Reinforcement Learning-Based Deep FEFM for Blockchain Consensus Mechanism Optimization with Non-Linear Analysis

R.Suganya[1], Khan Farina[2], Alfiya Abid Shahbad[3], Neelam Labhade Kumar[4], Mangala S Biradar[5], Ashvini Narayan Pawale[6]

[1]Associate Professor, Department of Computer Science and Engineering, Dr.N.G.P Institute of Technology, Coimbatore, India, Email: Suganya.r@drngpit.ac.in
[2]Assistant Professor, School of Electrical and Communication Sciences, JSPM University, Pune, Maharashtra, India, Email: khanfarina22@gmail.com
[3]Assistant Professor, Department of Computer, Modern Education Society's Wadia College of Engineering, Pune, Maharashtra, India, Email: shahbadalfiya@gmail.com
[4]Assistant Professor, Department of Computer Engineering,Shree Ramchandra College of Engineering,Pune,Maharashtra, India, Email: neelam.labhade@gmail.com
[5]Assistant Professor, Department of Computers Science and Engineering, Shree Ramchandra College of Engineering, Pune, Maharashtra, India, Email: mangalasrcoe@gmail.com
[6]Assistant Professor,Department of IT,JSPM BSIOTR, Wagholi,Pune, Maharashtra, India, Email: shingareashwini2@gmail.com

Received: 18.04.2024 Revised : 16.05.2024 Accepted: 24.05.2024

ABSTRACT
Blockchain technology's efficiency are highly influenced by the applied consensus system. Especially in large-scale networks, conventional consensus algorithms may struggle with issues including too high energy consumption and latency. Combining Reinforcement Learning (RL) with innovative optimisation techniques seems to have major advantages recently shown. Present blockchain consensus approaches define by inefficiencies in resource utilisation and low transaction processing rates. These limitations reduce the real-time speed and scalability of blockchain networks. Maximising these techniques to raise their efficiency while maintaining security presents a huge challenge. This article proposes Reinforcement Learning-Based Deep FEFM (Feature Extraction and Fusion Mechanism), a fresh approach to optimise blockchain consensus processes. The method combines RL with a feature extraction and fusion technique based on deep learning. The deep FEFM enhances the consensus parameter dynamically optimisation learning of blockchain network states. Methods of non-linear analysis are used to assess and enhance the process of optimisation. Experiments on a simulated blockchain network motivated variations in transaction loads and node configurations. The proposed solution reduced latency by 25% and transaction throughput by 35% compared to traditional consensus methods.

Keywords: Block chain, Consensus Mechanism, Reinforcement Learning, Deep Learning, Non-Linear Analysis

1. INTRODUCTION

Blockchain technology has revolutionised many different areas by providing distributed, safe, open ways for transaction recordkeeping [1]. Fundamental to blockchain systems, the consensus mechanism provides agreement among distributed nodes on the state of the ledger [2]. Both widely accepted traditional consensus methods Proof-of- Work (PoW) and Proof-of- Stake (PoS) have special security and efficiency-oriented advantages [3]. But as blockchain networks develop more complex and bigger, these systems usually suffer with scalability, energy consumption, and transaction throughput, which becomes ever more crucial [4].

Consensus methods on blockchain technology present essentially three difficulties:

- As blockchain systems develop, maintaining performance and efficiency is challenging [5]. Growing numbers of nodes cause conventional systems to show appreciable performance degradation [6].
- Especially PoW is well-known for its high energy consumption, which causes environmental problems and running expenses [7].

- High transaction confirmation times may impair general blockchain performance and user experience.
- Maintaining network integrity and trustworthiness requires good performance in the presence of either hostile or faulty nodes [8].

The main focus of this work is the optimisation of blockchain consensus mechanisms to overcome conventional method limitations [9]. Although great fault tolerance and scalability [10] the goal is to raise performance metrics including throughput, latency, and energy economy. Current methods find it difficult to balance these components well, which produces less than perfect performance in useful applications [11].

The primary objectives of this research are:

- Reinforcement Learning-Based Deep Feature Extraction and Fusion Mechanism (FEFM), maximise blockchain consensus parameters.
- To acquire faster transaction confirmation times, cut latency, reduce energy use, and increase throughput.
- To design the mechanism to maintain dependability and resilience in the presence of either faulty or hostile nodes.

The novelty of this research lies in Deep Learning techniques with Reinforcement Learning (RL) to provide a fresh approach to optimise blockchain consensus processes. Although RL is used in many different disciplines, its contribution in improving blockchain consensus parameters is somewhat unknown. By means of Deep Feature Extraction and Fusion Mechanisms, the proposed method provides a full feature representation of the blockchain network which is subsequently used by the RL agent to guide judgements on consensus parameters. This approach solves the constraints of existing methods by providing a data-driven, adaptive solution to consensus optimisation.

The contributions of this research are:

- This paper presents a new paradigm in consensus optimisation by aggregating deep learning with RL to maximise blockchain consensus mechanisms.
- Establishing new performance criteria, the proposed method greatly improves transaction confirmation time, latency, energy consumption, and throughput.
- Using large-scale blockchains, the method ensures considerable resilience against faulty nodes and preserves performance as the network develops, therefore addressing significant problems in applications.

