International Journal of Safety and Security Engineering
Vol. 16, No. 6, June, 2026, pp. 1221-1233
Journal homepage: http://iieta.org/journals/ijsse

Learning Node Trustworthiness in Permissioned Blockchain Consensus Using Reinforcement Learning

Mohammed-Lamine Daikha[1*], Radja Boukharrou[1], Ahmed-Chawki Chaouche[1], Jean-Michel Ilié[2]

[1] MISC Laboratory, University of Constantine 2 – Abdelhamid Mehri Ali Mendjeli Campus, Constantine 25000, Algeria
[2] LIP6, UMR 7606 Sorbonne Université 4 Place Jussieu, Paris 75005, France

Corresponding Author Email: lamine.daikha@univ-constantine2.dz

https://doi.org/10.18280/ijsse.160605

Received: 15 May 2026
Revised: 15 June 2026
Accepted: 22 June 2026
Available online: 30 June 2026

Keywords:
blockchain, validator trust assessment, reputation management, byzantine fault tolerance, reinforcement learning, Double Deep Q-Network, consensus-aware trust

ABSTRACT

Permissioned blockchain systems rely on consensus among authorized validators to ensure security and correctness. However, existing trust and reputation mechanisms are often static and fail to adapt to dynamic node behavior and adversarial conditions. Indeed, reinforcement learning (RL)–based approaches primarily focus on performance optimization without explicitly addressing trust evaluation. In this paper, we propose a trust-aware RL framework for permissioned blockchain consensus based on a Double Deep Q-Network (DDQN). The proposed approach models validator behavior through interaction with the blockchain environment and derives trust assessments from observable consensus outcomes. The framework is adaptive, consensus-aware, and generic, enabling its application across different permissioned consensus protocols. Experimental evaluation is conducted in a practical byzantine fault-tolerant-based permissioned blockchain setting, where validators are simulated using the Java Agent Development (JADE) platform. The results demonstrate stable learning behavior, with convergence achieved from episode 15 onward, and trust scores ranging from 57.59 to 79.85 across 20 validator nodes, producing clear stratification between highly reliable, moderately reliable, and high-risk validators. These results highlight the suitability of the proposed framework for trust-aware consensus evaluation in permissioned blockchain environments.

1. INTRODUCTION

Blockchain technology has emerged as a fundamental building block for decentralized applications, enabling secure, transparent, and tamper-resistant data sharing without relying on a trusted central authority [1]. Initially proposed as the underlying technology for cryptocurrencies, blockchain has evolved into a general-purpose distributed ledger supporting applications across diverse domains, including transportation, finance, the Internet of Things (IoT), smart cities, and healthcare systems [2, 3].

At the core of blockchain systems lies the consensus mechanism, which governs how distributed nodes agree on the validity and ordering of transactions. Classical consensus protocols such as proof of work often assume honest or economically rational participants [4, 5]. However, such assumptions are frequently unrealistic in private and dynamic environments. Nodes may behave selfishly, suffer from resource constraints, or act maliciously, potentially degrading consensus correctness, and overall system reliability. To address these challenges, trust and reputation management mechanisms have been widely studied in distributed systems [6]. Approaches such as Proof of Stake quantify node reliability based on historical behavior [7]. These mechanisms aim to reduce the influence of unreliable or malicious participants and support more informed decision-making in decentralized systems.

While public blockchains prioritize openness and economic incentives, many real-world applications require controlled participation, accountability, and predictable performance. This has driven the adoption of permissioned blockchains, where participants are authenticated and authorized by an organization or consortium [8]. Permissioned blockchains are particularly suitable for enterprise systems, IoT deployments, smart cities, and healthcare environments, where compliance, data governance, and low-latency consensus are essential [9, 10].

Consensus protocols in permissioned blockchains are commonly based on practical byzantine fault tolerant (PBFT) or crash fault tolerant (CFT) designs, which offer fast finality and high throughput under bounded adversarial assumptions [11, 12]. However, even in permissioned settings, node behavior can be abnormal or suspicious, due to workload fluctuations, misconfigurations, insider threats, or partial failures. Treating all authorized nodes equally during consensus may therefore lead to suboptimal validation decisions and reduced robustness.

Despite extensive research, existing blockchain trust models present notable limitations. Traditional reputation-based approaches rely on static historical metrics and exhibit

limited adaptability to dynamic and adversarial blockchain environments [13, 14]. Blockchain-specific trust frameworks improve transparency through on-chain data and smart contracts, yet they often depend on heuristic or rule-based mechanisms that fail to capture evolving node behavior [15, 16]. Although reinforcement learning (RL)–based approaches introduce adaptivity, trust is frequently modeled implicitly or simplified, limiting interpretability, graded correctness assessment, and proposer accountability [17, 18].

Recently, RL has gained increasing attention as a promising tool for enhancing blockchain systems through adaptive and data-driven optimization [19]. RL-based approaches have been applied to tune consensus parameters, improve block propagation, detect anomalous behavior, and optimize validator strategies.

Despite their potential, most existing RL-driven solutions focus primarily on performance optimization. Trust is often treated implicitly or reduced to a binary outcome, without accounting for graded correctness, long-term behavior, or proposer accountability within the consensus process [18].

In this work, we introduce a trust-aware RL framework for permissioned blockchain consensus based on a Double Deep Q-Network (DDQN). By addressing the non-stationary nature of consensus environments, the framework enables stable learning, interpretable trust assessment, and meaningful differentiation of validator behavior. Specifically, this work makes several contributions. First, we propose a DDQN framework that explicitly models validator behavior in permissioned blockchain consensus, combining graded correctness, fault contribution, proposer accountability, and penalty feedback into a unified and interpretable trust score. Second, we design a consensus confidence weighting procedure that adjusts trust updates based on agreement strength, enabling the framework to distinguish reliable consensus rounds. Third, we conduct an experimental validation in a PBFT-based permissioned blockchain environment simulated using Java Agent DEvelopment (JADE), demonstrating stable learning convergence, meaningful node stratification and successful identification of three injected byzantine nodes as the lowest-scoring validators. Finally, we present a baseline comparison and ablation study showing that the proposed trust formulation differentiates nodes far better than simple reputation averaging. Node rankings also remain stable across coefficient variations.

The remainder of this paper is organized as follows. Section 2 reviews related work on trust management and RL in blockchain systems. Section 3 presents the system model and problem formulation. Section 4 details the proposed RL–based trust evaluation framework. Section 5 describes the trust evaluation mechanism. Section 6 presents the implementation details and experimental results. Finally, Section 7 concludes the paper and outlines future research directions.

2. RELATED WORK

Trust management has been widely studied in distributed systems and has recently gained renewed attention in blockchain environments due to the decentralized and trustless nature of these networks [6]. Ensuring reliable node behavior, secure participation, and robust consensus remains a fundamental challenge, particularly in the presence of malicious or selfish actors. Existing solutions can be broadly classified into three main categories. The first category includes traditional trust and reputation models, which evaluate node reliability based on historical interactions, feedback aggregation, and reputation scores, and have been widely applied in peer-to-peer and multi-agent systems, such as EigenTrust [13] and PeerTrust [14]. However, these approaches often exhibit limited adaptability and scalability when directly transferred to blockchain networks. The second category comprises blockchain-specific trust management approaches that integrate trust evaluation into the blockchain architecture or consensus process using on-chain data, smart contracts, and immutable transaction records. Nevertheless, many of these methods rely on static or rule-based mechanisms and fail to capture the dynamic and evolving behavior of nodes [15, 16]. The third category focuses on RL-based consensus and validation mechanisms, where intelligent agents learn adaptive strategies for trust evaluation, node selection, and reward or punishment allocation [17, 20]. Despite their potential, existing RL-based solutions often rely on simplified trust representations or lack a comprehensive and systematic trust modeling framework.

