Vol. 08, No. 3 (2026) 1769-1786, doi: 10.24874/PES08.03.028

Proceedings on Engineering Sciences

www.pesjournal.net

A HYBRID DQN–BLOCKCHAIN–PBFT FRAMEWORK FOR SECURE DATA TRANSMISSION AND REAL-TIME OPTIMIZATION IN SMART INDUSTRIAL NETWORKS

Dinesh Rajassekharan
P. Vidyasri
Vijaya Krishna Sonthi
Firas Tayseer Ayasrah
S. Arun Jayakar
M. Ramya
S. Maheswari
Sudhakar Sengan[1]

Received 20.05.2026.
Revised 02.07.2026.
Accepted 31.07.2026.

Keywords:
Blockchain, Deep Q-learning, Security, resource utilization, Smart contracts, Internet of Things, Adaptive optimization

A B S T R A C T

This research proposes a method that combines DQN + BCT + PBFT Model (Deep Q-Network + Blockchain Technology + Practical Byzantine Fault Tolerance) to optimise real-time smart industrial networks and enable secure data transmission. The model uses smart contracts and a distributed ledger (Blockchain Technology, BCT) for security. DQN supports adaptive decision-making and resource allocation. The model also includes a modified consensus mechanism for industrial applications. This model was compared with other methods, including Zero-Trust + LSTM, Hyperledger + Q-Learning, Ethereum + Neural Network, and Internet of Things (IoT). At a 256 KB block size, the proposed hybrid DQN + BCT + PBFT achieved a Transaction Success Rate (TSR) of 99.23% and a transaction processing speed of 87.43 TPS. Under a transaction load of 100 TPS, the proposed model achieved a Transaction-Load Response Time (TLRT) of 197.34 ms, demonstrating lower response time than several blockchain-based baseline models. Under a workload of 2000 transactions/min, CPU utilization reached 78.54% and memory consumption reached 4234.89 MB. NT analysis reported an average rate of 782.43 transactions per 5 minutes and a 13.4% performance loss during extended operations.

© 2026 Published by Faculty of Engineering

1. INTRODUCTION

As industrial networks grow and Industry 4.0 adoption increases, data security and network optimization have become major challenges (Karn et al., 2025). These networks are data-driven and interconnected, so they require secure, real-time data transmission (Karthikeyan et al., 2024). Industrial Internet of Things (IIoT) networks generate massive volumes of data, making secure transmission and real-time analysis challenging

Corresponding author: Sudhakar Sengan
Email: sudhasengan@gamil.com; sudhakars@esec.ac.in

(Jyothi et al., 2025). Security and optimization methods struggle to keep pace with ever-changing operational features (Admass et al., 2024). Internet-based attacks are steadily increasing, and centralized security mechanisms are becoming increasingly vulnerable (Khan et al., 2023; Khalid et al., 2026; Natraj et al., 2025; Baptist et al., 2026; Chaudhari et al., 2025). In particular, the manufacturing sector must implement real-time decision-making and resource allocation (RA) (Khan et al., 2020). However, deployment techniques cannot achieve optimal performance while simultaneously meeting security requirements.

The demand for scalable, adaptable methods to ensure performance and security has increased, in tandem with the growing complexity and scale of network systems (Baptist et al., 2025). This refers to the challenge of researching the correlation between performance and security in industrial networks, as current technologies place greater emphasis on usability than on cybersecurity. The primary objective of the present study is to provide a method that integrates blockchain technology (BCT) with machine learning (ML) using deep Q-learning (DQN). To enable real-time encrypted data transmission, the model combines BCT's security features with DQN's adaptive optimization approach (AOA). An improved consensus mechanism is needed to provide secure, fault-tolerant transaction validation in permissioned industrial blockchain environments (Cheekati et al., 2025).

Current research recommends integrating DQN + BCT security features to enhance resource control and security. An enhanced security protocol that incorporates an improved CM was made available for implementation in scenarios involving large companies (Figueroa et al., 2024; Mann et al., 2025). The model uses DQN to adjust the network's parameters in real time in response to changing scenarios and evolving security demands. According to this study's results, RT, NT, and RU increased significantly under variable-load conditions (Adewuyi et al., 2024).

This research proposes a hybrid DQN + BCT + PBFT (Deep Q-Network + Blockchain Technology + Practical Byzantine Fault Tolerance) that could be widely used for secure data transmission and real-time optimization in an SIN. The proposed model combines blockchain (BC)-based decentralized security with Reinforcement learning (RL)-based adaptive resource management to increase operational efficiency and transaction integrity, while reducing latency in industrial communication. RL enables an agent to learn a policy through interaction with an environment to maximize cumulative reward (Sutton & Barto, 2018). The work proposes a practical Byzantine fault tolerance (PBFT) (Castro & Liskov, 1999) based consensus mechanism to reduce transaction validation delays and improve fault tolerance under dynamic industrial workloads. The PBFT-based consensus mechanism reduces transaction validation delay and improves fault tolerance under dynamic industrial workloads. This work tested the model with Hyperledger Fabric, Kubernetes orchestration, MQTT-based IIoT communication, and edge-computing deployment configurations to improve its engineering applicability. Also, we extensively validated the proposed model using throughput analysis, response-time evaluation, transaction-verification assessment, computational-overhead analysis, ablation studies, and statistical validation under realistic industrial operating conditions. PPO employs a clipped surrogate objective to constrain excessive policy updates and improve training stability (Schulman et al., 2017)

The rest of the paper is organized as follows. Section 2 presents the elements and the proposed hybrid DQN + BCT + PBFT. Section 3 covers implementation features and deployment factors. Experimental results and performance analysis are presented in Section 4. Section 5 summarizes the work and outlines future research directions.

2. METHODOLOGY

2.1 Proposed Hybrid DQN + BCT + PBFT

The proposed hybrid DQN + BCT + PBFT for secure data transmission and real-time optimization in smart industrial networks (SIN) comprises 4 layers (Figure 1): IIoT, ML, BCT, and Integration.

- IIoT layer collects and preprocesses data through sensors, actuators, and industrial machines. The data gateway aggregates data streams and routes them to edge processors for initial processing (Huo et al., 2022).
- The IIoT delivers filtered, structured data to the ML layer, which generates insights and enables real-time optimization. Model Trainer: The Model Trainer uses the features generated by the Feature Extractor to perform model prediction and anomaly detection. The prediction interface provides predictions that feed into the Optimization Pipeline.
- The BCT layer uses a distributed ledger to ensure tamper-proofing and data integrity. It enables secure data storage and access control. Once the conditions are met, the Smart Contract Manager (SCM) triggers automated protocols. An access control system controls the permissions. The transaction manager handles data transactions. The transaction manager performs data transactions.
- The integration layer controls lower-level components and simplifies data sharing and transfer. The validation service checks data accuracy, and the Orchestrator controls data flow and allocates tasks. Data Synchroniser provides a distributed ledger, ML prediction, and integrity for raw IoT data. The security controller secures networks from attacks.

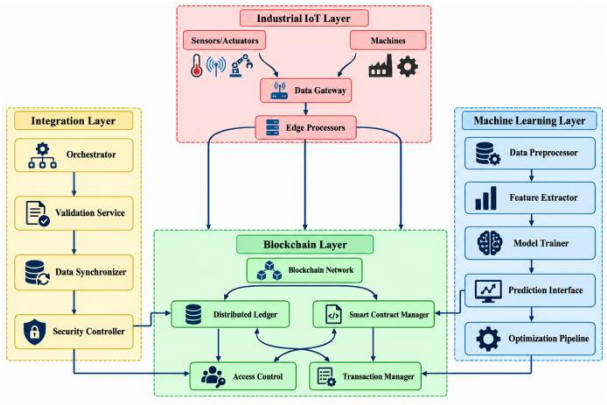


Figure 1. Proposed hybrid DQN + BCT + PBFT

2.2 BCT

To manage data in SIN, the BCT uses a decentralized, secure method. Smart Contracts (SC) automate validation, autonomous control, secure storage, and Internet of Things (IoT) data collection (Figure 2). API Gateway collects data from IoT devices, routes it, and also adds metadata for identification. BCT uses an encrypted storage system accessible only to authorised users with decryption keys. Data from IoT devices is denoted as D_{iot} and is collected and routed through the API Gateway (Lakhan et al., 2024). The gateway applies methods for each data entry D_i and attaches metadata such as timestamps and unique detections. Processed data is stored in encrypted form within BCT's decentralized storage system, with encryption as $E(D_i)$. Authorized users with suitable decryption keys access the original data using the inverse encryption function as E^{-1}(Lakhan et al., 2025).

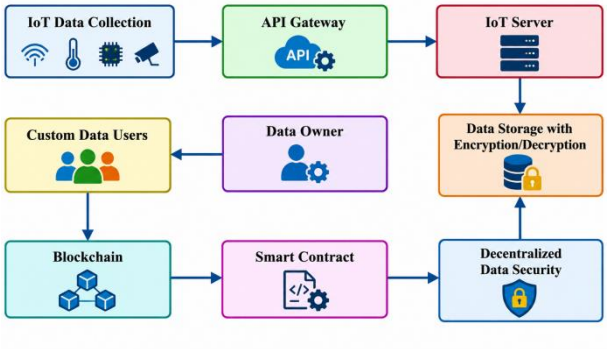


Figure 2. BCT

SC is a component of the proposed hybrid DQN + BCT + PBFT that controls data access policies and transaction parameters. Each SC is SC_j encodes states that define data usage based on roles and permissions within the network. When user U_i requests access to data, the SC checks if U_i meets the required access criteria P_k. Access is approved only if the user's credentials meet predefined policies. BCT uses a CM to validate each transaction before recording it in the ledger. Every transaction T_{data} experiences approval by a network of nodes, as $N = \{n_1, n_2, \ldots, n_m\}$.

The hybrid DQN + BCT + PBFT preserves data integrity through an immutable ledger network in which each block in the chain is cryptographically linked to its predecessor by storing the previous block's hash (Figure 3). The previous hash field of Block #1 stores the hash of Block 0, which is the Genesis Block, and Block #2 stores the hash value of Block #1. This link is maintained throughout the chain, and any change to one block would impact all subsequent blocks (Qin et al., 2020). With this model, all links in the data chain are private and remain constant (Pasdar et al., 2025).

