilized the dataset that was formed during the problem formulation, containing real Shimmer data, while also adding

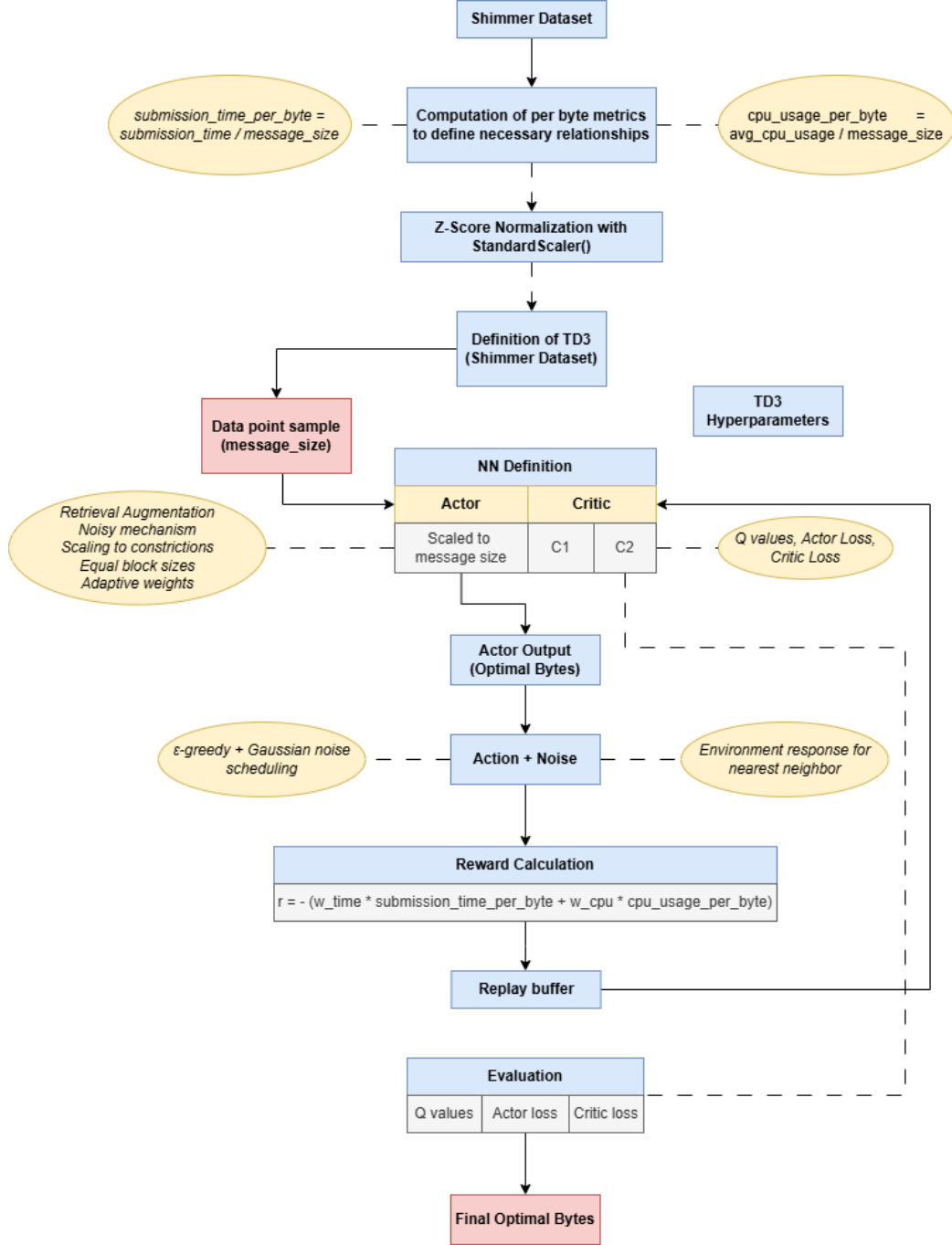


Figure 6: Operational Flow of the Offline TD3 basis

the calculated data of submission time per byte (t_b), and CPU utilization per byte (u_b). Moreover, for additional learning context, we included the category of bytes per block (m_b), which is formulated by dividing the total message size (M) by the block count (num_b), as also seen in equation (3).

$$m_b = \frac{M}{num_b} \quad (3)$$

To ensure stable and efficient learning, the utilized variables of the TD3 model undergo normalization to mitigate scaling disparities among the different implementing, utilizing the *StandardScaler* library, which is based on z-score standardization, as seen in the formula below (4), consisting of the raw implemented value (x), the mean of the utilized data category (μ), as well as the standard deviation (σ) of the respective data category.

$$z = \frac{x - \mu}{\sigma} \quad (4)$$

Considering the perspective of the end-user, in a realistic scenario, the only variable available would be the total message size to be submitted within the Shimmer DLT. Consequently, we have designed ABS-TD3 to receive only the total message size (M) as its respective input state, while enabling random sampling for each training iteration to reduce potential biases from temporal correlations or patterns, and to embrace generalization across diverse network conditions.

Once the size M enters the Actor network, it triggers the functionality of the first innovative mechanism integrated within our solution. Motivated by the work seen in [49], we enhance our Actor network with the Retrieval Augmentation (RA) mechanism. Generally, RA directs an agent to query an external memory of past experiences and retrieve relevant information that facilitates its decision-making process. In our case, the Actor is equipped with a key-value memory buffer, whose keys are two-dimensional (2D) pairs and specifically the total size (M) and the bytes per block (m_b). Similarly, the values store the corresponding submission time per byte (t_b) and CPU utilization per byte (u_b) that are observed in the offline training dataset. For each decision step, the current normalized size (M) is pushed through a linear layer for the formation of a query vector (q), while the related keys are pushed through their own dedicated layer for the generation of key embeddings (k_i). Considering the different scaling that is present within the data, and encouraged by the findings of [50] and [51], we apply l_2 -normalization in

both q and k_i (5) while including a small positive constant to avoid division by zero, and use their dot-product as a cosine similarity (6), in order for the respective similarity scores to depend only on angular alignment rather than vector length, as a means of stabilizing retrieval performance.

$$\hat{q} = \frac{q}{\|q\|_2 + \varepsilon}, \quad \hat{k}_i = \frac{k_i}{\|k_i\|_2 + \varepsilon} \quad (5)$$

$$sim_i = \hat{q} \cdot \hat{k}_i \quad (6)$$

Next, the model keeps only the K most similar keys with the highest similarity scores, with K being a tunable hyperparameter during the implementation of training. Then a soft-max with a small temperature (τ) (7) is applied over the collected K scores, for the creation of the related attention weights and the calculation of a weighted average (8), resulting in a 2D *read* vector, representing the typical submission times and CPU utilizations occurring for situations similar to the input state M .

$$\alpha_j = \frac{\exp(sim_{i_j}/\tau)}{\sum_{l=1}^K \exp(sim_{i_l}/\tau)}, \quad j = 1, \dots, K, \quad (7)$$

$$r = \sum_{j=1}^K \alpha_j v_{i_j}, \quad (8)$$

Finally, the *read* vector is correlated with the respective input state M and passes through a two-layer MLP, resulting in the utilization of the *tanh* activation function for the generation of the initial decision of m_b . Specifically, *tanh* provides a strictly bounded range of $[-1, 1]$, allowing the Actor network to explore increases and decreases around a neutral midpoint without risking out-of-bound or unrealistic selections. An overview of the Retrieval Augmentation process applied in our solution can be seen in Fig. 7

The initial selected action and the respective *read* vector are retrieved from the Actor network, with the action undergoing a noise infusion process for effective exploration and exploitation. Motivated by the works of [52], [53], and [54], we employed a combination of noise mechanisms, as well as a noise scheduling approach. Specifically, we employ a combination of epsilon-greedy (ϵ -greedy) and Gaussian noise (σ), both of which are decayed during the training process. For each episode, we calculate a progress ratio ϕ (9)

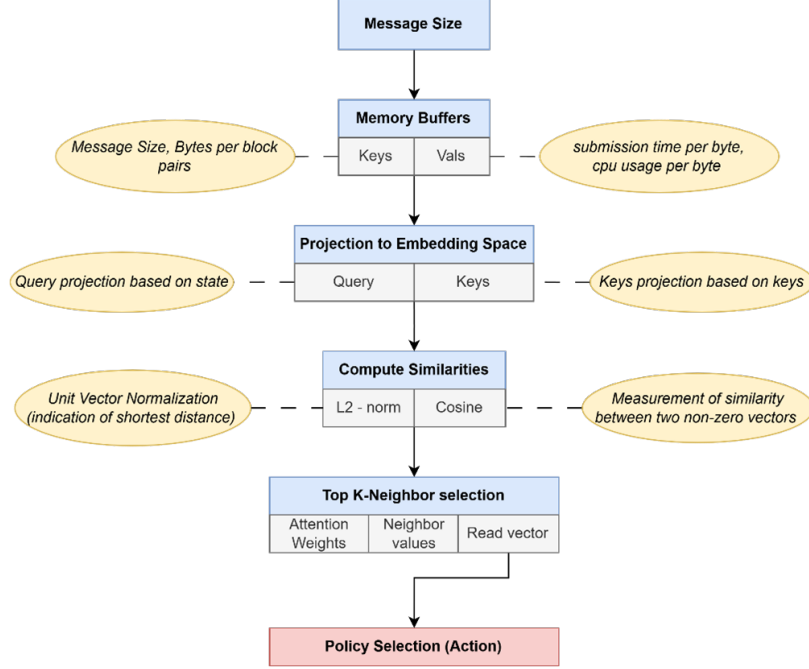


Figure 7: Operational Flow of Retrieval Augmentation

utilizing the current episode (e) and the total number of episodes (E), while subtracting both by 1 for each training iteration, and with the respective noise parameters starting with their beginning values (ϵ_0, σ_0), and decaying to their final values (ϵ_f, σ_f) (10).

$$\phi = \frac{e - 1}{E - 1} \quad (9)$$

$$\sigma = \sigma_0 + \phi \times (\sigma_f - \sigma_0), \quad \epsilon = \epsilon_0 + \phi \times (\epsilon_f - \epsilon_0) \quad (10)$$

Based on the respective approach and during action selection, for probability $\epsilon(\phi)$, the agent proceeds by ignoring the selected policy and selecting a random action uniformly from the default range of $[-1, 1]$. Otherwise, the model proceeds with the utilization of Gaussian noise by drawing a zero-mean sample from $N(0, \sigma^2)$, and adding it to the raw action selected by the agent.

