Proof of Learning Applied to Binary Neural Networks

Zoltán-Valentin Gyulai-Nagy[a]
Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

Keywords: Blockchain, Proof of Learning, Binary Neural Networks, Neural Network Quantization.

Abstract: This paper introduces a novel method that leverages binary neural networks (BNNs) for transaction validation on blockchains. Utilizing the computational capabilities of traditional Proof-of-Work systems, this approach generates multiple models suitable for real-world applications. BNNs are chosen for their smaller memory footprint, fitting well into blockchain validations and embedding within blocks. The method aligns with the Proof of Learning concept, requiring neural network training to create new blocks, while also incorporating computationally intensive heuristic approaches. Despite the lower precision of BNNs compared to traditional models, their reduced computational demand during inference is beneficial. The goal is to improve their precision through multiple training rounds and the use of evolutionary algorithms. This scalable approach can be customized to meet diverse application needs by allowing users to upload datasets for training specific models. Additionally, it is cost-effective as BNNs can be trained on low-cost devices, broadening access. This strategy aims to refine blockchain validation processes and produce usable models as a byproduct.

1 INTRODUCTION

Artificial intelligence's recent popularity gave birth to many new ideas and opportunities for us to enhance the average user experience around many applications. Much speculation exists in different domains about whether AI will replace the usual human workforce (Huang and Rust, 2018). All of these questions and developments point to the fact that we have started evolving into a new phase of machine learning where the general public is aware of its force and will try to embed it as much as possible in many mundane scenarios (Jarrahi, 2018). This will aid existing workflows to reduce the routine and repetitive tasks carried out by manual labor.

In this paper, we propose an approach that modifies the traditional principles of Proof of Work (PoW) (Gervais et al., 2016) blockchains and incorporates the concepts of Proof of Learning (PoL) (Jia et al., 2021) while changing and adapting the validation flow of the blocks to support additional training and inference algorithms. Our approach, which incorporates Binary neural networks (BNNs) instead of conventional neural network (NN) models, also benefits from generating lightweight models that can be seamlessly embedded into a blockchain. This approach also guarantees ownership of the models.

[a] https://orcid.org/0000-0001-6506-4679

Our contribution builds on previous methods (Liu et al., 2021) by introducing a new technique that:

- Stores BNN models within blocks and uses these models for block validation.
- Uses evolutionary algorithms to improve models, even when they have reached their relative best shape and performance

We will evaluate the proposed method to determine its practicality, focusing on three key aspects: the impact on network bandwidth usage, the throughput of the chain, and its ability to validate blocks and adjust the validation difficulty. These assessments will help us gauge the viability and efficacy of our approach.

This paper is organized as follows. Section 2 presents the existing knowledge in our research domain. Section 3 then formulates the specific research questions we aim to address. Our methodology, including a detailed description of our empirical approach, is outlined in Section 4. Section 5 showcases the results of our test scenarios, which were designed to support our research questions. Finally, Section 6 concludes the paper by summarizing our findings and provides a few directions for future research.

Gyulai-Nagy, Z.-V.
Proof of Learning Applied to Binary Neural Networks.
DOI: 10.5220/0013145800003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 411-418
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

2 LITERATURE REVIEW

Blockchain technology and neural networks have been focal areas of research in recent years, particularly in how they can be optimized and integrated into various applications, including transaction validations.

BNNs, which use binary values (-1 or 1) for weights and activations, reduce both memory footprint and computational complexity (Courbariaux and Bengio, 2016) , making them ideal for resource-limited devices like IoT and mobile phones (Lin et al., 2017). This efficiency is crucial for blockchain operations in decentralized environments.

Traditional PoW mechanisms are secure but consume significant energy (Satoshi, 2008). Research has explored methods to minimize resource usage, supported by investigations into alternative consensus mechanisms that lower computational demands (Micali et al., 1999). The concept of PoL, a derivative of the PoW concept, involves training a neural network as part of the validation process. (Jia et al., 2021). While existing approaches already create neural network model side products (Liu et al., 2021), our work enhances these concepts by incorporating BNNs which are more resource efficient (Mani et al., 2022).

The challenges and potential solutions in blockchain scalability have been highlighted, emphasizing the need for new approaches (Xie et al., 2019). BNNs can contribute to blockchain scalability and can be customized for various applications. Their ability to train on low-cost devices like Raspberry Pi provides further flexibility (Wang et al., 2023).