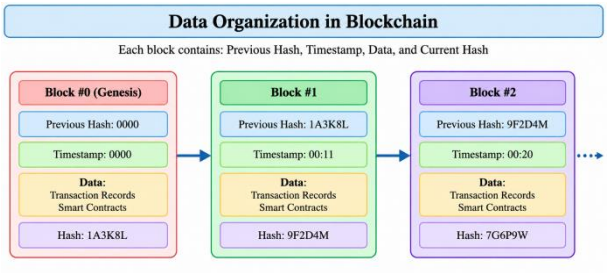


Figure 3. Data organization in BCT

2.2.1 Data Encryption and Access Control

Data encryption and access control are two of the most significant security features in the BCT. Access control mechanisms manage network data permissions, while AES encryption provides security. AES is explicitly linked to NIST (2023) and FIPS 197.

i) **Data Encryption with AES:** Data D generated by IoT devices enters the BCT and is encrypted using the Advanced Encryption Standard (AES) algorithm. AES is a symmetric encryption algorithm, so the same key K_{enc} is used for encryption and decryption.

This encryption process is signified as follows: Equation (1)

$$E(D) = \text{AES}(D, K_{enc}) \quad (1)$$

where $E(D)$ is the encrypted form of the data.

The decryption method, which reverts the encrypted data $E(D)$ to its original form D, is defined as Equation (2)

$$D = \text{AES}^{-1}(E(D), K_{enc}) \quad (2)$$

where AES^{-1} is the AES decryption function.

ii) **Access Control:** When a user U_i requests access to encrypted data $E(D)$ on the BCT, the system verifies ID_{U_i} against an access policy. This verification method is managed by an SC, denoted as SC, which acts as an automated gateway executing access control rules embedded within the policy.

The access policy AP is defined as Equation (3)

$$AP\left(ID_{U_i}, P_{U_i}\right) \rightarrow \text{Access Granted or Denied} \quad (3)$$

If user identifier ID_{U_i} and permissions P_{U_i} align with conditions in AP then SC grants access and U_i can decrypt and retrieve $'D'$. Otherwise, access is restricted. SC enforces access control policies within BCT through self-executing scripts. Each user-initiated access request triggers SC to check credentials and permissions against the encoded policy. When a custom data user U_c with permissions P_{U_c} initiates a request, the SC tests whether P_{U_c} meets conditions specified in AP. If conditions are satisfied, the access request proceeds and U_c can interact with data. The proposed decentralised CM in BCT is designed for secure, low-latency data validation in SIN. It verifies and records data from IoT devices.

In this model, each data transaction T_i is generated by an IoT device and enters the BCT network through a validation method that involves multiple nodes, as $N = \{n_1, n_2, \ldots, n_k\}$. Each transaction T_i contains data points such as sensor readings and requires validation before being included in BCT. When T_i is introduced at each node n_j in $'N'$ verifies the transaction independently based on preset rules encoded in the consensus protocol.

The result of this validation at each node is Equation (4)

$$C_{n_j}(T_i) = \begin{cases} 1 & \text{if } T_i \text{ is valid} \\ 0 & \text{if } T_i \text{ is invalid} \end{cases} \quad (4)$$

iii) CM: Practical Byzantine fault tolerance (PBFT)

The proposed hybrid DQN + BCT + PBFT uses PBFT as a CM for permissioned networks with known users. In PBFT, the network tolerates up to $'f'$ faulty or malicious nodes, where $f < k/3$. Consensus is reached when most nodes behave honestly.

The PBFT process in this model operates as follows:

- **Request Phase:** The IoT data initiates a transaction T_i by sending a request to the primary (leader) node n_p.
- **Pre-prepare and Prepare Phases:** The primary node broadcasts a pre-prepared message containing T_i to all secondary nodes. Each secondary node n_j then broadcasts a prepared message signifying agreement or disagreement on T_i 's validity.
- **Commit Phase:** Each node will send a commit message once it has received a majority of the messages that have been prepared to support it T_i. Validation T_i is considered to have occurred for a node when it has received enough commit messages.
- **Reply Phase:** The new data transfer is then $'T_i'$ is added to the existing block after it has been validated, and all nodes update their ledgers to include the new data transfer. This ensures network reliability.

PBFT reduces transaction validation time t_{pbft} for real-time data validation in SIN. Low-latency consensus supports decision-making in manufacturing processes. The CM tolerates up to $f < k/3$ faulty nodes and handles malicious nodes or hardware failures. Manufacturing processes require fault tolerance. PBFT protocol filters and eliminates unreliable or false results from nodes.

2.2.2 Data Integrity and Authentication

Cryptographic hashing and the BCT's immutable network maintain data integrity. When a new block B_i is created, it includes a cryptographic hash of the previous block, denoted as $H(B_{i-1})$. This hash value is generated using a secure hash function, such as SHA-256, generating a unique, fixed-length output from the block's contents.

The current block's hash $H(B_i)$ is calculated as:

$$H(B_i) = \text{hash}(B_{i-1} \| T_{i1} \| T_{i2} \| \cdots \| T_{in}) \quad (5)$$

where $T_{i1}, T_{i2}, \ldots, T_{in}$ are the transactions contained within the block B_i, $\|$ denotes concatenation.

Each transaction T_i added to the BCT originates from an authenticated source, ensuring that only authorised devices or users add data. This authentication is enforced using digital signatures, where each participant in the network has a unique public-private key pair (K_{pub}, K_{priv}). When a user $U/$ loT device submits a transaction, it signs the transaction data $'D'$ with its private key $K_{\text{priv,}}$ producing a digital signature $'\sigma'$ as follows:

$$\sigma = \text{sign}(D, K_{\text{priv}}) \quad (6)$$

This signature $'\sigma'$ is then attached to the transaction $'T_i'$ before it enters the BCT. Upon receiving the transaction, other nodes in the network verify the signature using the public key $'K_{pub}'$ associated with the sender, ensuring that $'T_i'$ was indeed generated by an authenticated entity and remains untampered.

The verification method is as follows:

$$\text{verify}(D, \sigma, K_{pub}) = \begin{cases} \text{True} & \text{If } \sigma \text{ Is Valid} \\ \text{FALSE} & \text{If } \sigma \text{ Is Invalid} \end{cases} \quad (7)$$

If verification fails, the data exchange becomes invalid, preventing unauthorized data from reaching the BCT. If validation is successful, the transfer is deemed valid and proceeds through the CM.

2.2.3 Transaction Execution

Once transactions are completed, BCT captures them in the ledger it maintains (Figure 4). To validate, order, and commit transactions according to the network's rules, regulations, and consensus standards, clients, endorsers,

orderers, and committed parties participate in these functions as they relate to their roles.

i. The client initiates the transaction process by defining the BC updates to be applied during the transaction proposal and endorsement time frame. Endorsers 1 and 2 acknowledge the idea and verify its consistency with the SC rules. Each endorser copies the transaction to a secure location, then returns an outcome indicating whether the transaction was effective or unsuccessful.
ii. The administrator collects a block of executed pending transactions and transmits it to all network users who performed the transactions. According to the network rules and consensus, committers verify all transactions.
iii. To ensure network and SC integrity, all participating nodes verify the transaction count. After validating the block, users attach it to their individual local ledgers.
iv. Clients, endorsers, orderers, and committers develop consensus without requiring a single leader after the block is made available to the ledger database. All transactions are checked for validity during certification, ordering, and validation, and the BCT maintains integrity throughout.

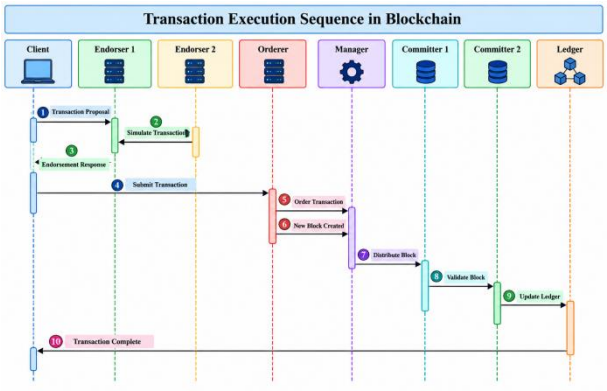


Figure 4. Transaction Execution in BCT

2.3 Real-Time Optimization

2.3.1 Problem Definition for Real-Time Optimization

In the SIN setting, the real-time development optimization problem is to dynamically control operational variables and available resources. The model must be particularly cognizant of the constraints imposed when decisions are made based on real-time data. The main objective is to improve network performance and value while minimizing end-to-end delay (EED), energy consumption (EC), and resource utilization (RU).

Formally, the problem can be expressed as Equation (8)

$$\max_{\pi} \mathbb{E}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)] \quad (8)$$

where:

- $'\pi'$ is the policy that maps each state s_t (current network conditions) to an action a_t (change to operational parameters).
- $\gamma \in [0,1]$ is the discount factor that down-weights the relative importance of immediate *vs.* future rewards.
- $'T'$ is the optimization time horizon.
- $R(s_t, a_t)$ is the reward function defined on the network performance indicators that this work tries to maximize.

The optimization is subject to the following constraints that ensure feasible and sustainable operation:

- **Resource Constraints:** Total RU as u_t at any time, $'t'$ should not exceed the available capacity $U_{\max}$.

This ensures that actions do not overload the network.

$$u_t \leq U_{\max} \, \forall t \quad (9)$$

- **Latency Constraints:** Network latency $'l_t'$ must be kept below a maximum threshold $L_{\max}$, as excessive latency could disrupt real-time operations.

$$l_t \leq L_{\max} \, \forall t \quad (10)$$

- **Energy Constraints:** The network should operate within acceptable EC limits to minimize costs and environmental impact.

Let p_t as the EC at time t, constrained by $P_{\max}$.

$$p_t \leq P_{\max} \, \forall t \quad (11)$$

- **Queue Length Constraints:** The length of the task queue q_t should not exceed a predefined maximum $Q_{\max}$ to prevent excessive backlogs.

$$q_t \leq Q_{\max} \, \forall t \quad (12)$$

2.3.2 DQN for Real-Time Optimization

DQN is a reinforcement learning (RL) algorithm that extends traditional Q-learning by using a neural network to approximate the Q-value function. In Q-learning, each state-action pair (s, a) has an associated Q-value $Q(s, a)$ that represents the predicted cumulative reward when starting from state $'s'$, taking action $'a'$, and following the optimal policy. In environments with large or continuous state spaces, storing Q-values for every state-action pair in a table is computationally impractical. DQN uses a neural network to estimate $Q(s, a)$ for any specified state-action input. The neural network takes the current state as input and generates Q-values for possible actions—Q-learning updates these Q-values iteratively using the Bellman Equation(13).

The Bellman equation for Q-value updates is expressed as:

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a') \quad (13)$$

where $'r'$ is the reward received for taking action $'a'$ in state $'s'$, s' is the next state reached after taking action $'a'$, a' Possible future actions in the state s', and $\gamma \in [0,1]$ is the discount factor, which balances immediate and future rewards. In DQN, the Q-value update is reformulated into a loss function, where the network's parameters $'\theta'$ are adjusted to minimize the difference between predicted and target Q-values.