The next step is focused on the transformation of the normalized range of $[-1, 1]$ into the actual range in bytes. To begin with, a known constraint

with the utilization of the Shimmer DLT is the maximum size of a block being $32KB$, requiring the definition of a dynamic upper bound of $b_{dyn} = \min(b_{MAX}, M)$. Consequently, the defined upper bound signals the agent that the respective action cannot exceed b_{MAX} , and it can not be longer than the size M . The corresponding upper bound is then utilized for the rescaling of the normalized $[-1, 1]$ into the actual range of $[1, b_{dyn}]$ in bytes by transforming it through standard linear interpolation given by the formula (11).

$$y = \frac{x - x_{min}}{x_{max} - x_{min}} \times (y_{max} - y_{min}) + y_{min} \quad (11)$$

Based on our available data and desired transformation outcomes, we know that $[x_{min}, x_{max}] = [-1, 1]$, $x = a_{noisy}$, and $[y_{min}, y_{max}] = [1, b_{dyn}]$, with y being the actual byte selection of the agent (b_{act}), and thus resulting in the formula (13).

$$b_{act} = \frac{a_{noisy} - (-1)}{1 - (-1)} \times ((b_{dyn} - 1) + 1) \quad (12)$$

$$= \frac{a_{noisy} + 1}{2} \times (b_{dyn} - 1) + 1 \quad (13)$$

Due to the size of the block required to be an integer number of bytes, the result b_{act} is rounded to the nearest whole byte. By taking into consideration the works of [55] and [22] on the impact of selecting an optimal block size to achieve high performance, we calculate the number of blocks (num_b) to be utilized, based on the agent's selection of optimal bytes per block. Also, we receive the ceiling of the calculation to ensure compliance with the defined upper bound. Moreover, we compute the final optimal bytes per block once more, in case the ceiling was needed (14).

$$num_b = \lceil \frac{M}{b_{act}} \rceil, \quad b_{opt} = \frac{M}{num_b} \quad (14)$$

As proven in several studies, implementing a Nearest Neighbor (NN) mechanism in Deep RL applications, although it's considered a traditional approach, provides effective stability during the learning process [56], [57], [58]. Generally, our approach of utilizing the 1-NN lookup mechanism, attempts to provide local smoothness and on meaningful dimensions related to size M and mb . Specifically, in complex and high-dimensional environments,

traditional 1-NN can underperform due to distance concentration, with the result of few points being defined as NNs to many queries, making a single neighbour unreliable [59]. Moreover, 1-NN is considered to be sensitive to noise and memory imbalance, including single noisy targets, degrading the quality of the resulting action from the agent, and destabilizing learning updates [60].

Therefore, in order to stabilize the learning process of our Retrieval Augmentation within the actor and estimate the respective t_b and u_b , resulting from the selected b_{opt} , we perform a 1-NN lookup within the training dataset. Specifically, through the creation of subsets, the lookup process is restricted to datapoints where the sizes M , match the current input state, with the respective low-dimensional and task-aligned approach mitigating high-dimensional imbalances. In case the current size M is not present within the data, a tunable parameter N is utilized for the lookup to focus on the N amount of closest sizes, that results in the limitation of imbalances and outliers. Consequently, from the selected pool of data, the lookup process is then focused on the datapoints where m_b is closest to the b_{opt} that was selected through the Retrieval Augmentation. Simultaneously, the read vector resulting from Retrieval Augmentation, assists the smoothing of any residual noise of the 1-NN lookup. Finally, the two per-byte metrics, t_b and u_b , are treated as a simulated environment response, which are then used for the calculation of the reward.

Originally, upon calculation of the reward, Deep RL algorithms, including TD3, aim toward the selection of policies that result in the maximization of the reward. However, our objective is focused on the efficiency of the submission process, thus minimizing t_b and u_b . Consequently, the calculated reward (r) is converted into a negative outcome, with the initial form of the reward function being shown in (15).

$$r = -(t_b + u_b) \quad (15)$$

During the investigation and formulation of the problem, it has been proven that on an average scale, a tradeoff between latency and CPU utilization is present, with the sum of the respective weights resulting in 1, and with the final form of the reward function being shown in (16).

$$w_{time} + w_{CPU} = 1, \quad r = -(w_{time} \times t_b + w_{CPU} \times u_b) \quad (16)$$

Considering the dynamic environment of the Shimmer DLT and the un-

predictable conditions of the network, ABS-TD3 needs to select policies that are adaptable to such conditions. Therefore, it utilizes the Retrieval Augmentation occurring in the Actor network for the construction of adaptive w_{time} and w_{CPU} implemented within the reward function. Specifically, a second lightweight MLP is integrated within the Actor and receives the same input as the action selection part, namely the size M and the *read* vector of the Retrieval Augmentation. Due to the existence of the weights tradeoff and their correlation, they are represented in a single logit, which is then passed into a *sigmoid* activation function since it naturally maps a value within the range of $(0, 1)$, which is the exact range utilized for the two weights. As a next step, we assign the resulting value to w_{time} , and then we calculate the remaining value through $w_{CPU} = 1 - w_{time}$. As a result, the w_{time} and w_{CPU} are extracted with the respective *read* vector and initial selected action, and are ready to be utilized within the reward function. Consequently, in the later stages of the agent’s updating, and specifically through the Twin Critics that are explained below, the network back-propagates through the reward function shown in (16) and the weight head contained within the Actor network, with the weight-based MLP learning retrospectively from the previous training episodes when to increase w_{time} and w_{CPU} in order to achieve the required efficiency. An overview of the adaptive weights mechanism is presented in Fig. 8.

The final part of the offline pre-training is dedicated to updating the model. Normally, Deep RL algorithms are designed for environments requiring the resolution of multi-step problems. However, our problem of efficient submission is considered a one-step problem. With each training iteration, a transition is constructed consisting of the related data that resulted during the training episode. Specifically, the input state M , the selected action b_{opt} , the *read* vector of the Retrieval Augmentation, the adaptive weights in their singular value w , and the reward r of the training episode. Moreover, a transition requires involving the next input state, however, due to the one-step nature of the problem, and considering that in a realistic scenario, the size of the next message to be submitted is unknown, the next input state is assigned to be the same size M as the original one. Finally, the last parameter of the transition is assigning *done* = *True*, to finalize the one-step training process. Then, the transition is stored in the replay buffer of the model in order for the twin Critic networks to proceed with the evaluation process by analyzing mini-batch samples of the stored data. The structure of the transition is presented as follows (17):

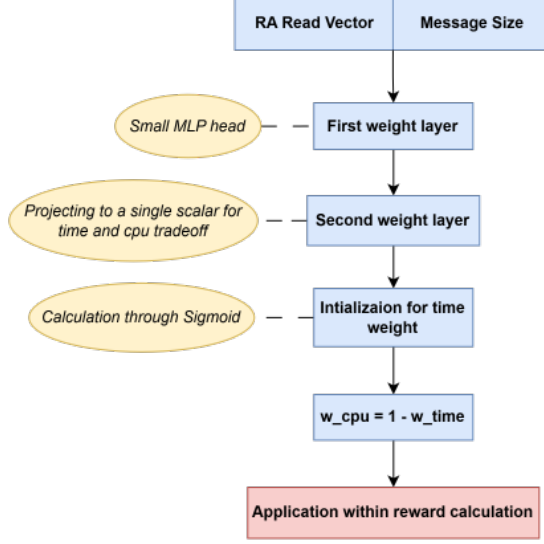


Figure 8: Operational Flow of the Adaptive Weights mechanism

$$transition = (M, b_{opt}, read, w, r, M, done = True) \quad (17)$$

As mentioned in subsection 3.3.1, the evaluation process and calculation of Q values within the twin Critic networks are based on the implementation of the Bellman equation. However, due to the development of a solution that is dedicated to a one-step problem, meaning $done = 1$, the Q value is equal to the reward of the episode. The respective correlation is seen in (18).

$$Q = r + \gamma \times (1 - done) \times \min(Q_1, Q_2), \quad Q = r \quad (18)$$

Even though ABS-TD3 is modified to undertake a one-step problem, with the Q values of each episode being its direct reward, we maintain the twin Critic (Q_1 and Q_2) architecture to benefit from TD3’s innovative features for suppressing over-estimation bias and stabilization of policy updates. Specifically, both Q_1 and Q_2 are two independent networks with different initialization by design, and we proceed by calculating the sum of their Mean Squared Error (MSE) loss based on the immediate reward of the training episode, over the data contained in the total batch size B (19).

$$\mathcal{L}_{critic} = \frac{1}{B} \sum_{i=1}^B [(Q_{1,i} - r_i)^2 + (Q_{2,i} - r_i)^2], \quad (19)$$

One of the main features of the TD3 model for training stabilization is the principle of policy delay, enabling the twin Critics to be frozen after a specific number of updates in order to proceed with the respective Actor updating. After each training episode of the single-step problem, the first critic Q_1 is fitted directly to the observed reward r , thus utilizing it for the calculation of the Actor loss as the negative B -mean of the Q_1 estimations on the Actor’s current outputs. The corresponding minimization signals Q_1 to maximize its predicted reward, and for the actor to select a b_{opt} and adaptive weights that will minimize submission times and CPU utilization, while utilizing the *read* vector of the Retrieval Augmentation for more effective and efficient mapping.

$$\mathcal{L}_{actor} = -\frac{1}{B} \sum_{i=1}^B Q_1(M_i, b_{opt_i}, w_i, read_i) \quad (20)$$

The last step in the training process of the TD3 model is dedicated to the soft-update of the target networks for both the twin Critics and Actor, which is achieved through the method of Polyak averaging (21). With the respective process, the target parameters (θ') are adjusted by infusing a small portion of the parameters of the live networks (θ) into the existing target θ' , and with the updating rate being controlled through a small constant (τ). As a result, the twin Critic networks are capable of comparing their estimations with the reference of their respective targets, and the Actor network obtains a stable, low-noise learning signal.

$$\theta' \leftarrow (1 - \tau) \theta' + \tau \theta \quad (21)$$

Finally, upon completion of the training process, we store the offline basis components in order to be re-deployed in the online transition. Specifically, the weights of the Actor and twin Critic networks, the replay buffer containing the knowledge obtained during the offline training, the scalars utilized for the normalization of the utilized values, and the memorized key-value matrices to re-enable Retrieval Augmentation.

3.3.3. ABS-TD3 Online Transition

The second step towards the implementation of ABS-TD3 is the integration of the online transition of the pre-trained basis. The main purpose of the particular transition is the utilization of the pre-trained components, in order for the TD3 model to interact effectively with the live environment of the Shimmer DLT. Moreover, to further enhance the adaptability of the model within the online and real-time conditions of the network, the online transition is equipped with Prioritized Experience Replay (PER) with exponential decay to facilitate exploitation, combined with a novelty bonus mechanism to enable and balance exploration. An overview of the online transition architecture is presented in Fig. 9.

To begin with, we proceed by loading all of the pre-trained components that were prepared during the offline training, namely the Actor and twin Critic weights, the respective scalars for the normalization of the utilized values, the replay buffer containing the pre-training transition knowledge and acting as a warm-up mechanism for the online transition, and the key-value matrices for the re-activation of Retrieval Augmentation. One additional action that will be presented in detail in the description of the architecture below is the implementation of a small priority constant to the transitions that were stored within the replay buffer during the offline pre-training, as part of the PER mechanism integration.

During the online transition, the replay buffer initially contains the experience acquired during the offline pre-training, and gradually incorporates the transitions created during the live interaction with the Shimmer DLT. This incorporation is achieved with the utilization of the PER and decay mechanisms, which essentially support the agent in replacing the older offline transitions with the new online experience that increases with every batch, while keeping all the collected knowledge sampleable. As a result, the utilized buffer is at first dominated by offline knowledge for stability, while gradually increasing its online experience and shifting into an adaptable behavior. Moreover, both NNs, namely the Actor and Twin Critics, are trained with the same blended sampling process for aligned updating and stable performance.