2.1 Traditional trust and reputation models

Early trust and reputation models were primarily developed for peer-to-peer and distributed systems, where the main objective was to assess the reliability of participants based on historical interactions and feedback exchanged among nodes. Seminal approaches such as EigenTrust [13] compute global trust values using eigenvector centrality derived from local satisfaction scores, enabling the identification of trustworthy peers in large-scale decentralized environments. Similarly, models such as PeerTrust [14] and Bayesian-based trust frameworks [21] rely on the aggregation of past behavior, transaction outcomes, and peer feedback to estimate node reliability and mitigate malicious activities. These approaches have proven effective in open peer-to-peer systems by promoting cooperation and isolating untrustworthy participants through reputation-based mechanisms.

However, when applied to blockchain environments, traditional trust and reputation models exhibit notable limitations. First, they typically employ static or slowly evolving trust metrics, which limits their ability to adapt to rapidly changing node behavior and sophisticated adversarial strategies commonly observed in blockchain networks. Second, these models are not inherently consensus-aware, as they do not consider the outcome or strength of consensus rounds, nor do they distinguish between benign disagreements and malicious deviations during block validation and agreement processes. As a result, their direct integration into blockchain-based systems may lead to inaccurate trust assessments and reduced robustness under dynamic and adversarial conditions.

2.2 Trust management in blockchain systems

With the emergence of blockchain technology, a growing body of research has investigated trust-aware and reputation-driven mechanisms to enhance consensus reliability and system robustness. Several studies have proposed reputation-enhanced consensus mechanisms, where voting power, validator selection, or leader election is dynamically adjusted based on the historical behavior of participating nodes, aiming to mitigate the impact of malicious or selfish actors [22]. In

parallel, blockchain-based trust management frameworks have been introduced to record trust evidence, behavioral metrics, and reputation scores in an immutable and auditable manner, leveraging the transparency and tamper-resistance of distributed ledgers to improve accountability and traceability [23]. Despite these advances, recent survey studies indicate that the majority of existing blockchain trust models rely on heuristic weight assignments, predefined thresholds, or externally computed reputation values that are not learned from behaviors data [24, 25].

While such approaches enhance resilience against certain attack vectors, they generally lack adaptivity and struggle to capture complex and evolving behavioral patterns exhibited by nodes over time. Furthermore, proposer accountability and validation responsibility are often either overlooked or treated as binary events, failing to reflect the nuanced influence of proposer behavior on consensus outcomes and overall system trustworthiness.

2.3 Reinforcement learning for consensus validation

RL has been increasingly adopted in blockchain systems as a means to enhance performance, security, and adaptability under dynamic and adversarial conditions [26]. Early RL-based approaches mainly focused on optimizing low-level consensus parameters, such as block generation intervals, transaction throughput, block propagation delay, or mining strategies, by modeling blockchain operation as a sequential decision-making process [27, 28]. With the advent of deep RL, more advanced solutions have been proposed to address complex challenges, including anomaly detection, selfish mining mitigation, and intelligent oracle selection, by learning behavioral patterns directly from on-chain and network-level data [29, 30]. More recently, research has shifted toward adaptive and trust-aware consensus mechanisms that leverage RL and multi-agent RL, particularly in permissioned and IoT-enabled blockchain environments, where nodes exhibit heterogeneous capabilities and dynamic trustworthiness [18, 31].

These studies demonstrate that RL is effective in capturing evolving node behavior and adapting consensus decisions accordingly. However, in most existing works, trust is either evaluated implicitly, embedded as an internal control variable, or approximated through reward shaping rather than modeled as an explicit, interpretable metric. As a result, critical aspects such as consensus agreement strength, graded correctness of validation decisions, and proposer responsibility are often overlooked or simplified, limiting the transparency and explainability of trust evolution. In contrast, only a limited number of studies attempt to integrate RL with a formal and comprehensive trust model that explicitly quantifies these factors, highlighting a clear research gap that motivates the proposed approach.

2.4 Comparison with existing approaches

To systematically position existing trust and consensus solutions, Table 1 summarizes their key characteristics with respect to adaptivity, learning capability, consensus awareness, trust granularity, proposer accountability, and explicit trust representation. These dimensions are particularly relevant for blockchain systems operating in dynamic and adversarial environments, where node behavior, consensus outcomes, and validation responsibilities continuously evolve.

Traditional reputation systems, such as EigenTrust, compute global trust values from historical interactions and provide explicit and interpretable reputation scores; nevertheless, they neither adapt to dynamic consensus processes nor account for validation responsibility within blockchain environments [13]. Reputation-based distributed ledger consensus frameworks incorporate trust or reputation into validator selection, thereby introducing a degree of consensus awareness; however, they typically rely on static or slowly evolving trust metrics, thereby constraining their responsiveness to behavioral changes [31]. More tightly coupled solutions, such as GT-BFT mechanisms, embed trust evaluation directly into the consensus process by dynamically assessing node behavior to influence block proposal and validation. While this achieves stronger integration between trust and consensus, such approaches still lack learning-based adaptivity [32].

Similarly, trust models designed to enhance blockchain consensus security propose formal trust metrics to identify and exclude unreliable participants, yet they do not incorporate advanced learning mechanisms capable of capturing complex or evolving behavior patterns [33]. Blockchain-specific trust frameworks, including VeriBlock, leverage the immutability and transparency of distributed ledgers to store and audit trust evidence, supporting explicit trust computation; however, trust is generally treated as a static or externally derived quantity rather than as a learned variable [23]. In contrast, RL–based consensus optimization approaches introduce adaptivity by learning optimal consensus parameters or miner strategies through interaction with the environment, but they often focus on system-level performance objectives and do not explicitly model trust, consensus confidence, or proposer accountability in an interpretable manner [27, 28].

Table 1. Comparison of trust and consensus approaches

Approach	Adaptive	RL-Based	Consensus-Aware	Graded Trust	Proposer Accountability	Explicit Trust Score
EigenTrust [13]	No	No	No	No	No	Yes
Reputation-based DLT consensus (RCF) [32]	Partial	No	Yes	Partial	No	Yes
Trust model for blockchain consensus [33]	Partial	No	Yes	Partial	No	Yes
GT-BFT trust consensus [34]	Yes	No	Yes	Yes	Partial	Yes
VeriBlock trust framework [23]	Partial	No	Partial	Partial	No	Yes
RL consensus optimization [27, 28]	Yes	Yes	Partial	No	No	No
Adaptive RL trust [18]	Yes	Yes	Partial	Partial	No	Partial

More recent adaptive RL-based trust approaches, particularly in IoT and multi-agent blockchain settings, extend RL to dynamically assess node behavior; however, trust is frequently treated implicitly or embedded within reward functions rather than represented as an explicit, graded trust metric [18]. Overall, the comparison reveals a clear trade-off between interpretability and adaptivity: traditional and blockchain-based trust frameworks offer explicit trust scores and auditability but lack learning capabilities, whereas RL-based approaches provide adaptivity and behavioral awareness while often omitting formal trust representation and accountability mechanisms. This gap highlights the need for comprehensive trust-aware frameworks that integrate adaptive learning with explicit, interpretable trust modeling and consensus-aware responsibility attribution.

Overall, existing trust and consensus solutions exhibit a clear gap between adaptive learning capabilities and explicit, consensus-aware trust modeling, motivating the need for approaches that jointly address trust interpretability, dynamic behavior, and accountability. In the next sections, we introduce the proposed RL–based trust framework, describing its trust model, learning components, and consensus-aware design.

3. MODELING AND FORMALIZATION

Modeling and formalization involve creating precise, mathematical, or logical representations of systems to analyze, simulate, or verify their behavior. This process transforms abstract concepts or informal descriptions into rigorous structures (models) using formal languages, enabling validation, simulation, and transformation into executable code.

3.1 System model

We consider a permissioned blockchain network composed of a fixed set of validating nodes $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$. All nodes are authenticated and authorized to participate in the consensus process. The blockchain evolves through discrete consensus rounds indexed by r.

