LearningChain: A Highly Scalable and Applicable Learning-Based Blockchain Performance Optimization Framework

Jishu Wang, *Member, IEEE*, Yaowei Wang, Xuan Zhang, *Member, IEEE*, Zhi Jin, *Fellow, IEEE*, Chao Zhu, *Member, IEEE*, Linyu Li, *Student Member, IEEE*, Rui Zhu, *Member, IEEE*, and Shenglong Lv, *Student Member, IEEE*

***Abstract*—Blockchain is a trans-generational technology that is gradually introduced and applied in many fields because of its characteristics such as tamper-proof, traceability, and decentralization. However, the performance bottlenecks of blockchain have been one factor that hinders its practical application. This paper proposes a blockchain performance optimization framework (called LearningChain). We use a temporal convolution network to predict the transaction arrival rate of the blockchain and propose an ensemble learning-based method and a meta-learning-based method to train a blockchain performance prediction model, respectively. We design a performance scoring mechanism to dynamically tune the configuration parameters of the blockchain to optimize the blockchain performance. In addition, we collect and contribute a blockchain performance dataset (called HFBTP) for other researchers to research. The sufficient experimental results and analysis show that LearningChain can effectively optimize blockchain performance. The quantitative and qualitative comparisons with related work demonstrate the superiority and innovation of our work, LearningChain reaches state-of-the-art, is highly applicable, scalable, and can be applied to many practical blockchain-based application scenarios and different blockchain platforms. LearningChain can be complemented with other existing blockchain performance optimization tools and methods to further enhance the effectiveness of blockchain performance optimization.**

***Index Terms*—Blockchain, blockchain performance prediction model, blockchain performance optimization, machine learning.**

Manuscript received 14 August 2023; revised 3 November 2023; accepted 23 December 2023. Date of publication 28 December 2023; date of current version 15 April 2024. This work was supported by the National Key Research and Development Program of China under Grant No. 2020AAA0109400; Science Foundation of Young and Middle-aged Academic and Technical Leaders of Yunnan under Grant No. 202205AC160040; Science Foundation of Yunnan Jinzhi Expert Workstation under Grant No. 202205AF150006; Major Project of Yunnan Natural Science Foundation under Grant No. 202302AE09002003; Knowledge-driven Smart Energy Science and Technology Innovation Team of Yunnan Provincial Department of Education ; Science and Technology Project of Yunnan Power Grid Co., Ltd. under Grant No.YNKJXM20222254; Open Foundation of Yunnan Key Laboratory of Software Engineering under Grant No. 2023SE101; the 14th Postgraduate Research Innovation Project of Yunnan University (KC-2222958). The associate editor coordinating the review of this article and approving it for publication was A. Veneris. *(Corresponding author: Xuan Zhang.)*

Jishu Wang and Yaowei Wang are with the School of Information Science and Engineering, Yunnan University, Kunming 650091, China (e-mail: cswangjishu@hotmail.com; wangyaoweis@gmail.com).

Xuan Zhang, Rui Zhu, and Shenglong Lv are with the School of Software, Yunnan Key Laboratory of Software Engineering, Yunnan University, Kunming 650091, China (e-mail: zhxuan@ynu.edu.cn; rzhu@ynu.edu.cn; 675176214@qq.com).

Zhi Jin and Linyu Li are with the Key Laboratory of High Confidence Software Technologies, Ministry of Education, and the School of Computer Science, Peking University, Beijing 100871, China (e-mail: zhijin@pku.edu.cn; xltx_youxiang@qq.com).

Chao Zhu is with BYD Company Ltd., Shenzhen 518119, China (e-mail: 704026774@qq.com).

Digital Object Identifier 10.1109/TNSM.2023.3347789

I. Introduction

With the rapid development of the information age and the rise of emerging technologies such as artificial intelligence. The public is becoming concerned about the security of private data, and many organizations and departments are increasingly demanding efficiency and security in data sharing. Blockchain as a distributed and integrated technology, is naturally suited to solve many existing problems (e.g., untrustworthy environment [1], privacy protection [2], data traceability and tracking [3]) because of its tamper-evident, decentralized and traceable characteristics. In recent years, blockchain has received increasing attention and has gained rapid development in theoretical innovation and application implementation.

Nowadays, blockchain is integrated into many information systems and services, include but are not limited to healthcare (privacy protection [4], privacy authentication [5], information sharing [6]), intelligent transportation systems (Internet of Vehicles [7], electronic toll collection systems [8], traffic big data [9]), smart grid (energy auctions [10], log management [11], energy trading [12]), Internet of Things (IoT) (device authentication [13], cloud computing [14], fog computing [15]), federated learning [16].

Even though blockchain has been used in numerous fields, performance bottlenecks in blockchain (especially throughput and latency) are one of the significant factors that hinder its further development. In blockchain, there are many parameters (e.g., block size, number of nodes, network bandwidth, block packing interval, etc.) that can affect the performance of blockchain. In Bitcoin [17], its throughput is limited to 7 transactions per second (TPS) because it uses a block size of 1 MB and a block packing interval of 10 minutes. As a result, the Bitcoin network has a large number of transactions that are blocked and cannot be confirmed quickly. On the other hand, the consortium blockchain has higher throughput because it tends to have more flexible configuration space, but

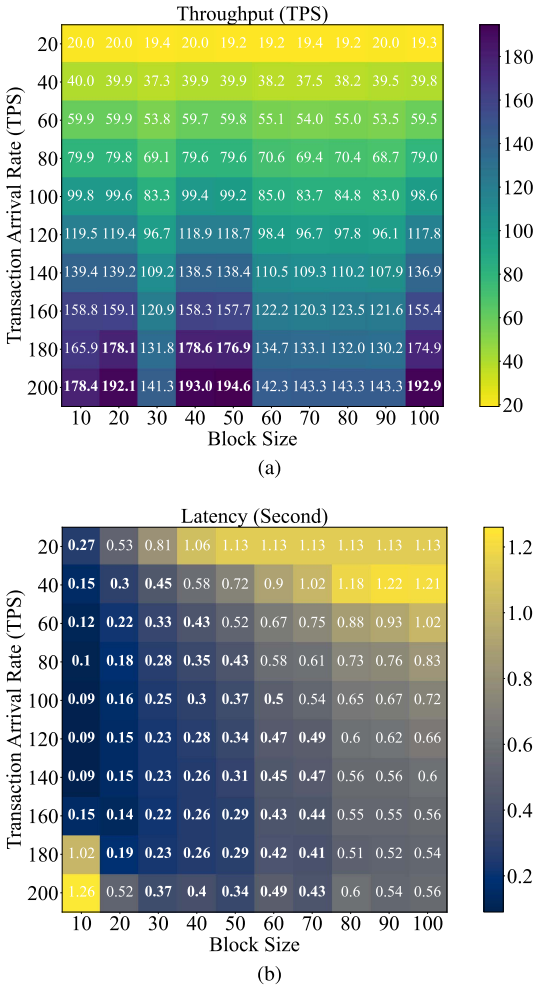


Fig. 1. How blockchain performance varies for different transaction arrival rates and block size. (a) Throughput (b) Latency.

this also leads to a higher barrier to use and makes it difficult for blockchain maintainers to quickly optimize blockchain performance.

In recent years, researchers have started to optimize the blockchain performance from different optimization directions (improvements to consensus algorithms [18], [19], [20], blockchain sharding [21], [22], and traditional optimization algorithms-based [23], [24], [25]). Existing work has achieved good performance, but their scalability and applicability will be constrained because improvements to consensus algorithms and blockchain sharding will face a certain technical threshold and various complexities requirements (e.g., the requirements of different scenarios, the customization requirements of blockchain users) when blockchain are actually deployed, and thus may not be fully applicable to various application scenarios.

It is necessary and highly valuable to optimize the performance of blockchain by tuning blockchain parameters. We conduct a set of experiments to verify the variation of throughput and latency by varying the block size on Hyperledger Fabric, each set of experiments was performed 1000 times and the final average was taken. The results of the experiments are shown in Fig. 1. Setting different transaction arrival rates and block sizes yields different blockchain performance. How to dynamically tune the blockchain parameters so that the blockchain performance is optimal is the main motivation of this paper.

Recently, a few researchers [26] have used machine learning methods to tune blockchain parameters to optimize blockchain performance, this can effectively complement and synergize existing research efforts to optimize the performance of the blockchain more efficiently. However, the accuracy of their proposed blockchain performance prediction model can be further optimized, and the scalability of these methods is limited (a large amount of data is required for training, which may lead to additional data collection and training costs).

Ensemble learning is a feasible method to improve predictive accuracy because it achieves better predictive performance, stronger generalization, and stability than a single base model by constructing and combining multiple different base models [27]. However, a broader and practical situation also needs to be considered, which is that collecting large-scale blockchain performance data is difficult, time-consuming, and labor-intensive. Therefore, training a relatively accurate prediction model with few blockchain performance data will effectively improve the applicability and scalability of blockchain performance optimization methods. With the rapid development of meta-learning, many researchers [28], [29], [30], [31] have been investigating how to improve the training effectiveness and efficiency of meta-learning with few sample data in recent years. Meta-learning is a machine learning method that aims to equip models with the ability to learn, utilizing existing knowledge and experience to quickly adapt to new tasks without requiring a large amount of data or training [28]. The advantage of meta-learning for training on few sample data is significant, but it is of concern that there is still little research on using meta-learning for blockchain, which may be due to the small number of datasets available for research in the current blockchain domain, this shows the urgency and importance of the work in this paper.