Essentially, the online transition maintains key components of the architectural flow that were implemented during the offline training. Specifically, the size of the message M to be submitted within the Shimmer DLT is used as an input state, and as a result, retrieving the normalized action selec-

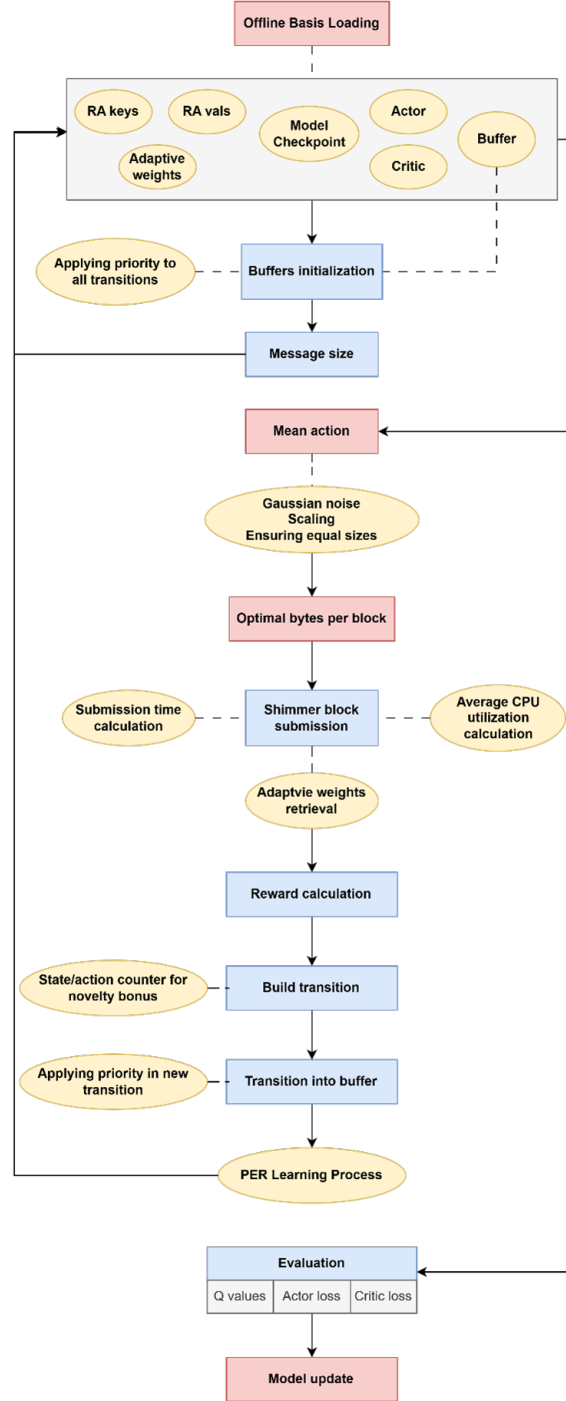


Figure 9: Operational Flow of the TD3 Online Transition

tion, accompanied by the related weight initialization and the *read* vector. Regarding noise mechanisms, due to the purpose of the online transition to realistically interact with the real-time conditions of the Shimmer DLT, we remove the ϵ -greedy approach to avoid the selection of random or extreme policies, and maintain the standard approach of Gaussian noise, slightly enhanced, to enable exploration during the live interaction. The scaling process and definition of a dynamic range in bytes remain as is, and utilize them for the extraction of the actual b_{opt} that the M needs to split into.

Then, each block is prepared and submitted to the Shimmer DLT, utilizing the scripts provided by the IOTA Foundation. The interaction with the respective DLT can be achieved by connecting either to an official Shimmer public endpoint or the known address of an operating Hornet node. For the block to be prepared successfully, a tag identifier and the respective strings of data are required to be converted into a hexadecimal format, and then be placed as parameters in the specialized block creation command *build_and_post_block()*. The submission latency and CPU utilization calculation begin and conclude before and after the initialization of the corresponding command, with the respective metrics being used to retrieve the related t_b and u_b . Similarly to the offline pre-training, the extracted adaptive w_{time} and w_{CPU} , and the respective per-byte metrics are utilized for the calculation of the reward r for the submission process of size M . Once r is calculated, we proceed by constructing the related transition in the format presented in (17).

Past the corresponding point, we proceed by implementing the process of PER with exponential decay, as well as the integration of a novelty bonus mechanism, motivated by the works of [61], [62], [63], and [64]. Generally, PER is an effective exploitation approach, designed to sample transitions in proportion to their Temporal Difference (TD) error, signaling the agent to focus on transitions that can potentially lead to boosted learning progress. However, such a process can trigger a plateaued training progress, with the model maintaining focus only on the learned experiences with an attached high priority, escaping the possibility of further learning. The integration of an exponential decay resolves the corresponding learning plateau by decaying the priority of the experiences that were utilized during the learning process while still propagating value out of the learned information, and by prioritizing newly stored transitions. Nevertheless, a combined mechanism of PER and exponential decay is still an exploitation-based strategy that can potentially lead to neglecting favorable policies of the available state-action

range. Thus, a count-based novelty bonus is implemented to promote the balance between exploitation and exploration. An overview of the implemented process is presented in Fig. 10.

To begin with, a state-action counter ($N(M, b_{opt})$) is constructed, with the main purpose of driving and assisting the process of the count-based novelty bonus, by counting the times that b_{opt} was selected for size M . Simultaneously, both for the pre-stored experiences within the buffer and the newly defined transitions appended into it, an initial priority is assigned to each one of them, equal to $\max(|r| + \epsilon, p_{min})$. By utilizing the absolute reward, the model guarantees non-negative priorities, to later convert them into probabilities. Moreover, a small constant ϵ is added to ensure that each transition receives an initial priority that can be used during the learning process, while utilizing a small priority floor p_{min} , guaranteeing the existence of a safety net that prevents priorities from vanishing and retaining the possibility of being sampled again.

Next, when the replay buffer contains at least B amount of experiences, a training batch is constructed through the conversion of the stored priorities into probabilities, and with the drawing of low variance samples from it. Specifically, the priority value p_i from each transition is collected, with the respective sequence being stored in a *NumPy* vector. Each priority is then raised to the power of α in order to maintain control over extreme outliers and ordering, while dividing the priority by their resulting sum to perform conversion into their probability mass function, as seen in (22).

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha} \quad (22)$$

We then proceed by producing their respective Cumulative Distribution Function (CDF) through $C_k = \sum_i P(i)$, for efficient inverse-transform sampling, combined with systematic resampling for the retrieval of the minibatch to be used for the updating of the Actor and Critic networks and achieve low variance. Specifically, a uniform random number is added within $[0, \frac{1}{B}]$ to the spaced grid of $(\frac{0}{B}, \frac{1}{B}, \dots, \frac{B-1}{B})$, and with each uniform deviate being mapped into an index through a binary search on the CDF. As a result, the corresponding transitions are extracted and form the minibatch that will update the model's networks.

We continue by implementing non-uniform sampling with Importance Sampling (IS) weights, formulated through $IS_i = (N \times P(i))^{-\beta}$, with N representing the replay buffer size of the model and $P(i)$ being the probability

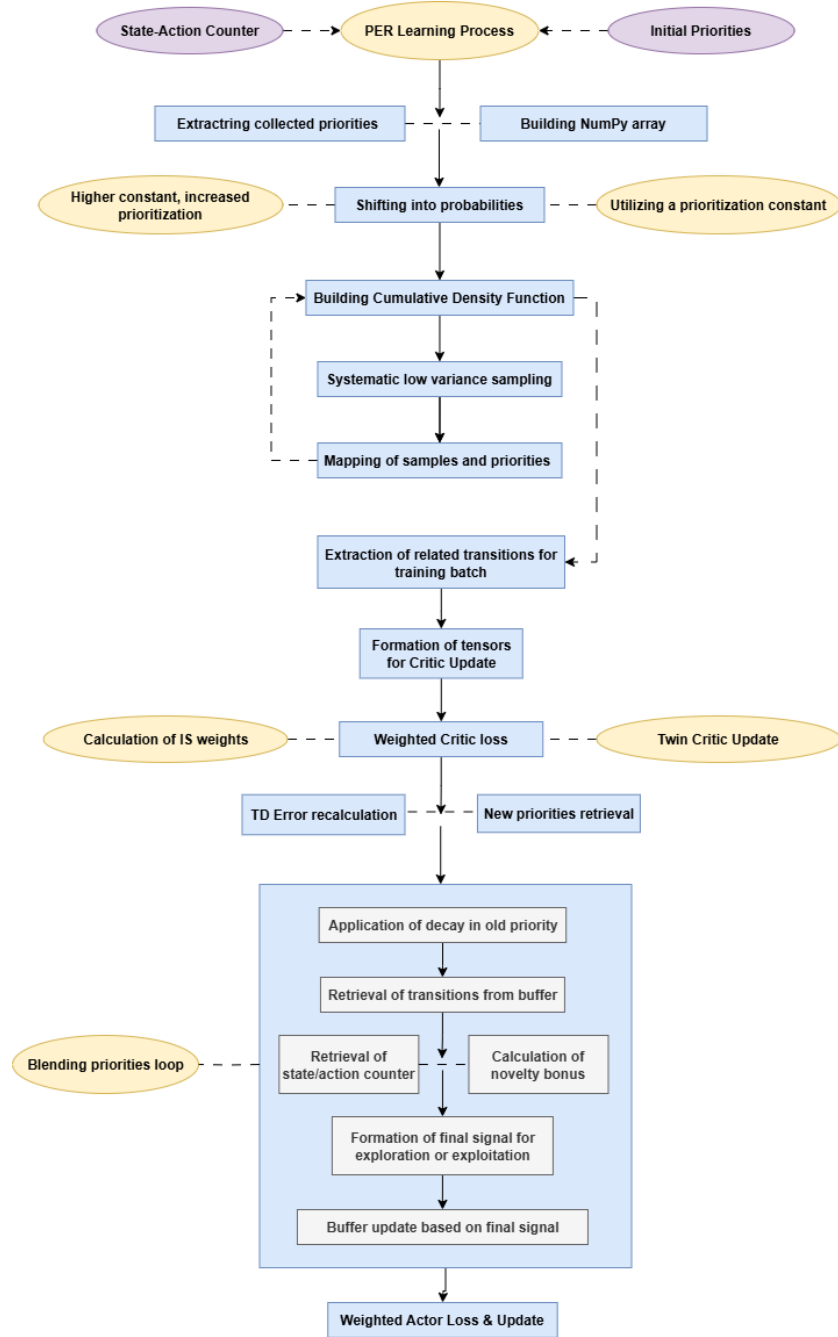


Figure 10: Operational Flow of the PER Learning Process

that the specific transition i was chosen, and β being a tunable parameter that controls the correction of the bias, as a means of preventing frequently replayed transitions from dominating the updating process. After normalizing the weights by their maximum, we proceed by updating the twin Critic networks, with the difference of applying a weighted MSE loss (23) to control the sampling bias and balance the features of a uniform replay buffer and the variance reduction of PER.