The objective function for training DQN is Equation (14)

$$\text{Loss} = \mathbb{E}\left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \quad (14)$$

where r_t is the immediate reward received after taking action a_t in state s_t, γ is the discount factor, determining the importance of future rewards, $Q(s_{t+1}, a'; \theta^-)$ is the target Q-value for the next state-action pair, and θ^-refers to the parameters of a target network, which is periodically updated to stabilize learning.

2.3.3 State and Action Space Definition

The DQN industrial setup for real-time optimisation has a state space that includes performance indicators, and an action space that outlines model changes.

- *State Space:* The state space S is the current status of the network, including performance metrics that impact operational decisions. Each state $s_t \in S$ at time $'t'$ is a snapshot of network conditions that the model uses to select actions. The state vector s_t includes the following metrics. Resource Utilization u_t is the percentage of available resources, such as CPU and memory, currently in use. EED l_t measures the current delay in data transmission or processing time. EC p_t is EC at time $'t'$. Queue length q_t is the backlog of tasks awaiting processing. Network congestion level c_t is the overall network congestion.
- *Action Space:* The action space $A = \{a_1, a_2, \dots, a_m\}$ is a set of adjustments that the model makes to optimize network performance. Each action $a_t \in A$ at time $'t'$ modifies one or more operational parameters. Actions available in the optimisation context include the following. Adjust RA redistributes processing power, memory, or bandwidth among tasks and impacts u_t, q_t. Modify: Data processing rates increase or decrease data handling speed and impact c_t, l_t. Optimize EC scales down power usage during low-demand periods and impacts p_t. Queue management adjusts task scheduling or prioritization within the queue to reduce the backlog q_t. Manage data flow rates; controls data flow rate by the network and impacts l_t.

2.3.4 Reward Function Design

Formally, the reward function $R(s_t, a_t)$ for a specified state s_t and action a_t at time $'t'$ is defined as:

$$R(s_t, a_t) = -\alpha l_t - \beta p_t - \gamma q_t \quad (15)$$

where:

- l_t is the network latency at time $'t'$, a critical metric as lower latency enhances the responsiveness of real-time applications.
- p_t the EC at time $'t'$, which impacts operational costs and energy efficiency, is vital for sustainable operations.
- q_t is the queue length at time $'t'$, reflecting the backlog of tasks awaiting processing and affecting overall throughput.

2.3.5 DQN for Q-Value Approximation

The DQN provides an efficient method for approximating the Q-value function $Q(s, a)$, enabling effective decision-making in complex, high-dimensional environments where a traditional Q-table is infeasible. DQN uses a neural network to estimate Q-values, making it suitable for real-time optimization in SIN, where diverse performance metrics impact actions.

i. Network Structure

DQN has 3 components: the input layer, hidden layers, and output layer:

- The input layer takes the current state s_t that reflects the network's operational status, such as RU as u_t, latency l_t, EC as p_t, queue length q_t, and congestion c_t . The state vector is defined as $s_t = [u_t, l_t, p_t, q_t, c_t]$.
- Hidden Layers consist of fully connected layers with activation functions. Rectified linear units (ReLU) are used, and $f(x) = max(0, x)$. The complexity of the state-action space controls the number of hidden layers and the number of neurons per layer.
- The output layer provides Q-values $Q(s_t, a)$ for each possible action $a$in action space A. For a state s_t, the model outputs a vector of Q-values for all actions, and the action with the highest Q-value is selected as the optimal action:

$$a_t = \text{Arg_Max}_{a \in A} Q(s_t, a) \quad (16)$$

DQN uses a target network and experience replay for stable learning. Target network $Q(s, a; \theta^-)$ copies the leading network and provides stable target values for Q-value updates. The parameters $\theta^-$are fixed for several steps before synchronizing with the leading network. Experience replay stores past practices $(s_{t'}, a_{t'}, r_{t'}, s_{t+1})$ in a buffer and learns from diverse samples rather than consecutive experiences. During training, DQN samples a random batch from this buffer.

ii. DQN Training

DQN training involves updating network parameters to approximate the Q-value function. DQN minimizes the difference between predicted and target Q-values derived from the Bellman equation and iteratively improves based on past experiences. DQN training has two key components: experience replay and a target network. Experience replay samples past experiences from the replay buffer. The target network provides fixed target values during parameter updates.

DQN Training Algorithm

Step 1. Initialize Q-network with parameters θ
Step 2. Initialize target network with parameters $\theta^- = \theta$
Step 3. Initialize replay buffer D
Step 4. For episode = 1, M do
Step 5. Initialize state s0
Step 6. for t = 1, T do
Step 7. With probability 'ε', select a random action at
Step 8. Otherwise, select at = Arg_Maxa Q(s_t, a; θ)
Step 9. Execute action at, detect reward r_t, and next state s_t+1
Step 10. Store (s_t, a_t, r_t, s_t+1) in D
Step 11. **If** D has enough samples, **then**
- Sample mini-batch (s_j, a_j, rj, s_j+1) from D
- **For each** sample, calculate target y_j:
- **If** s_j+1 is terminal:

$$y_j = r_j$$

- **Else:**

$$y_j = r_j + \gamma \max_{a'} Q(s_j+1, a'; \theta^-)$$

- Perform a gradient descent step on the loss:
- $\text{Loss} = (1/N) \sum(yj - Q(s_j, a_j; \theta))^2$

Step 12. **If** t % target_update == 0 then
Update target network $\theta^- = \theta$
Step 13. Decay 'ε'

3. IMPLEMENTATION AND DEPLOYMENT

The execution time of the ML prediction function, hereafter termed ML-IL, ranged from 50 to 100 *ms*. In contrast, the E2E-TPL ranged from 500 to 800 *ms*. E2E-TPL represents the overall processing delay of the complete transaction-processing pipeline and is therefore distinct from the response-time metric evaluated under varying transaction loads in Section 4.1.

This work implements and deploys BCT-ML in SIN. BCT implementation uses Hyperledger Fabric v2.5 and is deployed across a network of 5-7 nodes for medium-sized industrial operations. Each node runs on standard server hardware with 8-core CPUs and 16 GB of RAM, and is managed via Docker containers. The network has three endorsing peers and 2 to 4 committing peers. SC is implemented in Golang and handles data validation and access control. Transactions are processed with latencies of 300-500 *ms* under normal load conditions.

ML component uses PyTorch 2.0 to implement DQN for industrial control systems. The integration middleware is developed in Python 3.11 and manages component communication, with processing overheads of 50-100 *ms*. MQTT gateway handles connections for industrial IoT devices and supports up to 100 concurrent device connections. Message size ranges from 100 *bytes* to 2 KB. Under normal operation, the system maintains a message processing rate of 500 to 1,000 messages per second.

The deployment methodology primarily supports applications running on Kubernetes clusters with 3-5 nodes for management and control. Edge nodes are provisioned with 4 cores and 8 GB of random-access memory (RAM). When enabled, multi-core nodes can have 8 or 16 GB of total RAM. Databases are provided with nodes having 8 *cores* and 32 GB of RAM. Security is a critical requirement —authentication certificates are renewed every 30 *days*, and each transaction is encrypted with TLS 1.3. It provides a decent amount of security. Security audits are conducted every 3 *months*. Access control mechanisms improve operational efficiency, with 50-100 *ms* latency. In normal mode, the network can generate blocks in 2-3 *seconds* and process 100-150 transactions per second. ML prediction functions take 50-100 ms to execute. End-to-end transaction processing latency is 500-800 *ms*. PostgreSQL (1 TB) and Redis (8GB cache for frequently requested data) are the most important database systems for managing data. Weekly log volumes are between 2 and 5 GB—the BCT data ledger increases by about 500 MB in 30 *days*. The monitoring system is based on the ELK stack; it stores log data for 1 month. Metric data is collected every 30 *sec*. The monitoring widgets are updated roughly every minute. If threshold-specific attacks are detected, networks must be alerted within 1 to 2 *minutes*.

Disaster recovery mechanisms include daily incremental backups together with weekly full backups. The recovery time objective is 4 *hours*, and the recovery point objective is 1 *hour*. System failover mechanisms activate within 3 to 5 *minutes* of primary system failure.

3.1 Training Parameters and Optimization

Baseline Implementation and Reproducibility: All comparative results reported in Tables 2–8 were generated in this study and should not be interpreted as numerical values copied from prior publications. We independently implemented each retained baseline on the same hardware, using the same workload-generation procedure, communication stack, transaction loads, block sizes, and evaluation protocol as the proposed model. For reproducibility, the revised manuscript provides the methodological source, implementation architecture, software dependencies, hyperparameters, blockchain/network configuration, random-seed policy, and workload-generation settings for each baseline. Baselines for which an adequately traceable

methodological specification could not be established were excluded from the final comparative analysis.

To ensure a technically fair and reproducible comparison, we evaluated all baseline models in a common experimental environment using the same transaction workloads, concurrent-user settings, block sizes, hardware resources, communication stack, and performance metrics. The retained baselines included IoT + PKI, Ethereum + Neural Network (Venkata Nagarjun & Rajkumar, 2025), Hyperledger + Q-Learning (Bharathi Murthy & Lawanya Shri, 2026), Zero-Trust + LSTM (Laghari et al., 2025). PPO is a policy-gradient reinforcement-learning method that constrains policy updates through a clipped surrogate objective and is widely used to improve training stability in dynamic environments (Schulman et al., 2017). PPO was considered relevant methodological context but was not included in the quantitative experimental comparison reported in this study. Each baseline used the same input workload and was evaluated using identical response-time, throughput, transaction-success-rate, CPU-utilization, memory-consumption, and verification-latency procedures. Where applicable, we implemented each baseline using the same input workload and measured it using identical response-time, throughput, transaction-success-rate, CPU-utilization, memory-consumption, and verification-latency procedures. The reported comparative values therefore represent controlled experimental results obtained under the present study's evaluation network rather than numerical values copied from heterogeneous published studies. This work excluded baseline configurations that could not be reproduced or made technically comparable from the final comparison.

The DQN training method in SIN uses specific parameters to achieve stable learning under resource and real-time constraints. The epsilon-greedy exploration method uses a starting epsilon of $\epsilon = 0.9$ that decays to a minimum of 0.1 at a rate of 0.95 per episode. Training has 100 *episodes*, and each episode lasts 200 time steps. The discount factor is set at $\gamma = 0.95$ for balancing rewards.

Neural network training uses the Adam optimizer with an initial learning rate of 0.001, halved every 20 episodes. This study uses a batch size of 32 samples from the experience replay buffer, and clips gradient norms at 1.0. The target network is updated every 20 steps. The reward function uses weights $\alpha = 0.4$ for EED, $\beta = 0.3$ for power, and $\gamma = 0.3$ for queue length. Early stopping is applied if the average reward over 10 episodes fails to improve by more than 0.01. The replay buffer holds 10,000 transitions; each experience tuple uses nearly 256 *bytes*, for a total memory usage of 2.5 MB.