In particular, to better optimize the blockchain performance by tuning blockchain parameters, we spend several months collecting a large-scale Hyperledger Fabric performance dataset (HFBTP) for the training of blockchain performance prediction model and other researchers to research. We propose a blockchain performance optimization framework (LearningChain). To the best of our knowledge, this is the first work to use ensemble learning and meta-learning to tune blockchain parameters for blockchain performance optimization, and HFBTP is also the largest-scale blockchain performance dataset at this stage.

Our main contributions are highlighted as follows:

- We collect and contribute HFBTP. At present, such datasets are extremely scarce (which proves that they are difficult and time-consuming to collect). HFBTP could be a new benchmark dataset in such studies, and with HFBTP, researchers can validate the accuracy of their proposed approaches.
- We propose LearningChain, which includes 3 modules (the prediction of transaction arrival rate, the modeling of blockchain performance, and the scoring mechanism for

blockchain performance optimization). LearningChain is not dependent on a specific blockchain platform and has relatively high scalability and applicability to be used in different blockchain application scenarios and studies.
- The sufficient experimental results and analysis show that LearningChain achieves state-of-the-art (SOTA) on different metrics for multiple datasets. The qualitative comparison and analysis with related work fully demonstrate the innovation and effectiveness of the work in this paper, and LearningChain applies to blockchain-based application scenarios and research problems.

The rest of this paper is organized as follows: in Section II, we discuss some related work. In Section III, we introduce the overall framework and details of LearningChain. Section IV introduces the design of the related experiments and fully evaluates and analyzes the performance and features of LearningChain. Finally, Section V concludes this paper.

II. Related Work

As blockchain begins to be used in many fields, its performance issues are gradually beginning to gain the attention of researchers. Currently, researchers have focused on different perspectives to optimize blockchain performance.

1) Improvement of Consensus Algorithms. Gao et al. [18] proposed a blockchain-based distributed data system for industrial IoT and optimized the throughput and latency of the blockchain using a piecewise hash graph consensus algorithm. Zhang et al. [19] improved the consensus algorithm used by the Hyperledger Fabric, thus balancing the performance and security of the consensus network. Xu et al. [20] proposed an adaptive extended blockchain with transaction deduplication function, and achieved higher throughput. When users have high transaction requirements in the network, blockchain will expand to meet the requirements, when transaction requirements are low, blockchain contracts to save communication and storage costs.

2) Blockchain Sharding. Cai et al. [21] proposed a sharding system that improves performance through collaboration-based sharding while protecting the security of each shard. Li et al. [22] proposed a new sharding system to improve the throughput of blockchain, which periodically migrates active accounts from heavily loaded shards to less loaded shards, thus dynamically balancing the transaction load on different shards.

3) Traditional Optimization Algorithms-Based: Jamil et al. [23] used a particle swarm optimization algorithm to analyze the relationship between throughput and latency in Hyperledger Fabric to find the optimal transaction arrival rate to optimize throughput versus latency. Chen et al. [24] constructed a real private Ethereum IoT network and probabilistically fitted the performance for latency. Wilhelmi et al. [25] analyzed numerical blockchain performance results and used Markov Chain to select the optimal block size to optimize latency.

4) Machine Learning-Based Optimization. Wang et al. [26] used Multilayer Perceptron (MLP) to train blockchain performance prediction model and Long Short-Term Memory (LSTM) to predict transaction arrival rates. This is our previous work where we proposed a blockchain performance optimization scheme based on deep learning, but we did not consider the collection of blockchain performance datasets to be difficult and time-consuming, the usability of the prediction model at small sample data sizes, and the fact that the proposed methods still have space for further optimization, which is the main research motivation of this paper.

However, based on the above work, there are still some refinements that can be made. We provide a detailed comparison in Table IX.

III. LearningChain System Model

In this section, we introduce the overall framework of LearningChain, including the prediction model of transaction arrival rate, ensemble-learning blockchain performance prediction model, meta-learning blockchain performance prediction model, and scoring optimization mechanism.

A. Overall Framework

Our proposed LearningChain blockchain performance optimization framework is shown in Fig. 2, which contains 3 main modules.

1) Blockchain System: A running blockchain system generates a lot of performance data, including throughput and latency. Due to differences in transaction arrival rates and configuration parameters (e.g., block size, number of nodes, etc.), blockchain performance data can vary. This module is responsible for collecting transaction arrival rate and blockchain performance data of a certain size, thus providing the basis for training machine learning models.

2) Learning Model: In this module, a temporal convolution network (TCN) is used to predict the arrival rate of transactions at a certain point in the future, while we choose either an ensemble-learning model or a meta-learning model to predict the performance of the blockchain based on the size of the data volume.

3) Blockchain Performance Optimization: In this module, we propose a blockchain scoring optimization mechanism. We input the predicted transaction arrival rate and different blockchain configurations into the trained blockchain performance prediction model and score them. After that, the configuration with the highest score is adjusted to the blockchain, thus completing the dynamic optimization of blockchain performance.

In addition, the specific optimization process of LearningChain is shown in Fig. 3, which can be divided 3 into steps.

1) Preparation: To effectively tune the configuration parameters of the blockchain using machine learning methods to optimize the blockchain performance, a certain amount of data needs to be prepared for training first. In this module, two types of data need to be collected, i.e., historical time-series data on transaction arrival rate and blockchain performance dataset. In addition, to make the optimization effect match the expectations of blockchain maintainers (researchers) as much as possible, the weight (throughput and latency) of

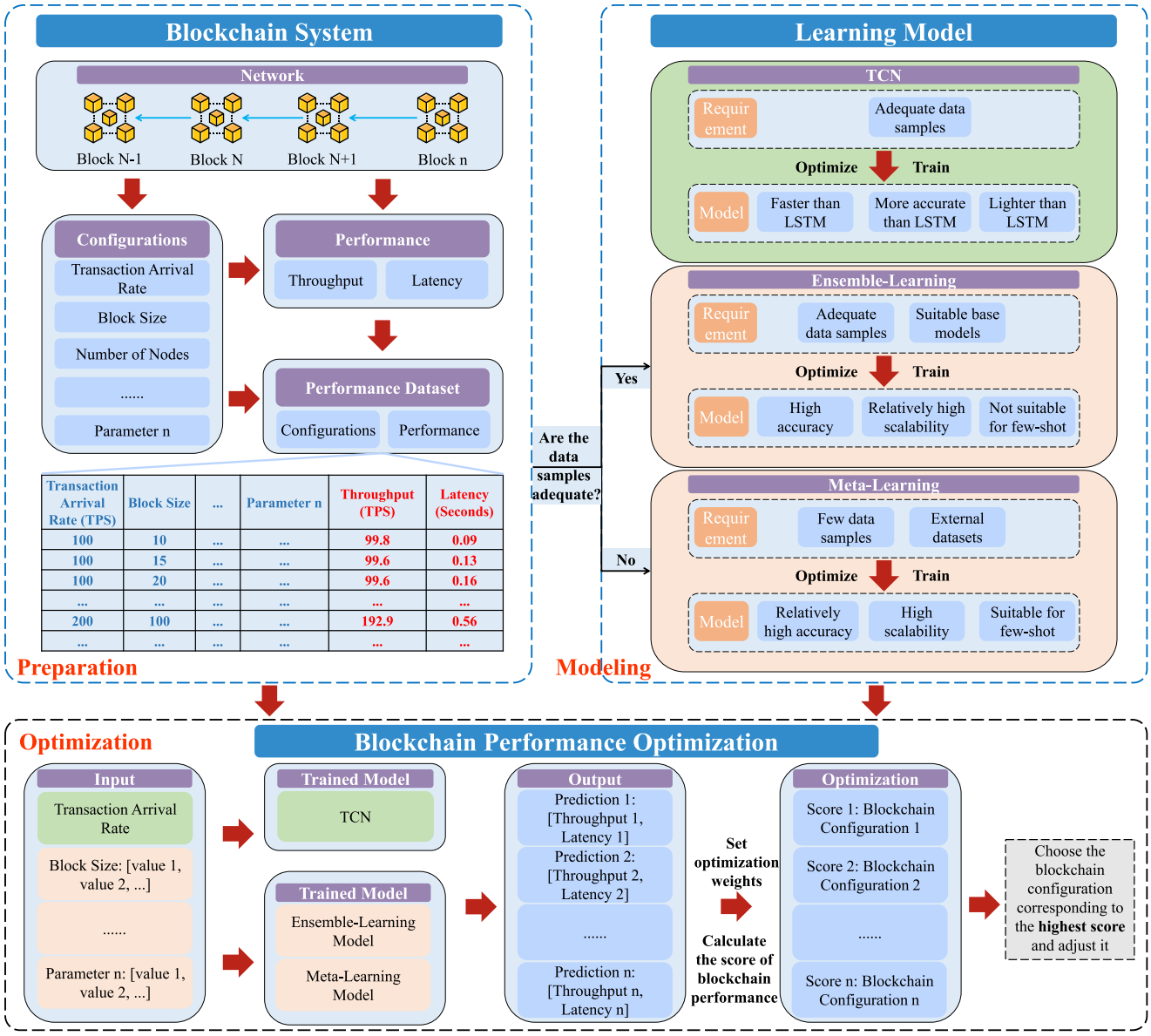


Fig. 2. The overall framework of LearningChain.

optimization needs to be set according to the actual needs. In short, by setting different weights, the effect of optimization makes a difference.