$$\mathcal{L}_{critic} = \frac{1}{B} \sum_{i=1}^B IS_i[(Q_{1,i} - r_i)^2 + (Q_{2,i} - r_i)^2], \quad (23)$$

Upon completion of the twin Critics updating, the TD-errors (δ) of each sample are re-evaluated, while also applying the same concept of adding a small constant ϵ and a p_{min} (24), to ensure that the newly formed priorities will still maintain value for the learning process of the TD3 model. The re-evaluation process of the sampled transitions begins by applying the exponential decay factor ρ in the old priority p_i^{old} , while ensuring that the p_{min} is maintained and guaranteeing that the influence of older transitions is gradually decreased without removing the possibility of re-sampling it (25).

$$p_{new} = \max(|\delta| + \epsilon, p_{min}) \quad (24)$$

$$p_i^{dec} = \max(\rho \times p_i^{old}, p_{min}) \quad (25)$$

Then we continue with the initialization of the count-based novelty bonus by retrieving the respective transition from the buffer, containing the input-state M , and selected action b_{opt} , while utilizing the pre-defined counter $N(M, b_{opt})$. Overall, the novelty bonus enhances the exploration mechanism of the model by investigating actions that have never been selected or have rarely been observed during the online interaction, with the respective novelty being controlled by a constant n and faded as the appearance of a (M, b_{opt}) pair increases. Also, we add a constant of $(+1)$ to avoid division by zero, and give a finite bonus to unseen pairs of (M, b_{opt}) in order to exploit potential effective learning outcomes that have not been yet observed, by promoting the possibility of exploring the respective pairs at least once, and avoiding possible instabilities (26).

$$nov(M, b_{opt}) = \frac{n}{\sqrt{N(M, b_{opt}) + 1}} \quad (26)$$

Motivated by the principle of maximum prioritization, we define a learning signal for each transition as the larger of its TD-error and novelty bonus. This strategy ensures that the model is always driven by whichever of the two factors, meaning exploitation or exploration, is most informative (27). After every update, we compare the decayed priority p_i^{dec} with the newly formed signal, and decide whether the transition should retain its exploitation-exploration status or undergo a priority decay (28). Consequently, the replay buffer shifts its focus between consolidating valuable learned experiences and discovering unexplored policies, promoting adaptation and stability for the model’s on-line interactions.

$$signal = \max(p_{new}, \text{nov}(M, b_{opt}), p_{min}) \quad (27)$$

$$p_{fin} = \max(p_i^{dec}, signal) \quad (28)$$

Last but not least, just like the Critic loss calculation, the Actor loss is calculated through a weighted negative B -mean of the Q_1 estimations (29), and closing with the last step of the update through Polyak averaging, which remains identical with the offline pre-training, and with the respective replay buffer and weights of the Actor and Critic networks to be stored for further use.

$$\mathcal{L}_{actor} = -\frac{1}{B} \sum_{i=1}^B IS_i \times Q_1(M_i, b_{opt_i}, w_i, read_i) \quad (29)$$

4. Evaluation Results

The corresponding section presents the implementation specifications integrated within ABS-TD3, including its performance results compared to three (3) baseline metrics, namely maximum, minimum, and random uniform policy, including the performance occurring from utilizing fixed weights within the reward function.

4.1. Implementation Specifications

For the implementation of ABS-TD3 and to proceed with the experimentation process, we employed two (2) hardware devices. Firstly, for the training process and experimentation, we employed a Lenovo Legion Slim 5 with 32GB RAM, integrated with the CPU AMD Ryzen 7 7840HS and

3.80 GHz clock speed, and the GPU NVIDIA RTX 4070 with 8GB VRAM. Secondly, for the interaction with Shimmer we utilized a dedicated Horner node, powered by an RPi 4B model with 8GB RAM, and using a Broadcom BCM2711 System-on-Chip (SoC) with ARM Cortex-A72 CPU running at 1.5 GHz CPU, and an external SSD drive with 480GB memory. To enable remote interaction between the two hardware devices and treat our Horner node as a client for our block submission process, we operated the node in HTTPS mode. However, as mentioned above, the specific step of utilizing a Horner node is not required, and the end-user can proceed through the public endpoint of the Shimmer DLT.

Moreover, the *Python* language was utilized for the programming of our proposed solution and with the integration of the *PyTorch* libraries for the construction of the model. Similarly, for the calculation of the submission latency and CPU utilization for the block submission process, the *time* and *psutil* libraries were used, respectively.

Starting with the offline basis of the model, the architecture of the Actor and twin Critics employs two fully connected layers of 128 and 64 units, utilizing the ReLU activation function. Regarding the key dimensions implemented within the structure of the TD3 model, we implement one (1) state input, specifically size M , one (1) action output b_{opt} , two (2) for the *read* vector representing the pairs of t_b and u_b , and two (2) for the adaptive weights, representing the w_{time} and w_{CPU} . For the selection of the hyperparameters that enable the proper training of our model, we utilized the *Optuna* library, a well-known optimization framework that enables the detection of favorable parameters within search spaces for a specific number of running trials, while stopping early poorly performing combinations through a built-in pruning mechanism [65]. As stated in [66], RL training and hyperparameter selection can often lead to overfitting if a fixed seed is used. Therefore, considering the unpredictability of the network, we deliberately conducted one random search for 100 trials, and *Optuna*’s *Hyperband* pruner enabled, as the model was tested with different hyperparameter combinations with the goal of maximizing the reward for 2000 episodes as a minimum training effort, and with the combinations showing learning potential moving up to 10000 training episodes.

With the corresponding process, we retrieved the values for the Actor and Critic learning rates, the size of the replay buffer and batch size, the respective policy noise and noise clipping rate, the policy delay for the updating of the Actor network, the constant controlling the Polyak averaging in

the soft-updating process, and the variable γ located within the Q function. Regarding the γ , although the final version of our Q value is equal to the reward r due to our one-step problem, we maintained it in our search grid for consistency purposes. In the *Optuna* process, we have also implemented the three (3) hyperparameters related to the Retrieval Augmentation, namely the variable K , the implemented embedded dimension, and the nearest neighbor variable for the traditional k-NN lookup. Regarding the selection of the noise intensity within the model, our choices for the Gaussian mechanism are motivated by the works of [19] and [67] for wide coverage and policy smoothing, and for the scheduling of the ϵ -greedy mechanism, it was based on the empirical findings of [68].

For the online transition, we maintain the key hyperparameters found during the utilization of *Optuna* for the offline pre-training process. In terms of the strategy followed for noise infusion in the online transition, and as mentioned in subsection 3.3.3, we maintained only a Gaussian mechanism with value motivated by the works of [68] for effective exploration on demanding continuous-controlled tasks. Moreover, our selection of hyperparameter values related to our PER implementation with exponential decay, namely the prioritization exponent, the importance sampling power, and the decaying factor, including the strategy of adding small positive constants to ensure the possibility of sampling, were motivated by the empirical findings of [61] and [62]. Finally, the selection for our count-based novelty bonus was motivated by the range investigated in [63] and the stability shown in [69] for effective exploration. A summative table presenting the hyperparameters used in both the offline and online versions of our model can be seen in Table 1.

4.2. Offline Pre-training Results

The first part regarding the implementation of ABS-TD3 was the offline pre-training of our basis, utilizing the dataset that was structured during the problem formulation presented in subsection 3.2.2. For the offline basis, we input the message size M , to receive b_{opt} , based on the reward calculated through the minimization of the t_b and u_b metrics, while attempting to balance their respective average tradeoff. For the training process, we executed 10000 episodes, while applying a policy delay of 2, meaning that the Actor network was updated every 2 training steps compared to the twin Critics.

In Fig. 11, we showcase the reward plot occurring from the offline pre-training of our basis. Due to the noise occurring from the variance of the

Table 1: Hyper-parameters used in ABS-TD3.

Parameter	Offline	Online
Episodes	10 000	—
Replay-buffer capacity	20 000	same
Mini-batch size	64	same
Discount factor γ	0.955863208093027	same
Target-network smoothing τ	0.08403864920196714	same
Policy-delay	2	same
Policy-noise	0.2786358488144805	same
Noise clip	0.10935969452948402	same
Gaussian schedule	$0.5 \rightarrow 0.2$	fixed 0.6
ε -greedy schedule	$1.0 \rightarrow 0.1$	—
Actor LR	$2.8094754201816465 \times 10^{-4}$	same
Critic LR	$5.572539383999804 \times 10^{-4}$	same
RA K	13	same
RA embedded dimension	32	same
k Nearest Neighbors	40	same
Prioritization exponent	—	0.6
Importance sampling	—	0.4
Minimum priority floor p_{min}	—	1×10^{-3}
Small prioritization constant	—	1×10^{-5}
Priority decay factor	—	0.7
Novelty bonus	—	0.4
Max bytes per block	32 768	same

offline network data, we have implemented a moving average on a window of every 1000 episodes, to clearly depict the resulting trends of the training. From the beginning of the training and up to episode 6000, the model is capable of exploiting the static data for the discovery of policies that are capable of minimizing the per-byte metrics, with the upward average trend beginning aggressively and being more stable as the training progresses. Past that particular point, there is a slower but steady improvement, suggesting that the Actor network can still uncover some value from the offline data, possibly being close to information saturation. Overall, the respective trend curve presents a stable and high-quality baseline that can be used effectively for the online transition.

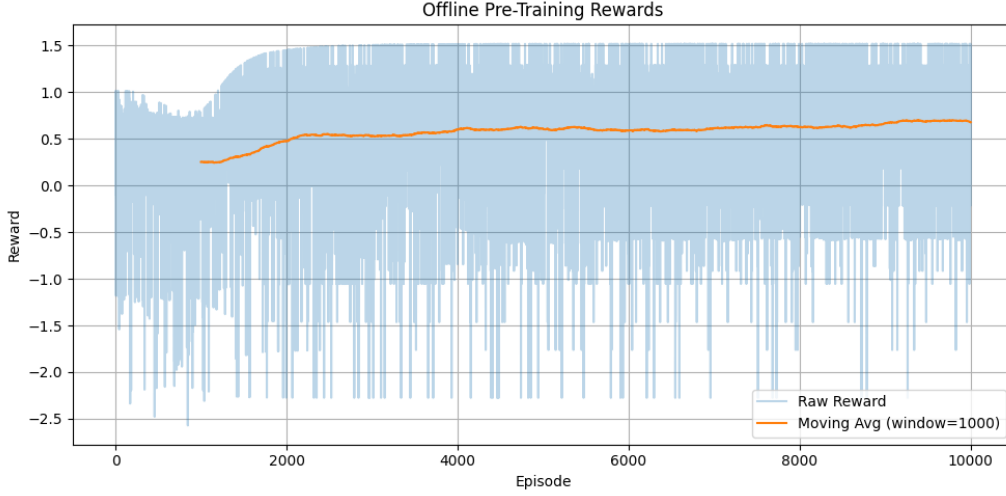


Figure 11: Offline Pre-training Reward Trends

To further ensure that the model successfully reacts to the learning procedure through the offline pre-training before we initialize the online transition, we should also evaluate the respective Actor and Critic losses presented in Fig. 12.