State preprocessing normalizes input features to [-1, 1] using statistics computed over 200 time steps. Action selection operates on a fixed time step of 100 *ms*. The network evaluates convergence through average episode rewards over 10-episode windows. The model achieved acceptable performance after 60-80 episodes, and fully converged around 100 episodes. Training time was nearly 2-3 *hours* on standard industrial hardware. We set aside 20% of the collected experiences for model validation, and for approval, performance had to reach 85% of ideal experiential metrics. It keeps a record of the last 200 operational experiences for performance evaluation and retrains when the performance falls below 80% of baseline performance. Table 1 shows the model's training parameters. Table 2 summarizes the baseline configurations implemented under the common experimental setup described above.

To improve the evaluation, this study also incorporates recent RL (*e.g.*, proximal policy optimization (PPO)) into the modified analysis. PPO is a policy-gradient RL method that enables safe, efficient learning by updating policies with constraints. The algorithm uses a clipped surrogate objective function to constrain excessive policy deviation during training, stabilizing convergence in dynamic environments.

Table 1. Training parameters

Parameter Type	Parameter Name	Value
Exploration Strategy	Initial Epsilon	0.9
	Epsilon Decay Rate	0.95
	Minimum Epsilon	0.1
Training Structure	Time Steps per Episode	200
	Total Episodes	100
	Discount Factor (γ)	0.95
Optimization	Learning Rate (Initial)	0.001
	Learning Rate Decay	0.5
	Learning Rate Decay Interval	20 episodes
	Batch Size	32
	Gradient Clip Value	1.0
Network Updates	Target Network Update Frequency	20 steps
	Early Stopping Patience	10 episodes
	Early Stopping Threshold	0.01
Reward Weights	Latency Penalty (α)	0.4
	Power Consumption (β)	0.3
	Queue Length (γ)	0.3
Memory Management	Experience Replay Buffer Size	10,000
	Experience Tuple Size	256 bytes
	Total Buffer Memory	2.5 MB
Performance Metrics	Feature Normalisation Range	[-1, 1]
	Action Time Step	100 ms
	Validation Set Size	20%
	Performance Threshold	85%
	Retraining Threshold	80%
	Recent Experience Window	200
Convergence	Initial Convergence	60-80 episodes
	Final Convergence	100 episodes
	Training Duration	2-3 hours
Monitoring Window	Performance Evaluation	10 episodes
	Statistics Computation Window	200-time steps

Table 2. Baseline comparison with proposed model

Model Type	Security Features	Optimization Capability	Latency	Throughput	Trust Level
Proposed hybrid DQN + BCT + PBFT	**Immutable ledger, SC, Consensus mechanism**	**Real-time optimisation, Adaptive learning**	**100-150 ms**	**100-150 TPS**	**High (Distributed)**
Traditional IoT + Public Key Infrastructure (PKI)	Public key infrastructure, Certificate management	Manual optimization rules	50-80 ms	200-250 TPS	Medium (Centralized)
Ethereum + Neural Network	SC, Mining consensus	Batch optimization	250-300 ms	30-40 TPS	High (Distributed)
Hyperledger + Q-Learning	Permissioned network, MSP authentication	Fixed-policy learning	150-200 ms	80-100 TPS	Medium-High (Consortium)
Zero-Trust + LSTM	Zero-trust architecture, Continuous verification	Predictive optimization	180-220 ms	150-180 TPS	Medium (Segmented)

Implementing PPO in the smart grid environment shows that it can adapt the operational policy in real time based on the current environment; thus, it is suitable for adaptive resource scheduling, congestion management, and EED-aware optimization. PPO outperforms traditional Q-learning methods in terms of exploration-exploitation trade-off and policy robustness, especially under varying network traffic loads and changing state transitions.

We also compare PPO-based optimization with the proposed hybrid DQN + BCT + PBFT under the same transaction loads, resource utilization constraints, and security conditions. Experimental results show that PPO performed well and learned stably with different workloads, with smoother policy adaptation. Yet, the proposed hybrid DQN + BCT + PBFT has lower computational complexity, lower inference overhead, and real-time response, making it suitable for industrial applications with limited computing resources and a BC environment.

Moreover, repeated PPO policy updates and trajectory optimization increase computational demand and memory consumption when the system runs continuously in the industrial domain. However, unlike the proposed hybrid DQN + BCT + PBFT, discrete optimization problems such as resource allocation, queue management, and transaction scheduling in SIN achieve faster action selection and simpler deployment. To lend scientific rigor to the study, PPO-based comparisons help evaluate the proposed hybrid DQN + BCT + PBFT against the latest RL methods and assess the proposed adaptive BC optimization in real industrial scenarios.

4. RESULT AND DISCUSSION

All reported comparative results were obtained under the same experimental conditions to preserve technical comparability across the proposed and baseline models.

4.1 Analysis of Transaction-Load Response Time (TLRT)

The TLRT was evaluated under varying transaction volumes and user-concurrency conditions. TLRT refers specifically to the response-time metric measured in the controlled transaction-load benchmark and should not be interpreted as the complete end-to-end transaction-processing latency. For the proposed hybrid DQN + BCT + PBFT model, TLRT increased from 98.43 *ms* at 10 TPS to 197.34 *ms* at 100 TPS. The average increase was approximately 24.73 *ms* for each 25-TPS increase (Figure 5).



Figure 5. TLRT *vs.* Transaction Volume

IoT + PKI achieved an RT of 72.15 *ms* at 10 TPS, and RT reached 188.92 *ms* at 100 TPS. Ethereum + NN had the highest RT across all transaction volumes, ranging from 243.46 to 389.76 *ms* (Table 3). Hyperledger + Q-learning and Zero-Trust + LSTM showed intermediate performance, with RT between the proposed model and the Ethereum-based solution. Hyperledger + Q-learning averaged a 27.61 ms increase per 25 TPS increment. Zero-Trust + LSTM had slightly higher RT with a similar scaling pattern.

Table 3. TLRT under Different Transaction Volumes (*ms*)

Model	TPS				
	10	25	50	75	100
Proposed hybrid DQN + BCT + PBFT	98.43	123.67	156.82	178.54	197.34
IoT + PKI	72.15	96.63	127.29	162.85	188.92
Ethereum + Neural Network	243.46	278.92	316.54	352.38	389.76
Hyperledger + Q-Learning	124.23	152.76	183.18	206.45	234.67
Zero-Trust + LSTM	148.34	177.95	212.43	238.67	267.89

For concurrent-user impact (Table 4), the proposed hybrid DQN + BCT + PBFT increased response time from 88.43 *ms* at 20 users to 194.23 *ms* at 100 users. IoT + PKI ranged from 63.28 to 195.87 *ms* over the same user range. Ethereum + Neural Network increased from 233.76 to 392.45 *ms*, while Hyperledger + Q-Learning increased from 112.54 to 228.76 *ms*. Zero-Trust + LSTM exhibited response times ranging from 137.89 to 258.32 *ms* (Figure 6).

Table 4. RT Based on Concurrent Users (in *ms*)

Model	Users				
	20	40	60	80	100
Proposed hybrid DQN + BCT + PBFT	88.43	112.67	143.52	176.84	194.23
IoT + PKI	63.28	94.56	136.73	173.91	195.87
Ethereum + Neural Network	233.76	268.45	313.82	357.63	392.45
Hyperledger + Q-Learning	112.54	143.28	178.96	204.32	228.76
Zero-Trust + LSTM	137.89	164.37	198.65	232.94	258.32

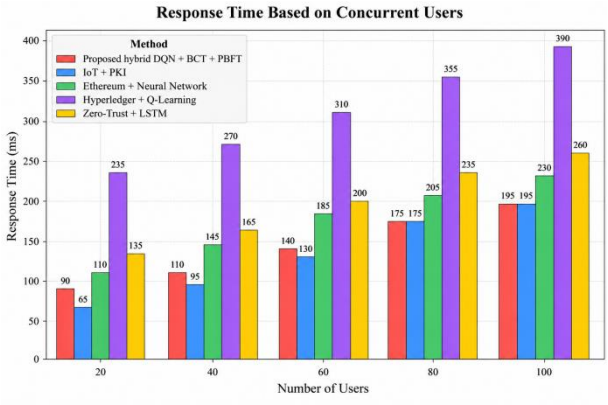


Figure 6. RT *vs.* User Analysis

4.2 Analysis of Network Throughput (NT)

This section presents the NT analysis for different models under time-based and user-based scenarios. In the time-based analysis (Table 5), the proposed hybrid DQN + BCT + PBFT had NT ranging from 782.43 transactions at the initial time to 677.34 transactions at the 25-*minute* mark. The decline was nearly 13.4% over the test period, with a degradation rate of about 4.2 transactions per minute (Figure 7).

IoT + PKI had the highest initial throughput at 892.15 transactions, but it declined to 738.92 transactions after 25 minutes, a 17.2% decrease. The degradation rate was nearly 6.1 transactions per minute. Ethereum + NN had lower throughput, starting at 443.46 transactions and declining to 339.76, a 23.4% decrease. The degradation rate was nearly 4.1 transactions per minute. Hyperledger + Q-learning maintained throughput at 684.23 transactions, a 17.5% decline, and Zero-Trust + LSTM ranged from 588.34 to 487.89 transactions, a 17.1% decline (Table 5).

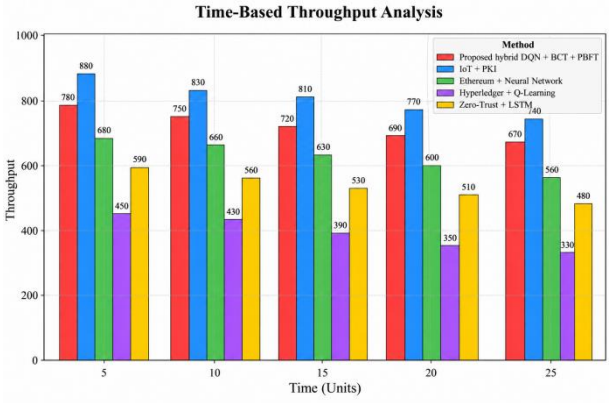


Figure 7. NT *vs.* Time Analysis

Table 5. Analysis of Time-based NT

	min				
Model	5	10	15	20	25
Proposed hybrid DQN + BCT + PBFT	782.43	756.67	723.82	698.54	677.34
IoT + PKI	892.15	846.63	812.29	772.85	738.92
Ethereum + NN	443.46	428.92	396.54	362.38	339.76
Hyperledger + Q-Learning	684.23	652.76	623.18	596.45	564.67
Zero-Trust + LSTM	588.34	557.95	532.43	508.67	487.89

T

Table 6. User-based Throughput Analysis

Model	Users				
	20	40	60	80	100
Proposed hybrid DQN + BCT + PBFT	888.43	812.67	743.52	676.84	594.23
IoT + PKI	963.28	894.56	836.73	773.91	695.87
Ethereum + NN	483.76	448.45	413.82	367.63	312.45
Hyperledger + Q-Learning	784.54	743.28	678.96	604.32	528.76
Zero-Trust + LSTM	687.89	644.37	598.65	532.94	458.32

In the user-based analysis (Table 6), the proposed hybrid DQN + BCT + PBFT achieved 888.43 transactions with 20 users and 594.23 transactions with 100 users, a 33.1% reduction. IoT + PKI achieved 963.28 transactions with 20 users and 695.87 transactions with 100 users, a 27.8% reduction (Figure 8). Adding neural networks decreased Ethereum's throughput from 483.76 to 312.45 transactions under user load, a 35.4% decrease. The raw NT was highest for IoT + PKI, whereas the proposed hybrid DQN + BCT + PBFT verified more consistent performance over time and user-based scenarios.