Similar to the PBFT protocol, a proposer node is selected at each consensus round to propose a candidate block. Each validator node n_i independently evaluates the block and issues a local decision:

$$d_i^r \in \{0,1\},$$

where, $d_i^r = 1$ denotes that node n_i considers the block valid, and $d_i^r = 0$ otherwise.

After collecting all local decisions, the network produces a final consensus decision.

$$D^r \in \{0,1\},$$

which represents the global outcome of the round. The final decision may be obtained through a majority or weighted voting mechanism.

Each node is characterized by a set of operational and behavioral attributes, including historical reputation, reward and punishment scores, communication latency, computational resource usage (CPU time and memory consumption), and past participation in consensus rounds.

The objective of the system is to continuously evaluate the trustworthiness of nodes based on their observed behavior over time, while remaining robust against faulty or malicious participants.

The framework considers three categories of node misbehavior as a threat model. Byzantine nodes deliberately cast votes contrary to the correct block decision in every round. Selfish nodes behave correctly only when it serves their local interest, producing intermittent and unpredictable faults. Transiently faulty nodes deviate due to resource constraints or misconfigurations rather than malicious intent. The system assumes that the fraction of byzantine nodes does not exceed $f \leq \left\lfloor \frac{N-1}{3} \right\rfloor$, consistent with PBFT safety guarantees [11].

The adversary cannot forge consensus messages or observe the agent's internal Q-values. Trust evaluation relies exclusively on observable consensus outcomes, ensuring robustness without requiring prior knowledge of attacker identity.

3.2 Reinforcement learning formulation

The node behavior assessment problem is modeled as a RL task, where an agent learns to assess node behavior through interactions with the blockchain environment.

State Space

At each step, the agent observes a state vector $s_i^r \in \mathbb{R}^{11}$ associated with node n_i in round r, defined as:

$$s_i^r = [\text{NodeID, RoundNb,Reputation, RewardScore, PunishmentFactor, Latency, CPU, Memory, } d_i^r, D^r, \text{ProposerID}]$$

Continuous features are normalized to improve training stability, while categorical attributes are encoded numbers.

Action Space

The agent selects a binary action:

$$a_i^r \in \{0,1\},$$

which represents the agent's predicted *local decision* for node n_i. This prediction is compared against the final consensus decision to evaluate correctness.

Reward Function

The agent is rewarded based on the correctness of its prediction relative to the final consensus decision:

$$R_i^r = \begin{cases} +1, & \text{if } a_i^r = D^r, \\ -1, & \text{otherwise.} \end{cases}$$

This formulation encourages the agent to align its predictions with the global consensus while penalizing inconsistent behavior.

3.3 Trust modeling

Trust is computed at the end of each consensus round based on the observed behavior of nodes during that round. For each node n_i, the system maintains the following cumulative variables over multiple consensus rounds:

P_i: number of participations,

C_i: cumulative correct contribution,

F_i: cumulative faulty contribution,

R_i: proposer responsibility score,

Π_i: accumulated penalty.

The problem addressed in this work is to learn a policy π^* that accurately predicts node behavior while enabling the computation of reliable trust scores that distinguish honest nodes from faulty or malicious ones.

4. PROPOSED TRUST-AWARE REINFORCEMENT LEARNING FRAMEWORK

This section presents the proposed RL-based trust evaluation framework for permissioned blockchain systems. The method leverages a DDQN to model node behavior and dynamically evaluates trust based on consensus participation, decision accuracy, and proposer responsibility.

4.1 Motivation for Double Deep Q-Networks

In the considered blockchain environment, nodes operate under dynamic and partially observable conditions, where behavioral patterns may change over time due to workload, network latency, or adversarial actions. Classical rule-based trust mechanisms fail to adapt to such non-stationary dynamics.

Deep Reinforcement Learning (DRL) enables an agent to learn optimal decision policies through interaction with the environment. However, standard DQN are known to suffer from Q-value overestimation, particularly in noisy environments such as decentralized blockchain systems.

To address this limitation, the proposed framework adopts a DDQN architecture, which decouples action selection from action evaluation. This design improves training stability and yields more reliable Q-value estimates, making it well suited for trust assessment in permissioned blockchain networks.

A natural question concerns why RL is preferred over supervised learning for this task. Three fundamental reasons justify this choice. First, node behavior in permissioned blockchain environments is non-stationary: behavioral patterns evolve over time due to workload fluctuations, network conditions, and adversarial actions, making it infeasible to define a fixed labeled training set that remains valid across all consensus rounds. Second, no ground-truth trust labels are available at deployment time; the only observable feedback is the final consensus decision D^r, which the agent receives as a reward signal rather than a supervised target. Third, trust is not a property of a single interaction but emerges from an accumulated trajectory of consensus participations — a sequential decision process that is inherently better captured by RL than by single-pass classification or regression.

4.2 Architecture of the trust-aware Double Deep Q-Networks framework

Figure 1 illustrates the architecture of the proposed trust-aware framework integrating a DDQN with a permissioned blockchain environment. The architecture is composed of three main components: the blockchain environment, the DDQN agent, and the trust evaluation and rating module.

The blockchain environment models the consensus process executed by a set of authorized validator nodes. In each consensus round, nodes generate local validation decisions, and a final agreement is reached using the PBFT protocol. The environment provides state observations, reward signals, and consensus outcomes (i.e., whether the consensus round succeeds or fails), which are used by the DDQN agent during training. When the consensus decision is valid ($D^r = 1$), a new block is appended to the blockchain.

The DDQN agent is responsible for learning an optimal policy to predict node decision behavior from observed blockchain states. To improve training stability and sample efficiency, the agent employs an experience replay memory that stores transitions of the form (s_r, a_i^r, r, s_{r+1}) collected through interaction with the blockchain environment. Mini-batches are randomly sampled from this replay memory during training, which breaks temporal correlations between consecutive samples and reduces variance in gradient updates.

The agent has two neural networks: an online network and a target network. The online network is used for action selection and policy learning, estimating the action-value function $Q(s, a; \theta)$. At each step, the selected action is determined by an ϵ-greedy policy based on the online network outputs. The target network, parameterized by θ^-, is a delayed copy of the online network and is updated periodically to provide stable Q-value targets during training. Following the DDQN formulation, the target value y_t used to update the online network is computed by decoupling action selection from action evaluation. Specifically, the online network selects the best action for the next state. The online network parameters are then updated by minimizing the temporal-difference loss between the predicted Q-value and the target value. This separation between action selection and evaluation mitigates Q-value overestimation and significantly improves the stability and convergence of the learning process.

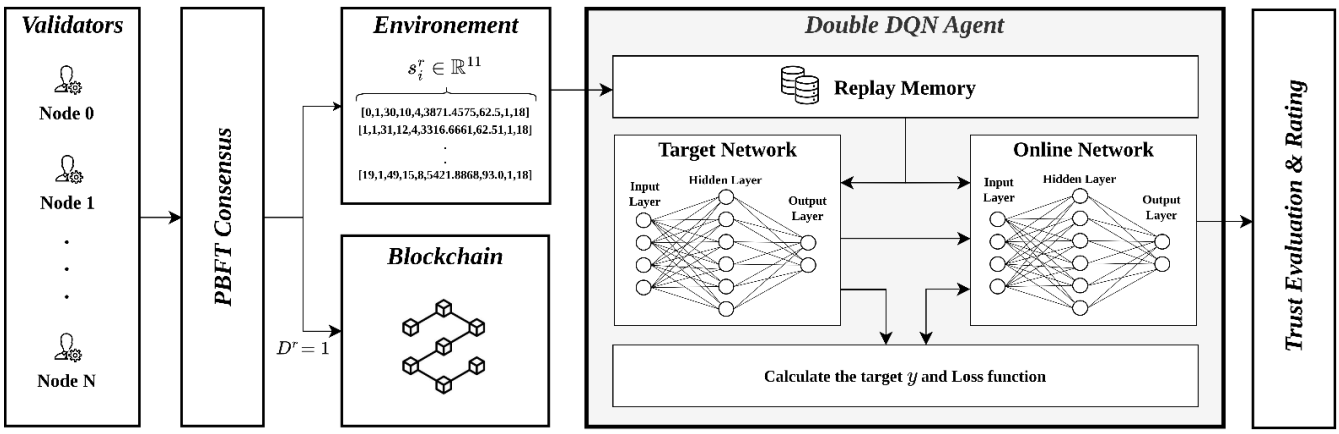

Figure 1. Architecture of the proposed trust-aware Double Deep Q-Networks (DDQN) framework

Importantly, trust evaluation is driven by the online DDQN network, which directly determines node trust ratings through its interaction with the blockchain environment. Trust scores are computed from consensus outcomes resulting from online network decisions, including graded correctness, faulty behavior, proposer responsibility, and accumulated penalty feedback across consensus rounds.