2) Modeling: After obtaining datasets, we first use TCN to complete the prediction of the transaction arrival rate, which will help the subsequent blockchain performance optimization. Meanwhile, we propose blockchain performance prediction models based on ensemble learning and meta-learning for different situations, respectively. The ensemble learning-based blockchain performance prediction model is used when the data available for training is sufficient, otherwise, the meta-learning-based blockchain performance prediction model is used. In real scenarios, the transaction arrival rate varies from time to time. For example, in an intelligent transportation system, if the traffic flow is considered as the transaction arrival rate, the transaction arrival rate is significantly higher during peak hours than during other hours. We need to predict the transaction arrival rate in some future period in advance (e.g., after 5 minutes) and input it as a parameter into the trained blockchain performance prediction model to optimize the blockchain performance more accurately and effectively.

3) Optimization: After training with the dataset, the transaction arrival rate prediction model and the blockchain performance prediction model have achieved high prediction accuracy. Taking the optimization of blockchain performance after 5 minutes as an example, the complete optimization process is as follows. We first predict the transaction arrival rate after 5 minutes and input it into the trained blockchain performance prediction model along with each blockchain configuration available for tuning, at which point we will get different predicted blockchain performances. Next, we can calculate the score for each blockchain performance with the performance scoring optimization mechanism and the

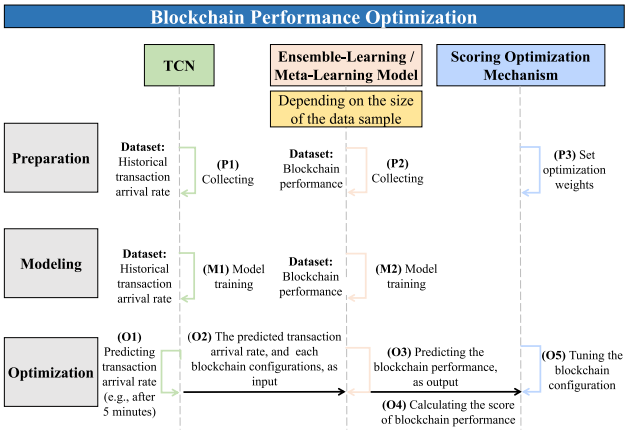


Fig. 3. The blockchain performance optimization process of LearningChain.

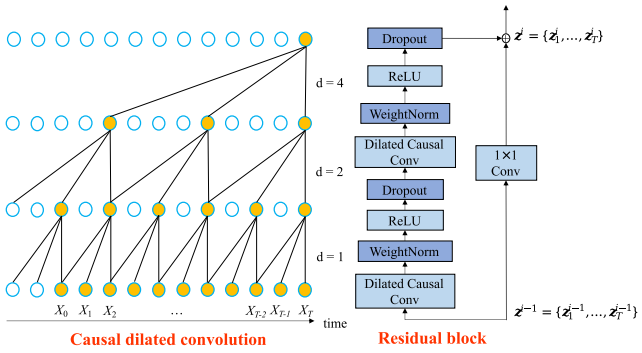


Fig. 4. The framework of TCN.

setting of weights. Finally, we select and tune the blockchain configuration with the highest score.

The details of each module and process are introduced in the following.

B. Prediction of Transaction Arrival Rate

To optimize the blockchain performance more effectively, we first need to predict the transaction arrival rate in a certain period in the future (e.g., peak hours, when the optimization benefits are often higher) and complete the adjustment of the blockchain configuration parameters in advance, to achieve the blockchain performance optimization.

In this part, we introduce how to use TCN to predict the short-term transaction arrival rate of the blockchain. To obtain a more accurate prediction effect and thus improve the scalability and usability of blockchain performance optimization, we predict the transaction arrival rate for a future period based on a given 60-minute historical transaction arrival rate. Bai et al. [32] introduced convolutional neural networks to time series modeling and proposed TCN for processing time-series data. TCN model mainly consists of two parts, which are the causal dilated convolution and the residual block, which are shown in Fig. 4.

Recurrent neural networks and their variants perform recursive processing of long sequences, which leads to long training times and gradient disappearance or explosion problems.

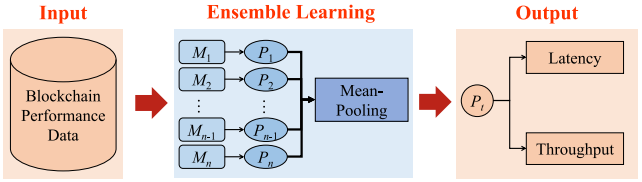


Fig. 5. The framework of ensemble-learning blockchain performance prediction model.

However, convolutional neural networks can compute in parallel and can expand the receptive field by increasing the depth of the convolutional layers. The causal dilated convolution operator in a temporal convolutional network can be defined as

$$x \star f = \sum_{k=0}^{K-1} f_k x_{t-d\times k} \tag{1}$$

where $x \in \mathbb{R}^T$ denotes a one-dimensional time series, $f \in \mathbb{R}^K$ denotes the convolution kernel. The parameter d denotes the dilation factor and $\star$ denotes the convolution operation. By increasing the dilation factor d, the receptive field in the time domain grows exponentially, which enables the temporal convolutional network to capture longer time series information by stacking only a smaller number of convolutional layers.

C. Blockchain Performance Prediction Model

In this section, we provide a detailed description of the blockchain performance prediction model. The purpose of the model is to predict the blockchain performance based on blockchain configuration data. Formally, we define the blockchain configurations data as $\{X_1, X_2, \ldots, X_N\}$, where each sample $X_i \in R^{1\times C}$, and C is the attribute dimension of the blockchain configurations data. The blockchain performance is represented by $\{Y_1, Y_2, \ldots, Y_N\}$, where each sample is used to predict two performance metrics, i.e., throughput and latency, and therefore Y_i represents throughput or latency. N represents the total number of blockchain configurations data samples.

To adapt to more application scenarios, we propose two blockchain performance prediction models. When the amount of data is sufficient, high-accuracy prediction results are obtained by the ensemble-learning model, but since collecting blockchain performance datasets of sufficient size is both time-consuming and difficult, using ensemble-learning is costly in many scenarios, which limits the application scenarios of using machine learning methods to optimize blockchain performance. This is a typical few-shot learning problem (i.e., training a relatively accurate model with dozens or even fewer data samples), so we introduce meta-learning to solve this problem.

1) Ensemble-Learning Blockchain Performance Prediction Model: The ensembling process is shown in Fig. 5. First, multiple different base models are trained, and then the results of these base models are averaged to obtain the final result. Where, M_1, M_2,..., M_n represent different base models, P_1, P_2,..., P_n represent the outputs of these base models, n

represents the number of base models, and P_t represents the output of the entire model.

To ensure the effectiveness of ensemble learning, the base models must have good performance. If the performance of a base model is poor, the overall prediction performance of the model will also decrease. In addition, the base models should be relatively independent and diverse so that ensemble learning can capture complementary information between the base models. Therefore, we choose five models with good performance and different implementation principles, including **MLP**, KNeighborsRegressor (**KNR**), GradientBoostingRegressor (**GBR**), BaggingRegressor (**BR**), and RandomForestRegressor (**RFR**). Among them, MLP is implemented by building a three-layer fully connected neural network, in each layer of MLP, we use ReLU [33] as the activation function, and the loss function of the model is defined as:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \widehat{Y}_i\right)^2 \tag{2}$$

among them, where N represents the number of samples, Y_i represents the true label, and $\widehat{Y}_i$ represents the predicted label. The other base models are implemented using the library provided by scikit-learn.[1]

2) Meta-Learning Blockchain Performance Prediction Model: The essence of few-shot learning is pre-learning, which transfers experience from a set of similar tasks, allowing the model to establish a robust prediction model with only a few samples in a new task. One solution is to treat few-shot learning as a meta-learning problem, where Model-Agnostic Meta-Learning (MAML) is a representative and excellent meta-learning method that can effectively address the few-shot problem [28]. Compared with other meta-learning methods, MAML is a task-agnostic learning method with almost no assumptions about the form of the model, introducing only a small number of parameters and adopting mainstream optimization processes such as gradient descent. In other words, MAML can find the optimal initial parameters for the network model, so that good results can be obtained in few-shot tasks with only a few gradient update steps.[2] Based on the characteristics and advantages of MAML, we use it to solve the problem that blockchain cannot be quickly optimized for performance without sufficient scale data so that a usable blockchain performance prediction model can be trained with a few performance data samples to quickly optimize blockchain transaction performance. The framework of meta learning-based blockchain performance prediction model is shown in Fig. 6.