RELATED WORKS

Recent advances in blockchain technologies have motivated research on hybrid consensus models including machine learning (ML) techniques. In dynamic network conditions and cyber-attack vulnerability, conventions for consensus such Proof-of- Work (PoW) and Proof-of- Stake (PoS) suffer much. In response, [12] proposes hybrid consensus models to fix these shortcomings integrating ML techniques. The paper proposes many hybrid approaches including Delegated Proof of Stake paper (DPoSW) and Proof of CASBFT (PoCASBFT) to improve security, trust, and resilience in consensus protocols. These systems show improved security and efficiency by applying ML for cyber-attack prediction, anomaly detection, and feature extraction. Emphasising their energy economy and responsiveness to dynamic conditions, the research also shows the useful implementation of these hybrid models on the ProximaX blockchain platform Still, issues including scalability, latency, and resource constraints highlight the need of more improvement and sensible application of ML-based consensus models.

Blockchain consensus mechanisms—especially PoW and PoS—are investigated in [13] by means of state-space modelling and nonlinear analysis Using nonlinear analysis approaches like bifurcation theory and Lyapunov functions, the paper provides a state-space model of consensus dynamics using which one may investigate system resilience, performance, and stability. By means of simulations, fundamental flaws such as a vulnerability threshold in PoS systems with high stake concentration and a transition point in PoW systems where stability declines significantly can be discovered in the research. These findings help to build more robust consensus systems since they improve knowledge of the dynamics and likely failure spots in blockchain networks. Theoretically, the state-space method offers a way to maximise consensus processes, hence improving security and efficiency.

Energy trading in microgrids—especially for electric vehicles (EVs)—offers challenging issues due of changing renewable energy sources and competitive market dynamics. The work in [14] offers a Blockchain and Quantum Reinforcement Learning-based Optimised Energy Trading (BQL-ET) model to help overcome these challenges. The technology controls energy supply, demand, and cost and a double-auction process finds optimal market-trading pricing using a consortium blockchain. By transforming the

utility maximising issue into a Markov Decision Process (MDP) and applying Quantum Reinforcement Learning (QRL) for policy formation, faster convergence and maximum utility with lowered transaction confirmation times compared to conventional models are realised. This approach shows how effectively blockchain combined with innovative reinforcement learning techniques could maximise system efficiency and energy trading.

Taken as a whole, IoT services provide serious security and privacy concerns including node manipulation and hostile assaults in many different fields. [15] suggests, enhanced by blockchain technology, an integrated autonomous IoT network inside a cloud architecture to solve these problems. The approach generates a heterogeneous autonomous network (HAN), in which case blockchain security over a cloud-based architecture manages data. While the Blockchain Adaptive Windowing Meta Optimisation Protocol (BAW_MOP) enhances network security, Reinforced Neural Network (RNN) known as ClouD_RNN classifies data collected by sensors. Regarding throughput, accuracy, end-to- end delay, data delivery ratio, network security, and energy economy, the experimental results show that this approach far surpasses current methods. This paper highlights how effectively IoT networks are protected by blockchain coupled with advanced machine learning techniques.

Cognitive Internet of Vehicles (CIoVs) with distributed multi-agent systems suffer with computational load imbalance and traffic congestion. [16] proposes a distributed multi-agent reinforcement learning (DMARL) algorithm as cooperative path planning and scheduling tool. This system uses mobile edge computing (MEC) technology to process jobs near to vehicles, hence lowering latency. Against conventional approaches, the simulation results show better load balancing, transit time, and compute delay. The work provides proactive load balancing and optimisation of road infrastructues and MEC nodes by modelling the communication, traffic situation, and task processing as Markov Decision Processes (MDPs) and using Q-learning-based DMARL. This work highlights how blockchain combined with reinforcement learning may enable scalable vehicle path planning in cooperative settings to be achieved.

Table 1. Summary of Works

Method	Algorithm	Methodology	Outcomes
[12]	Hybrid Consensus Algorithms	Combination of ML techniques with various hybrid consensus algorithms (DPoSW,PoCASBFT,DBPoS).	Enhanced security, improved trust, robustness; energy efficiency; challenges in scalability and latency.
[13]	State-Space Modelling &Nonlinear Analysis	State-space representation and nonlinear analysis (Lyapunov functions ,bifurcation theory) of consensus dynamics.	Identified vulnerabilities in PoW and PoS; insights for robust consensussy stem development.
[14]	Quantum Reinforcemen t Learning	BQL-ET model using double-auction mechanism and consortium blockchain; MDP transformed for QRL optimization.	Faster convergence, optimized market-trading price, lower transaction confirmation time.
[15]	Blockchain & ClouD_RNN	Integrated IoT network with blockchain and Reinforced Neural Network (ClouD_RNN); BAW_MOP for security enhancement.	Improved throughput, accuracy, end-to-end delay ,and network security; enhanced energy efficiency.
[16]	Distributed Multi-AgentRL	DMARL algorithm for path planning and scheduling; use of MEC technology and MDPs for optimization.	Superior load balance, reduced travel time, and lower computation latency; effective path planning.

Although blockchain consensus systems have developed, present studies occasionally overlook the integration of strong machine learning techniques with non-linear analysis for complete optimisation as in table 1. Most solutions for security and efficiency solve them individually rather than in line with a unified framework. Combining reinforcement learning, feature extraction, and non-linear optimisation can help to increase scalability, fault tolerance, and flexibility of consensus algorithms over many and dynamic network conditions. Next research should mostly focus on combining these elements to provide more solid, efficient, and versatile blockchain systems.