This design ensures that trust scores faithfully capture observed node behavior rather than abstract model inference, thereby enhancing interpretability, and robustness. The resulting trust ratings provide a reliable measure of node credibility and can be readily integrated into higher-level consensus optimization and governance mechanisms.

4.3 Q-network architecture

The Q-network is implemented as a fully connected feedforward neural network. It consists of an input layer corresponding to the state dimension, followed by two hidden layers with 64 neurons each and Rectified Linear Unit (ReLU) activation functions. The output layer contains two linear neurons, representing the estimated Q-values for each action. Formally, the Q-function approximation is defined as:

$$Q(s, a; \theta) \approx f_\theta(s),$$

where, θ denotes the trainable network parameters.

To stabilize training, a target network $Q'(s, a; \theta^-)$ is maintained as a delayed copy of the online network. The target network parameters θ^- are periodically updated by copying the weights from the online network.

An experience replay buffer $\mathcal{D}$ stores transitions of the form:

$$(s_r, a_i^r, r, s_{r+1}, \mathbb{I}_{\text{done}}),$$

which are uniformly sampled during training. This mechanism breaks temporal correlations between consecutive experiences and improves sample efficiency.

The DDQN target is computed as:

$$y_r = r + \gamma Q'\left(s_{r+1}, \operatorname{argmax}_{a'} Q(s_{r+1}, a'; \theta), \theta^-\right),$$

where, γ is the discount factor.

The loss function is defined using the Huber loss to reduce sensitivity to outliers:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[\ell(y_r - Q(s, a; \theta))].$$

For the exploration strategy, an ϵ-greedy exploration policy is employed to balance exploration and exploitation. At each step, the agent selects a random action with probability ϵ, and the greedy action otherwise.

The exploration rate decays exponentially across episodes:

$$\epsilon_{k+1} = \max(\epsilon_{\min}, \epsilon_k \cdot \lambda),$$

where, $\lambda \in [0,1]$ controls the decay speed.

4.4 Training procedure

The training process iteratively interacts with the blockchain environment over multiple episodes. During each episode, the agent observes node states, predicts actions, receives rewards based on consensus correctness, and updates the Q-network using sampled experiences. Algorithm 1 summarizes the complete training procedure.

Algorithm 1. DDQN-Based Trust Learning

```
Initialize replay buffer D
Initialize online network Q(s, a; θ)
Initialize target network Q'(s, a; θ⁻)
for each episode do
    Reset environment and obtain initial state s_0
    while episode not terminated do
        Select action a_i^r using ε-greedy policy
        Execute a_i^r, observe reward r and next state s_{r+1}
        Store (s_r, a_i^r, r, s_{r+1}) in D
        Sample mini-batch from D
        Update θ using Double DQN loss
        Periodically update target network θ⁻
    end while
end for
```

5. EVALUATION MECHANISM

This section describes how the trust variables defined in Section 0 are aggregated to compute a quantitative trust score for each node.

5.1 Consensus confidence weighting procedure

Let ρ^r denote the agreement ratio in consensus round r, defined as the fraction of nodes whose local decisions match the final decision:

$$\rho^r = \frac{|\{n_i : d_i^r = D^r\}|}{|\{n_i : n_i \text{ participates in } r\}|}.$$

A confidence weight w^r is derived as:

$$w^r = \begin{cases} 1.0, & \rho^r \geq 0.8, \\ 0.7, & 0.6 \leq \rho^r < 0.8, \\ 0.4, & \rho^r < 0.6. \end{cases}$$

The confidence weights are designed to reflect different levels of agreement among validator nodes during each consensus round. When the agreement ratio ρ^r is high ($\rho^r \geq 0.8$) the consensus outcome is considered highly reliable. Therefore, the maximum confidence weight $w^r = 1.0$ is assigned. For moderate agreement levels ($0.6 \leq \rho^r < 0.8$), a reduced weight of 0.7 is used to account for the increased uncertainty while still recognizing a majority alignment. When the agreement ratio falls below 0.6, the decision reliability becomes weak, and a lower weight of 0.4 is applied to limit the influence of potentially unstable or conflicting validation outcomes. This tiered weighting strategy allows the trust model to emphasize strong consensus while attenuating the impact of uncertain rounds.

5.2 Correct and fault contributions

Using the confidence weight w^r, the cumulative correct and fault contributions of node n_i are computed as:

$$C_i = \sum_r \mathbb{I}\,(d_i^r = D^r) \cdot w^r,$$
$$F_i = \sum_r \mathbb{I}\,(d_i^r \neq D^r) \cdot w^r.$$

The corresponding rates are obtained by normalization:

$$\text{CorrectRate}_i = \frac{C_i}{P_i}, \quad \text{FaultRate}_i = \frac{F_i}{P_i}.$$

5.3 Proposer responsibility

When node n_i acts as proposer in round r, it assumes additional responsibility. The proposer contribution is defined as:

$$R_i = \sum_r \mathbb{I}\,(p^r = i) \cdot \begin{cases} +w^r, & D^r = 1, \\ -w^r, & D^r = 0. \end{cases}$$

The normalized proposer rate is:

$$\text{ProposerRate}_i = \frac{R_i}{P_i}.$$

5.4 Penalty rate

Let Π_i^r denote the penalty incurred by node n_i in round r. The normalized penalty rate is defined as:

$$\text{PenaltyRate}_i = \frac{\sum_r \Pi_i^r}{P_i \cdot \max(\Pi)}.$$

5.5 Final trust score

The final trust score of a node n_i is computed as:

$$\text{Trust}_i = 100 \times (\alpha \cdot \text{CorrectRate}_i - \beta \cdot \text{FaultRate}_i + \delta \cdot \text{ProposerRate}_i - \gamma \cdot \text{PenaltyRate}_i)$$

where, α, β, δ, and γ are weighting coefficients.

The trust score is clipped to the interval [0,100] to ensure numerical stability and interpretability.

By combining RL outcomes with consensus-aware trust metrics, the proposed mechanism achieves three key objectives: adaptability to evolving node behavior, robustness against transient faults, and accountability of proposer nodes. This makes the approach particularly suitable for consortium and private blockchain environments.

6. IMPLEMENTATION AND EXPERIMENTATION

This section presents an experimental evaluation of the proposed DDQN-based trust assessment framework. The evaluation analyzes the learning dynamics of the agent, the impact of exploration–exploitation trade-offs, and the resulting trust score evolution. All observations are supported by plots generated during training and testing.

Experiments are conducted on a permissioned blockchain network composed of $N = 20$ validator nodes operating over sequential consensus rounds ($R = 600$), producing one record per node per round, yielding ($20 \times 600 = 12000$) interaction traces in total. The validators are simulated using the JADE platform [35], where each node runs within its own isolated container to enable accurate measurement of resource consumption, including CPU and memory usage. A Practical PBFT consensus protocol is employed, in which one node is selected as proposer in each round and the remaining nodes submit local decisions regarding block validity. During this process, the DDQN agent interacts with the JADE simulation environment in an online manner. At each consensus round, the environment generates state observations and consensus outcomes in real time, which are immediately fed to the agent as experience transitions. This design reflects a realistic deployment scenario where the agent learns continuously from live consensus activity rather than from a pre-collected static dataset.