In meta-learning, we divide the dataset into $D_{meta-train}$ and $D_{meta-test}$, where $D_{meta-test}$ represents our target dataset with very few samples, and $D_{meta-train}$ is an external dataset consisting of data types similar to $D_{meta-test}$. The set of tasks $p(T)$ is constructed by meta-training set $D_{meta-train}$,

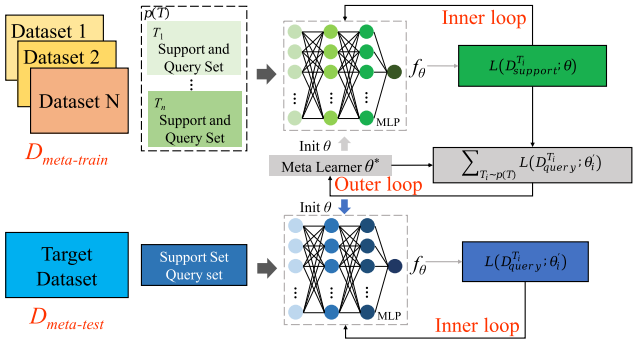


Fig. 6. The framework of ensemble-learning blockchain performance prediction model.

and each task contains only a few samples. Then, a batch of learning tasks T_b is sampled from $p(T)$, where each task $T_i \in T_b$ consists of a support dataset $D_{support}$ and a query dataset D_{query}. Because the MAML method is model-agnostic, we still choose MLP as the base model. During the meta-training phase, MAML is trained using a double optimization process, which includes inner and outer loop updates [34] that operate on a batch of related tasks at each iteration. In the inner loop, the base model f_θ is initialized with parameters θ given by the Meta Learner, and gradient descent is performed on $D_{support}$ for each task to update the model. The loss function L in the inner loop is consistent with Eq. (2), to optimize the model's parameters on $D_{support}$ to achieve better performance on the D_{query}.

In the inner loop update, given a model f_θ parameterized by θ, the gradient update of θ after one iteration on task T_i can be expressed as follows:

$$\theta_i' = \theta - \alpha\nabla_\theta L_{T_i}(f_\theta) = \theta - \alpha\nabla_\theta L\left(D^{T_i}_{support};\theta\right) \tag{3}$$

where α represents the learning rate of the inner loop, L represents the loss function, and $D^{T_i}_{support}$ represents task T_i randomly sampled from $D_{support}$. The above gradient update can be performed multiple times. The goal of the Meta Learner is to obtain meta-knowledge from all tasks $T_i \sim p(T)$ to optimize the parameter θ, so that the model can quickly adapt to new tasks within a few gradient steps on the support set and achieve maximum performance on the query set. The parameters θ^* of the Meta Learner are obtained by minimizing the following loss:

$$\theta^* = \text{argmin}_\theta \sum_{T_i\sim p(T)} L\left(D^{T_i}_{query};\theta_i'\right) \tag{4}$$

in outer loop updates, we perform meta-optimization using Eq. (4) to obtain θ^*. Specifically, we update θ using stochastic gradient descent with back-propagated loss:

$$\theta^* \leftarrow \theta - \beta\nabla_\theta \sum_{T_i\sim p(T)} L\left(D^{T_i}_{query};\theta_i'\right) \tag{5}$$

where β is the learning rate for outer loop updates. The outer loop update updates θ_i' based on the performance of θ on $D^{T_i}_{query}$, a batch of meta-training tasks.

[1] https://scikit-learn.org/stable/

[2] Ensemble learning methods usually require a large amount of training data to fit the actual distribution of the data, which improves the generalization ability and reduces the risk of overfitting. However, the data samples are not adequate will lead to accuracy decreases significantly.

TABLE I
LIST OF NOTATIONS

Notations	Description
BPC_i	The i^{th} blockchain parameters configuration
Thr_i	The throughput corresponding to BPC_i (TPS)
Lat_i	The latency corresponding to BPC_i (Second)
Thr^{Min}	BPC corresponding to the minimum Thr
Thr^{Max}	BPC corresponding to the maximum Thr
Lat^{Min}	BPC corresponding to the minimum Lat
Lat^{Max}	BPC corresponding to the maximum Lat
$Score_i$	The performance score corresponding to BPC_i
$Score_i^{Thr}$	The $Score_i$ (Thr) corresponding to BPC_i
$Score_i^{Lat}$	The $Score_i$ (Lat) corresponding to BPC_i
$Weight^{Thr}$	The weight corresponding to $Score_i^{Thr}$
$Weight^{Lat}$	The weight corresponding to $Score_i^{Lat}$
$Ratio$	The ratio of Thr to Lat volatility
$Thres^{Lat}$	The maximum tolerable Lat threshold
$Thres^{Ratio}$	The maximum threshold for $Ratio$

As we can see, the update to θ is through the gradient of the gradient (using θ to compute θ_i' in Eq. (3) and using θ_i' to update θ in Eq. (5)) during meta-training. During meta-testing, given a new task T_j from $D_{meta-test}$, we fine-tune and optimize the model θ^j by performing inner loop updates on $D_{query}^{T_j}$. Note that outer loop updates are not present during meta-testing.

D. Blockchain Performance Scoring Optimization Mechanism

In the previous subsection, we introduce the proposed learning-based blockchain performance prediction models. To further enhance the usefulness and value of the models and to guide the application in real scenarios, we further implement the optimization of blockchain performance based on it. The symbols used in this subsection are shown in Table I.

We define a score function that is expressed as

$$Score_i = Weight^{Thr} * Score_i^{Thr} + Weight^{Lat} * Score_i^{Lat} \tag{6}$$

for each BPC_i, we can calculate its $Score_i^{Thr}$ and $Score_i^{Lat}$. Therefore, we can find out the BPC_i that obtains the optimal performance according to $Score_i$. To better match the needs of realistic scenarios and thus allow blockchain maintainers to effectively control the bias of performance (e.g., pursuing lower latency, or pursuing higher throughput), the weighting factor $Weight^{Lat}$ and $Weight^{Thr}$ is given to $Score_i^{Lat}$ and $Score_i^{Thr}$, respectively.

To ensure the fairness and effectiveness of the weight setting, it needs to satisfy the following Eq. (7).

$$1 = Weight^{Thr} + Weight^{Lat} \tag{7}$$

$$Score_i^{Thr} = \frac{Thr_i - Thr^{Min}}{Thr^{Max} - Thr^{Min}} \tag{8}$$

$Score_i^{Thr}$ is 0 when Thr_i is equal to Thr^{Min} and 1 when Thr_i is equal to Thr^{Max}.

$$Score_i^{Lat} = \frac{Lat_i - Lat^{Max}}{Lat^{Min} - Lat^{Max}} \tag{9}$$

$Score_i^{Lat}$ is 0 when Lat_i is equal to Lat^{Max} and 1 when Lat_i is equal to Lat^{Min}.

Blockchain maintainers can manually set the weights to match the needs of real-world scenarios. However, the use of manual settings is often blind and cannot be adjusted in real time according to the actual situation. Therefore, we optimize and improve the scoring mechanism to make it more in line with the needs of realistic scenarios and enhance its scalability. We propose a weight-setting strategy to complement this so that when blockchain maintainers do not set weights manually, they can also flexibly accomplish the optimization of blockchain performance.

The primary strategy is *Ratio*. we can set $Weight^{Thr}$ and $Weight^{Lat}$ according to *Ratio*, which is expressed as

$$Ratio = \frac{Thr^{Max} - Thr^{Min}}{Lat^{Max} - Lat^{Min}} \tag{10}$$

according to *Ratio*, the degree of fluctuation of throughput and latency can be effectively determined. When *Ratio* is large, it means that boosting $Weight^{Thr}$ can achieve significant results:

$$Weight^{Thr} = \begin{cases} Weight_1^{Thr}, \text{ if } Ratio \geq Thres^{Ratio}; \\ Weight_2^{Thr}, \text{ otherwise} \end{cases} \tag{11}$$

in addition, the secondary strategy is the judgment and weight set by $Thres^{Lat}$. which can be expressed as

$$Weight^{Lat} = \begin{cases} Weight_1^{Lat}, \text{ if } Lat^{Min} \leq Thres^{Lat}; \\ Weight_2^{Lat}, \text{ otherwise} \end{cases} \tag{12}$$

when Lat^{Min} is less than or equal to $Thres^{Lat}$, higher throughput should be pursued as much as possible (e.g., set $Weight^{Thr}$ to 1 and $Weight^{Lat}$ to 0), while when Lat^{Min} is greater than $Thres^{Lat}$, the latency should be reduced as much as possible (e.g., set $Weight^{Thr}$ to 0 and $Weight^{Lat}$ to 1).

With such a design, we can further refine the setting way of $Weight^{Thr}$ and $Weight^{Lat}$ to avoid blind (meaningless) weight setting, so that the scoring mechanism for blockchain performance is more scalable and feasible.

The blockchain parameters corresponding to the highest score $Score^{Best}$ are the optimal blockchain parameters corresponding to the next time slice. The specific process is shown in Algorithm 1. We divide the execution flow of Algorithm 1 into 4 steps, details are given as follows.