Using BNNs, which are trainable on inexpensive devices and operable on low-cost FPGAs, could reduce financial barriers to blockchain adoption (Beck et al., 2016) . Despite their lower accuracy, iterative improvements and FPGA devices can enhance validation efficiency and overall performance.

3 PROBLEM STATEMENT AND RESEARCH QUESTIONS

Blockchain technology has revolutionized the way transactions are processed and validated, providing a decentralized and secure framework for various applications. However, the traditional consensus mechanisms used in blockchain networks, such as PoW (Satoshi, 2008), suffer from several limitations. These include high computational complexity, energy consumption, and scalability issues, which might stale the widespread adoption and sustainability of blockchain technology (Beck et al., 2016).

The PoW consensus mechanism requires nodes to solve complex mathematical puzzles to validate transactions and create new blocks, leading to significant computational overhead and energy consumption (O'Dwyer and Malone, 2014). As the blockchain network grows, the computational requirements for PoW increase, resulting in longer transaction confirmation times and limited scalability (Croman et al., 2016). Moreover, the high energy consumption associated with PoW has raised concerns about the environmental impact of blockchain technology (de Vries, 2018).

To tackle existing challenges in blockchain consensus mechanisms, BNNs offer an efficient solution due to their lower computational complexity and memory needs compared to traditional neural networks (Courbariaux and Bengio, 2016). Integrating BNNs into a PoL consensus mechanism could address the shortcomings of PoW, enabling more scalable and energy-efficient transaction validation.

This paper discusses a framework for repurposing blockchains' processing power. We assume the reader is familiar with basic concepts related to neural networks and blockchains. We focus only on the methodology of generating useful artifacts for the real world and providing the key to linking blocks in the blockchain as a full solution would require larger documentation.

However, the application of BNNs in the context of blockchain consensus mechanisms is still in its early stages, and several research questions need to be addressed to fully realize the potential of this approach. The following research questions guided the investigation into the application of BNNs for blockchain transaction validation:

RQ1. How can BNNs be effectively integrated into the blockchain consensus mechanism to enable efficient and secure transaction validation?

RQ2. What are the optimal architectures and training strategies for BNNs in the context of blockchain transaction validation?

RQ3. Can the proposed approach maintain the possibility of increasing the difficulty of generating new blocks?

RQ4. What is the bandwidth consumption while also encoding neural networks in the blockchain?

RQ5. What is the throughput of the chain, when compared to a highly used blockchain?

By addressing these research questions, this study aims to provide a comprehensive understanding of BNNs application in blockchain transaction validation and contribute to the development of efficient, scalable, and sustainable consensus mechanisms for blockchain networks.

4 METHODOLOGY

Our proposed modification to the traditional POW framework involves two main algorithm steps: the validation step and conflict resolution with other nodes. They can include BNNs in the workflow to have models as a side product, but we also need to change what data a block contains.

A blockchain has the properties of being a continuous flow of blocks that are linked together. In our implementation, this single feature will not be enough; we must have two main features to trace other blocks. One is the previous block (`previous_hash`), and the second is a block mined in the past that uses the same BNN model as in the current block. This is done by including the hash of the previous block (`previous_model_hash`) in the current block to indicate the continuity.

These fields are essential for the blockchain's operation, maintaining integrity, and facilitating advanced features like machine learning integration. Further details on the computation and utilization of these fields will be discussed in subsequent sections. Additionally, constants defining each miner node's parameters are updated throughout the blockchain's lifecycle to ensure fair model distribution and mining difficulty.

In the following sections we are going to provide further information about the purpose and role of these constants.

4.1 Tracking Model Differences

Model shape-related differences are crucial in determining when to generate new model shapes, assessing the number of weights per layer, connection types, and total layers. While adding a single neuron to a new model and reusing weights from a previous generation might seem efficient to avoid recomputation, it could risk the blockchain's integrity. Ensuring significant model differences is essential to prevent reusing neuron weights across model families.

Another metric, the values associated with the model's weights and biases, also plays a critical role. Modifying a single layer by reusing values from a previous model might improve accuracy, but to prevent cyclical changes and ensure model integrity, it's necessary to track modifications. Assigning additional values to each binary weight to indicate if both possible values (-1 and 1) have been used can help compare model generations and enforce a minimum difference threshold. This ensures that any new model with improved accuracy results from genuine computation rather than superficial changes.