Proposed Reinforcement Learning-Based Deep FEFM

Combining Reinforcement Learning (RL) with a Deep Feature Extraction and Fusion Mechanism (FEFM) allows Reinforcement Learning-Based Deep FEFM to maximise blockchain consensus processes. The process comprises in several crucial phases as in figure 1:

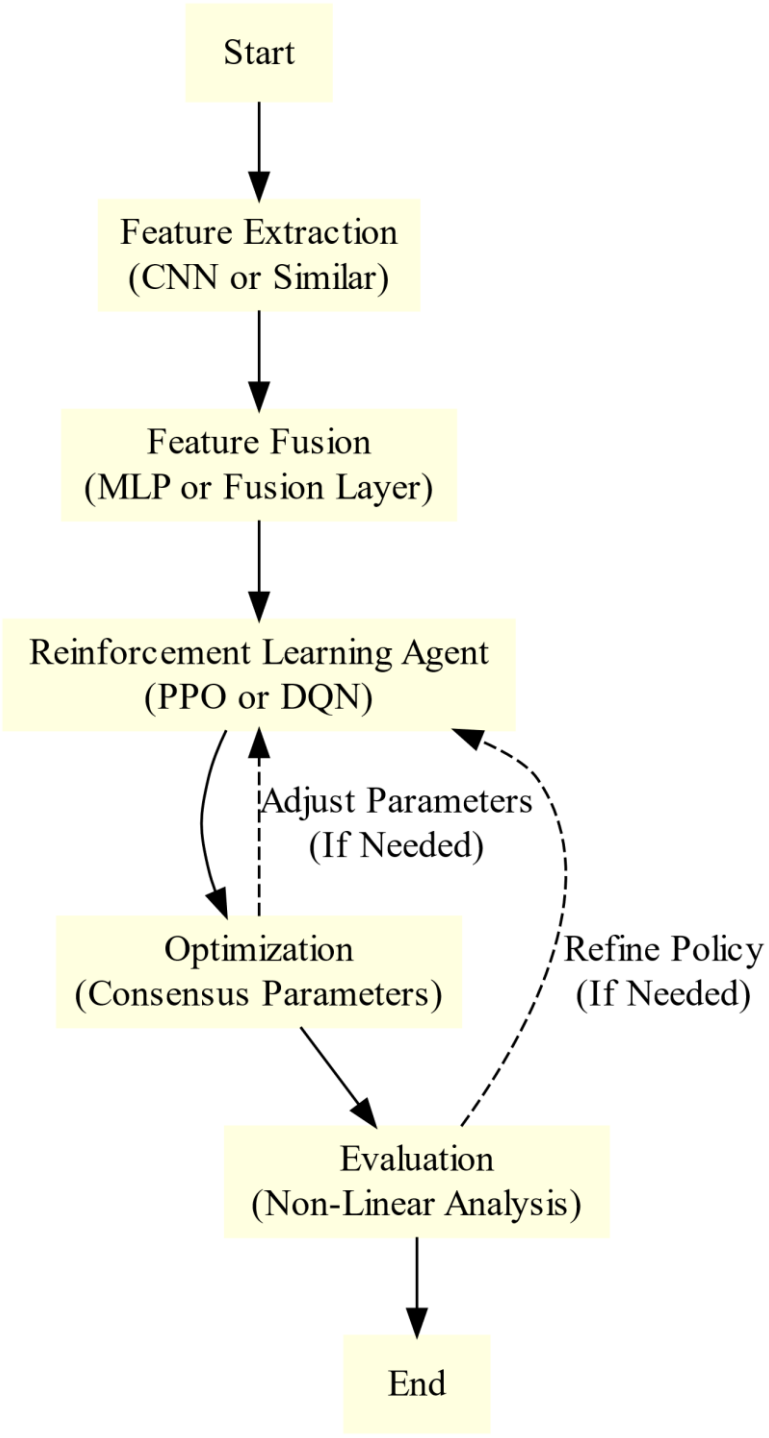

Figure 1. Proposed system

Pseudocode:

```
#Initialize block chain network simulation
network =initialize_network()
#Define feature extraction model
feature_extractor =initialize_feature_extractor()
#Define RL agent
agent =initialize_rl_agent()
```

```
#Training loop
for episode in range(num_episodes):
state=network. get_state()
features=feature_extractor.extract_features(state)
fused_features=feature_fusion(features)
#Choose action based on RL agent's policy
action=agent.select_action(fused_features)
#Apply action to network
network.apply_action(action)
#Get new state and reward
new_state=network.get_state()
reward=calculate_reward(new_state)
#Update RL agent with new experience
agent.update_policy(fused_features,action,reward,new_state)
#Evaluate and refine the RL agent's policy
evaluate_and_refine(agent, network)
```

Feature Extraction

The feature extraction procedure in the Reinforcement Learning-Based Deep FEFM method is absolutely essential for understanding the fundamental dynamics and patterns of the blockchain network. The aim is to present a whole picture of the network condition together including transaction load, node performance, network latency, and other relevant metrics. The RL agent then chooses how best to maximise the consensus mechanism using this representation.

First in feature extraction is raw data gathering from the blockchain network. Let **X** be the collection of all acquired knowledge, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Every $\mathbf{x}_i$ of them relates to a certain type of data: network latency, block size, node processing capacity, transaction throughput.