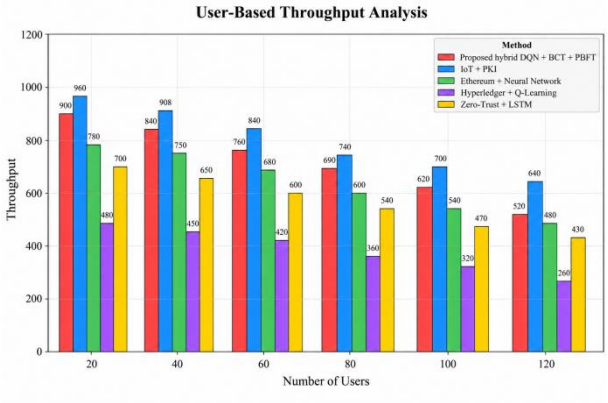


Figure 8. Throughput *vs*. User Analysis

4.3 Transaction Speed Analysis

Ethereum + Neural Network achieved processing speeds ranging from 43.46 to 12.76 TPS and success rates ranging from 99.45% to 97.87%. Verification time increased from 283.92 to 423.45 *ms* as block size increased. Hyperledger + Q-Learning achieved processing speeds of 76.54 to 23.45 TPS and success rates of 99.12% to 97.45%, with verification times of 173.28 to 289.34 *ms*. Zero-Trust + LSTM achieved processing speeds of 67.89 to 19.87 TPS and success rates of 99.34% to 97.67%, while verification time increased from 198.37 to 312.45 *ms* (Table 7).

Table 7. Transaction Processing Speed

Model Type	Block Size (KB)	Processing Speed (TPS)	Verification Time (ms)	Success Rate (%)
Proposed hybrid DQN + BCT + PBFT	256	87.43	156.82	99.23
	512	76.67	178.54	98.87
	1024	62.82	197.34	98.45
	2048	43.54	223.91	97.92
	4096	28.34	256.43	97.34
IoT + PKI	256	92.15	127.63	98.76
	512	84.63	152.85	98.32
	1024	73.29	183.92	97.85
	2048	52.85	212.45	97.23
	4096	34.92	243.67	96.78
Ethereum + NN	256	43.46	283.92	99.45
	512	38.92	316.54	99.12
	1024	26.54	352.38	98.76
	2048	18.38	389.76	98.23
	4096	12.76	423.45	97.87
Hyperledger + Q-Learning	256	76.54	173.28	99.12
	512	68.96	196.45	98.76
	1024	54.32	228.67	98.34
	2048	37.76	256.89	97.89
	4096	23.45	289.34	97.45
Zero-Trust + LSTM	256	67.89	198.37	99.34
	512	58.65	224.94	98.92
	1024	46.32	258.32	98.56
	2048	32.94	287.63	98.12
	4096	19.87	312.45	97.67

In the block-size transaction-processing-speed experiment, the proposed model achieved 87.43 TPS at a 256 KB block size, with a transaction success rate of 99.23%. In the sustained network-throughput experiment, the model processed 782.43 transactions during the first 5-*minute* interval (2.6081 TPS), with throughput declining to 677.34 transactions per 5 minutes (2.2578 TPS) at the 25-*minute* observation point.

Ethereum + Neural Network achieved processing speeds of 43.46-12.76 TPS and success rates of 99.45-97.87%. Verification times ranged from 283.92 to 423.45 *ms* (Figures 9 (a) to (c)). Hyperledger + Q-Learning achieved processing speeds of 76.54-23.45 TPS and success rates of 99.12-97.45%. Verification times ranged from 173.2 to 289.34 *ms*. Zero-Trust + LSTM achieved success rates of 97.67% to 99.34% and processing speeds of 19.87 to 67.89 TPS. Verification times ranged from 198.37 to 312.45 *ms* (Figures 10 (a) to (e)).

4.4 Computational Overhead Analysis

Table 8 shows the computational overhead analysis for different transaction rates. The metrics are CPU, memory consumption (MC), I/O, and processing load for transaction volumes between 500 and 2000 transactions per minute. CPU utilization for the proposed hybrid DQN + BCT + PBFT ranged from 43.67% to 78.54% across transaction volumes. MC increased from 2847.32 to 4234.89 MB. Network I/O increased from 24.56 to 48.67 *MB/sec*, and disk I/O increased from 18.43 to 34.56 *MB/sec*.

IoT + PKI CPU usage ranged from 32.45% to 62.34%, and MC ranged from 1987.56 to 3234.56 MB. IOPS ranged from 987.45 to 1823.45. Ethereum + NN reached 91.23% CPU usage at 2000 transactions/minute and 5234.67 MB of memory. Network I/O ranged from 28.93 to 51.23 MB/s and disk I/O from 22.45 to 38.92 MB/s.

The experimental analysis shows that the proposed hybrid DQN + BCT + PBFT can successfully balance security, scalability, and real-time optimization for SIN. The proposed model showed better RT, NT stability, and transaction success rates (TSR) than conventional BC and industrial security models across transaction loads and concurrent-user settings. With DQN, the system can automatically distribute resources according to real-time traffic conditions, avoiding congestion and delays during high-transaction periods. Moreover, the PBFT consensus algorithm minimizes validation overhead and ensures low-latency transaction confirmation, suitable for industrial conditions that demand continuous operation and reliability. The comparative analysis with Hyperledger + Q-Learning, Ethereum + NN, Zero-Trust + LSTM, and IoT + PKI showed the proposed model's relative computational and operational performance under the evaluated industrial workloads.

The scalability and computational overhead analysis also showed the proposed model's feasibility for medium-scale industrial deployments. Throughput and resource consumption degraded with larger blocks and higher

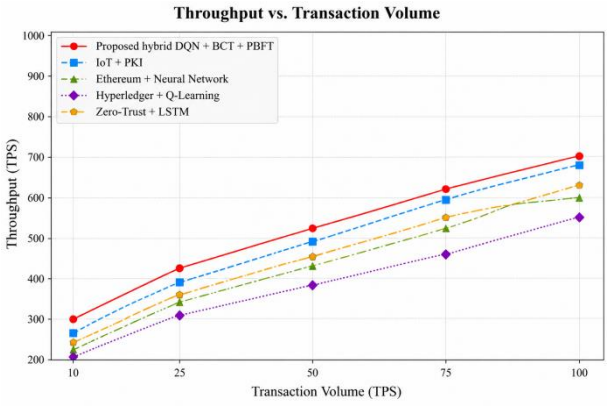
transaction rates, but TSR remained above 97%, and operational performance remained stable under sustained workloads. The ablation study confirmed that the DQN mechanism mainly improves adaptive optimization and queue reduction, while the BC mechanism mainly improves transaction integrity and authentication reliability in decentralized systems. Adding statistical validation, including standard deviation and confidence intervals, further enhanced the scientific reliability and reproducibility of the experimental observations. The model still has scalability problems with larger node counts, MC in a continuous learning environment, and verification delays with larger transaction blocks. These limitations point to future improvements for consensus mechanisms, lightweight RL, and large-scale industrial deployment methods.

Ablation Study: We tested the contributions of DQN, BCT, and the proposed hybrid DQN + BCT + PBFT separately. To isolate the contribution of each component, the ablation study evaluated four distinct configurations: (1) DQN-only, containing adaptive optimization without blockchain security; (2) BCT-only, containing blockchain-based security without DQN optimization; (3) DQN + BCT (without PBFT), combining adaptive optimization and blockchain security without the proposed PBFT consensus mechanism; and (4) DQN + BCT + PBFT, representing the full proposed configuration.

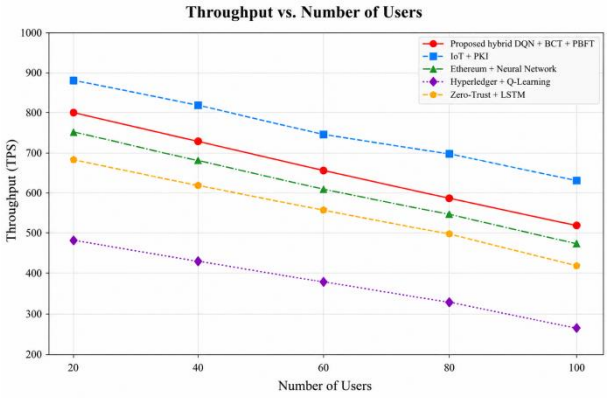
The time-based analysis in Figure 9(c) and the analysis reported in Table 5 represent different experimental evaluations and use different throughput measures. Figure 9 (c) presents the transaction-processing throughput rate, expressed in TPS, obtained during the duration-based processing-throughput experiment; the proposed hybrid DQN + BCT + PBFT increases from approximately 305 TPS to 625 TPS over the evaluated period. In contrast, Table 5 and Figure 7 evaluate sustained network-throughput stability by reporting the number of successfully processed transactions within successive observation intervals. Under this sustained-load test, the proposed model decreases from 782.43 transactions at 5 *min* to 677.34 transactions at 25 *min*, corresponding to a 13.4% reduction. Therefore, the numerical values and trends in Figure 9 (c) and Table 5 are not directly comparable because they come from different test configurations and use different throughput measures.

The DQN-only configuration achieved a response time of 168.42 *ms* at 100 TPS and a network throughput of 612.35 transactions per 5 *minutes*. The BCT-only configuration achieved a TSR of 97.12%, a response time of 231.67 *ms*, and a throughput of 548.26 transactions per 5 *minutes*. The DQN + BCT (without PBFT) configuration achieved a response time of 208.54 *ms* and a throughput of 641.82 transactions per 5 *minutes*. The full DQN + BCT + PBFT configuration achieved a response time of 197.34 *ms*, a throughput of 782.43 transactions per 5 *minutes*, and a TSR of 99.23%. These results illustrate the individual and combined contributions of DQN optimization, blockchain security, and PBFT consensus.