These metrics also regulate blockchain difficulty, adapting to hardware advancements or increased network participation. By enforcing dataset and model size constraints, and ensuring significant differences between models, the blockchain maintains its stability and security.

4.2 Training with Evolutive Algorithms

BNNs, with their two-value quantized neuron weights, are designed for simplicity and reduced computational demands, making them ideal for resource-limited settings. However, this quantization can lead to overfitting and reduced accuracy. To counteract this, we employ evolutionary algorithms to optimize BNNs by evolving multiple generations of models, enhancing their accuracy and generalizability.

BNNs can be trained initially using conventional backpropagation methods. This process forms the initial population of models. However, to enhance the accuracy and generalizability of the models, we use evolutive algorithms that iteratively refine the population. Evolutive algorithms are particularly useful in mitigating overfitting, as they can explore a broader range of model configurations and select the best-performing ones. The core principles of evolutive algorithms include crossover and mutation. These concepts are implemented to generate new models from the existing population. Crossover involves selecting a percentage of weights from each layer of two parent models and interchanging them to create new offspring models. This process allows the combination of features from different models, potentially leading to better performance. The percentage of weights selected for crossover is defined by the variable `population_cross_percent`. Nodes across the network use this variable to determine the proportion of weights to be exchanged between models.

Mutation introduces variability by randomly altering a percentage of weights within a model. This helps maintain diversity in the population and prevents premature convergence to suboptimal solutions. The percentage of weights subject to mutation is controlled by the variable `population_mutate_percent`. This ensures that each model undergoes a predefined level of mutation, fostering exploration of the solution space.

We also apply evolutive algorithms over generating the shape of models. It is similar in implementation to the previous algorithm, but besides the crossover and mutation percentages, we use an additional variables to manage the generation of new models. It defines the maximum number of neuron connections a model can have. This constraint

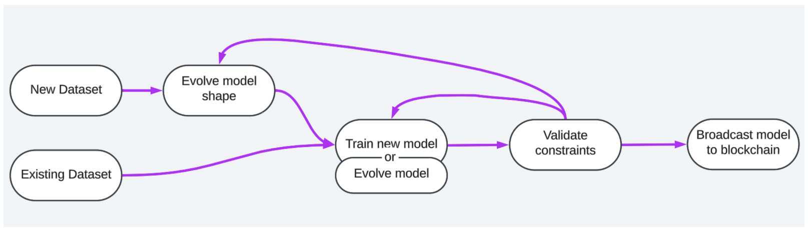

Figure 1: Proposed block creation flow.

is vital for maintaining a manageable model size and computational feasibility. A predefined variable, `max_connection_count`, sets the upper limit on the number of connections in a model, ensuring that models remain within reasonable complexity limits.

When a model becomes overfitted, it may perform well on the training data but poorly on unseen data, leading to a loss of accuracy. Evolutive algorithms address this issue by generating and evaluating multiple iterations of models. Through crossover and mutation, the algorithm explores various configurations, identifying models with better generalizability and improved accuracy.

This optimization helps prevent overfitting by producing models that perform well on unseen data, though it requires substantial computational resources. The distributed nature of blockchain allows these resources to be shared across miner nodes, spreading the computational load.

4.3 Electing a Model to Train

In blockchain contexts involving BNNs, it's crucial to establish rules for training and integrating these networks into blocks incrementally, ensuring continuous availability of diverse models for efficient computation across globally linked miner nodes (Fiszelew A. and R, 2003). The process begins when an end user uploads a dataset, which is then distributed among nodes, allowing miners to train and validate new models based on these datasets.

As fresh datasets are introduced, new neural network models are generated using evolutionary algorithms, which select and train novel model shapes not yet present in the blockchain. This process is regulated by a `min_families` constant, determining the number of model variations allowed in the network. Following this, models may also be trained using conventional backpropagation, with network-wide constants `min_population` and `min_family_acc` setting thresholds for the number and minimum accuracy of these models. All models must exhibit sufficient differences, tracked by `min_family_diff`.

Further optimization of these models is achieved through evolutionary algorithms, using the best models from each family as a base for generating new iterations. Acceptance of new models into the network is contingent on meeting `min_population_acc` and `min_population_diff` criteria, ensuring that only the most effective models are retained and built upon.