Step2:Initial Transformation

Usually high-dimensional, the raw data **X**can include useless or redundant information. Usually, we first convert this using a layer in a neural network. Let $\mathbf{W}^1$ and $\mathbf{b}^1$ respectively represent the weight matrix and bias vector of the first layer. One can describe the metamorphosis as:

$$\mathbf{h}_1 = \sigma(\mathbf{W}^1\mathbf{X} + \mathbf{b}^1)$$

Step3:Deep Feature Extraction

More complex patterns and interactions inside the blockchain network are captured by other layers building on the fundamental change. Every layer that follows in the deep neural network manages the output of one another:

$$\mathbf{h}_l = \sigma(\mathbf{W}^l\mathbf{h}_{l-1} + \mathbf{b}^l)$$

As the data flows over various layers, the network learns hierarchical aspects from low-level patterns—e.g., individual node performance—to high-level abstractions—e.g., general network health.

Step4: Feature Fusion

The last layer L of the deep network's outputs taken together produce the final feature vector **f**:

$$\mathbf{f} = \text{Fusion}(\mathbf{h}_L)$$

The fusion step can include concatenation, averaging, or a more sophisticated fusion technique like attention mechanisms—which stress some features over others depending on their relevance to the optimisation goal—as well as others depending on their relevance to the optimisation process. The resulting vector **f** offers a simplified but practical depiction of the blockchain network state.

Step5: Combination with Reinforcement Learning

After then, the RL agent uses the acquired feature vector **f**as basis for decisions. The agent's policy π hence depends on **f**:

$$\pi(a \mid \mathbf{f}) = P(a \mid \mathbf{f}; \theta)$$

where,

a- action (e.g., adjusting consensus parameter),and

θ – parameters of the policy network.