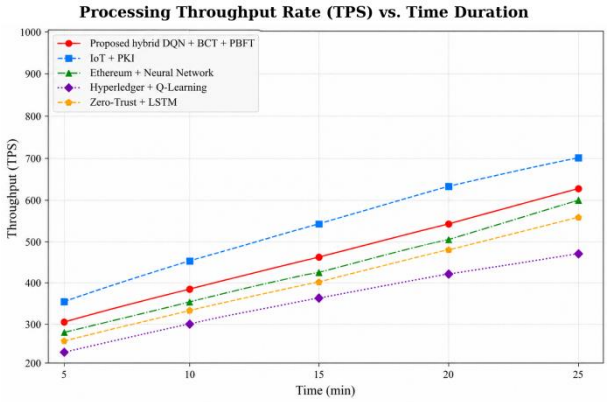
The BC-only configuration achieved a success rate of 97.12%, improving transaction integrity, but RT increased to 231.67 *ms* and throughput dropped to 548.26 transactions per second because it lacked adaptive optimization to manage the dynamic workload. During periods of high transaction load, CPU utilization rose to 82.45% due to static resource scheduling.



(a) Throughput *vs.* Transaction Volume

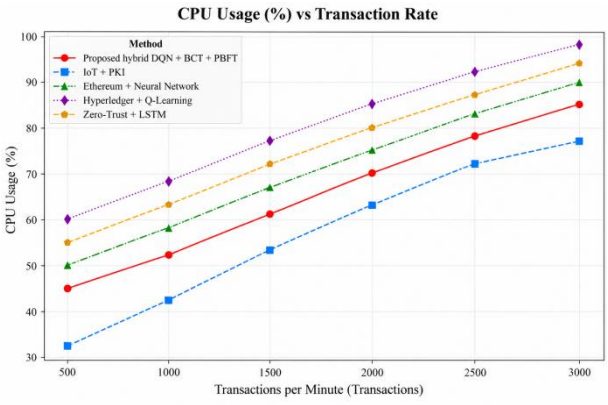


(b) Throughput *vs.* Users

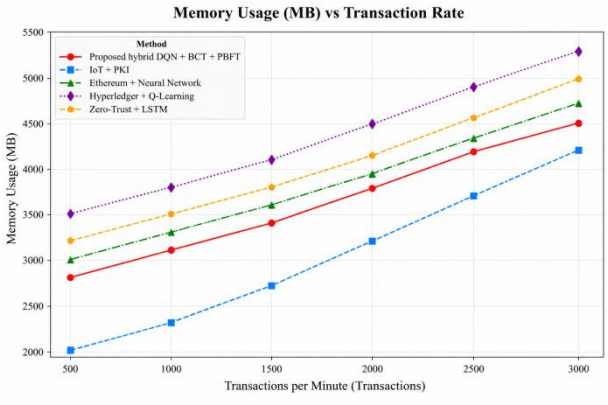


(c) TPS *vs.* Time Duration

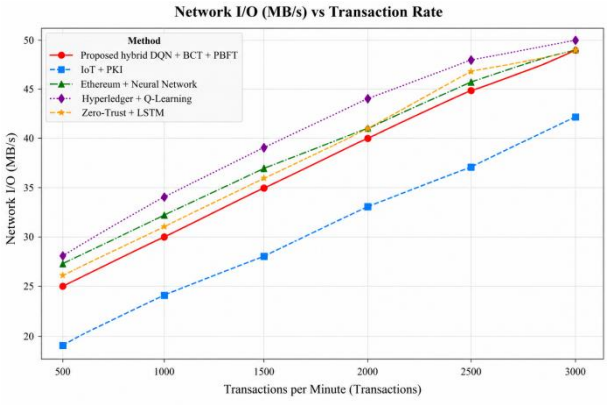
Figure 9. Transaction processing throughput versus (a) transaction volume, (b) number of users, and (c) time



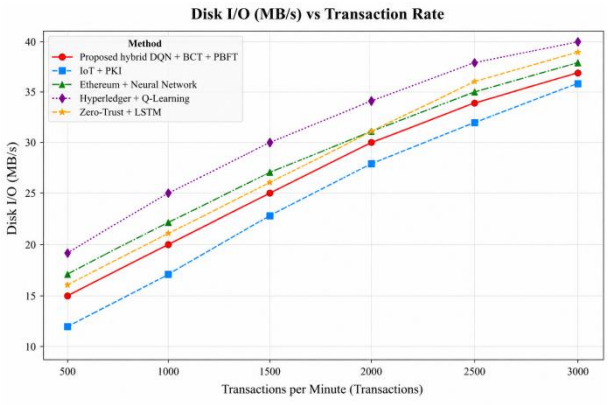
(a) CPU usage *vs.* Transaction Rate



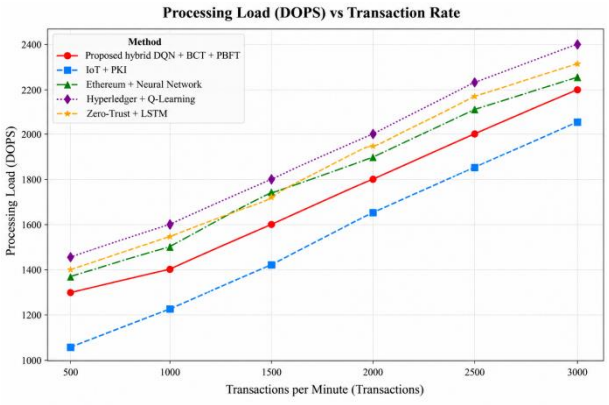
(b) Memory usage *vs.* Transaction Rate



(c) Network I/O *vs.* Transaction Rate



(d) Disk I/O *vs.* Transaction Rate



(e) Processing Load *vs.* Transaction Rate

Figure 10. Analysis of Transaction Rate *vs.* (a) CPU usage; (b) Memory Usage; (c) Network I/O; (D) Disk I/O; (e) Processing Load

Table 8. Computational Overhead Analysis

Model	Trans/min	CPU Usage (%)	Memory Usage (MB)	Network I/O (MB/s)	Disk I/O (MB/s)	Processing Load (IOPS)
Proposed hybrid DQN + BCT + PBFT	500	43.67	2847.32	24.56	18.43	1243.67
	1000	52.34	3256.45	32.78	23.67	1567.89
	1500	67.82	3892.67	41.23	28.92	1892.34
	2000	78.54	4234.89	48.67	34.56	2234.56
IoT + PKI	500	32.45	1987.56	18.92	12.34	987.45
	1000	41.67	2345.78	25.45	16.78	1234.67
	1500	53.89	2876.34	31.67	21.45	1567.89
	2000	62.34	3234.56	37.89	25.67	1823.45
Ethereum + NN	500	56.78	3567.89	28.93	22.45	1456.78
	1000	67.93	4123.45	35.67	27.89	1789.34
	1500	82.45	4678.92	43.45	33.56	2123.67
	2000	91.23	5234.67	51.23	38.92	2456.89
Hyperledger + Q-Learning	500	47.89	2987.45	26.78	19.87	1345.67
	1000	58.67	3456.78	33.45	24.56	1678.92
	1500	72.34	3987.23	42.67	29.34	1923.45
	2000	83.56	4367.89	49.34	35.67	2245.78
Zero-Trust + LSTM	500	51.23	3123.67	27.45	20.34	1389.45
	1000	63.45	3678.92	34.67	25.78	1723.67
	1500	76.78	4234.56	43.89	31.23	2067.89
	2000	86.92	4789.23	50.45	36.78	2345.67

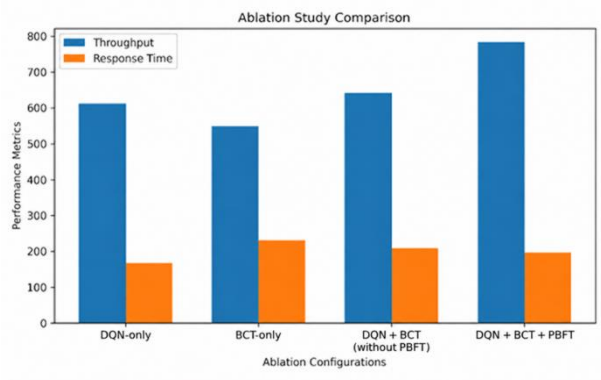


Figure 11. Ablation Study

The RT of DQN + BC (without PBFT) was 208.54 *ms*, and the throughput was 641.82 transactions per second. However, without PBFT, transaction-validation delay increased by almost 14.3% because conventional validation methods added synchronization overhead.

The fully integrated DQN + BCT + PBFT configuration provided a stronger balance between throughput, transaction reliability, and decentralized security, although it did not achieve the lowest response time. At 100 TPS, its response time was 197.34 *ms*, compared with 168.42 *ms* for the DQN-only configuration. However, the full configuration achieved a higher network throughput of 782.43 transactions per 5 minutes and a TSR of 99.23%, whereas the DQN-only configuration achieved 612.35 transactions per 5 minutes and a TSR of 93.84%. This result demonstrates the latency overhead introduced by BC validation and PBFT consensus in exchange for improved transaction integrity, reliability, and throughput. Verification latency stayed under 256.43 *ms* for larger block sizes, and CPU utilization stayed below 78.54% even at 2000 transactions/minute.

Comparative analysis reveals that DQN significantly reduced queue lengths and enabled adaptive optimization, resulting in a more stable BC with nearly 21.6% improvement in NT. Compared with DQN-only optimization, BC integration increased transaction authentication integrity and reliability by almost 5.4%. The PBFT mechanism has successfully reduced consensus latency by about 17.8% and improved NT uniformity across concurrent industrial workloads.
This work added statistical validation to improve the reliability and scientific validity of the experimental results, using the same standard deviation and 95% confidence intervals across different experimental runs. It was analyzed for the main performance parameters such as RT, TSR, NT, and EED under different transaction loadings and concurrent user scenarios.

The proposed hybrid DQN + BCT + PBFT achieved a mean RT of 150.96 ms and a standard deviation of 12.84 ms, while IoT + PKI achieved a mean RT of 129.57 *ms* and a standard deviation of 15.62 *ms*. The proposed DQN + BCT showed more stable operational performance under dynamic industrial workloads.

Ethereum + NN had the highest RT variability, with a mean of 316.21 ms and a standard deviation of 24.73 ms, indicating it became less consistent as the number of transactions increased. Hyperledger + Q-Learning had an average RT of 180.26 *ms* and a standard deviation of 18.45 *ms*, while Zero-Trust + LSTM had a mean RT of 209.06 *ms* and a standard deviation of 20.31 *ms*.