To evaluate the framework's ability to identify adversarial behavior, three nodes n_{16}, n_{17}, and n_{18} are designated as byzantine during the simulation. These nodes systematically cast votes opposite to the correct block decision in most consensus round, representing worst-case adversarial behavior within PBFT's fault tolerance bound of $f \leq \left\lfloor \frac{N-1}{3} \right\rfloor = 6$, where $N = 20$.

The RL agent interacts with the environment at the granularity of a node-round pair. Each interaction step corresponds to one row in the dataset.

6.1 Dataset and training configuration

The dataset consists of approximately ~12,000 records, each representing a node participation in a consensus round. Each record includes operational metrics (latency, CPU usage, memory usage), behavioral indicators (local decision, proposer role), and historical trust-related attributes. Training is performed using:

- Number of episodes: E = 50
- Training steps per episode: equal to the number of dataset rows
- Replay buffer size: 20,000 transitions
- Batch size: 64
- Discount factor: $\gamma = 0.99$
- Learning rate: 10^{-3}
- Target network update interval: 1,000 steps

All continuous state features are normalized prior to training to ensure numerical stability.

6.2 Exploration strategy and epsilon decay

Figure 2 illustrates the evolution of the exploration rate ϵ across training episodes. The exploration rate is initialized at $\epsilon_0 = 1.0$ and decays exponentially with a factor of 0.95 per episode.

Over 50 training episodes, ϵ decreases from 1.0 to 0.0809, ensuring extensive exploration during early episodes and a gradual shift toward exploitation as training progresses. Specifically, ϵ falls below 0.5 after 14 episodes and below 0.2 after 33 episodes, indicating a controlled reduction of random actions.

This decay schedule enables stable policy learning by preventing premature convergence while allowing the agent to rely increasingly on learned Q-values in later training stages.

6.3 Training loss analysis

Figure 3 presents the training loss measured at each learning

step over approximately 1.2×10^5 updates. During early training, the loss exhibits high variance, with peak values reaching approximately 0.55, reflecting unstable Q-value estimates and limited replay buffer diversity. As training progresses, the loss rapidly decreases and remains consistently below 0.1 for the majority of steps, with bounded oscillations induced by stochastic minibatch sampling.

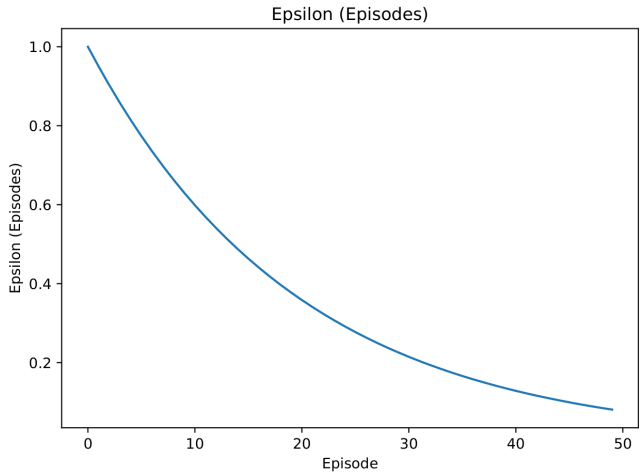

Figure 2. Evolution of the exploration rate (ϵ) over training episodes

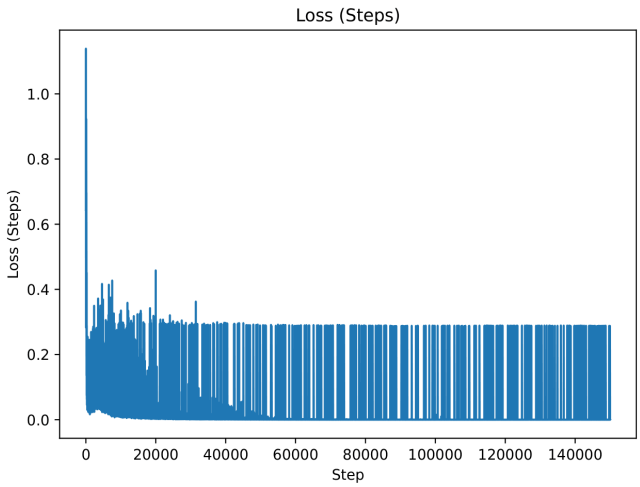

Figure 3. Training loss values measured at each learning step

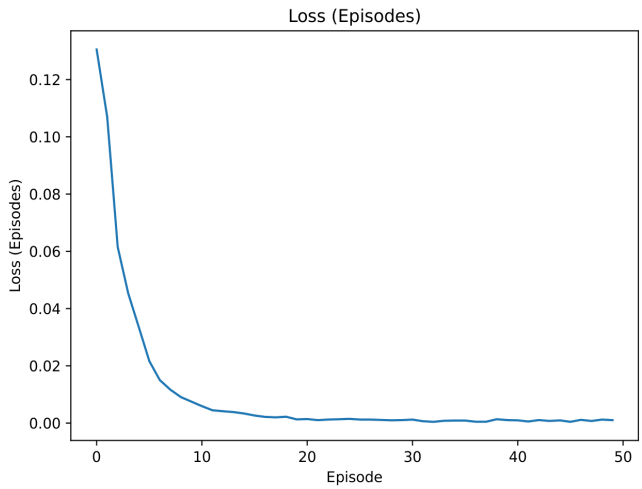

Figure 4. Average training loss per episode over 50 episodes

Figure 4 reports the average training loss aggregated per episode over 50 episodes. The loss decreases sharply from 0.1305 in the first episode to 0.0336 by episode 5, and falls below 0.01 after episode 9. Convergence is observed from approximately episode 15 onward, where the average loss stabilizes below 0.003 and remains within the range $[4.4 \times 10^{-4}, 1.35 \times 10^{-3}]$ for the remainder of training.

The absence of any increasing loss trend or divergence in later episodes indicates stable convergence of the DDQN and suggests that the learned value function does not suffer from overfitting. This behavior confirms the effectiveness of the target network, experience replay, and DDQN update mechanism in maintaining training stability.

6.4 Reward analysis

Figure 5 presents the cumulative reward obtained per episode over 50 training episodes. The reward curve shows a monotonic upward trend after the initial exploration phase, indicating progressive improvement in the agent's ability to predict node behavior consistent with the final consensus decision.

The cumulative reward increases from -178 in the first episode to 718 by episode 2, reflecting rapid policy adaptation once sufficient experiences are collected. From episode 5 onward, the reward exceeds 2000 and continues to grow steadily, reaching 6124 at episode 15 and 9316 by episode 30. In the final phase of training, the reward stabilizes with smaller incremental gains, culminating at 11032 in episode 50.

The absence of reward collapse or oscillatory behavior in later episodes indicates stable policy convergence. This sustained reward growth, combined with the stabilized loss values reported earlier, confirms that the DDQN successfully learns a consistent decision policy without overfitting or performance degradation.

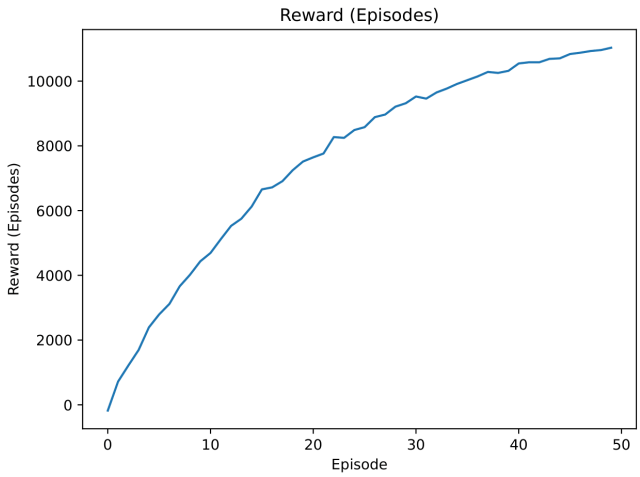

Figure 5. Cumulative reward obtained per episode over 50 training episodes

6.5 Q-value stability

Figure 6 illustrates the evolution of the average estimated Q-values at each training step. During the early learning phase, Q-values increase rapidly as the agent explores the state–action space and refines its value estimates. This transient growth reflects the correction of initially uninformative value predictions.