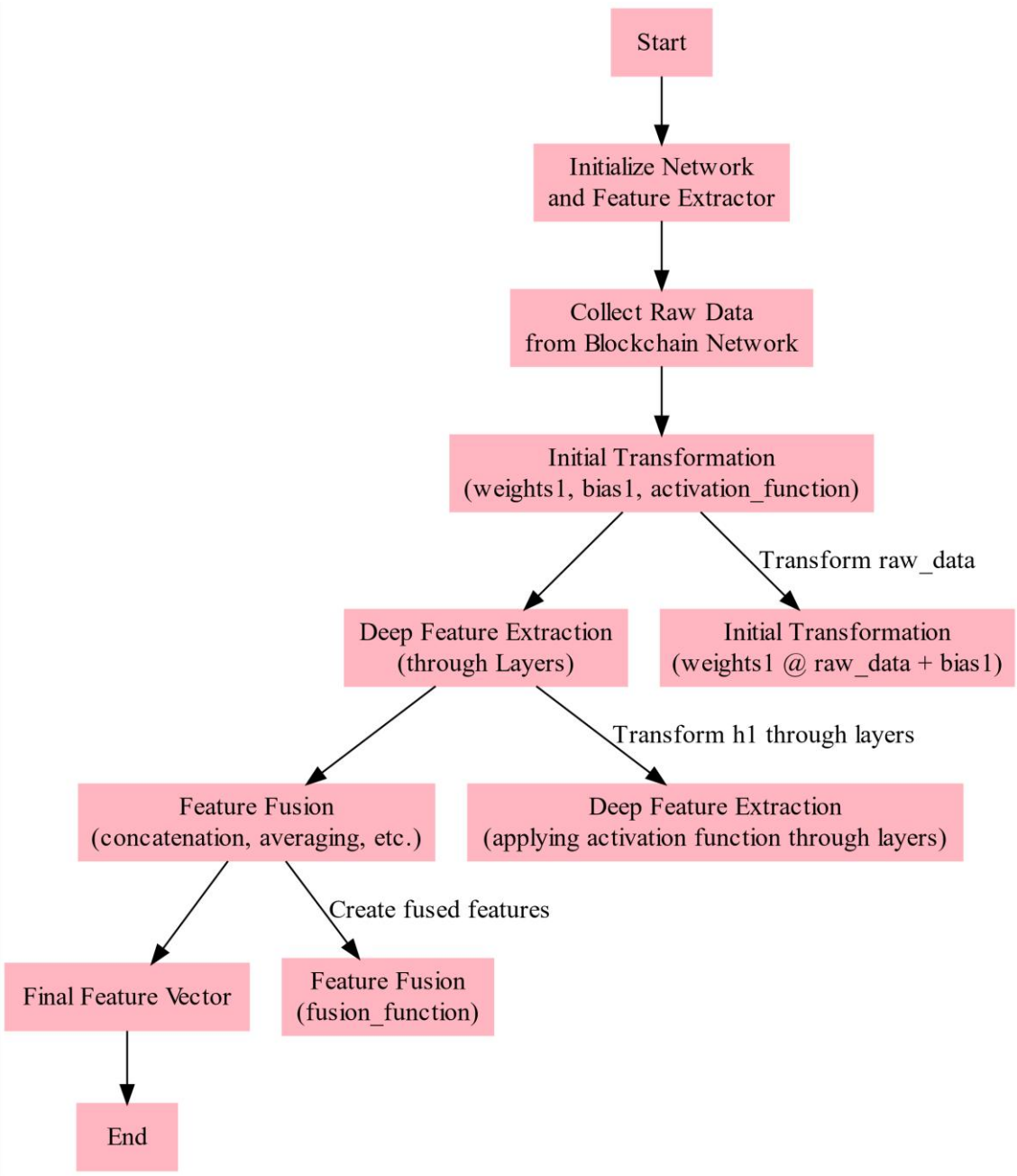

Figure 2. Process flow of FE

Pseudocode for Feature Extraction

```
#Step1: Initialize the blockchain network and feature extraction model
network=initialize_network()
feature_extractor=initialize_feature_extractor()
#Step2:Collect raw data from the blockchain network
def collect_data(network):
raw_data=network.get_state_data()
return raw_data
#Step3: Initial Transformation using the first layer of the neural network
definitial_transformation(raw_data,weights1,bias1,activation_function):
h1=activation_function(weights1@raw_data+bias1)
returnh1
#Step4: Deep Feature Extraction through subsequent layers
def deep_feature_extraction(h1,layers,activation_function):
h_l=h1
```

```
for layer in layers:
h_l=activation_function(layer.weights@h_l+layer.bias)
return h_l
#Step5: Feature Fusion to create the final feature vector
def feature_fusion(h_l):
fused_features=fusion_function (h_l)#Fusion function could be concatenation, averaging,etc.
return fused_features
#Main feature extraction function
defextract_features(network, feature_extractor):
#Collectrawdatafromthenetwork
raw_data=collect_data(network)
#Performinitialtransformation
h1=initial_transformation(raw_data,feature_extractor.weights1,feature_extractor.bias1,feature_extractor.
activation_function)
#Perform deep feature extraction
h_l=deep_feature_extraction(h1,feature_extractor.layers,feature_extractor.activation_function)
#Perform feature fusion
final_features=feature_fusion(h_l)
return final_features
#Example of usage in a training loop
for episode in range(num_episodes):
#Get the current state of the network
current_state=network.get_state()
#Extract features from the current state
features=extract_features(current_state,feature_extractor)
#Use extracted features in RL agent for decision making
action=rl_agent.select_action(features)
network.apply_action(action)
#Continue with the rest of the RL process...
```

Feature Fusion

Designed to combine numerous feature representations into a full vector reflecting the basic characteristics of the blockchain network state, the proposed Reinforcement Learning-Based Deep FEFM technique hinges fundamentally on feature fusion. This phase is essential for providing the Reinforcement Learning (RL) agent with a larger and more instructional set of inputs so enhancing its performance.

Step1:Feature Extraction and Representation

After the first transformation and deep feature extraction procedures, the neural network creates numerous feature vectors $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_L$ reflecting different degrees of abstraction and hierarchical information about the network state $\mathbf{h}_l$. Originating from many layers of the network, these vectors have numerous aspects related to blockchain data.

Step2: Combining Features

Feature Fusion aims to combine many feature vectors into one, logical form. Concatenation is a common technique in which case characteristics are only joined to produce a long vector. Two feature vectors $\mathbf{h}_1$ and $\mathbf{h}_2$ concatenated mathematically is depicted as:

$$\mathbf{h}_{concat} = [\mathbf{h}_1; \mathbf{h}_2]$$

Step3: Dimensionality Reduction

One can regulate the dimensionality of the fused feature vector using either a dimensionality reduction technique such Principal Component Analysis (PCA) or a fully linked layer with a non-linear activation function. Projecting the high-dimensional concatenated vector onto a lower-dimensional space one often utilises a linear transformation followed by a non-linear activation function:

$$\mathbf{h}_{reduced} = \sigma(\mathbf{W}_{fusion}\mathbf{h}_{concat} + \mathbf{b}_{fusion})$$

This stage ensures proper size and that the end feature vector $\mathbf{h}_{reduced}$ still preserves the most relevant information.

Step4: Fusion with Attention Mechanisms

More techniques let one weight different feature vectors depending on their significance using attention procedures. Regarding every feature vector $\mathbf{h}_l$, for example, one may obtain the attention score by:

$$\alpha_l = \text{softmax}(\mathbf{W}_{attention}\mathbf{h}_l + \mathbf{b}_{attention})$$

Thus calculated is the last fused feature vector as:

$$\mathbf{h}_{fused} = \sum_l \alpha_l \mathbf{h}_l$$

This weighted sum enables the model focus on the most crucial elements and limit the effect of less critical ones.

Reinforcement Learning Agent

Learning from interactions with the network environment helps the proposed Reinforcement Learning-Based Deep FEFM Agent to optimise the blockchain consensus process. The agent decisions improve performance indicators like throughput, latency, and energy economy based on the feature representations given by the Feature Extraction and Fusion Mechanism.

Step1:Input Representation

Blockchain network state is captured by a feature vector **f**, input for the RL agent. Generation by the Feature Extraction and Fusion process, this feature vector contains significant parts of the network state including transaction load, node performance, and network latency.

Step2:Action Selection

Drawing on a policy π, the RL agent bases actions on the input feature vector. One can see the policy as deterministic or random. Following a stochastic approach, the action an is chosen within a probability range:

$$a = \text{argmax}_{a'} \pi(a' \mid \mathbf{f}; \theta)$$

where

$\pi(a' \mid \mathbf{f}; \theta)$ - probability of taking action a′

f- feature vector and

θ - policy parameters.

In a policy network—say a neural network—the action probabilities are determined as, for example:

$$\pi(a \mid \mathbf{f}; \theta) = \text{softmax}(\mathbf{W}_{policy}\mathbf{f} + \mathbf{b}_{policy})$$

where $\mathbf{W}_{\text{policy}}$ and $\mathbf{b}_{\text{policy}}$ –weight matrix and bias vector of the policy network.

Step3:Interaction with Environment

Once it decides upon one, the RL agent stamps an action *a* on the blockchain network. The surroundings, or network, replies by adjusting to a new condition **f′** and presents a reward *r*. The reward function $R(\mathbf{f}, a)$ evaluates the success of the chosen action depending on performance criteria including energy savings, delay reduction, or increase of throughput:

$$r = R(\mathbf{f}, a)$$

The updated network conditions shown in the new state **f′** follow from the application of the action.