Regarding the Actor network, up to the first 1000 training steps, there is an aggressive boost in the predicted Q-values reflecting the selection of policies that exploit effectively the static dataset, as also shown in the beginning training episodes of the reward plot. Then, the respective actor curve rebounds, followed by a stable behavior for the next training steps, showcasing that in combination with the Critics recalibration, the Actor network proposes actions that the Critics perceive as beneficial, while preventing unstable updates.

Secondly, although the twin Critics begin with a tall spike due to the lack of learning experience, the loss plunges within the next thousand training steps, proving that their estimates start matching the empirical distribution of the dataset. As the training progresses, the respective curve becomes narrower, with minor spikes due to the possible appearance of rare state-action samples that fade immediately. Since the loss curve doesn't fluctuate, it suggests that there is a prevention of reward overestimation or lack of accuracy.

Generally, the Actor plot showcases a healthy learning process with long-

run stability, implying that the offline basis is capable of delivering effective policies, with the Critic loss presenting a fast, stable, and reliable signal for the upcoming online transition.

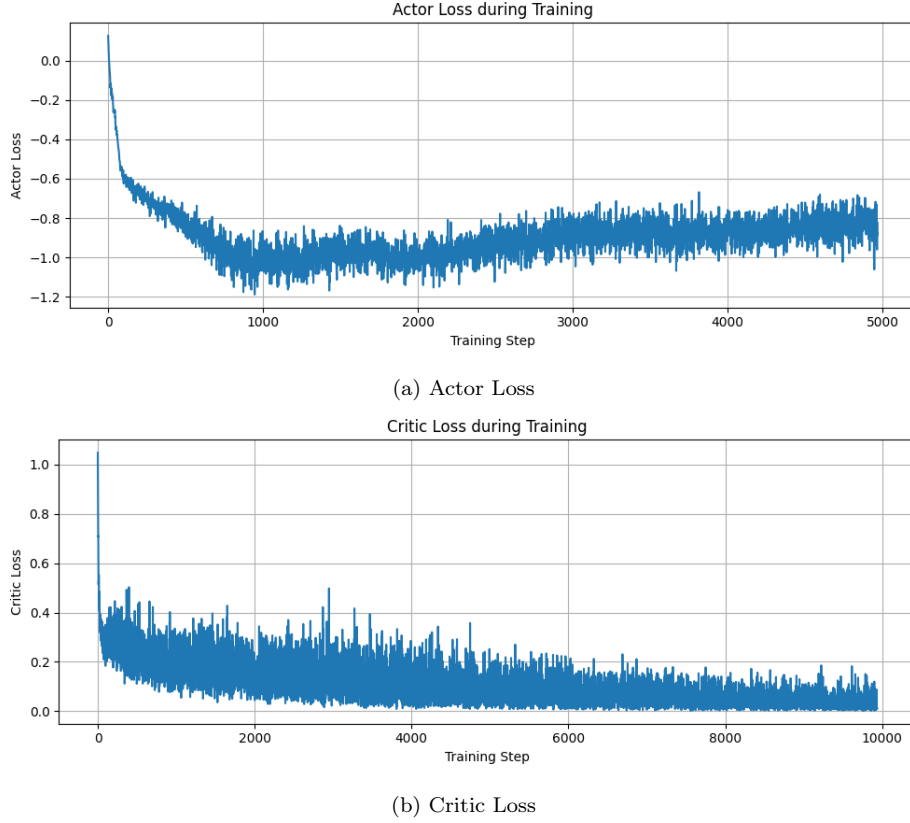


Figure 12: Presentation of Actor and Critic losses during offline pre-training.

4.3. Online Transition Results

As described in subsection 3.3.3, for the online transition, we load the pre-trained TD3 components for the live interaction with the Shimmer DLT. To evaluate the performance of our model, we proceeded by submitting thirty (30) different message sizes for ten (10) rounds, namely [5000, 8275, 11551, 14827, 18103, 21379, 24655, 27931, 31206, 34482, 37758, 41034, 44310, 47586, 50862, 54137, 57413, 60689, 63965, 67241, 70517, 73793, 77068, 80344, 83620, 86896, 90172, 93448, 96724, 100000]. Once again, each size is utilized as input for our model, and based on the selected output, the size is split into

equal-sized blocks that are submitted within Shimmer, while expecting that the model adapts to the network’s conditions through the PER and novelty bonus mechanisms, as well as with the adaptive weights within the reward function to balance exploitation and exploration.

As seen from the empirical findings of [70], one of the main objectives for the online transition of a TD3 model is for the agent to showcase a form of stability in its respective rewards. Similarly to the offline training, in order to detect clear trends from our online transition reward plot, which is presented in Fig. 13, we have applied a moving average on a window of every 30 submissions.

Considering that our reward function is calculated through the minimization of the respective per-byte metrics, the range of the rewards shown in the particular plot lies within $[-4.0715996546, 1.4883659557]$, with their raw form being in the range of $[0.0001650188, 0.0075764788]$, due to the extremely low values occurring from the calculation of the specific metrics.

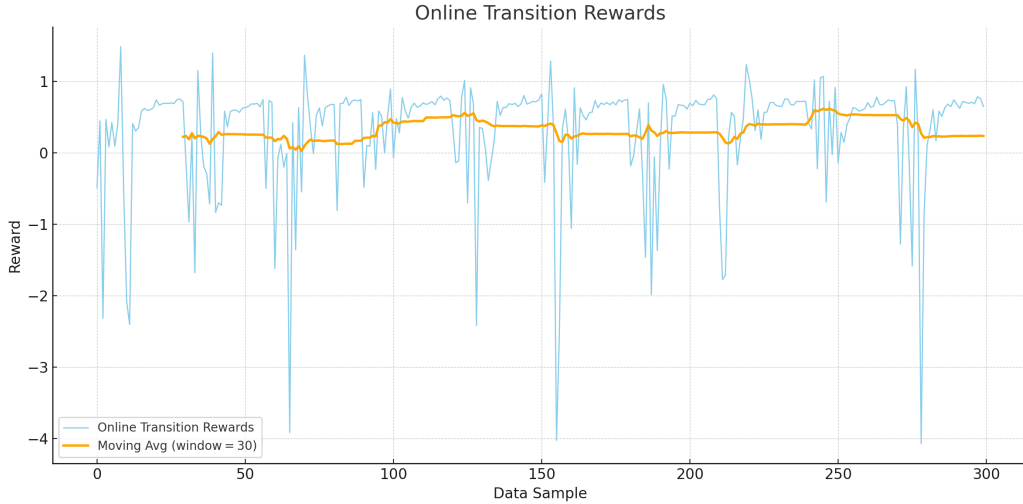


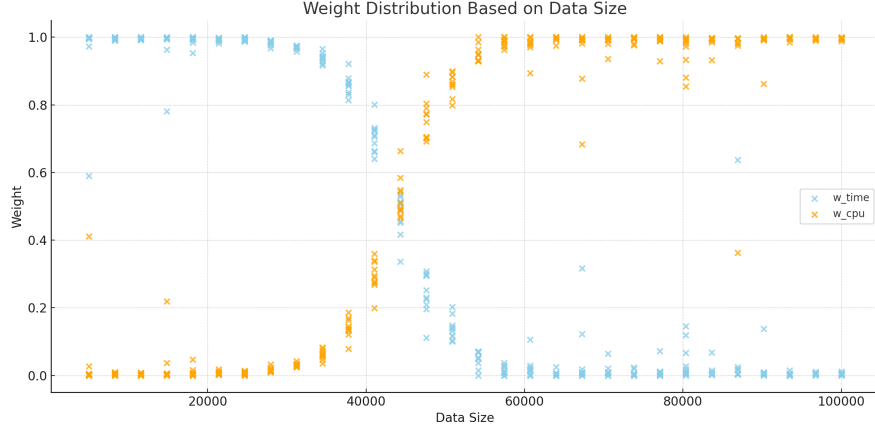
Figure 13: Online Transition Reward Trends

Generally, it is observed that the offline basis successfully affects the on-line transition process and provides a solid performance floor, since despite the presented fluctuations, the majority of the rewards remain positive and don’t drag the moving average below zero. Moreover, the presented fluctuation proves that our model balances exploitation and exploration with the investigation of new $M-b_{opt}$ pairs, and with the moving average showcasing

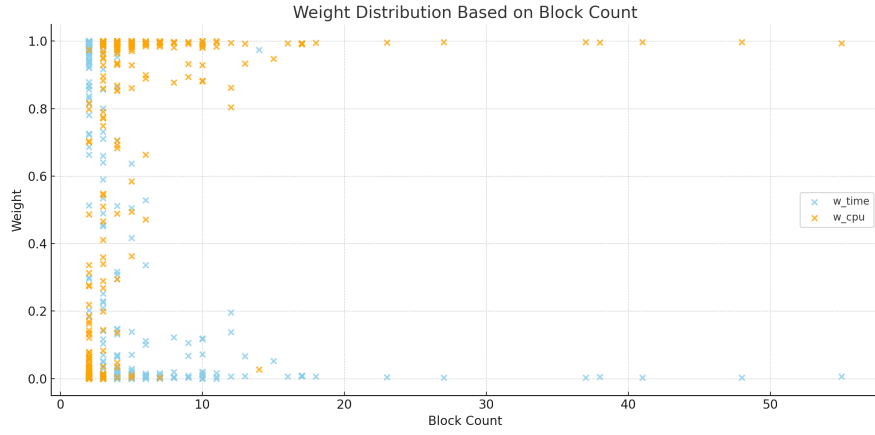
stability. Also, the frequent and stable rises that occur during the live interactions confirm the capability of the Critic networks to convert fresh experiences into valuable learning gains. Overall, these features indicate that our online transition achieves the safeguarding of the offline performance, utilizes the integrated exploration features for effective learning, and demonstrates that continued interaction with the Shimmer DLT is stable and adaptive.

Next, we proceed by evaluating the adaptive weighting mechanism for the minimization of t_b and u_b , which is presented in Fig. 14, with the presented plots being a clear validation of the average tradeoff that was presented during our problem formulation. Specifically, in order to have a proper view of the weight distribution, we plotted the respective data against the implemented message sizes and block counts, in both of which, there is a common pattern. For sizes up to almost 30000 bytes, the model clearly favors latency over CPU, recognizing that the size is small enough to be submitted quickly, and thus making selections that result in a low number of utilized blocks. For sizes up to almost 55000 bytes, we can see that the agent attempts to balance the tradeoff between latency and CPU utilization, while selecting actions that increase the number of utilized blocks accordingly. Finally, for sizes up to 100000 bytes, the model understands the impact of heavy computational payloads when a block reaches its maximum capacity, and as a result, makes selections that focus purely on saving computational resources by proposing sizes that increase even further the block distribution. Consequently, the Actor makes decisive and prioritized choices at the extremes while attempting to balance the in-between conditions, evaluating effectively the dynamic environment of the Shimmer DLT.