Figure 1 illustrates how generations are constructed on the blockchain.

4.4 Block Placement

In the previous subsection, we defined how models can be generated to satisfy the requirement of continuous mining operations. Every block will be placed in a predefined order. With many data sources, some models might be easier to compute than others. There is a need to ensure fairness in the distribution of the models to ensure that no dataset has an unfair advantage and ends up with many trained models while another one won't have none.

We could state that with the structure of a block pointing to a previous family member of a model while also pointing to the previous block, we could simultaneously generate multiple blockchains, but that is not desired as there could be too many chains to keep track of. There is a need for a scheduling mechanism that will prioritize the model generations based on how many models they already have in the blockchain.

The counter inside the `Model` type named `model_count` will provide information about the number of models inside of families. It will be incremented when a new model is added. The arrangement of the blocks will depend on this counter to be equal relative to the date when a dataset was added. In case there is a dataset with two families at the 5th generation on the blockchain and a new dataset is added, this would mean that the first two families will have in the counter 5, and the newly generated family from the new dataset will have 1 on the counter. For the next generation, all of the families will need to be present, with the increased counters being 6, 6, and 2.

The variable `max_family_generations` will track the number of models inside a family and limit the number of models available. When the maximum is reached, a new model schema will be generated with a new family, prioritizing datasets with fewer families.

4.5 Block Validation

Block validation is crucial in blockchain consensus mechanisms, ensuring network integrity and security, especially when integrating BNNs. This process tracks and reviews all previous models and changes, bolstering the network against attacks and manipulations.

Validation begins by checking a proposed block and its model against predefined criteria, such as sufficient computational changes from previous models to deter superficial or reused modifications. This includes verifying the integrity of model weights and biases, ensuring all changes are legitimate, and preventing cyclical value changes by tracking historical modifications to each weight. The `min_population_diff` variable is utilized to confirm these necessary differences.

Accuracy validation is also critical, guided by the `min_population_acc` variable, which sets the minimum accuracy threshold for model acceptance. If a model's evaluated accuracy meets or exceeds this threshold, the block is validated, ensuring only genuinely improved models are added to the blockchain.

The distribution of `min_population_acc` and `min_population_diff` among miners standardizes validation criteria across the network, enhancing decentralized security. Additional checks validate the block against established constraints for model families, populations, and generations, ensuring consistency with previous blocks.

Block validation also includes transaction verification, ensuring all transactions within a block are authentic, not duplicated, and comply with network rules. This encompasses checking transaction signatures, sender balances, and adherence to network constraints. Overall, block validation is central to maintaining the decentralization and trustlessness of the blockchain network, allowing each node to independently validate blocks and models without a central authority, preserving network integrity and immutability.

4.6 Conflict Resolution

In blockchain networks, conflicts can occur when multiple nodes simultaneously accept different blocks. To maintain network integrity and consistency, and to avoid forks, effective conflict resolution mechanisms are essential.

Blockchain protocols typically employ the "*longest chain rule*" to resolve conflicts, where nodes preferentially extend the chain with the most accumulated blocks, assuming it represents the most work done (PoW). However, in complex systems, especially those using BNNs for transaction validation, additional strategies may be necessary:

- *Block Propagation and Delay Mechanisms*: Nodes aim to rapidly propagate new blocks across the network. Implementing slight delays before a block is added can allow time for resolving any emerging conflicts.
- *Block Propagation and Delay Mechanisms*: Nodes aim to rapidly propagate new blocks across the network. Implementing slight delays before a block is added can allow time for resolving any emerging conflicts.
- *Penalization and Incentives*: To encourage nodes to contribute positively to network stability, those causing conflicts may face penalties, whereas those aiding in conflict resolution could receive incentives.
- *Local Block Consensus (LBC)*: Nodes achieve a localized consensus on the blockchain's state before broad propagation, reducing the likelihood of conflicts by aligning nodes' perspectives.

These strategies help blockchain networks efficiently manage conflicts, ensuring a unified, consistent blockchain version and maintaining the trust, security, and reliability of the system, particularly in advanced setups like those incorporating BNNs.