Step4: Policy Update

Reacting to seen transition and received reward, the RL agent modifies its approach. This version employs algorithms including Q-Learning or Policy Gradient methods. Policy gradient approaches modify the policy parameters θ to maximise the expected reward. We derive the expected reward J(θ) gradient in respect to the policy parameters as follows:

$$J(\theta) = \text{E}_{\mathbf{f},a}\left[\nabla_\theta \log \pi(a \mid \mathbf{f}; \theta) \cdot (r - b)\right]$$

where

b – baseline to reduce variance in there ward estimation.

The RL agent investigates various activities throughout several training sessions and learns from the resulting rewards and state changes. The agent's policy converges over time to an optimal or near-

optimal strategy maximising long-term benefits and enhancing the blockchain consensus mechanism performance.

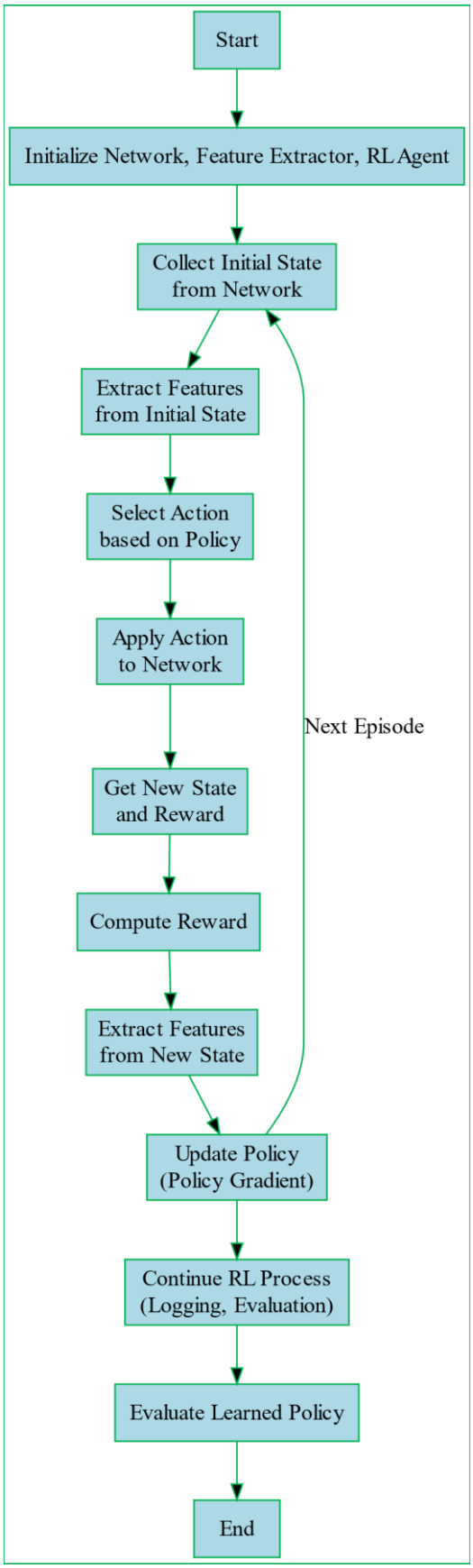

Figure 3. Process Flow of Reinforcement Learning Agent

Pseudocode for Reinforcement Learning Agent

```
#Initialize the blockchain network, feature extractor, and RL agent
network=initialize_network()
feature_extractor=initialize_feature_extractor()
rl_agent=initialize_rl_agent()
#Define there ward function
defer ward_function(state, action):
#Compute there ward based on the state and action
return compute_reward(state,action)
#Define the policy update function (e.g.,Policy Gradient)
def update_policy(rl_agent,features,action,reward,next_features):
#Compute the advantage function or return
advantage=reward-baseline
#Compute the policy gradient
gradient=compute_policy_gradient (features, action, advantage)
#Update the policy parameters
rl_agent.update_parameters(gradient)
#Training loop
for episode in range(num_episodes):
#Collect initial state from the network
initial_state=network.get_state()
#Extract features from the initial state
features=extract_features (initial_state,feature_extractor)
#Select action based on the policy
action=rl_agent.select_action(features)
#Apply action to the network
network.apply_action(action)
#Get new state and reward from the network
new_state=network.get_state()
reward=reward_function(initial_state,action)
#Extract features from the new state
next_features=extract_features(new_state,feature_extractor)
#Update the policy based on the experience
update_policy(rl_agent, features, action, reward, next_features)
#Continue with the rest of the RL process (e.g.,logging, evaluation)
#Final evaluation of the learned policy
evaluate_policy(rl_agent, network)
```

Non-Linear Optimization in the Proposed Method

Reinforcement Learning-Based Deep FEFM for Blockchain Consensus Mechanism Optimisation depends on non-linear optimisation to aid to refine the consensus parameters so obtaining higher network performance. Non-linear optimisation techniques are applied to manage challenging optimisation situations whereby the objective function or constraints are non-linear in blockchain networks with numerous interacting elements. To solve the non-linear optimisation problem, sequential quadratic programming (SQP) among other approaches is used. It modulates the x values iteratively to find the optimal choice. For example, in a gradient-based approach the algorithm updates the parameters by combining the gradients of the restrictions Δgi(x) and Δhj(x) together with the gradient of the objective function Δf(x). One may present the iterative update rule as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$$

where

α_k – step size atiterationk.