1) Predicting the Transaction Arrival Rate: The future transaction arrival rate is predicted based on the previous T transaction arrival rates, as shown in line 4 of Algorithm 1. In the case scenario considered in this paper, we set T to 12 and P to 1.

2) Predicting the Blockchain Performance: For each blockchain parameter that can be selected, we feed it into the trained blockchain performance prediction model to predict and save different blockchain performances. The number of nested loops is equal to the number of parameters that can be selected, thus traversing every possible combination of parameters, as shown in lines 5-11 of Algorithm 1.

3) Setting Weights: Based on the defined weight setting strategy, we set the weights to optimize blockchain performance

Algorithm 1 Dynamically Tuning Blockchain Parameters

1: **Input:**
2: The transaction arrival rate of last T time steps: $X_T = \{x_{t-T+1}, \ldots, x_t\}$; The trained TCN model for predicting transaction arrival rate $X_P = \{x_{t+1}, \ldots, x_{t+P}\}$ in the next P time steps: *TCN*; The trained blockchain performance prediction models: $Model_{Lat}$, $Model_{Thr}$; The set of blockchain parameters that can be selected: $P = \{P_1, \ldots, P_N\}$; The set of blockchain parameter P_i that can be selected: $P_i = \{P_i^1, \ldots, P_i^M\}$; $Thres^{Lat}$, $Thres^R$;
3: **Procedures:**
4: $X_P = TCN(X_T)$
5: **for** P_1^i in P_1 **do**
6: **for** $P_{\ldots}^i$ in $P_{\ldots}$ **do** // N loops (N parameters)
7: **for** P_N^i in P_N **do**
8: Add $Model_{Lat}(X_P, P_1^i, P_{\ldots}^i, P_N^i)$ to *Lat*, $Model_{Thr}(X_P, P_1^i, P_{\ldots}^i, P_N^i)$ to *Thr*, and $[P_1^i, P_{\ldots}^i, P_N^i]$ to *BPC*
9: **end for**
10: **end for**
11: **end for**
12: **if** $Ratio \geq Thres^{Ratio}$ **then**
13: Eq. (11)
14: **else if** $Lat^{Min} \leq Thres^{Lat}$ **then**
15: Eq. (12)
16: **else then**
17: $Weight^{Lat} = Weight_2^{Lat}$, Eq. (7)
18: **end if**
19: $i = 0$, $Score^{Best} = 0$, $BPC^{Best} = BPC_0$
20: **while** $i < BPC.Length()$ **do**
21: Eq. (6)
22: **if** $Score_i > Score^{Best}$ **then**
23: $Score^{Best} = Score_i$, $BPC^{Best} = BPC_i$
24: **end if**
25: $i++$
26: **end while**
27: Tuning the configuration of the blockchain to BPC^{Best}

more effectively and flexibly. As shown in lines 12-18 of Algorithm 1.

4) Optimizing the Blockchain Performance: The blockchain parameters are tuned to the optimal block parameters BPC^{Best}, as shown in lines 19-27 of Algorithm 1.

The time complexity of Algorithm 1 depends on 3 parts. Firstly, each layer of convolutions is $O(k * T * d^2)$, where k is the kernel size of the convolutions, T is the time series length, and d is the representation channel. Because of dilated convolution, $O(log_k T)$ layers of convolution operations are required to be stacked. So the time complexity of line 4 is $O(log_k T * k * T * d^2)$. Secondly, the number of loops depends on the number of blockchain parameters that can be selected, when the number of blockchain parameters is N, the number of loops is N. Our proposed performance prediction models only need to perform one forward propagation computation on new input data during the prediction phase, each step of the computation is based on the trained weights and biases, which are fixed during the training phase. Therefore, the time complexity of lines 5-11 is $O(M^N)$, because the number of blockchain parameters available for tuning tends not to be too large and the range of values for each parameter tends to be limited so that the total time complexity is acceptable. In summary, the time complexity of Algorithm 1 is $O(M^N)$.

IV. Experimental Evaluation and Analysis

In this section, we design different experiments to verify and analyze the performance and effectiveness of LearningChain. Qualitative comparisons are also made with related work to illustrate the innovation of LearningChain. All experiments were done on a Workstation (64-bit Intel Core i9-12900K 3.2GHz, 128GB RAM, and Windows 11 operation system).

A. Experimental Settings

1) Experimental Design: We design four experiments to fully and adequately demonstrate the performance, effectiveness, and feasibility of LearningChain: (1) The prediction performance on transaction arrival rate; (2) The performance of ensemble learning-based blockchain performance prediction model; (3) The performance of meta learning-based blockchain performance prediction model. (4) The effectiveness of scoring optimization mechanism.

2) Datasets: In experiment (1), we use a open-source dataset[3] for the prediction on transaction arrival rate. The dataset is a simulated dataset of transaction arrival rates obtained by processing real-world parking records, which contains two parameters, timestamp and transaction arrival rate. Therefore, this dataset can be used to validate the effectiveness of our proposed transaction arrival rate prediction model.

In experiments (2) and (3), we use an open-source dataset and our contributed blockchain performance dataset HFBTP to verify the performance of LearningChain: **BPD**[4] and **HFBTP**. BPD includes four sub-datasets, based on Hyperledger Fabric, which contains four parameters, transaction arrival rate, block size, latency, and throughput. To collect HFBTP, we deploy Hyperledger Fabric 2.3, and the blockchain performance testing tool is Hyperledger Caliper 0.5.0. We set up three Organizations, each containing three Peer nodes, it contains five parameters, [3, 5, 7, 9] Orderer nodes, [10, 15, 20, ..., 200] transaction arrival rate, [10, 15, 20, ..., 800] block size, latency, and throughput. HFBTP (24687 data in total) is larger than BPD (4120 data in total), BPD and HFBTP also can be used to validate the effectiveness of blockchain performance prediction models.

3) Comparable Methods: In experiment (1), the SOTA method [26] (LSTM-based) and Auto-Regressive Integrated Moving Average (**ARIMA**, a statistical temporal series prediction method) are compared with LearningChain.

In experiments (2) and (3), to further demonstrate the superiority and performance of LearningChain, the

[3]https://www.kaggle.com/datasets/loveffc/transaction-arrival-rate-blockchain-for-parking

[4]https://www.kaggle.com/datasets/loveffc/blockchain-performance

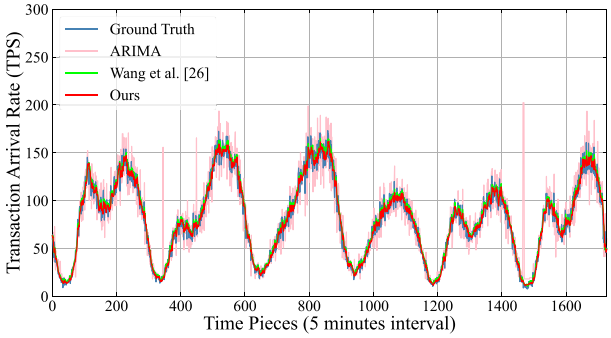


Fig. 7. The prediction performance of different models for transaction arrival rate (5 minutes period).

TABLE II
THE PREDICTION PERFORMANCE ON TRANSACTION ARRIVAL RATE

Time periods	Metrics	Methods		
		ARIMA	Wang et al. [26]	**Ours**
1 minute	MAE	648.05	617.30	**594.89**
	RMSE	853.60	799.69	**778.78**
	MAPE	8.35	8.19	**7.72**
2 minutes	MAE	993.69	923.72	**898.96**
	RMSE	1310.79	1195.51	**1179.93**
	MAPE	12.82	6.16	**5.68**
5 minutes	MAE	3831.86	1747.98	**1659.60**
	RMSE	15381.00	2260.08	**2174.94**
	MAPE	22.96	8.99	**8.16**

SOTA method [26] (MLP-based) and seven traditional machine learning models are compared with LearningChain: LinearRegression (**LR**), AdaBoostRegressor (**ABR**), DecisionTreeRegressor (**DTR**), **KNR**, **GBR**, **BR**, and **RFR**.

4) Evaluation Metrics: In experiments (1-3), we take the mean squared error (**MSE**) as the loss function of our model for training. We adapt the commonly used mean absolute error (**MAE**), mean absolute percentage error (**MAPE**), and root mean square error (**RMSE**) as evaluation metrics for regression.

The specific experimental details and evaluation analysis are detailed in the corresponding subsection. HFBTP and code open-sourced for this paper are available.[5]

B. The Performance of LearningChain

1) The Prediction Performance on Transaction Arrival Rate: In this experiment, the input data is mapped between [0, 1] using min-max normalization. We use 70% of the data for training, 20% of the data for testing, and the remaining 10% of the data for validating. We perform a grid search strategy on the validation set to locate the best hyperparameters. We use the Adam optimizer for training. The learning rate is set to 0.001. The hidden state dimension of TCN is set to 128, and the hidden layer is set to two layers. The batch size is set to 64, and early stopping is used to avoid over-fitting.

To fully validate that the proposed TCN-based transaction arrival rate prediction model has good generalization and robustness, we choose multiple periods (1, 2, and 5 minutes) for the experiments.