4.7 General Users

To better understand the operation of the proposed blockchain system, we identify three primary actors: end users, model trainers, and miners. End users perform peer-to-peer currency transactions, model trainers use the network's processing power to train BNN models in exchange for blockchain coins, and miners maintain network functionality by hosting nodes and validating blocks, earning rewards for their contributions.

The process begins when a user uploads a dataset or initiates a transaction. This dataset is used to generate models that help create blockchain blocks. Although blocks are secured via hashing to prevent data alteration, the byproducts that are pre-trained BNN models, are publicly available and useful.

We retain many concepts from traditional PoW blockchains but modify the validation and block generation steps to repurpose the blockchain's processing power for producing meaningful, real-world applicable products. This includes enhancements to the consensus mechanism, although a detailed exploration of these improvements is beyond the scope of this summary.

5 RESULTS AND DISCUSSION

In this section, we present the results from several test scenarios designed to evaluate the performance of our proposed blockchain system integrating BNNs. The tests focus on network communication speed, mining efficiency across different difficulty levels, conflict resolution speed, and the relationship between difficulty and transaction handling capacity.

To facilitate the testing of the proposed approach, we developed a small application written in Python that serves as a base. It implements the base API requests required for a miner node and can train and evaluate models using the previously mentioned methodology. To facilitate the communication between nodes, the simplist Flusk framework was used implementing a few endpoints.

When enough transactions are received by the nodes, the mining is automatically triggered, generating a new model family. In our implementation, we used Larq to simulate the training and usage of BNNs. It provides a wrapper over Tensorflow and can visualize and store models in their binary format.

We also implemented the evolutive algorithms to generate model schemas and new generations for the same family using the tflite file format provided by the Larq framework.

The following tests and results are carried out on a simple dockerized framework where multiple nodes were spanned to simulate how a real-world blockchain would work using the MNIST dataset (Lecun, 1998). The purpose is to validate the feasibility of the approach; real world tests might need to take into consideration a higher latency due to having different types of network providers with different network speed and specs in play.

5.1 Network Communication Speed

The first test case assesses the speed of network communications from node to node when transmitting new blocks. In these tests, no transaction content was used to provide insight into the requirements of the blockchain network. We measured the average latency and bandwidth usage across local network configurations using convolutional neural network (CNN) models with three convolutional layers and multiple parameter configurations. The models were encoded to use the optimized binary `tflite` (optimized TensorFlow file format) version. The results are summarized in Table 1.

Table 1: Network Communication Speeds.

Configuration	**Bandwidth Usage** (KB/block)
BNN 4,4K params	1.41
BNN 93.6K params	26.38
BNN 1.17M params	304.26

It can be seen that as the number of BNN parameters increases, the network bandwidth usage also increases. Our approach simply involves storing the resulting model directly on the block therefore the bandwidth is directly correlated to the size of the model. Compared to the memory footprint of other neural network models, ours is measured in kilobytes, while other models with a similar number of parameters typically start at a few megabytes (Pisarchyk and Lee, 2020). This means that when blocks are transmitted between nodes, the required bandwidth is reduced by using BNNs, allowing more space for transactions. Based on the research of (Singh and Vardhan, 2020), the optimal block size is 255 kilobytes, so models with only 100k parameters can easily fit onto blocks without taking up even one fifth of the space allocated for transactions resulting in more than 150 transactions to be stored along it.

5.2 Mining Efficiency at Different Difficulty Levels

The second test case examines the speed of mining a new block at varying difficulty levels. The difficulty level affects the computational effort required to mine a block, impacting the overall efficiency of the blockchain. We tested three difficulty levels: low, medium, and high, recording the average time taken to mine a block at each level.

Our test implementation keeps track of the mentioned minimum difference and accuracy variables when comparing previous models. For this test, we used a laptop with an M1 Pro processor. We trained five models with normal backpropagation, and afterward, the other models were optimized only with the evolutive algorithm. We used a multithreading of 8 threads, 10 models as population, mutation probability of 0.05 for 10 percent of the model schema, and cross probability of 10 percent. The MINST Handwritten digits dataset was used, and the model family we used had 4,440 parameters. For the accuracy evaluation of the new models, we used the non-quantified model instead of the binary file version, as the test script was written in Python. Our aim for this test is to showcase that the difficulty can be increased.

The difficulty was given by the difference and accuracy factors compared to previous generations. The results are shown in Table 3.

Table 2: Mining Efficiency at Different Difficulty Levels.