Figure 7 presents the average Q-values aggregated per

episode over 50 episodes. The average Q-value increases from 3.66 in episode 1 to 10.24 in episode 2 and exceeds 18.0 by episode 5. After episode 10, the Q-values converge toward a stable range centered around 20.0, with minor bounded oscillations.

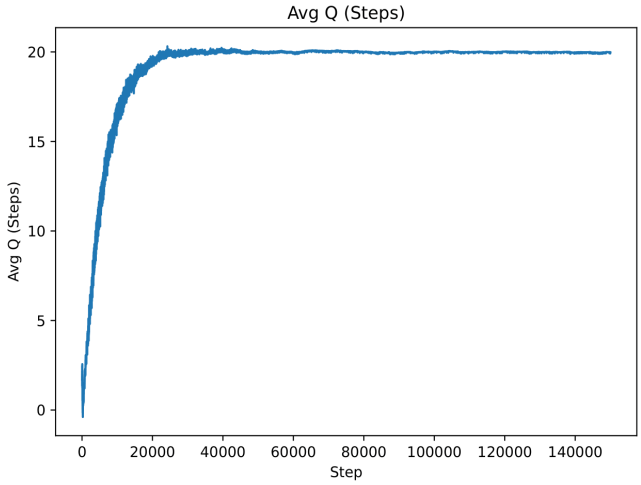

Figure 6. Average estimated Q-values at each training step

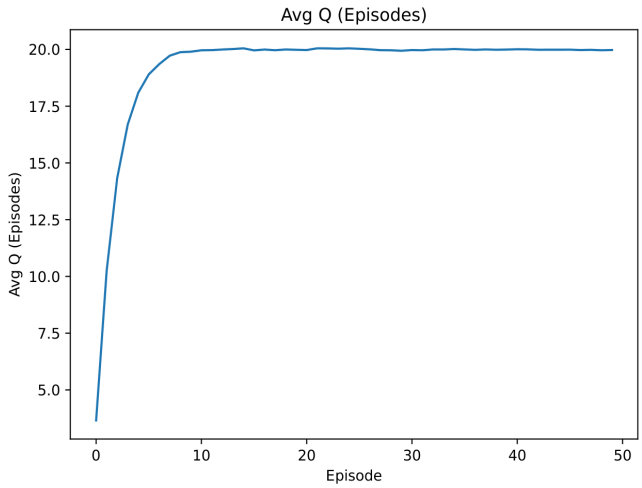

Figure 7. Average Q-values per episode over 50 training episodes

From episodes 10 to 50, the average Q-values remain tightly bounded within [19.94,20.05], indicating stable value estimation and the absence of divergence or overestimation. This stabilization confirms the effectiveness of the DDQN formulation, where decoupling action selection from value evaluation mitigates the overestimation bias commonly observed in standard DQN.

The convergence of Q-values, combined with the stabilized loss and increasing reward trends reported earlier, demonstrates that the learned policy reaches a consistent and reliable value function approximation suitable for trust evaluation in permissioned blockchain environments.

6.6 Trust score evolution and node rating analysis

This analysis examines the evolution of node trust scores at the end of training and explains how the proposed framework distinguishes reliable, risky, and potentially malicious nodes based on observed behavior.

Table 2 reports the final trust scores of the 20 evaluated nodes. The trust values are computed using the weighted trust formulation described previously, with the following parameters used throughout all experiments:

$$\alpha = 1.0, \beta = 1.2, \delta = 0.8, \gamma = 0.5$$

The parameter $\alpha = 1$ serves as the baseline weight for correct validation behavior. A slightly larger value $\beta = 1.2$ is used to penalize faulty decisions more strongly due to their higher impact on consensus reliability. The proposer contribution is weighted with $\delta = 0.8$ since block proposal occurs less frequently than validation actions. Finally, $\gamma = 0.5$ moderates the effect of accumulated penalties to avoid excessive trust degradation from occasional faults.

Table 2. Final trust scores of evaluated nodes

Node	Trust	Node	Trust
n_0	70.44	n_{10}	76.30
n_1	74.52	n_{11}	62.44
n_2	78.95	n_{12}	68.29
n_3	74.20	n_{13}	73.06
n_4	74.65	n_{14}	66.88
n_5	74.35	n_{15}	67.20
n_6	74.01	n_{16}^*	62.23
n_7	76.45	n_{17}^*	60.95
n_8	68.29	n_{18}^*	57.59
n_9	79.85	n_{19}	76.02

Note: * Byzantine nodes deliberately injected during simulation.

The obtained trust scores exhibit a clear stratification of node behavior. Nodes such as n_9, n_2, n_7, n_{10}, and n_{19} achieve trust values above 76, indicating consistently correct participation, low fault rates, and positive proposer contributions. These nodes can be classified as highly reliable validators and are suitable candidates for critical consensus roles.

A second group of nodes, including n_0, n_1, n_3, n_4, n_5, and n_{13}, obtain trust scores in the range 70–75. These nodes demonstrate generally correct behavior but exhibit occasional faults or penalty accumulation. They are considered moderately reliable and may require monitoring under stricter consensus conditions.

At the lower end of the trust spectrum, the three byzantine nodes n_{16}, n_{17}, and n_{18}, deliberately injected with adversarial behavior during the simulation, receive the lowest trust scores of 62.23, 60.95, and 57.59 respectively, resulting from high graded fault rates, repeated penalties, and limited positive proposer impact. The framework detects and penalizes their systematic adversarial voting through the fault contribution F_i and penalty rate components of final trust score function, without requiring any prior knowledge of node identity or attack strategy. Node n_{11} (with score 62.44) exhibits intermittent faulty behavior consistent with a selfish or transiently faulty node, demonstrating the framework's ability to differentiate between deliberate and non-deliberate misbehavior. Such nodes should be excluded or down-weighted in the consensus process.

Overall, the results demonstrate that the proposed DDQN–based framework effectively captures long-term node behavior and provides an adaptive, interpretable trust assessment for robust consensus management in permissioned blockchain systems.

6.7 Baseline comparison

To justify the complexity of the proposed trust formulation, we compare it against a simple reputation baseline that computes node trust as the plain fraction of correct validation decisions, without confidence weighting, proposer accountability, or penalty rate:

$$\text{SimpleReputation}_i = \frac{\text{correct decisions}_i}{\text{total participations}_i} \times 100$$

Table 3 reports the trust scores produced by both methods on the same dataset. The simple reputation baseline assigns nearly identical scores to all nodes, with a total range of only 2.83 points and a standard deviation of 0.80. Critically, it assigns an average score of 88.05 to the three byzantine nodes n_{16}, n_{17}, and n_{18}, compared to 89.30 for the top honest nodes, resulting in a gap of only 1.25 points that renders byzantine detection effectively impossible.

In contrast, the proposed DDQN trust framework produces a range of 22.26 points and a standard deviation of 6.33, representing a 7.9× improvement in score spread. The average trust score of byzantine nodes drops to 60.26, compared to 78.42 for top honest nodes, yielding a separation gap of 18.16 points, which represents a 14.5× improvement over the baseline. This result confirms that the multi-component trust formulation, which integrates fault contribution, proposer accountability, and penalty rate alongside correct-rate weighting, is essential for meaningful node differentiation and adversarial detection in permissioned blockchain environments.