The experimental results are shown in Fig. 7 and Table II. Compared with two different models, our proposed model achieves a smaller prediction error, which can effectively capture short-term and long-term temporal dependencies.

In addition, the computational performance of our proposed method transaction arrival rate prediction method is superior compared to existing SOTA work. The training time (epoch), inference, and CPU memory of [26] are 2.69s, 0.53s, and 370MB, respectively. The training time (epoch), inference, and CPU memory of our proposed model are 1.49s, 0.53s, and 361MB, respectively. The shorter training time and smaller memory footprint of our proposed method make it more scalable and practical.

In real-world scenarios, the prediction period of the transaction arrival rate needs to be reasonably considered and set as it coincides with the tuning period of the blockchain parameters. When the transaction arrival rate in a specific scenario is more stable over some time, the prediction period of the transaction arrival rate should be set to a larger value to avoid too frequent parameter adjustments, thus reducing the cost of blockchain parameter adjustments. When the transaction arrival rate fluctuates frequently within a period, the prediction period should be set to a smaller value, to effectively ensure the effect of blockchain performance optimization.

2) The Performance of Ensemble-Learning Blockchain Performance Prediction Model: We use the Adam optimizer for training. The learning rate is 0.001, and the batch size is 32. The reported results are the average of 5-fold cross-validation.

The results of the experiments conducted on data set BPD are shown in Table III. The adequate experimental results show that our proposed ensemble-learning-based blockchain performance prediction model obtains SOTA in most of the experimental results. This demonstrates our proposed method can lead to better and more comprehensive prediction models by making full use of the information learned from each base model and improving the effectiveness and feasibility of blockchain performance optimization.

Notably, traditional machine learning models also achieved relatively accurate results for the task of throughput prediction, probably because the task falls under the more traditional category of numerical regression.

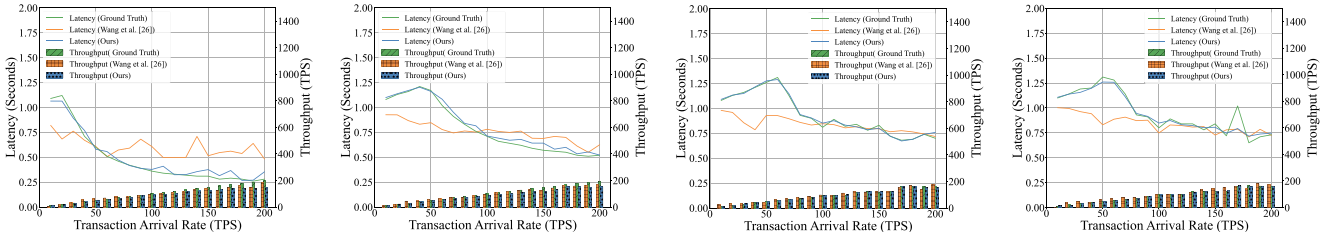
Our method obtains a significant improvement in all the latency prediction results compared to the existing SOTA work. On the other hand, since the range of values for throughput is larger than that for latency, the error in throughput is also higher than the error in latency in the experimental metrics (MAE, and RMSE).

In addition, to further validate the performance of our proposed ensemble learning-based blockchain performance prediction model, we also conduct comprehensive experiments on the HFBTP. The experimental results are shown in Table IV and Fig. 8. Compared with existing SOTA work, our proposed ensemble learning-based blockchain performance prediction

[5]https://github.com/JishuWang/LearningChain

TABLE III
THE PREDICTION PERFORMANCE OF DIFFERENT MODELS ON BPD

Tasks	Methods	BPD-1			BPD-2			BPD-3			BPD-4		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Latency	LR	0.494	0.638	56.54	0.542	0.787	64.98	0.619	0.952	69.28	0.585	0.865	75.89
	ABR	0.347	0.518	35.10	0.329	0.496	41.22	0.414	0.656	46.46	0.368	0.566	45.50
	DTR	0.123	0.223	11.89	0.178	0.299	31.94	0.242	0.392	36.27	0.191	0.329	28.08
	KNR	0.094	0.194	8.195	0.132	0.253	18.79	0.176	0.360	18.74	0.148	0.272	17.53
	GBR	0.105	0.195	9.073	0.138	0.246	18.52	0.191	0.343	22.34	0.164	0.291	19.44
	BR	0.115	0.249	8.623	0.156	0.315	19.50	0.178	0.380	17.75	0.208	0.397	21.97
	RFR	0.150	0.233	15.48	0.176	0.287	31.43	0.245	0.414	38.37	0.213	0.323	33.43
	Wang et al. [26]	0.101	0.199	7.870	0.130	0.255	17.97	0.144	0.316	14.54	0.158	0.278	20.50
	Ours	**0.084**	**0.179**	**6.301**	**0.115**	**0.245**	**14.67**	**0.135**	**0.291**	**13.49**	**0.131**	**0.260**	**14.31**
Throughput	LR	5.583	7.437	8.565	5.866	7.837	10.41	6.744	9.381	12.11	6.681	9.114	11.55
	ABR	4.820	6.191	8.683	4.411	6.161	7.764	5.544	7.414	10.28	6.278	7.778	10.46
	DTR	2.644	4.631	4.490	3.238	5.212	3.563	3.645	5.933	**4.167**	3.844	6.144	4.112
	KNR	2.781	4.601	3.671	4.287	6.255	9.675	4.681	6.811	10.36	4.883	6.820	10.36
	GBR	3.125	5.157	4.686	5.002	7.058	7.372	3.583	5.641	4.471	3.642	5.629	4.161
	BR	**2.130**	5.403	**2.055**	2.637	5.300	**3.018**	4.177	8.203	4.722	3.538	6.842	**3.701**
	RFR	2.550	4.517	4.333	2.860	4.650	3.265	3.484	5.784	4.190	3.526	5.601	3.849
	Wang et al. [26]	5.026	6.730	7.880	4.067	6.409	5.925	4.905	7.562	7.197	5.082	7.664	7.003
	Ours	2.224	**3.941**	2.854	**2.608**	**4.529**	3.764	**3.250**	**5.436**	4.631	**3.408**	**5.390**	4.548



(a) (b) (c) (d)

Fig. 8. The prediction performance of blockchain performance prediction model on HFBTP. (a) block size = 50 (b) block size = 100 (c) block size = 150 (d) block size = 200.

TABLE IV
THE PREDICTION PERFORMANCE OF DIFFERENT MODELS ON HFBTP

Tasks	Methods	MAE	RMSE	MAPE
Latency	Wang et al. [26]	0.029	0.050	4.441
	Ours	**0.023**	**0.046**	**3.656**
Throughput	Wang et al. [26]	5.543	8.141	7.118
	Ours	**2.384**	**5.192**	**2.823**

model has higher accuracy and the visualization results show that our proposed model is more stable.

3) The Performance of Meta-Learning Blockchain Performance Prediction Model: In this experiment, we consider that the smaller the number of training samples required, the better the scalability of the proposed meta-learning-based blockchain prediction model. Therefore, in the comparison experiments in this section, we consider the sample size and training effect together, we set the sample size of the experiments to 20-shot, and in this case, we conduct a full experimental comparison and analysis with other existing approaches or related work.

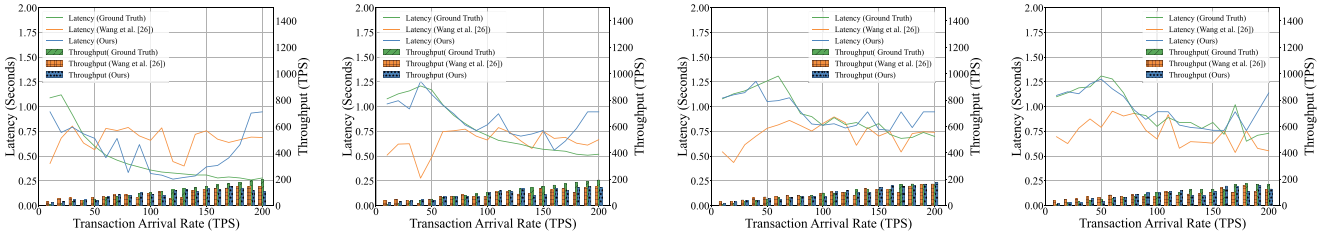
We only provide 20 data samples for model training, the results of the experiments conducted on BPD are shown in Table V. In such a case, the advantages of our proposed meta-learning-based blockchain performance prediction model are very obvious, and SOTA is achieved on all experimental results, and the error has been reduced to a low level, which can meet the needs of practical scenarios and validates the feasibility and effectiveness of our proposed approach compared to existing approaches and related work.

The prediction performance of the existing SOTA work and different traditional machine learning models is very poor when the amount of data available for training is very small. Compare this to the case in the experiment (2) where sufficient data was provided for training to obtain relatively accurate predictions. In this experiment, because the amount of data available for training is only 20, only our proposed meta-learning-based blockchain performance prediction model achieves relatively low prediction errors, which fully demonstrates the clear advantages and high scalability of our proposed method when only a small amount of blockchain performance data is available.