Building upon the aforementioned results, Fig. 15 presents the respective distribution of b_b , based on the input size M . Specifically, as the size M grows, so does the b_b selection window of our model, proving the effectiveness of the adaptive weights mechanism, while keeping the respective distribution from exploding. Moreover, the particular distribution showcases that when more weight is assigned to CPU efficiency, then the agent proceeds by selecting a smaller b_b , thus resulting in more blocks. Similarly, when latency is favored, larger b_b sizes are favored, and as a result, fewer blocks for faster submissions. The widening yet well-bounded b_b window confirms that the policy does not rely on a fixed rule set, but it adjusts the respective fragmentation of M to achieve the best latency versus CPU tradeoff. The corresponding adjustment shows that the agent is capable of evaluating the real-time conditions of Shimmer while adapting the respective selection



(a) Weights based on message size



(b) Weights based on block count

Figure 14: Presentation weights distribution in the online transition.

window accordingly.

4.4. Overall Performance Evaluation

In the corresponding subsection, we will proceed by comparing the performance of ABS-TD3 in terms of achieved efficiency in total latency and average CPU utilization during the block submission process, compared to four different baseline metrics, utilizing the same range of message sizes that were used in the implementation process:

- Maximum policy: Submitting data using the maximum capacity of the block, while adding an additional block for a potential remainder. In

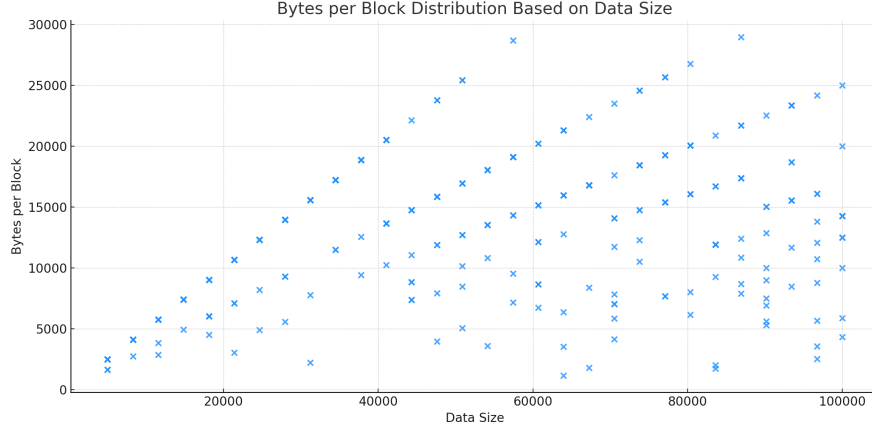


Figure 15: Distribution of Bytes per Block based on the Message Size

order to avoid overhead errors during submission, we defined the block sizes at 32000 Bytes.

- Random policy: Submitting data through random fragmentation of the total message size, based on a dynamic limit within the permitted Byte range, considering the size M and the maximum block size of 32KB.
- Minimum policy: Submitting data using the minimum capacity of the block, while adding an additional block for a potential remainder. For the specific baseline, we defined the block sizes at 128 Bytes.
- Fixed weights policy: Submitting data without the adaptive mechanism within our proposed solution, and by integrating fixed weights within the reward function, namely 0.6 for total latency and 0.4 for CPU, as a means of forcing the balancing of the two per-byte metrics.

In all cases, to ensure proper comparison and clarity for the CPU utilization, we have capped the percentages at 100%, for the submission cases where CPU utilization exceeded the maximum percentage due to the system's turbo-boost features and utilization of additional cores. Overall, 100% CPU is defined as full saturation of the utilized computational resources for the block submission process. Generally, in modern multi-core and turbo boost systems, the collection of readings that are above the 100% mark is frequent during short frequency bursts, and with the capping process being

implemented to maintain comparable results throughout the experimentation process. Practical consequences of exceeding the corresponding limit are related to the consumption of higher dynamic power and thermal events [71], as well as CPU throttling [72], resulting in a degrading end-user experience. Moreover, we begin by presenting the initial performance results, through comparative box plots, as presented in Fig. 16

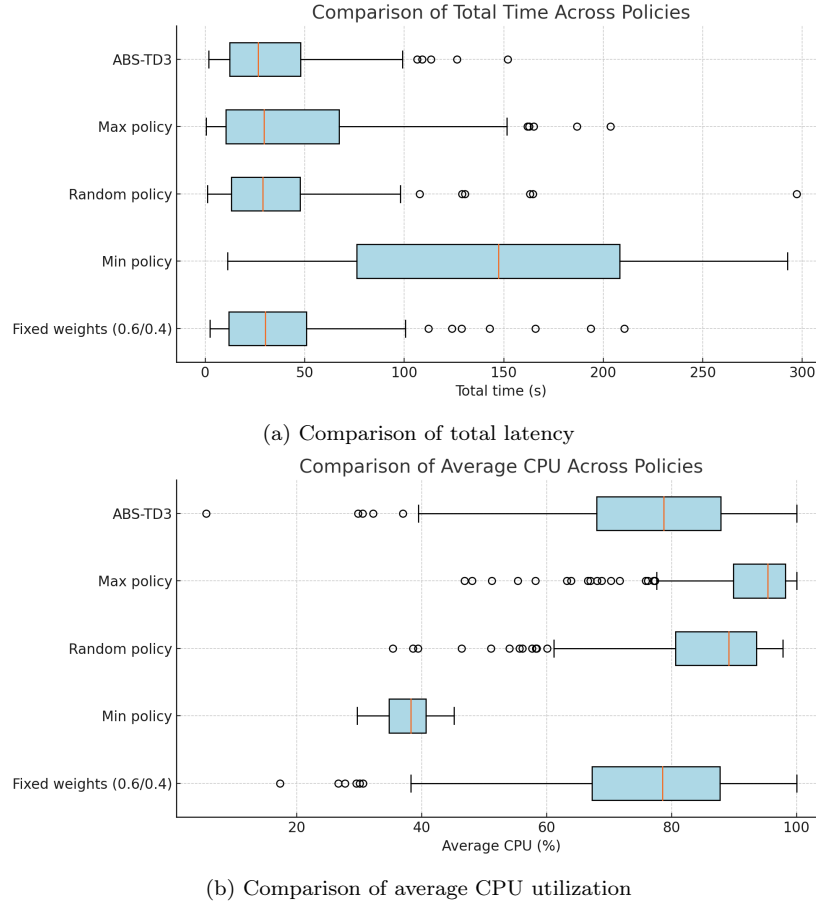


Figure 16: Presentation of total latency and average CPU performance based on median values.

Starting with the comparison in terms of total latency, ABS-TD3 achieves the most efficient median submission time with a value of ~ 27 seconds and with a tight inter-quartile range, signifying efficiency in terms of speed, as well as consistency. The maximum and random policies, including the fixed-

weights approach, achieve a median latency within the range of 29 – 30 seconds, with increased variability considering the appearance of several outliers. Finally, the minimum policy, as expected from our initial observations, achieves the highest latency with a value of ~ 147 seconds.

Regarding the average CPU utilization, we observe a mix of outcomes. As expected, the heaviest payload is achieved by the maximum policy, with its median located slightly underneath 100%, with the random policy following closely. The minimum policy achieves the most efficient CPU utilization, maintaining its median value around 38%, but as seen in the total latency plot, at the sacrifice of significantly longer submission time. ABS-TD3 and the fixed-weight policy share similar medians of $\sim 78.5\%$, with our proposed solution being somewhat tighter, signifying a bit more stability in terms of utilizing computational resources.

To perceive clear indications regarding the overall performance of ABS-TD3, through a single interpretable metric, we proceed by utilizing a combination of approaches validated by the works of [73], [74], and [75]. Specifically, as also approached in [73], we calculate the relative improvement provided by our proposed solution through the formulas (30) and (31).

$$implat = \frac{base_{lat} - model_{lat}}{base_{lat}}, \quad (30)$$

$$imp_{CPU} = \frac{base_{CPU} - model_{CPU}}{base_{CPU}} \quad (31)$$

Then, following the reliability theory regarding two independent parallel components as seen in the works of [74] and [75], we calculate the residual gaps through the utilization of the calculated relative improvements. Moreover, it is stated that their joint residual gap is calculated through their respective product, with the complement of that product being translated into the singular system reliability, thus its overall performance metric (32).

$$OP = 1 - (1 - implat)(1 - imp_{CPU}) \quad (32)$$

The corresponding results of the median, mean, and overall performance of our proposed ABS-TD3 compared to the respective baseline values in percentages and provided in Table 2 and Table 3.

As presented, the experimental results show that our proposed solution, ABS-TD3, provides the best overall performance and balance of total latency and computational resources during block submission in the Shimmer

Table 2: Median overall performance of ABS-TD3 compared to baseline policies.

Policy	Median Time (s)	Median CPU (%)	Median OP (%)
ABS-TD3	26.69	78.72	-
Max policy	29.61	95.39	25.6
Random policy	28.91	89.16	18.5
Min policy	147.34	38.25	62.7
Fixed weights (0.6/0.4)	30.25	78.55	11.6

Table 3: Mean overall performance of ABS-TD3 compared to baseline policies.

Policy	Mean Time (s)	Mean CPU (%)	Mean OP (%)
ABS-TD3	33.64	76.30	-
Max policy	44.10	92.42	37
Random policy	34.54	85.49	13
Min policy	143.96	37.73	52.7
Fixed weights (0.6/0.4)	36.27	76.15	7.1

DLT. Specifically, its submission time, based on both the median and mean values, is faster across all the baseline policies, while keeping CPU utilization demands at a moderate level, avoiding saturation as observed in the extreme cases of the maximum and random policies, and managing both resources efficiently compared to the minimum policy. Regarding the fixed-weight baseline, although it requires almost the same CPU demands as our proposed solution, it lacks in terms of submission time efficiency. Moreover, it is clearly observed that the maximum and random policies utilize the CPU almost to its fullest capacity for the preparation of the respective blocks to be submitted, while achieving moderate latency. Also, although the minimum policy is the most efficient in terms of CPU utilization, it presents extremely slow submission times due to the overhead of submitting multiple small-sized blocks into Shimmer.

However, the overall performance of our proposed ABS-TD3 signifies its effectiveness, delivering a 11.6% and 7.1% efficiency compared to integrating fixed weights within the reward function, with even better optimizations occurring compared to performing random fragmentations of the data to be submitted, while avoiding the CPU and latency penalties of the maximum and minimum policies, respectively. Consequently, it is proven that our so-

lution, ABS-TD3, is capable of yielding faster completion times, smoother CPU utilizations, with more consistent and balanced performance for the interactions of the end-users with DAG-based DLTs like Shimmer.

5. Discussion

This work proposes ABS-TD3, an enhanced and adaptive TD3 agent, focusing on the optimization of block sizes in Shimmer, as an attempt to enable smooth data exchange between IoT mechanisms and DCE ledgers, while assisting in the efficient interactions of the end-users. Starting with the initial scope of our research, the analysis of empirical studies dedicated to optimized submission of transactions-blocks within decentralized networks, showed that the majority of the proposed solutions are focused on the modification of the internal mechanisms that consist the network itself. Although such generalization attempts to provide an optimized network behavior as a whole, it doesn't guarantee stable conditions for all individual interactions. Therefore, we proceeded by researching and developing a solution that provides efficient and adaptable data submission from the perspective of the end-user, by taking into consideration key network factors that cannot be affected by them.