Table 3. Baseline comparison: Simple reputation vs. proposed Double Deep Q-Network (DDQN) trust

Node	Simple Reputation	DDQN Trust
n_0	87.97	70.44
n_1	88.87	74.52
n_2	89.27	78.95
n_3	89.50	74.20
n_4	88.33	74.65
n_5	89.67	74.35
n_6	88.97	74.01
n_7	88.80	76.45
n_8	87.80	68.29
n_9	89.83	79.85
n_{10}	88.93	76.30
n_{11}	87.93	62.44
n_{12}	89.17	68.29
n_{13}	87.50	73.06
n_{14}	87.83	66.88
n_{15}	89.10	67.20
n_{16}^*	89.23	62.23
n_{17}^*	87.93	60.95
n_{18}^*	87.00	57.59
n_{19}	87.90	76.02
Range	**2.83**	**22.26**
Standard deviation	**0.80**	**6.33**

Note: * Byzantine nodes deliberately injected during simulation.

6.8 Ablation study on trust coefficients

To assess the sensitivity of the trust formulation to coefficient selection, four configurations of $(\alpha,\ \beta,\ \delta,\ \gamma)$ are evaluated on the same dataset. The value of α is always fixed at 1.0 to serve as a normalization reference, allowing β, δ, and γ to be interpreted as relative weights with respect to the baseline contribution of correct validation behavior. Beyond the default setting (1.0, 1.2, 0.8, 0.5), three variants are considered: a higher fault penalty ($\beta = 1.5$) to reflect stricter consensus requirements, a reduced proposer weight ($\delta = 0.4$) to limit the influence of block proposal frequency, and a higher penalty rate ($\gamma = 1.0$) to enforce stronger punishment for accumulated faults.

Table 4 reports the resulting trust scores across all configurations. Despite variations in coefficient values, the relative ranking of nodes remains highly consistent: n_9 and n_2 consistently occupy the top two positions across all settings, while the three byzantine nodes n_{16}, n_{17}, and n_{18} remain at the bottom in all configurations. The maximum rank displacement observed across all nodes is 4 positions, with a mean displacement of 1.35, confirming that the trust ordering is stable and not an artifact of a specific parameter choice. These results validate that the proposed trust formulation captures genuine behavioral differences rather than coefficient-dependent artifacts, and that the default configuration represents a balanced and robust parameterization.

Table 4. Ablation study: Trust scores under different coefficient configurations

Node	Default (1.0, 1.2, 0.8, 0.5)	High β (1.0, 1.5, 0.8, 0.5)	Low δ (1.0, 1.2, 0.4, 0.5)	High γ (1.0, 1.2, 0.8, 1.0)
n_0	70.44	69.14	67.34	51.92
n_1	74.52	73.49	73.57	62.40
n_2	78.95	78.04	76.13	66.63
n_3	74.20	73.36	71.12	56.11
n_4	74.65	73.46	72.25	60.92
n_5	74.35	73.56	74.35	62.19
n_6	74.01	73.01	73.20	61.42
n_7	76.45	75.40	74.79	64.97
n_8	68.29	66.94	66.36	50.32
n_9	79.85	79.11	76.84	66.80
n_{10}	76.30	75.29	73.96	63.02
n_{11}	62.44	61.13	62.23	41.76
n_{12}	68.29	67.35	66.96	48.51
n_{13}	73.06	71.62	70.75	59.77
n_{14}	66.88	65.54	66.07	49.67
n_{15}	67.20	66.24	66.39	47.52
n_{16}^*	62.23	61.31	61.71	37.86
n_{17}^*	60.95	59.64	58.94	35.18
n_{18}^*	57.59	56.00	57.59	34.53
n_{19}	76.02	74.70	73.34	64.07

Note: * Byzantine nodes deliberately injected during simulation.

6.9 Discussion

The stable convergence observed from episode 15 onward, with a final cumulative reward of 11,032, confirms that RL-based trust learning is feasible without pre-labeled data. This is a fundamental advantage over static reputation models such as EigenTrust [13] and PeerTrust [14], which cannot update their trust estimates in response to behavioral shifts observed across consensus rounds.

The 22.26-point spread across 20 nodes produce three operationally distinct tiers that directly inform governance decisions: nodes scoring above 76 are suitable for critical proposer roles, those in the range 70–75 require monitoring, and those below 63 should be excluded or down-weighted. This degree of differentiation is not achievable by threshold-

based approaches such as RCF [32] or GT-BFT [34], which classify nodes into binary or coarse-grained trust categories without learning-based refinement.

The clear separation between the top and bottom tiers is driven in part by the proposer responsibility component (Penalty rate), which distinguishes nodes that lead successful consensus rounds from those that repeatedly propose invalid blocks. This dimension is absent in VeriBlock [23], which records trust evidence on-chain but treats proposer behavior as a binary event, and in RL-based optimization approaches [27, 28], which focus on system-level performance without modeling proposer accountability explicitly.

Compared to adaptive RL trust models in IoT settings [18], the proposed framework produces fully explicit and interpretable trust scores derived from observable consensus outcomes rather than embedding trust implicitly within reward functions, making the scores directly usable for validator governance without requiring access to the agent's internal state.

In contrast, the proposed framework simultaneously satisfies key trust and learning requirements. It is fully adaptive and RL-based, explicitly integrates consensus outcomes into trust evaluation, supports graded trust computation, and directly incorporates proposer accountability. Trust scores are computed exclusively from observable blockchain events, ensuring interpretability while remaining responsive to dynamic and adversarial behavior. This combination enables a more comprehensive and reliable assessment of node credibility compared to existing approaches, as confirmed by the experimental results.

7. CONCLUSIONS

This paper presented a trust-aware RL framework for permissioned blockchain systems, aimed at overcoming the limitations of static reputation models and performance-oriented learning approaches. By integrating a DDQN with an explicit and consensus-aware trust evaluation mechanism, the proposed framework enables adaptive, interpretable, and data-driven assessment of validator behavior. Trust scores are derived from graded decision correctness, fault contribution, proposer responsibility, and penalty feedback, ensuring that trust reflects actual participation in the consensus process rather than abstract performance indicators. Operationally, nodes with trust scores below 65 should be excluded from proposer selection or subjected to increased validation scrutiny, while nodes scoring above 76 are suitable candidates for critical consensus roles requiring high reliability. Moreover, privacy is reinforced by assessing trust exclusively through consensus outcomes, avoiding direct exposure of transaction-level data.

Experimental evaluation in a PBFT-based permissioned blockchain environment demonstrated stable learning dynamics, effective exploration control, and clear differentiation of validator trust levels under dynamic conditions. The results indicate that jointly combining RL with explicit trust modeling improves robustness against faulty and selfish behavior while preserving transparency and interpretability.

Several promising research directions emerge from this work. First, the framework can be extended toward multi-agent reinforcement learning (MARL), enabling decentralized or cooperative trust learning among validators and more accurately capturing the strategic interactions inherent in permissioned blockchain environments. Second, future studies may explore fully online and continual trust learning, allowing the model to adapt continuously to evolving node behavior without relying on offline trace collection. Third, although the trust learning and scoring mechanisms rely on generic consensus signals such as validation correctness, proposer behavior, and penalty feedback, the experimental validation in this work is restricted to a PBFT-based setting. Extending the framework to other permissioned consensus protocols, such as Proof of Stake and Delegated Proof of Stake, and empirically validating its generality across these protocols, constitutes an important direction for future work. Finally, the robustness of the proposed approach can be further evaluated under advanced adversarial and strategic threat models, such as collusion, coordinated attacks, and long-term deceptive strategies. Investigating these scenarios will contribute to the design of more resilient, secure, and dependable trust-aware consensus systems.