We also conduct experiments on HFBTP. The experimental results are shown in Table VI and Fig. 9. Our proposed meta-learning-based blockchain performance prediction model still achieves superior prediction results compared to existing SOTA work. Fully comprehensive experiments on multiple datasets show that our proposed model is still working even when the amount of data available for training is very small,

TABLE V
THE PREDICTION PERFORMANCE OF DIFFERENT MODELS ON BPD (20-SHOT)

Tasks	Methods	BPD-1			BPD-2			BPD-3			BPD-4		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Latency	LR	0.494	0.638	56.54	0.542	0.787	64.98	0.619	0.952	69.28	0.585	0.866	75.89
	ABR	0.347	0.518	35.10	0.329	0.496	41.22	0.415	0.657	46.46	0.369	0.567	45.50
	DTR	0.348	0.518	35.07	0.353	0.525	44.42	0.426	0.676	45.16	0.371	0.570	43.57
	KNR	0.348	0.514	37.81	0.290	0.449	47.15	0.385	0.639	59.28	0.370	0.578	68.83
	GBR	0.348	0.518	35.12	0.335	0.507	40.85	0.416	0.663	44.94	0.369	0.570	44.19
	BR	0.345	0.512	36.09	0.299	0.471	38.24	0.399	0.637	46.97	0.319	0.506	42.55
	RFR	0.345	0.513	36.00	0.316	0.486	40.26	0.399	0.645	45.70	0.358	0.551	47.08
	Wang et al. [26]	0.421	0.532	56.30	0.398	0.580	57.48	0.484	0.721	72.31	0.535	0.751	87.59
	Ours	**0.185**	**0.304**	**20.02**	**0.216**	**0.364**	**26.34**	**0.229**	**0.465**	**28.35**	**0.241**	**0.411**	**31.45**
Throughput	LR	111.1	129.2	113.3	26.36	33.70	26.67	27.54	36.10	27.85	26.82	35.10	26.83
	ABR	76.29	87.32	78.02	44.29	56.35	38.84	44.17	56.04	38.64	44.01	55.87	38.20
	DTR	76.29	87.32	78.02	43.79	55.87	38.33	44.32	56.20	38.79	43.66	55.52	37.84
	KNR	76.32	87.34	78.05	48.27	59.31	47.10	48.35	59.21	47.12	48.96	59.88	47.35
	GBR	76.29	87.32	78.02	43.88	55.93	38.45	43.97	55.88	38.43	44.24	56.12	38.44
	BR	76.30	87.33	78.03	44.76	56.70	39.83	44.96	56.74	39.83	45.84	57.61	40.56
	RFR	76.30	87.33	78.03	45.00	56.94	40.11	45.43	57.21	40.49	45.77	57.55	40.58
	Wang et al. [26]	63.43	70.13	71.58	74.56	83.40	79.07	77.57	86.30	83.21	77.60	86.58	82.07
	Ours	**12.30**	**17.37**	**15.29**	**8.201**	**11.74**	**11.21**	**11.29**	**15.74**	**15.69**	**11.11**	**15.49**	**14.26**



(a) (b) (c) (d)

Fig. 9. The prediction performance of blockchain performance prediction model on HFBTP (20-shot). (a) block size = 50 (b) block size = 100 (c) block size = 150 (d) block size = 200.

TABLE VI
THE PREDICTION PERFORMANCE OF DIFFERENT MODELS ON HFBTP (20-SHOT)

Tasks	Methods	MAE	RMSE	MAPE
Latency	Wang et al. [26]	0.183	0.244	26.36
	Ours	**0.083**	**0.124**	**13.09**
Throughput	Wang et al. [26]	19.41	24.98	38.94
	Ours	**13.81**	**19.00**	**19.29**

TABLE VII
THE OPTIMIZATION RATE OF OPTIMAL BLOCK SIZE ON LATENCY

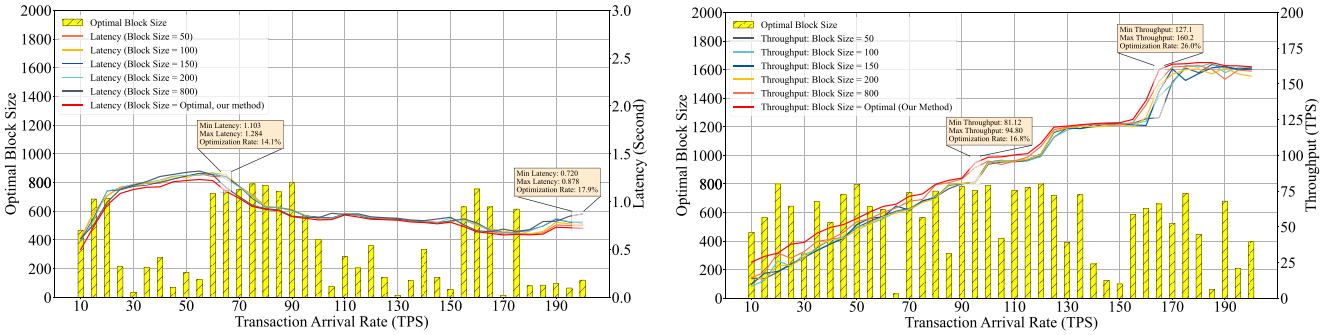
Block Size	50	100	150	200	800	**Optimal (Ours)**
Average Latency (Seconds)	0.915	0.920	0.920	0.923	0.917	**0.871**
Optimization Rate (%)	4.800	5.326	5.326	5.633	5.016	

which will effectively reduce the cost and threshold of the blockchain performance optimization approach proposed in this paper, thus further enhancing the scalability of this work.

4) The Effectiveness of Scoring Optimization Mechanism for Blockchain Performance: In the previous content, we propose two learning-based blockchain performance prediction models and prove that these models have excellent prediction results through sufficient experiments and analyses. Therefore, this part demonstrates experimentally that our proposed blockchain performance scoring optimization mechanism can efficiently and feasibly accomplish the optimization of blockchain performance.

We first verify that our proposed blockchain performance scoring optimization mechanism can find and tune the optimal blockchain parameters to obtain the optimal performance. In this experiment, instead of using a weight adjustment strategy, we set the weights manually to fully validate the effectiveness of our proposed method and to ensure the uniformity of the experimental parameters.

We choose five block sizes (50, 100, 150, 200, 800) as control experiments. When we have latency as the primary optimization goal, we set $Weight^{Lat} = 1$, and $Weight^{Thr} = 0$, the experimental results are shown in Fig. 10 (a). For the same transaction arrival rate, setting different block sizes (and other parameters that may affect blockchain performance) will yield different performances. Compared to setting block sizes manually, our proposed method can dynamically choose the optimal block size for different transaction arrival rates, thus achieving the lowest latency. For example, when the transaction arrival rate is 30 TPS, the lowest latency can



(a) (b)

Fig. 10. The performance of scoring optimization mechanism. (a) Latency as the primary optimization goal ($Weight^{Lat} = 1$, $Weight^{Thr} = 0$) (b) Throughput as the primary optimization goal ($Weight^{Lat} = 0$, $Weight^{Thr} = 1$).

TABLE VIII
THE OPTIMIZATION RATE OF OPTIMAL BLOCK SIZE ON THROUGHPUT

Block Size	50	100	150	200	800	**Optimal (Ours)**
Average Throughput (TPS)	91.50	91.77	91.89	91.89	94.87	**98.72**
Optimization Rate (%)	7.890	7.573	7.432	7.432	4.058	

be obtained by setting the block size to 310; when the transaction arrival rate is 35 TPS, the block size needs to be adjusted to 675 to achieve the lowest latency. As shown in Table VII, our proposed method improves the latency by about **5%**. It is worth noting that our proposed method yields very significant latency optimization (**14.1%** and **17.9%**, respectively) when the transaction arrival rate is 65 TPS and 200 TPS, respectively. When we have throughput as the primary optimization goal, we set $Weight^{Thr} = 1$, and $Weight^{Lat} = 0$, the experimental results are shown in Fig. 10 (b). Compared to setting block size manually, our proposed method can dynamically choose the optimal block size for different transaction arrival rates, thus achieving the highest throughput. For example, when the transaction arrival rate is 80 TPS, the highest throughput can be obtained by setting the block size to 750; when the transaction arrival rate is 85 TPS, the block size needs to be adjusted to 315 to achieve the highest throughput. As shown in Table VIII, our proposed method improves the throughput by about **7%**. It is worth noting that our proposed method yields very significant throughput optimization (**16.8%** and **26.0%**, respectively) when the transaction arrival rate is 95 TPS and 165 TPS, respectively. The reason for the above may be that there is a pronounced difference in performance corresponding to different block sizes when the transaction arrival rate is the above value, which may be because there is indeed a pronounced difference in the actual performance, or it may be due to the potential bias that arises from the collection of blockchain performance data by Hyperledger Caliper. This also illustrates that our proposed method relies on dataset quality (dataset size, fidelity). Therefore, providing sufficiently detailed blockchain performance datasets will aid research in this field.