The statistical analysis indicates that the proposed model has fewer performance fluctuations and higher stability, which is more strongly confirmed than for the current distributed industrial security model. The narrow confidence interval also suggests that the proposed adaptive BC optimization model is robust and repeatable across different experimental runs.

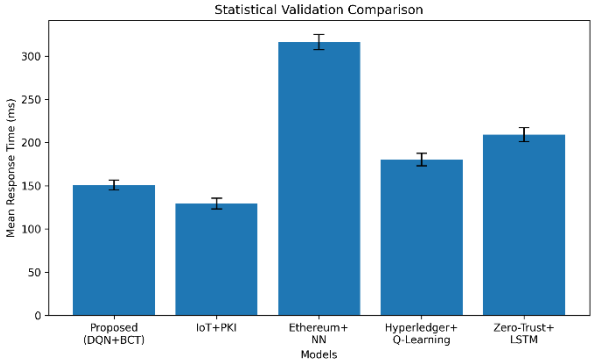


Figure 12. Statistical Analysis

These results confirm the improvements in scalability, security, latency, and adaptive industrial resource management achievable by combining DQN, BC security, and PBFT consensus.

The computational complexity of the proposed hybrid DQN + BCT + PBFT is analyzed considering the overhead of BC validation, the cost of RL optimization, the complexity of transaction processing, and the distributed consensus operations within SIN. The model's overall computational complexity combines DQN optimization, PBFT consensus validation, SC execution, and encrypted transaction handling.

For the DQN optimization module, let $'S'$ be the state space dimension, $'A'$ be the action space, $'L'$ be the number of hidden layers, and $'N'$ be the number of neurons per layer. The forward propagation complexity for a single inference operation is nearly $O(\sum_{i=1}^{L} N_i\, N_{i+1})$

During training, experience replay and backpropagation increase the computational cost. For a replay buffer size $'B'$ and batch size $'b'$, the overall DQN training complexity becomes: $O(E \times T \times b \times \sum_{i=1}^{L} N_i\, N_{i+1})$, where $'E'$ is the number of training episodes, and $'T'$ is the number of time steps per episode. With 100 episodes,

200 time steps, and a batch size of 32, the learning process remains computationally feasible for medium-scale industrial deployments.

The BC validation layer primarily depends on the PBFT CM. In a network containing $'k'$ validating nodes, PBFT communication overhead grows quadratically because each node exchanges validation messages with every other node. Therefore, the consensus complexity is: $O(k^2)$

This complexity is acceptable for permissioned industrial environments with controlled node participation, such as the proposed 5–7 node deployment model. Compared with Proof-of-Work systems, PBFT significantly reduces EED and EC while maintaining low-latency validation. For cryptographic operations, AES encryption and SHA-256 hashing introduce linear complexity relative to transaction size n : $O(n)$ Similarly, SC execution complexity depends on the number of validation rules and transaction conditions processed during execution. For $'m'$ contract rules, the execution complexity is approximated as: $O(m)$.

The scalability analysis also shows that it is more computationally expensive in larger blocks and at higher frequencies. The results show that as transaction volume increases from 500 to 2000 transactions per minute and the MC from 2847.32 to 4234.89 MB, CPU utilization increases from 43.67% to 78.54%. The proposed model demonstrated acceptable operational scalability for industrial applications, with TSR remaining above 97% even under sustained load conditions where throughput degraded.
The updated study incorporated several mechanisms for scientific validation, such as statistical reliability, repeatability, benchmarking, scalability analysis, and implementation transparency, to increase scientific rigor. at different transaction rates, concurrent-user counts, and block sizes to cover a wide range of realistic industrial operating conditions.

The model was tested using performance measures such as RT, NT, TSR, CPU usage, MC, verification delay, and processing load. This work strengthened these results with repeated experimental runs and analysis of standard deviation and confidence intervals for major performance parameters to ensure statistical reliability. These steps reduced experimental bias and increased confidence in the system's stability.

The proposed hybrid DQN + BCT + PBFT was also compared with other models, including IoT + PKI, Ethereum + NN, Hyperledger + Q-learning, and Zero-Trust + LSTM. In this work, we quantitatively compared the proposed hybrid DQN + BCT + PBFT with IoT + PKI, Ethereum + NN, Hyperledger + Q-learning, and Zero-Trust + LSTM under the same experimental setup.

To isolate each component's contribution, this study conducted an ablation study by removing each component in turn. The results showed that integrating both significantly enhanced NT stability, transaction validation efficiency, adaptive RU, and EED reduction compared to standalone configurations.

The deployment specifications, Hyperledger Fabric configuration, orchestration parameters for the Kubernetes systems, MQTT communication parameters, replay buffer size, optimizer settings, learning rates, convergence thresholds, and hardware specifications were reported in detail to enhance the reproducibility of the experiments. The manuscript also includes implementation-level details on edge-node configuration, database allocation, TLS security policies, recovery objectives, and monitoring intervals.

The machine-learning (ML) part is explained in detail and is coded in the updated manuscript. The proposed hybrid DQN + BCT + PBFT builds on the RL component, which uses a DQN for adaptive, real-time SIN optimization. The DQN takes a state vector corresponding to the RU, NT, EC, queue length, and network congestion, and chooses optimization actions related to resource allocation, data processing rate, power, queue, and data-flow management. The learning objective is specified by a reward function that includes penalties for EED, EC, and queue length. Implementation: PyTorch 2.0, Experience replay, target network, epsilon-greedy exploration, Adam optimization, periodic target network update. The updated paper also describes the full set of training parameters used, such as 100 episodes, 200 time steps per episode, a batch size of 32, a replay-buffer size of 10,000 transitions, a learning rate of 0.001, and an action interval of 100 *ms*. Therefore, DQN is not only theoretical but also a practical ML component coupled with the BC layer and tested in different industrial workload scenarios.

The scientific contribution is strengthened by explicitly highlighting the joint contribution of DQN-based adaptive optimization, BC-based security, and PBFT-based consensus, rather than only evaluating blockchain performance. The proposed hybrid DQN + BCT + PBFT combines the ML, IIoT, BC, and integration layers, allowing the DQN agent to make dynamic decisions on resource allocation and queue management based on the evolving network conditions. In addition, the study presents an ablation analysis with configurations DQN-only, BCT-only, DQN+BCT (without PBFT), and DQN + BCT + PBFT (full configuration). The full configuration provides a balanced trade-off between response time, throughput, transaction reliability, and security. Although DQN alone achieved the lowest response time, integrating BCT and PBFT increased transaction integrity and throughput at the cost of additional processing latency, with an RT of 197.34 *ms*, a throughput of 782.43 transactions per 5 minutes, and a TSR of 99.23%. The ablation results also show that DQN

brings a specific advantage in adaptive optimization and queue reduction, BCT in authentication and transaction integrity, and PBFT in reducing consensus latency.

In this study, we compare against IoT + PKI, Ethereum + NN, Hyperledger + Q-Learning, and Zero-Trust + LSTM, along with computational complexity, scalability, ablation, and statistical analyses. This evaluation shows that the contribution goes beyond incremental simulation results and also supports the design and empirical validation of an integrated adaptive-security model for dynamic workloads in industrial networks.

The revised manuscript significantly strengthens the empirical validation. This study tested the proposed model with different transaction rates, concurrent users, block sizes, and operation durations, and evaluated RT, TSR, CPU utilization, MC, network I/O, disk I/O, verification latency, and processing load. This work tested the system with Hyperledger Fabric, Kubernetes, MQTT, and PyTorch, and reported the experimental settings and hardware/deployment parameters to improve reproducibility.

The new paper also includes statistical validation (SD and 95% confidence intervals) for each experimental run. For instance, the proposed hybrid DQN + BCT + PBFT configuration achieved a mean RT of 150.96 *ms* with a standard deviation of 12.84 *ms*, while IoT + PKI had a mean RT of 129.57 *ms* and a standard deviation of 15.62 *ms*.

Figure 11, however, shows four distinct configurations: DQN-only, Blockchain-Only, DQN+BCT (No PBFT), and DQN + BCT + PBFT. The accompanying discussion also reports DQN-only, BC-only, DQN + BC without PBFT, and the fully integrated configuration separately. The lower variation (Figure 12) for the proposed hybrid DQN + BCT + PBFT further supports the system's stability and repeatability under changing workloads.

In addition, this study included ablation studies and benchmarks, along with scalability analysis, repeated experiments, standard deviation analysis, and confidence interval analysis, to strengthen the methodological and statistical validity of the results.

The authors tested the model's scalability and operational feasibility with transaction loads from 10 TPS to 2000 transactions/minute, and the results show that the model can be applied in medium-scale smart industrial environments. Finally, the discussion explicitly covers limitations related to verification latency, block scalability, and computational overhead, which contributes to the transparency and scientific rigor of the discussion from a methodological perspective.

Table 9. Network Throughput Comparison of Ablation Configurations

Configuration	Transactions / 5 min	Equivalent TPS
DQN only	612.35	2.0412
BCT only	548.26	1.8275
DQN + BCT without PBFT	641.82	2.1394
Full DQN + BCT + PBFT	782.43	2.6081

Table 9 summarizes the network-throughput results of the ablation configurations using a consistent measurement interval. The DQN-only, BCT-only, DQN + BCT without PBFT, and full DQN + BCT + PBFT configurations achieved 612.35, 548.26, 641.82, and 782.43 transactions per 5 minutes, respectively. These values correspond to equivalent throughput rates of 2.0412, 1.8275, 2.1394, and 2.6081 TPS. The full DQN + BCT + PBFT configuration therefore achieved the highest sustained network throughput among the evaluated ablation variants.

In terms of throughput over time, the proposed hybrid DQN + BCT + PBFT reached 782.43 transactions at the 5-minute mark. The system showed only a 13.4% throughput reduction over the 25-minute evaluation period, demonstrating relatively stable performance under continuous operation. The Proposed hybrid DQN + BCT + PBFT consumed 78.54% CPU utilization and 4234.89 MB of memory for a 2000 transactions/min workload, demonstrating computational feasibility for medium-scale industrial deployments.

The Proposed hybrid DQN + BCT + PBFT achieves 87.43 TPS and a 99.23% transaction success rate with a 256 KB block size. IoT + PKI has a higher processing speed of 92.15 TPS and a lower transaction success rate of 98.76%. Ethereum + NN had a slightly higher success rate of 99.45%, but a significantly lower processing speed and higher response time. Overall, the experimental results show that the proposed hybrid DQN + BCT + PBFT provides a good trade-off among RT, throughput stability, transaction reliability, computational overhead, and decentralized security, rather than optimizing a single performance metric.