REFERENCES

[1] Nakamoto, S. (2008). Bitcoin: A Peer-To-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

[2] Casino, F., Dasaklis, T.K., Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telematics and Informatics, 36: 55-81. https://doi.org/10.1016/j.tele.2018.11.006

[3] Zheng, Z.B., Xie, S.A., Dai, H.N., Chen, X.P., Wang, H.M. (2018). Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 14(4): 352-375. https://doi.org/10.1504/IJWGS.2018.095647

[4] Garay, J., Kiayias, A., Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. In Advances in Cryptology - EUROCRYPT 2015, Springer, Berlin, Heidelberg, pp 281-310. https://doi.org/10.1007/978-3-662-46803-6_10

[5] Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S. (2016). On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3-16. https://doi.org/10.1145/2976749.2978341

[6] Bellini, E., Iraqi, Y., Damiani, E. (2020). Blockchain-based distributed trust and reputation management systems: A survey. IEEE Access, 8: 21127-21151. https://doi.org/10.1109/ACCESS.2020.2969820

[7] Ge, L., Wang, J., Zhang, G.F. (2022). Survey of consensus algorithms for proof of stake in blockchain. Security and Communication Networks, 2022(1): 2812526. https://doi.org/10.1155/2022/2812526

[8] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, pp. 1-15. https://doi.org/10.1145/3190508.3190538

[9] Dorri, A., Kanhere, S.S., Jurdak, R., Gauravaram, P. (2017). Blockchain for IoT security and privacy: The case study of a smart home. In 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops),

Kona, HI, USA, pp. 618-623. https://doi.org/10.1109/PERCOMW.2017.7917634
[10] Azaria, A., Ekblaw, A., Vieira, T., Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. In 2016 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, pp. 25-30. https://doi.org/10.1109/OBD.2016.11
[11] Castro, M., Liskov, B. (1999). Practical byzantine fault tolerance. In the Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, pp. 1-14. https://css.csail.mit.edu/6.824/2014/papers/castro-practicalbft.pdf.
[12] Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I. (2019). HotStuff: BFT consensus with linearity and responsiveness. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, Toronto, ON, Canada, pp. 347-356. https://doi.org/10.1145/3293611.3331591
[13] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H. (2003). The Eigentrust algorithm for reputation management in p2p networks. In Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, pp. 640-651. https://doi.org/10.1145/775152.775242
[14] Xiong, L., Liu, L. (2004). PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering, 16(7): 843-857. https://doi.org/10.1109/TKDE.2004.1318566
[15] Yuan, Y., Wang, F.Y. (2018). Blockchain and cryptocurrencies: Model, techniques, and applications. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48(9): 1421-1428. https://doi.org/10.1109/TSMC.2018.2854904
[16] Alharby, M., Aldweesh, A., van Moorsel, A. (2018). Blockchain-based smart contracts: A systematic mapping study of academic research (2018). In 2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB), Fuzhou, China, pp. 1-6. https://doi.org/10.1109/ICCBB.2018.8756390
[17] Li, B.Y., Chenli, C.H., Xu, X.W., Shi, Y.Y., Jung, T. (2019). DLBC: A deep learning-based consensus in blockchains for deep learning services. arXiv preprint arXiv:1904.07349. https://doi.org/10.48550/arXiv.1904.07349
[18] Padia, S., Vaidya, D., Mangrulkar, R. (2025). Adaptive trust consensus for blockchain IoT: Comparing RL, DRL, and MARL against naive, collusive, adaptive, byzantine, and sleeper attacks. arXiv preprint arXiv:2512.22860. https://doi.org/10.48550/arXiv.2512.22860
[19] Jameel, F., Javaid, U., Khan, W.U., Aman, M.N., Pervaiz, H., Jäntti, R. (2020). Reinforcement learning in blockchain-enabled IIoT networks: A survey of recent advances and open challenges. Sustainability, 12(12): 5161. https://doi.org/10.3390/su12125161
[20] Gutierrez, R., Villegas-Ch, W., Govea, J. (2025). Adaptive consensus optimization in blockchain using reinforcement learning and validation in adversarial environments. Frontiers in Artificial Intelligence, 8: 1672273. https://doi.org/10.3389/frai.2025.1672273
[21] Jøsang, A., Ismail, R., Boyd, C. (2007). A survey of trust and reputation systems for online service provision. Decision Support Systems, 43(2): 618-644. https://doi.org/10.1016/j.dss.2005.05.019
[22] Zhuang, Q.W., Liu, Y., Chen, L.S., Ai, Z.P. (2019). Proof of reputation: A reputation-based consensus protocol for blockchain based systems. In Proceedings of the 1st International Electronics Communication Conference, Okinawa, Japan, pp. 131-138. https://doi.org/10.1145/3343147.3343169
[23] Pal, S., Hill, A., Rabehaja, T., Hitchens, M. (2022). VeriBlock: A blockchain-based verifiable trust management architecture with provable interactions. In 2022 International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, pp. 1-7. https://doi.org/10.1109/ICCCN54977.2022.9868875
[24] Hasan, O., Brunie, L., Bertino, E. (2022). Privacy-preserving reputation systems based on blockchain and other cryptographic building blocks: A survey. ACM Computing Surveys (CSUR), 55(2): 1-37. https://doi.org/10.1145/3490236
[25] Din, I.U., Khan, K.H., Almogren, A., Zareei, M., Diaz, J.A.P. (2024). Securing the metaverse: A blockchain-enabled zero-trust architecture for virtual environments. IEEE Access, 12: 92337-92347. https://doi.org/10.1109/ACCESS.2024.3423400
[26] Wiering, M.A., Van Otterlo, M. (2012). Reinforcement Learning. Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27645-3
[27] Goh, Y., Yun, J., Jung, D., Chung, J.M. (2022). Secure trust-based delegated consensus for blockchain frameworks using deep reinforcement learning. IEEE Access, 10: 118498-118511. https://doi.org/10.1109/ACCESS.2022.3220852
[28] Suganya, R., Farina, K., Shahbad, A.A., Kumar, N.L., Biradar, M.S., Pawale, A.N. (2024). Reinforcement learning-based deep FEFM for blockchain consensus mechanism optimization with non-linear analysis. Journal of Computational Analysis and Applications (JoCAAA), 33(5): 118-130. https://eudoxuspress.com/index.php/pub/article/view/457.
[29] Zhang, H.Y., Li, S.K., Bao, H., Cao, Y.P., Li, J.B. (2026). A trust-aware and cost-optimized blockchain oracle selection model with deep reinforcement learning. Future Generation Computer Systems, 182: 108489. https://doi.org/10.1016/j.future.2026.108489
[30] Sheetal, A.P., Khalaf, M.I., Zaidi, A., Dutta, A.K., Alam, M.S., Yogi, K.S., Pathak, N., Abdullayev, A., Krishna, V.B.M. (2025). Blockchain-based anomaly detection in vehicular ad-hoc networks using deep reinforcement learning. Transactions on Emerging Telecommunications Technologies, 36(10): e70282. https://doi.org/10.1002/ett.70282
[31] Islam, T., Bappy, F.H., Zaman, T.S., Sajid, M.S.I., Pritom, M.M.A. (2024). MRL-PoS: A multi-agent reinforcement learning based proof of stake consensus algorithm for blockchain. In 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, pp. 0409-0413. https://doi.org/10.1109/CCWC60891.2024.10427777
[32] Mohsenzadeh, A., Bidgoly, A.J., Farjami, Y. (2022). A novel reputation-based consensus framework (RCF) in

distributed ledger technology. Computer Communications, 190: 126-144. https://doi.org/10.1016/j.comcom.2022.04.015

[33] Shalini, R., Manoharan, R. (2022). Trust model for effective consensus in blockchain. EAI Endorsed Transactions on Scalable Information Systems, 9(5): 1-8. https://pdfs.semanticscholar.org/dfdd/75b9939c7e4e62be10e60704ce44512d8a5e.pdf.

[34] Xi, J.W., Xu, G.S., Zou, S.H., Yue, Y.L., Cai, B.S., Li, G.Q. (2026). A global trust-based blockchain lightweight consensus mechanism. Blockchain: Research and Applications, 7(1): 100322. https://doi.org/10.1016/j.bcra.2025.100322

[35] Bellifemine, F., Bergenti, F., Caire, G., Poggi, A. (2005). Jade — A java agent development framework. In Multi-Agent Programming. Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer, Boston, MA, pp. 125-147. https://doi.org/10.1007/0-387-26350-0_5