Moreover, it is well known that deep RL technologies are mainly used for complex robotic systems or dynamic infrastructures, such as algorithms related to gaming platforms or AR/VR applications, as well as multi-step perspectives, with limited proposals to other practical fields. Consequently, through our work, we prove that such technologies can be adapted accordingly, and depending on the requirements of the problem to be solved. A specific example of such a case is the proof that, with proper configuration (e.g., Retrieval Augmentation, adaptive weights, PER, and novelty bonus), deep RL technologies and their powerful features can be utilized effectively in complex network environments like Shimmer. Similarly, it has also been proven that multi-step algorithms can be used just as effectively for applications requiring single-step approaches with sufficient results.

Additionally, the case of adapting the TD3 algorithm for a single-step problem, while implementing its offline-to-online transition, caused challenges regarding the immediate feedback provided by the generated action, with the possibility of the respective latency and CPU being tied to short-lived queues of the fresh online transitions, causing initial confusion for the respective correlations. Apart from the aspect of adaptability, we attempted

to resolve the specific issue through the integration of the adaptive weights in the offline pre-training in order to avoid rising CPU metrics when such events occur, as well as through the integration of PER for improved exploitation. Also, the maintenance of the Actor and Twin Critics parameters in both the offline and online transitions of the agent assisted in the respective challenge while ensuring stability in case of the appearance of brief noise from such events.

Considering the results of the proposed mechanism, our solution is defined by the proper modification of the TD3 algorithm to provide efficient and adaptive performance, depending on the real-time conditions of Shimmer. The utilization of Retrieval Augmentation boosted the learning process of the model through the utilization of static, but real-world network data. The adaptive weights mechanisms enhanced the adaptability of the model in selecting policies that favor either latency minimization or optimization of computation resources, or attempts to balance the two factors. The combination of PER with exponential decay and novelty bonus further enhances the adaptability of the model within the conditions of Shimmer, while maintaining exploration for the discovery of effective policies. Overall, through our results, the integration of the aforementioned mechanisms in a unified structure has proven that it can provide sufficient performance, without the need to modify the internal structure of Shimmer.

Finally, our choice to conduct research in Shimmer has been under careful consideration. Specifically, up until recently, IOTA and Shimmer have been among the most innovative DLTs, optimized for data leveraging and sharing within IoT applications, due to their by-design efficient architecture. Recent decisions of the IOTA Foundation have led to the fundamental restructuring of the network, escaping from its original vision of a fee-less L1 network for efficient data sharing, and shifting into a network resembling traditional chain-based features and powered exclusively by fee-based smart contract interactions. Consequently, the complexity of interacting with the network for IoT applications has increased drastically, triggering reactions in both the research and its dedicated community. However, alternative solutions have been presented, with Shimmer remaining operational, without the official support of the IOTA Foundation, and turning it into a community-maintained DLT, and as a result, enabling us to proceed with our research. Also, community initiatives have started progressing as an attempt to continue the original vision of IOTA, such as the Tangle Application Network (TAN), which is currently in its early stages of structuring.

6. Conclusion

In this paper, we introduce ABS-TD3, an adaptive offline-to-online TD3 agent for the proper block sizing, mitigation of congestion, and confirmation latency on Shimmer, as a potential solution for smooth IoT data flow in DCE frameworks, while preserving seamless end-user interactions. For the specific work, we have analyzed real Shimmer network data to understand the factors that affect the interaction of a user with Shimmer, as well as combining different types of novel mechanisms for efficient learning and adaptability.

Specifically, we developed an offline basis with the core integrated mechanisms being Retrieval Augmentation, with its respective outcome being used for the formulation of adaptive weights in the reward function. Secondly, we used the offline basis as the transition into an online setting, which will interact with the Shimmer DLT, and integrated with PER and exponential decay, and a count-based novelty bonus, for enhanced adaptability and balancing of exploitation-exploration.

The evaluation results indicate that the purpose of our model is achieved effectively, compared to four different baseline scenarios, namely maximum, minimum, random, and fixed-weight policies, with the model proving better submission times and optimized CPU utilization, or a balance between the two factors. Consequently, our adaptive RL-powered block sizing agent lays the initial groundwork for smooth data flows between IoT assets and DCE ledgers.

Regarding future work, our main priority is the implementation of the proposed solution in a practical DCE use-case scenario, with the possibility of integration within the context of a DPP application, in order to prove its effectiveness in fulfilling essential requirements of the DCE domain. Secondly, for research purposes, we intend to proceed by adapting and evaluating the respective solution into the SAC algorithm, which is the direct competitor of TD3, based on the empirical findings of the research community. Finally, we will expand the experimentation process by detecting potential differences in alternative dynamic scenarios, specifically by investigating the performance correlations of the submitting device with the node endpoints, such as the utilization of Hornet nodes with different specifications and the submission of blocks through the public endpoint of Shimmer.

Acknowledgments

This work received funding from the European Union’s Horizon Europe Research and Innovation Programme under Grant Agreement No. 101070181. This paper only reflects the authors’ views, and the Commission is not responsible for any use that may be made of the information it contains. The publication of the article in OA mode was financially supported by HEAL-Link.

References

- [1] F. Corsini, N. M. Gusmerotti, M. Frey, Fostering the circular economy with blockchain technology: insights from a bibliometric approach, *Circular economy and sustainability* 3 (4) (2023) 1819–1839.
- [2] R. Soltani, M. Zaman, R. Joshi, S. Sampalli, Distributed ledger technologies and their applications: A review, *Applied Sciences* 12 (15) (2022). doi:10.3390/app12157898.
URL <https://www.mdpi.com/2076-3417/12/15/7898>
- [3] S. Park, H. Kim, Dag-based distributed ledger for low-latency smart grid network, *Energies* 12 (18) (2019). doi:10.3390/en12183570.
URL <https://www.mdpi.com/1996-1073/12/18/3570>
- [4] S. Nowacki, G. M. Sisik, C. M. Angelopoulos, Digital product passports: Use cases framework and technical architecture using dlt and smart contracts, in: 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), IEEE, 2023, pp. 373–380.
- [5] M. Asadi, M. A. J. Jamali, A. Heidari, N. J. Navimipour, Botnets unveiled: A comprehensive survey on evolving threats and defense strategies, *Transactions on Emerging Telecommunications Technologies* 35 (11) (2024) e5056.
- [6] M. Asadi, A. Heidari, N. Jafari Navimipour, A new flow-based approach for enhancing botnet detection efficiency using convolutional neural networks and long short-term memory, *Knowledge and Information Systems* 67 (7) (2025) 6139–6170.

- [7] M. Waleed, K. E. Skouby, S. Kosta, Idia: Iota and decentralized identifiers assisted authentication in smart oceans, in: 2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2024, pp. 414–420. doi:10.1109/WiMob61911.2024.10770467.
- [8] F. Guo, X. Xiao, A. Hecker, S. Dustdar, A theoretical model characterizing tangle evolution in iota blockchain network, *IEEE Internet of Things Journal* 10 (2) (2022) 1259–1273.
- [9] S. Rochman, J. E. Istiyanto, A. Dharmawan, V. Handika, S. R. Purnama, Optimization of tips selection on the iota tangle for securing blockchain-based iot transactions, *Procedia Computer Science* 216 (2023) 230–236.
- [10] S. Abdullah, J. Arshad, M. M. Khan, M. Alazab, K. Salah, Prised tangle: A privacy-aware framework for smart healthcare data sharing using iota tangle, *Complex & Intelligent Systems* 9 (3) (2023) 3023–3041.
- [11] B. Bellaj, A. Ouaddah, E. Bertin, N. Crespi, A. Mezrioui, Drawing the boundaries between blockchain and blockchain-like systems: A comprehensive survey on distributed ledger technologies, *Proceedings of the IEEE* 112 (3) (2024) 247–299.
- [12] H. Hellani, L. Sliman, A. E. Samhat, E. Exposito, Computing resource allocation scheme for dag-based iota nodes, *Sensors* 21 (14) (2021). doi:10.3390/s21144703.
URL <https://www.mdpi.com/1424-8220/21/14/4703>
- [13] K. Liu, M. Jourenko, M. Larangeira, Reducing latency of dag-based consensus in the asynchronous setting via the utxo model, in: 2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), IEEE, 2023, pp. 294–304.
- [14] C. Magrini, J. Nicolas, H. Berg, A. Bellini, E. Paolini, N. Vincenti, L. Campadello, A. Bonoli, Using internet of things and distributed ledger technology for digital circular economy enablement: The case of electronic equipment, *Sustainability* 13 (9) (2021). doi:10.3390/su13094982.
URL <https://www.mdpi.com/2071-1050/13/9/4982>

- [15] A. Heidari, Z. Amiri, M. A. J. Jamali, N. Jafari, Assessment of reliability and availability of wireless sensor networks in industrial applications by considering permanent faults, *Concurrency and Computation: Practice and Experience* 36 (27) (2024) e8252.
- [16] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, J. Morimoto, Deep learning, reinforcement learning, and world models, *Neural Networks* 152 (2022) 267–275.
- [17] Z. Amiri, A. Heidari, N. Jafari, M. Hosseinzadeh, Deep study on autonomous learning techniques for complex pattern recognition in interconnected information systems, *Computer Science Review* 54 (2024) 100666.
- [18] F. Maleki, M. A. J. Jamali, A. Heidari, Unmanned aerial vehicle routing based on frog-leaping optimization algorithm, *Scientific Reports* 15 (1) (2025) 11249.
- [19] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: *International conference on machine learning*, PMLR, 2018, pp. 1587–1596.
- [20] F. Wilhelmi, S. Barrachina-Muñoz, P. Dini, End-to-end latency analysis and optimal block size of proof-of-work blockchain applications, *IEEE Communications Letters* 26 (10) (2022) 2332–2335.
- [21] P. Camponês, H. Domingos, Dynamic optimization of the latency throughput trade-off in parallel chain distributed ledgers, in: *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024, pp. 226–234.
- [22] N. Singh, M. Vardhan, Computing optimal block size for blockchain based applications with contradictory objectives, *Procedia Computer Science* 171 (2020) 1389–1398, third International Conference on Computing and Network Communications (CoCoNet’19). doi:<https://doi.org/10.1016/j.procs.2020.04.149>.
- [23] A. Dutta, N. I. Rafin, M. A. A. Dewan, M. G. R. Alam, Robb: Recurrent proximal policy optimization reinforcement learning for optimal block formation in bitcoin blockchain network, *IEEE Access* 12 (2024) 31287–31311. doi:[10.1109/ACCESS.2024.3369896](https://doi.org/10.1109/ACCESS.2024.3369896).