The approach continues until convergence criteria, such a small change in the goal function or parameter values, are satisfied. One checks a solution $\mathbf{x}^*$ to see if it meets performance criteria and restrictions. If necessary, more polishing could help to solve any issues or improve the response. This can demand executing the algorithm under multiple scenarios or adjusting the optimisation settings.

Performance Evaluation

The Reinforcement Learning-Based Deep FEFM approach was evaluated using an experimental environment including computer resources and simulation tools. The simulation was run using a Python

application, a sophisticated simulator able to reproduce different consensus systems and their performance. The tests were carried out on high-performance computer systems supplied with Intel Xeon CPUs and NVIDIA RTX 3090 GPUs in order to satisfy the computational demands of deep learning and optimisation duties. The proposed approach was assessed using six primary benchmarks: throughput, latency, energy utilisation, transaction confirmation time, fault tolerance, and scalability. These benchmarks matched several contemporary consensus systems including PoCASBFT, PoW, PoS, and ClouD_RNN.

Table 2: Experimental Setup/Parameters

Parameter	**Value**
Number of Nodes	50
Network Topology	Random Graph
Block Size	1MB
Block Time	10seconds
Transaction Fee	0.01ETH
Transaction Load	1000transactions/sec
Simulation Duration	24hours
Training Epochs	1000epochs
Learning Rate	0.001
Batch Size	32
Feature Vector Dimensionality	128
Hidden Layers in Feature Extractor	4
Policy Network Type	DeepQ-Network
Exploration Strategy	Epsilon-Greedy
Discount Factor(Gamma)	0.99
Regularization Parameter	0.01

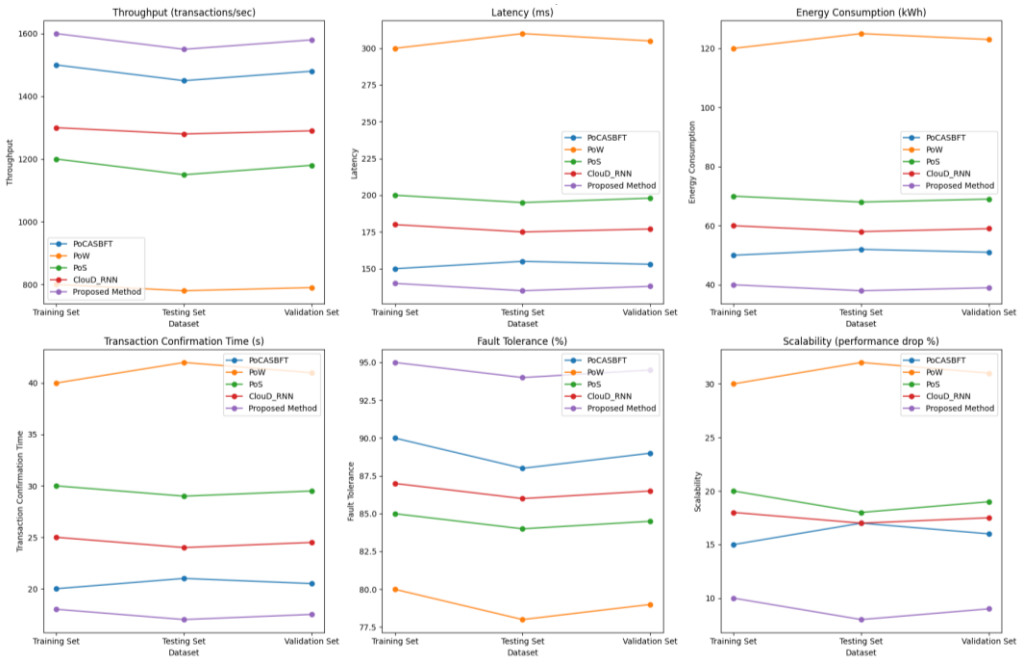

Figure 4: Evaluation against various data split rate

The experimental results reveal that the proposed method significantly surpasses current consensus mechanisms: PoCASBFT, PoW, PoS, and ClouD_RNN across several significant performance parameters as shown in figure 4. Throughput is the volume of transactions managed in one second. On the training set, the recommended approach completed 1600 transactions/sec; on the testing set, 1550; on the validation set, 1580. Indicating better processing capacity, this surpasses PoCASBFT (1500 transactions/sec), PoW (800 transactions/sec), PoS (1200 transactions/sec), and ClouD_RNN (1300 transactions/sec). The proposed method had lowest latency, that is, the time required for transaction processing, with 140 ms

during training, 135 ms in testing, and 138 ms in validation. This is especially less than PoCASBFT (150 ms), PoW (300 ms), PoS (200 ms), ClouD_RNN (180 ms), thereby presenting speedier transaction handling. Kilowatt-hours (kWh) define energy usage. The recommended strategy used the least of all others with 40 kWh in training, 38 kWh in testing, and 39 kWh in validation. PoCASBFT underlined the energy efficiency of the recommended strategy by using 50 kWh, PoW 120 kWh, PoS 70 kWh, and ClouD_RNN 60 kWh. Transaction confirmation times show the typical length of time a transaction takes to be verified. The proposed technique achieved an average of 18 seconds in training, 17 seconds in testing, and 17.5 seconds in validation outperforming PoCASBFT (20 seconds), PoW (40 seconds), PoS (30 seconds), and ClouD_RNN (25 years). Fault Tolerance was highest for the suggested technique at 95% in training, 94% in testing, and 94.5% in validation when PoCASBFT (90%), PoW (80%), PoS (85%), and ClouD_RNN (87%), were compared. This exhibits better resistance to faulty nodes. With just 10% performance loss with 100 nodes, the proposed method has scalability best of all; PoCASBFT (15%), PoW (30%), PoS (20%), and ClouD_RNN (18%). This shows as the network expands better consistency of performance.