Difficulty Level	Average Time to Mine (seconds)
Low (diff +0.5%, acc +1%)	106.39
Medium (diff +1%, acc +2%)	433.08
High (diff +1.3%, acc +5%)	1737.16

As expected, higher difficulty levels significantly increase the time required to mine a block. These results highlight the need for a balanced difficulty adjustment mechanism to maintain a steady block generation rate, which is crucial for network stability.

These test scenarios provide an understanding of the performance characteristics of our proposed blockchain system. Future work will involve further optimization and empirical testing to refine these mechanisms and ensure their robustness in real-world applications.

Our approach offers more flexibility in setting block mining difficulty and, consequently, the time required to mine a block. This becomes evident when compared to Bitcoin's average block mining time of 10 minutes (Mariem et al., 2020) and Ethereum's 12.05 seconds (Rouhani and Deters, 2017). It's worth noting that Ethereum, being a Proof of Stake blockchain, naturally operates faster than traditional blockchains, which rely on computational power to create blocks. Our method allows for customizable difficulty settings, potentially bridging the gap between these different blockchain types.

5.3 Comparative Analysis

One key metric to consider is throughput, which allows for comparison with other blockchains and their performance. It provides information about the number of requests that can be processed within a specific time-frame. Previous research (Schäffer et al., 2019) has dived deeper into a few methodologies that provide meaningful comparison values for blockchains.

Obviously, quite a few values need to be taken into consideration when trying to benchmark blockchains. One of them is the difficulty, which aims to increase the processing time required to mine a new block. Its value is increased or decreased based on the total available processing power of the network to ensure the security of the chain.

To provide a meaningful comparison, we illustrate in Table 3 the throughput of the Ethereum blockchain and ours. In the tests, we used docker containers containing one worker node. Afterward, we sent 300 transactions to them, and we waited to see what time was required to compute each block. The test was run on a MacBook Pro with an M1 Pro CPU. Benchmarking for the Ethereum chain was done via go-ethereum and Caliper (cal, 2021). For both chains, the number of transactions configured to be stored in each block is set to 146, which is the default value for Geth.

Table 3: Local Blockchain Throughput at Different Difficulty Levels.

Local Blockchain	Type	Difficulty	Avg. Throughput (TPS)
BNN Chain	PoL	Low	1.47
BNN Chain	PoL	Medium	0.43
Ethereum (GEth)	PoA	Auto	26.30
Ethereum (GEth)	PoW	Auto	12.70

It can be seen from the table that our blockchain has a lower throughput for the previously mentioned difficulty configurations. It can support less transactions in a single node configuration when compared to Ethereum. The difficulty can be set to be even smaller than in our "Low" configuration, although it would not make sense as checking the accuracy increase can occur too randomly.

According to (Fan et al., 2020), the throughput of a blockchain improves with the utilization of multiple nodes, a principle that holds true for our chain as well, since blocks can be mined concurrently. In summary, this implementation is intended to leverage significant computational power, making its advantage of having usable BNNs after mining each block more apparent when deployed on a large scale rather than for individual use, as demonstrated in the tests.

6 CONCLUSIONS AND FUTURE WORK

In conclusion, our approach presents a viable method for generating neural network models as a byproduct of encoding blocks within a blockchain. The integration of BNNs into the blockchain framework not only enhances transaction validation but also offers a novel way to utilize computational resources more efficiently.

Our experimental results demonstrate that the proposed system can handle various aspects of blockchain operations, such as network communication, block mining, and transaction throughput, effectively. These findings indicate the potential of our approach to improve the scalability and efficiency of blockchain networks while simultaneously generating valuable machine learning models.

Our approach addressed RQ1 by integrating BNN training into block generation, storing lightweight models on the block, and linking to previous models and blocks for consistency. Validation steps ensure miners meet required accuracy and model differences, confirming computational effort.

For RQ2, evolutionary algorithms emerged as optimal for training BNNs, preventing overfitting and avoiding local minima. These algorithms also aid in shaping models, increasing diversity and suitability for specific datasets.

Regarding RQ3, our tests indicate that altering miner constants to define minimum model differences and accuracy can increase the computational difficulty of generating new blocks. While BNNs store binary weight values to save bandwidth, our findings for RQ4 show that blocks with empty transactions use less than 1MB of bandwidth.