- [24] S. Yuan, J. Li, J. Liang, Y. Zhu, X. Yu, J. Chen, C. Wu, Sharding for blockchain based mobile edge computing system: A deep reinforcement learning approach, in: 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6. doi:10.1109/GLOBECOM46510.2021.9685883.
- [25] Z. Zhai, S. Shen, Y. Mao, An explainable deep reinforcement learning algorithm for the parameter configuration and adjustment in the consortium blockchain, *Engineering Applications of Artificial Intelligence* 129 (2024) 107606. doi:<https://doi.org/10.1016/j.engappai.2023.107606>.
- [26] A. Heidari, N. Jafari Navimipour, M. A. Jabraeil Jamali, S. Akbarpour, Securing and optimizing iot offloading with blockchain and deep reinforcement learning in multi-user environments, *Wireless Networks* 31 (4) (2025) 3255–3276.
- [27] J. Zheng, K. Li, N. Mhaisen, W. Ni, E. Tovar, M. Guizani, Federated learning for online resource allocation in mobile edge computing: A deep reinforcement learning approach, in: 2023 IEEE Wireless Communications and Networking Conference (WCNC), 2023, pp. 1–6. doi:10.1109/WCNC55385.2023.10118940.
- [28] T. Zhou, M. Xu, D. Qin, X. Nie, X. Li, C. Li, Computing offloading based on td3 algorithm in cache-assisted vehicular noma–mec networks, *Sensors* 23 (22) (2023). doi:10.3390/s23229064. URL <https://www.mdpi.com/1424-8220/23/22/9064>
- [29] Q. Sun, R. Zha, L. Zhang, J. Zhou, Y. Mei, Z. Li, H. Xiong, Crosslight: Offline-to-online reinforcement learning for cross-city traffic signal control, in: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 2765–2774.
- [30] D. Liang, Y. Zhang, Y. Liu, Episodic reinforcement learning with expanded state-reward space, in: *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS '24*, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2024, p. 1192–1200.
- [31] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, T. Hester, Challenges of real-world reinforcement learning:

- definitions, benchmarks and analysis, *Machine Learning* 110 (9) (2021) 2419–2468.
- [32] I. Amores-Sesar, C. Cachin, We will dag you, in: *European Symposium on Research in Computer Security*, Springer, 2024, pp. 276–291.
 - [33] P. S. Nzakuna, V. Paciello, A. Lay-Ekuakille, A. K. Lusala, S. D. Iacono, Empirical analysis of the iota tangle ledger in the stardust stage, in: *2024 IEEE International Symposium on Measurements & Networking (M&N)*, IEEE, 2024, pp. 1–6.
 - [34] C. Fan, S. Ghaemi, H. Khazaei, Y. Chen, P. Musilek, Performance analysis of the iota dag-based distributed ledger, *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 6 (3) (2021) 1–20.
 - [35] I.-C. Lin, P.-C. Tseng, P.-H. Chen, S.-J. Chiou, Enhancing data preservation and security in industrial control systems through integrated iota implementation, *Processes* 12 (5) (2024) 921.
 - [36] S. Müller, A. Penzkofer, N. Polyanskii, J. Theis, W. Sanders, H. Moog, Reality-based utxo ledger, *Distributed Ledger Technologies: Research and Practice* 2 (3) (2023) 1–33.
 - [37] S. T. Muntaha, Q. Z. Ahmed, F. A. Khan, P. I. Lazaridis, Hybrid blockchain-based multi-operator resource sharing and sla management, *IEEE Open Journal of the Communications Society* (2024).
 - [38] J. Wang, J. Yang, B. Wang, Dynamic balance tip selection algorithm for iota, in: *2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Vol. 5, IEEE, 2021, pp. 360–365.
 - [39] P. Sedi Nzakuna, V. Paciello, A. Lay-Ekuakille, A. Kuti Lusala, S. Dello Iacono, A. Pietrosanto, From iota tangle 2.0 to rebased: A comparative analysis of decentralization, scalability, and suitability for iot applications, *Sensors* 25 (11) (2025). doi:10.3390/s25113408.
URL <https://www.mdpi.com/1424-8220/25/11/3408>
 - [40] X. Shen, Comparison of ddpq and td3 algorithms in a walker2d scenario, in: *2023 International Conference on Data Science, Advanced Algorithm*

- and Intelligent Computing (DAI 2023), Atlantis Press, 2024, pp. 148–155.
- [41] X. Yang, J. Song, X. Zhang, D. Wang, Adaptive spiking td3+ bc for offline-to-online spiking reinforcement learning, in: 2024 International Joint Conference on Neural Networks (IJCNN), IEEE, 2024, pp. 1–6.
 - [42] G. Macaluso, A. Sestini, A. D. Bagdanov, Small dataset, big gains: Enhancing reinforcement learning by offline pre-training with model-based augmentation, in: Computer Sciences & Mathematics Forum, Vol. 9, MDPI, 2024, p. 4.
 - [43] P. Srinivasan, W. Knottenbelt, Offline reinforcement learning with behavioral supervisor tuning, in: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI '24, 2024. doi:10.24963/ijcai.2024/545.
URL <https://doi.org/10.24963/ijcai.2024/545>
 - [44] Y. Duan, Continuous control-based load balancing for distributed systems using td3 reinforcement learning, Journal of Computer Technology and Software 3 (6) (2024).
 - [45] S. Liu, An evaluation of ddpq, td3, sac, and ppo: deep reinforcement learning algorithms for controlling continuous system, in: 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023), Atlantis Press, 2024, pp. 15–24.
 - [46] P. Li, D. Chen, Y. Wang, L. Zhang, S. Zhao, Path planning of mobile robot based on improved td3 algorithm in dynamic environment, Heliyon 10 (11) (2024).
 - [47] M. Patil, B. Wehbe, M. Valdenegro-Toro, Deep reinforcement learning for continuous docking control of autonomous underwater vehicles: A benchmarking study, in: OCEANS 2021: San Diego–Porto, IEEE, 2021, pp. 1–7.
 - [48] A. Ravi, L. Bai, J. Lian, J. Dong, T. Kuruganti, Federated deep reinforcement learning for decentralized vvo of btm ders, in: 2024 56th North American Power Symposium (NAPS), IEEE, 2024, pp. 1–6.

- [49] A. Goyal, A. Friesen, A. Banino, T. Weber, N. R. Ke, A. P. Badia, A. Guez, M. Mirza, P. C. Humphreys, K. Konyushova, et al., Retrieval-augmented reinforcement learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 7740–7765.
- [50] H. Steck, C. Ekanadham, N. Kallus, Is cosine-similarity of embeddings really about similarity?, in: Companion Proceedings of the ACM Web Conference 2024, WWW '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 887–890. doi:10.1145/3589335.3651526. URL <https://doi.org/10.1145/3589335.3651526>
- [51] S. Wannasuphoprasit, Y. Zhou, D. Bollegala, Solving cosine similarity underestimation between high frequency words by l2 norm discounting, in: Findings of the Association for Computational Linguistics: ACL 2023, 2023, pp. 8644–8652.
- [52] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, Abstract demonstrations and adaptive exploration for efficient and stable multi-step sparse reward reinforcement learning, in: 2022 27th international conference on automation and computing (ICAC), IEEE, 2022, pp. 1–6.
- [53] J. Xue, S. Zhang, Y. Lu, X. Yan, Y. Zheng, Bidirectional obstacle avoidance enhancement-deep deterministic policy gradient: A novel algorithm for mobile-robot path planning in unknown dynamic environments, Advanced Intelligent Systems 6 (4) (2024) 2300444.
- [54] R. Fox, A. Pakman, N. Tishby, Taming the noise in reinforcement learning via soft updates, in: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI'16, AUAI Press, Arlington, Virginia, USA, 2016, p. 202–211.
- [55] H. Honar Pajooch, M. A. Rashid, F. Alam, S. Demidenko, Experimental performance analysis of a scalable distributed hyperledger fabric for a large-scale iot testbed, Sensors 22 (13) (2022). doi:10.3390/s22134868. URL <https://www.mdpi.com/1424-8220/22/13/4868>
- [56] P. Zhao, L. Lai, Minimax optimal q learning with nearest neighbors, IEEE Transactions on Information Theory (2024).

- [57] J. Shen, L. F. Yang, Theoretically principled deep rl acceleration via nearest neighbor function approximation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 9558–9566.
- [58] D. Shah, Q. Xie, Q-learning with nearest neighbors, *Advances in Neural Information Processing Systems* 31 (2018).
- [59] R. Feldbauer, T. Rattei, A. Flexer, scikit-hubness: Hubness reduction and approximate neighbor search, *Journal of Open Source Software* 5 (45) (2020) 1957. doi:10.21105/joss.01957.
- [60] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, A. Khraisat, Enhancing k-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications, *Journal of Big Data* 11 (1) (2024) 113.
- [61] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, in: International Conference on Learning Representations (ICLR), ICLR, 2016, pp. 1–21.
- [62] M. Brittain, J. Bertram, X. Yang, P. Wei, Prioritized sequence experience replay, in: *Advances in Neural Information Processing Systems, (NeurIPS 2019): Deep Reinforcement Learning Workshop, NeurIPS, 2019*, pp. 1–18.
- [63] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, et al., Never give up: Learning directed exploration strategies, in: 12th International Conference on Learning Representations (ICLR 2024), ICLR, 2024, pp. 1–28.
- [64] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, *Advances in neural information processing systems* 29 (2016).
- [65] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2623–2631.

- [66] T. Eimer, M. Lindauer, R. Raileanu, Hyperparameters in reinforcement learning and how to tune them, in: International conference on machine learning, PMLR, 2023, pp. 9104–9149.
- [67] S. Fujimoto, D. Meger, D. Precup, Off-policy deep reinforcement learning without exploration, in: International conference on machine learning, PMLR, 2019, pp. 2052–2062.
- [68] M. Plappert, R. Houthoofd, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, M. Andrychowicz, Parameter space noise for exploration, in: 6th International Conference on Learning Representations, ICLR, 2018, pp. 1–18.
- [69] Q. Zheng, M. Henaff, A. Zhang, A. Grover, B. Amos, Online intrinsic rewards for decision making agents from large language model feedback, arXiv preprint arXiv:2410.23022Preprint accepted for publication in RLC 2025 (2024).
- [70] A. Beeson, G. Montana, Improving td3-bc: Relaxed policy constraint for offline learning and stable online fine-tuning, in: 3rd Offline RL Workshop: Offline RL as a "Launchpad", NeurIPS, 2022, pp. 1–12.
- [71] C. Jin, X. Bai, X. Zhang, X. Xu, Y. Tang, C. Zeng, A measurement-based power consumption model of a server by considering inlet air temperature, *Energy* 261 (2022) 125126.
- [72] P. A. Misra, M. F. Borge, Í. Goiri, A. R. Lebeck, W. Zwaenepoel, R. Bianchini, Managing tail latency in datacenter-scale file systems under production constraints, in: Proceedings of the Fourteenth EuroSys Conference 2019, 2019, pp. 1–15.
- [73] T. Temizöz, C. Imdahl, R. Dijkman, D. Lamghari-Idrissi, W. van Jaarsveld, Deep controlled learning for inventory control, *European Journal of Operational Research* (2025).
- [74] A. Hoyland, M. Rausand, System reliability theory: models and statistical methods, John Wiley & Sons, 2009.
- [75] S. Sakamoto, Three dimensions of productivity, in: Beyond World-Class Productivity: Industrial Engineering Practice and Theory, Springer, 2010, pp. 67–80.