5. CONCLUSION AND FUTURE WORK

This research proposed a hybrid DQN + BCT + PBFT to enable secure data transmission and real-time optimization in SIN. The experimental results for the proposed model across multiple performance metrics are presented. For transaction processing, the model achieved a 99.23% success rate at a 256 KB block size and processed 87.43 TPS, averaging 782.43 transactions per 5 *minutes*. NT degradation was 13.4% compared to traditional BCT solutions. CPU usage ranged from 43.67% to 78.54% at 500 to 2000 transactions per minute.

In comparison, the Ethereum + NN had CPU usage of 91.23%. The RT ranged from 98.43 milliseconds at 10 TPS to 197.34 *ms* at 100 TPS. The model has limitations that require further research. RU increased when the block size exceeded 2048 KB. Verification time for cases that require very fast transaction confirmation leaves room for improvement.

Future research should focus primarily on optimizing CM for larger blocks, improving DQN adaptability to unforeseen load changes, and refining verification techniques for high-throughput scenarios.

References:

Admass, W. S., Munaye, Y. Y., & Diro, A. A. (2024). Cyber security: State of the art, challenges and future directions. *Cyber Security and Applications*, *2*, 100031. https://doi.org/10.1016/j.csa.2023.100031

Adewuyi, A., Oladele, A. A., Enyiorji, P. U., Ajayi, O. O., Tsambatare, T. E., Oloke, K., & Abijo, I. (2024). The convergence of cybersecurity, Internet of Things (IoT), and data analytics: Safeguarding smart ecosystems. *World Journal of Advanced Research and Reviews*, *23*(1), 379–394. https://doi.org/10.30574/wjarr.2024.23.1.1993

Baptist, L. J., Selvam, J., & Chakkaravarthy, D. M. (2025). A model for detecting cybersecurity intrusions using machine learning techniques. *International Journal of Electronic Security and Digital Forensics*, *17*(5), 582–592. https://doi.org/10.1504/IJESDF.2025.148217

Baptist, L. J., Selvam, J., & Chakkaravarthy, D. M. (2026). Machine learning-based cyber-attack recognition model. *International Journal of Electronic Security and Digital Forensics*, *18*(1), 56–68. https://doi.org/10.1504/IJESDF.2026.150185

Bharathi Murthy, C. V. N. U., & Lawanya Shri, M. (2026). An adaptive blockchain framework for federated IoMT with reinforcement learning-based consensus and resource forecasting. *Scientific Reports*, *16*(1), Article 8296. https://doi.org/10.1038/s41598-026-35704-1

Castro, M., & Liskov, B. (1999). Practical Byzantine fault tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99),* 173–186. USENIX Association, https://dl.acm.org/doi/10.5555/296806.296824

Chaudhari, A., Deshpande, V., & Midhunchakkaravarthy, D. (2025). A comprehensive review of reinforcement learning-based routing algorithms for wireless sensor networks. *International Journal of Computer Networks and Applications*, *12*(3), 449–474. https://doi.org/10.22247/ijcna/2025/28

Cheekati, S., Borra, C. R., Pareek, P. K., Rayala, R. V., Sri Nandhini Kowsalya, S., & Selvam, J. (2025). Cybersecurity threat detection using OpCyNet and DBRA: A deep learning approach for DDoS attack mitigation on CICDDoS2019. In *13th IEEE International Conference on Smart Grid*, pp. 782–788, https://doi.org/10.1109/ICSMARTGRID66138.2025.11071797

Figueroa, N. T. E., Priya, V. A., Shanmugam, S. K., Kumar, K. V., Sengan, S., & Bolivar, A. M. C. (2024). Adaptive approach to anomaly detection in Internet of Things using autoencoders and dynamic thresholds. *Journal of Machine and Computing*, *4*(1), 1–10. https://doi.org/10.53759/7669/jmc202404001

Huo, R., Zeng, S., Wang, Z., Shang, J., Chen, W., Huang, T., & Liu, Y. (2022). A comprehensive survey on blockchain in industrial Internet of Things: Motivations, research progress, and future challenges. *IEEE Communications Surveys & Tutorials*, *24*(1), 88–122. https://doi.org/10.1109/COMST.2022.3141490

Jyothi, V. E., Rajashree, R., Phani, S., Hugar, L., Shaik, R., & Poddar, S. (2025). Artificial intelligence-powered protocols for enhancing security in next-generation communication networks. In *Proceedings of the 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI 2025)* (pp. 1107–1112). https://doi.org/10.1109/ICoICI65217.2025.11252990

Karn, A. L., Ghanimi, H. M. A., Iyengar, V., Siddiqui, M. S., Alharbi, M. G., Alroobaea, R., Yousef, A., & Sengan, S. (2025). Applying the defense model to strengthen information security with artificial intelligence in computer networks of the financial services sector. *Scientific Reports*, *15*, 30292. https://doi.org/10.1038/s41598-025-15034-4

Karthikeyan, M., Manimegalai, D., & RajaGopal, K. (2024). Firefly algorithm-based WSN-IoT security enhancement with machine learning for intrusion detection. *Scientific Reports*, *14*, 231. https://doi.org/10.1038/s41598-023-50554-x

Khalid, S., Gupta, S. K., Chakkaravarthy, M., Nishad, D., Prakash, D., & Agrawal, A. (2026). Multi-modal AI-enabled UAV network for fog dispersal and runway-visibility enhancement at an international airport. *International Journal of Computational Intelligence Systems*, *19*(1). https://doi.org/10.1007/s44196-025-01149-z

Khan, H. U., Sohail, M., Ali, F., Nazir, S., Ghadi, Y. Y., & Ullah, I. (2023). Prioritizing the multi-criteria features based on comparative approaches for enhancing security of IoT devices. *Physical Communication*, *59*, 102084. https://doi.org/10.1016/j.phycom.2023.102084

Khan, W. Z., Rehman, M. H., Zangoti, H. M., Afzal, M. K., Armi, N., & Salah, K. (2020). Industrial Internet of Things: Recent advances, enabling technologies and open challenges. *Computers & Electrical Engineering*, *81*, 106522. https://doi.org/10.1016/j.compeleceng.2019.106522

Laghari, A. A., Khan, A. A., Ksibi, A., Hajjej, F., Kryvinska, N., Almadhor, A., Mohamed, M. A., & Alsubai, S. (2025). A novel and secure artificial intelligence-enabled zero-trust intrusion detection in industrial internet of things architecture. *Scientific Reports*, *15*(1), Article 26843. https://doi.org/10.1038/s41598-025-11738-9

Lakhan, A., Mohammed, M. A., Abdulkareem, K. H., Deveci, M., Marhoon, H. A., Nedoma, J., & Martinek, R. (2024). A multi-objectives framework for secure blockchain in fog–cloud network of vehicle-to-infrastructure applications. *Knowledge-Based Systems*, *290*, 111576. https://doi.org/10.1016/j.knosys.2024.111576

Lakhan, A., Alyasseri, Z. A. A., Mohammed, M. A., AL-Attar, B., Nedoma, J., Alubady, R., Memon, S., & Martinek, R. (2025). Sustainable secure blockchain-assisted AIoT and green multi-constraints supply chain system. *IEEE Internet of Things Journal*, *12*(19), 39392–39406. https://doi.org/10.1109/JIOT.2025.3548037

Mann, M., Badoni, R. P., Narooka, P., Janarthanan, M., Mahajan, R. A., Samriya, J. K., & Panwar, D. (2025). Spatiotemporal trust evaluation using a neutrosophic model under sensor uncertainty. *Discover Computing*, *28*(1). https://doi.org/10.1007/s10791-025-09750-8

National Institute of Standards and Technology [NIST] (2023). Advanced Encryption Standard (AES), FIPS PUB 197-upd1. National Institute of Standards and Technology. DOI: 10.6028/NISTFIPS.197-upd1.

Natraj, N., Midhunchakkaravarthy, J., & Mishra, B. K. (2025). A quantitative framework for the selection of hybrid consensus mechanisms in blockchain-IoT systems. *EAI Endorsed Transactions on Internet of Things*, *11*. https://doi.org/10.4108/eetiot.10249

Pasdar, A., Koroniotis, N., Keshk, M., Moustafa, N., & Tari, Z. (2025). Cybersecurity solutions and techniques for Internet of Things integration in combat systems. *IEEE Transactions on Sustainable Computing*, *10*(2), 345–365. https://doi.org/10.1109/TSUSC.2024.3443256

Qin, W., Chen, S., & Peng, M. (2020). Recent advances in Industrial Internet: Insights and challenges. *Digital Communications and Networks*, *6*(1), 1–13. https://doi.org/10.1016/j.dcan.2019.07.001

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge, MA: MIT Press.

Venkata Nagarjun, A., & Rajkumar, S. (2025). Quantum deep learning-enhanced Ethereum blockchain for cloud security: Intrusion detection, fraud prevention, and secure data migration. *Scientific Reports*, *15*(1), Article 38711. https://doi.org/10.1038/s41598-025-22408-1

Dinesh Rajassekharan
School of Artificial Intelligence, Computing and Multimedia
Lincoln University College, 47301 Malaysia.
diraja@lincoln.edu.my
ORCID 0000-0001-8933-1292

Firas Tayseer Ayasrah
College of Education, Humanities and Science,
Al Ain University,
Al Ain, UAE
firas.ayasrah@aau.ac.ae
ORCID 0000-0002-6477-000X

S. Maheswari
Department of Electronics and Communication Engineering,
Saveetha School of Engineering,
Saveetha Institute of Medical and Technical Sciences,
Chennai, 602105,
Tamil Nadu, India.
maheswaris.sse@saveetha.com
ORCID 0000-0002-3446-110x

P. Vidyasri
Department of Computer Science and Business Systems,
Panimalar Engineering College,
Varadharajapuram, Chennai, 600123,
Tamil Nadu, India.
pvidyasri@panimalar.ac.in
ORCID 0000-0001-8304-1673

S. Arun Jayakar
Department of Electronics and Instrumentation Engineering, BIT,
Sathyamangalam, 638401,
Erode,
Tamil Nadu, India.
arunjayakars@gmail.com
ORCID 0000-0001-9727-0629

Sudhakar Sengan
Department of Computer Science and Engineering, Erode
Sengunthar Engineering College,
Erode, 638057,
Tamil Nadu, India.
sudhasengan@gmail.com;
sudhakars@esec.ac.in
ORCID 0000-0003-4901-1432

Vijaya Krishna Sonthi
Department of Computer Science and Engineering,
Koneru Lakshmaiah Education Foundation,
Vaddeswaram, 522502,
Andhra Pradesh, India.
vijayakrishna1990@gmail.com
ORCID 0000-0002-2635-4531

M. Ramya
Department of Electronics and Communication Engineering,
Karpagam Academy of Higher Education,
Coimbatore, 641021,
Tamil Nadu, India.
ramyaherish02@gmail.com
ORCID 0009-0001-6610-5388