Lastly, for RQ5, although our system demonstrates lower throughput compared to Ethereum, it still efficiently handles a significant number of transactions. This makes it suitable for large-scale applications, despite longer block creation times.

Future research should explore several avenues to deepen understanding. While our experiments used controlled, dockerized environments, assessing model performance in real-world blockchain networks is crucial for a realistic evaluation of resilience and applicability. Additionally, the concept of distributed model training, where mining pools collaboratively train different model segments, could enhance creation efficiency and leverage blockchain's decentralized nature.

Extending our approach to larger-scale blockchain networks would also be an important step. This involves ensuring that the system can handle increased transaction volumes and more extensive network participation without compromising performance. By addressing these future directions, we plan to enhance the practical utility and scalability of our approach, making it a valuable contribution to the intersection of blockchain technology and machine learning.

REFERENCES

(2021). Hyperledger caliper [online]. Available: https://hyperledger.github.io/caliper/.

Beck, R., Czepluch, J. S., et al. (2016). Blockchain - the Gateway to Trust-Free Cryptographic Transactions. In *ECIS*, page Research Paper 153.

Courbariaux, M. and Bengio, Y. (2016). Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830.

Croman, K. et al. (2016). On Scaling Decentralized Blockchains - (A Position Paper). In *FC 2016*, volume 9604 of *LNCS*, pages 106–125. Springer.

de Vries, A. (2018). Bitcoin's Growing Energy Problem. *Joule*, 2:801–805.

Fan, C., Ghaemi, S., Khazaei, H., and Musilek, P. (2020). Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, 8:126927–126950.

Fiszelew A., Britos P. V., P. G. and R, G.-M. (2003). Automatic generation of neural networks. *Revista Eletrônica de Sistemas de Informação*, 2:1–6.

Gervais, A. et al. (2016). On the Security and Performance of Proof of Work Blockchains. In *CCS 2016*, page 3–16.

Huang, M.-H. and Rust, R. T. (2018). Artificial intelligence in service. *Journal of Service Research*, 21(2):155–172.

Jarrahi, M. H. (2018). Artificial intelligence and the future of work: Human-ai symbiosis in organizational decision making. *Business Horizons*, 61(4):577–586.

Jia, H., Yaghini, M., et al. (2021). Proof-of-Learning: Definitions and Practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1039–1056.

Lecun, Y. (1998). The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

Lin, X., Zhao, C., and Pan, W. (2017). Towards accurate binary convolutional neural network. *CoRR*, abs/1711.11294.

Liu, Y. et al. (2021). Proof of Learning (PoLe): Empowering neural network training with consensus building on blockchains. *Computer Networks*, 201:108594.

Mani, V. et al. (2022). Performance comparison of CNN, QNN and BNN deep neural networks for real-time object detection using ZYNQ FPGA node. *Microelectronics Journal*, 119:105319.

Mariem, S. B., Casas, P., et al. (2020). All that Glitters is not Bitcoin – Unveiling the Centralized Nature of the BTC (IP) Network. In *NOMS 2020*, pages 1–9.

Micali, S., Rabin, M., and Vadhan, S. (1999). Verifiable random functions. In *FOCS'99*, pages 120–130.

O'Dwyer, K. J. and Malone, D. (2014). Bitcoin mining and its energy footprint. In *ISSC 2014/CIICT 2014)*, pages 280–285.

Pisarchyk, Y. and Lee, J. (2020). Efficient memory management for deep neural net inference.

Rouhani, S. and Deters, R. (2017). Performance analysis of ethereum transactions in private blockchain. In *ICSESS 2017*, pages 70–74.

Satoshi, N. (2008). Bitcoin: A peer-to-peer electronic cash system.

Schäffer, M. et al. (2019). Performance and scalability of private ethereum blockchains. In *BPM 2019*, pages 103–118. Springer.

Singh, N. and Vardhan, M. (2020). Computing optimal block size for blockchain based applications with contradictory objectives. *Procedia Computer Science*, 171:1389–1398. Third International Conference on Computing and Network Communications (CoCoNet'19).

Wang, E., Davis, J. J., et al. (2023). Enabling Binary Neural Network Training on the Edge. *ACM Trans. Embed. Comput. Syst.*, 22(6).

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., and Liu, Y. (2019). A survey on the scalability of blockchain systems. *IEEE Network*, 33(5):166–173.