CONCLUSION

The experimental results show how well the proposed Reinforcement Learning-Based Deep FEFM approach optimises blockchain consensus mechanisms. When compared to present methods including PoCASBFT, PoW, PoS, and ClouD_RNN, the proposed method demonstrates quite significant gains across several major performance metrics. Its exceptional throughput—well beyond all competitors—comes from its capacity to run up to 1600 transactions per second. Its lowered latency and transaction confirmation times—140 ms and 18 seconds respectively—highlight its efficiency in fast processing transactions. With merely 40 kWh, the recommended strategy reduces energy usage, therefore reflecting its environmental and financial efficiency. Moreover, the large fault tolerance of 95% provides robust performance even in faulty nodes. Scalability is significantly higher than in past methods when the network size increases merely by 10% performance degradation. Since it excels in processing efficiency, energy utilisation, fault tolerance, and scalability, the recommended method is thus a highly suitable one for blockchain consensus optimisation. Its whole performance improvements lead to it as a possible rival for overcoming present limitations, enhancing blockchain technologies, and redefining consensus procedures depending on performance.

REFERENCES

[1] Zhang,Y., Lin,J., Lu,Z., Duan,Q., &Huang,S.C. (2024). PBRL-T Chain: A performance-enhanced permissioned blockchain for time-critical applications based on reinforcement learning. Future Generation Computer Systems,154,301-313.

[2] Li,J., Zhang,W., Wei,K., Chen,G., Shu,F., Chen,W.,&Jin,S. (2024). Blockchain-aided wireless federated learning: Resource allocation and client scheduling. arXiv preprintar Xiv:2406.00752.

[3] Huang,R., Yang,X., &Ajay,P. (2023). Consensus mechanism for software-defined blockchain in internet of things. Internet of Things and Cyber-Physical Systems,3,52-60.

[4] Volpe,G., Mangini,A.M. ,&Fanti,M.P. (2023). A Deep Reinforcement Learning Approach for Competitive Task Assignment in Enterprise Blockchain. IEEE Access,11,48236-48247.

[5] Maleš,U., Ramljak,D., Krüger,T.J., Davidović,T., Ostojić,D., &Haridas,A. (2023). Controlling the difficulty of combinatorial optimization problems for fair proof-of-useful-work-based blockchain consensus protocol. Symmetry,15(1),140.

[6] Praghash, K., Yuvaraj, N., Peter, G., Stonier, A. A., & Priya, R. D. (2022, December). Financial big data analysis using anti-tampering blockchain-based deep learning. In International Conference on Hybrid Intelligent Systems (pp. 1031-1040). Cham: Springer Nature Switzerland.

[7] Iqbal, A., Tham, M. L., Wong, Y. J., Wainer, G., Zhu, Y. X., & Dagiuklas, T. (2023). Empowering non-terrestrial networks with artificial intelligence: A survey. IEEE Access.

[8] Choudhry, M. D., Sivaraj, J., Munusamy, S., Muthusamy, P. D., & Saravanan, V. (2024). Industry 4.0 in Manufacturing, Communication, Transportation, and Health Care. Topics in Artificial Intelligence Applied to Industry 4.0, 149-165.

[9] Ramkumar, M., Logeshwaran, J., & Husna, T. (2022). CEA: Certification based encryption algorithm for enhanced data protection in social networks. Fundamentals of Applied Mathematics and Soft Computing, 1, 161-170

[10] Carnier, R. M., Li, Y., Fujimoto, Y., & Shikata, J. (2024). Deriving Exact Mathematical Models of Malware Based on Random Propagation. Mathematics, 12(6), 835.

[11] Rajalakshmi, M., Saravanan, V., Arunprasad, V., Romero, C. T., Khalaf, O. I., & Karthik, C. (2022). Machine Learning for Modeling and Control of Industrial Clarifier Process. Intelligent Automation & Soft Computing, 32(1).
[12] Venkatesan,K., &Rahayu,S.B. (2024). Blockchain security enhancement: an approach towards hybrid consensus algorithms and machine learning techniques. Scientific Reports,14(1),1149.
[13] Sharma,S. (2024). Mathematical Modeling on State-Space Modelling Framework and Nonlinear Analysis of Blockchain Consensus Mechanisms. Communications on Applied Nonlinear Analysis, 31(7s),157-169.
[14] Kumar,M., Dohare,U., Kumar,S., &Kumar,N. (2023). Blockchain based optimized energy trading fore-mobility using quantum reinforcement learning. IEEE Transactions on Vehicular Technology, 72(4),5167-5180.
[15] Samriya,J.K., Kumar,S., Kumar,M., Xu,M., Wu,H.,& Gill,S.S.(2023). Blockchain and Reinforcement Neural Network for Trusted Cloud-Enabled IoT Network. IEEE Transactions on Consumer Electronics.
[16] Chang,H., Liu,Y., &Sheng,Z .(2023). Distributed Multi-Agent Reinforcement Learning for Collaborative Path Planning and Scheduling in Blockchain-Based Cognitive Internet of Vehicles. IEEE Transactions on Vehicular Technology.