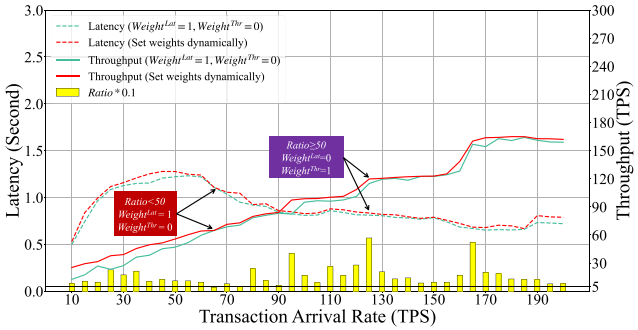


Fig. 11. The flexibility of our proposed scoring optimization mechanism.

Moreover, we verify that our proposed mechanism has higher flexibility because our proposed mechanism can dynamically adjust the weights. we set $Thres^{Ratio} = 50$, $Thres^{Lat} = 1$, $Weight_1^{Thr} = 1$, $Weight_1^{Lat} = 1$, $Weight_2^{Thr} = 0.5$, and $Weight_2^{Lat} = 0.5$, the experimental results are shown in Fig. 11. In this experiment, we set the weights according to the proposed strategies. Firstly, when *Ratio* is greater than or equal to $Thres^{Ratio}$, it means that the benefits of optimizing throughput are much greater than the benefits of optimizing latency, so we set $Weight^{Thr} = 1$ and $Weight^{Lat} = 0$. When *Ratio* is less than $Thres^{Ratio}$ and Lat^{Min} is less than $Thres^{Lat}$, it means that the benefits of optimizing throughput are not significant, so we set $Weight^{Thr} = 0$ and $Weight^{Lat} = 1$, we take latency as the main optimization orientation. We choose the *BPC* corresponding to Lat^{Min} as the optimal blockchain performance parameters configuration.

In the scoring optimization method proposed in [26], the settings of $Weight^{Lat}$ and $Weight^{Thr}$ are inflexible, and they are not adjusted based on the judgment of specific situations, but our proposed method has higher flexibility and can be judged based on pre-set situations (from the perspective of latency thresholds or the benefits of performance tuning), thus flexibly adjusting the setting of weights. This also proves that our proposed blockchain performance scoring optimization approach has better scalability and flexibility, and is more suitable for blockchain performance tuning in different practical scenarios.

TABLE IX
THE COMPARISON WITH RELATED WORK

Work	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Jamil et al. [23]	✓	✓	✓	✓	✗	✗	✻	✗
Chen et al. [24]	✗	✗	✓	✗	✓	✗	✻	✗
Wilhelmi et al. [25]	✗	✗	✓	✗	✓	✗	✻	✗
Wang et al. [26]	✓	✓	✓	✗	✓	✓	✗	✗
Ours	✓	✓	✓	✓	✓	✓	✓	✓

C_1: Not limited to specific blockchain platforms. **C_2:** Optimization of throughput. **C_3:** Optimization of latency. **C_4:** High scalability. **C_5:** Open-sourced code. **C_6:** Open-sourced dataset. **C_7:** Suitable for few-shot. **C_8:** High optimization flexibility.
Note: ✓ is "**Yes**" (related to this content), ✗ is "**No**" (not related to this content). ✻ is "**Unknown**" (can not be identified).

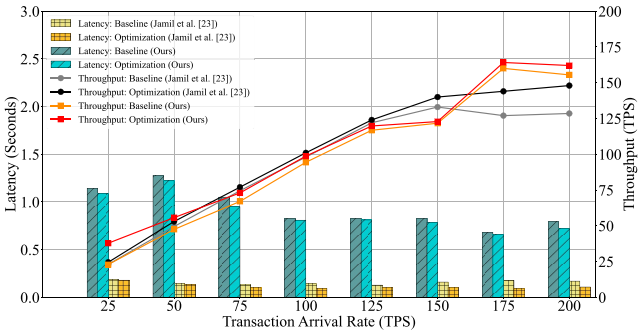


Fig. 12. The performance optimization effect of LearningChain and [23].

C. Significance and Comparison

To fully demonstrate the innovation and effectiveness of LearningChain, we compare it with four related works. For convenience, we qualitatively compare LearningChain with related work (traditional optimization algorithms and machine learning-based optimization) in terms of eight characteristics. The qualitative comparison results are shown in Table IX. First, in previous experiments, we demonstrate in full comparison with [26] that LearningChain has higher predictive performance, scalability, and optimization flexibility and that it can be applied to the case of a few sample data.

In addition, to further demonstrate the performance and innovativeness of LearningChain, we compare it quantitatively with [23]. To ensure the fairness of the comparison as much as possible, we choose the result in [23] (Hyperledger Fabric 1.4.1, Hyperledger Caliper 2.0.0, two organizations, two peers, one client) that is closest to the experimental setting of LearningChain. Since blockchain performance is affected by the performance of computing devices, there will be some differences in blockchain performance in different experimental environments. For this reason, we mainly compare the respective baselines (block sizes are 100 and 200, in [23] and LearningChain, respectively) with the optimized performance. The comparison results are shown in Fig. 12, both methods achieve optimization of blockchain performance, but when the transaction arrival rate is small (25 to 125 TPS), the optimization effect of [23] is not obvious compared to its baseline. In contrast, our proposed method achieves more significant optimization results most of the time, especially when the transaction arrival rate is small. In [23], the blockchain performance optimization is mainly done by intelligently controlling the transaction arrival rate, while in LearningChain, the dynamic blockchain performance optimization is mainly done by adjusting the blockchain parameters (e.g., block size). Therefore, the two optimization methods do not conflict and can be effectively integrated to further optimize the performance of the blockchain. The comparison with related work effectively demonstrates the innovation, effectiveness, and scalable of LearningChain.

D. Discussion, Analysis, and Limitations

Maintainers and users of different blockchain platforms only need to collect a certain number of performance datasets to use LearningChain for blockchain performance optimization. Therefore, LearningChain is not limited to a specific blockchain platform. LearningChain is easy to understand and does not require much of a technical threshold. In addition, LearningChain can be integrated and complemented with other existing optimization methods (e.g., improvement of consensus algorithms, blockchain sharding, etc.) to further improve the effectiveness of optimization. For the consensus-based performance optimization method, Zhang et al. [19] proposed the election of representative nodes to complete the packing and confirmation of blocks, which reduces the number of master nodes, thus reducing the block broadcasting time and optimizing the performance of the blockchain. When the number of representative nodes is at a small scale, our method can quickly complete the block size adjustment on this basis, thus further optimizing the blockchain performance. As for the sharding-based performance optimization method, for a transaction, since it can be verified and confirmed by only a small portion of the nodes in the blockchain network [21], the block size can also be adjusted quickly for this portion of the nodes due to the limited number of nodes, thus further optimizing the blockchain performance.

Since the larger the block size (the longer the block broadcast time), the probability of forking rises once a large number of transactions occur when the nodes in the blockchain network have a poor network environment. Therefore, in this paper, the main scenario we are oriented to is the consortium blockchain with a limited number of main nodes and more trustworthy, for example, Ethereum (adopts the delegated proof of stake consensus mechanism), and Hyperledger Fabric (adopts the Raft consensus mechanism). Since the number of nodes is limited and nodes tend to be served by computationally powerful devices, the overhead of completing a blockchain parameter adjustment is relatively small.

For public blockchain, especially Bitcoin, which adopts the proof of work consensus mechanism, due to the presence of a large number of mining (master) nodes with different computational capabilities and network bandwidths, frequent blockchain parameter adjustments may cause the blockchain

network to fork, resulting in anomalies in the ledger maintained by individual nodes. In addition, due to the maintenance of a large number of high-value cryptocurrencies, there may be cases of malicious nodes doing evil. As a result, frequent resizing of blocks (especially increasing the block size) may expose the blockchain to the risk of forking, which can lead to security problems. Therefore, our current proposed method may not be suitable for public blockchain platforms with many nodes widely distributed worldwide.

In realistic scenarios, we will minimize the impact of blockchain configuration adjustment on the blockchain network by controlling the time of block size adjustment (i.e., the frequency of adjustment) and limiting the range of adjustable parameters (e.g., block size up to 800), to keep the blockchain running stably.

V. Conclusion

In this paper, to further improve the accuracy and efficiency of blockchain performance optimization, we propose a blockchain performance optimization framework (LearningChain). We use TCN to predict the transaction arrival rate for a certain period in the future and use it as a subsequent input. We propose an ensemble learning-based method and a meta-learning-based method to train a blockchain performance prediction model, respectively. These two methods are each applicable to different data sample sizes. On this basis, we design a performance scoring mechanism to dynamically optimize the configuration parameters of the blockchain to improve the blockchain performance. In addition, we collect and contribute a blockchain performance dataset for other researchers to research. The adequate experimental results and analysis, as well as qualitative and quantitative comparisons with related work, demonstrate the validity and feasibility of our proposed method. Our proposed method applies not only to Hyperledger Fabric but also to other blockchain platforms with a similar nature.

In future work, we will study the parameters affecting blockchain performance more deeply, further find the relationship between them, and apply them to some suitable scenarios to solve some urgent problems. In any case, the work still has some shortcomings that can be further investigated in the future, including but not limited to, reducing the load and cost caused by blockchain parameter tuning, specific performance optimization schemes for public blockchains with a large number of nodes, and more efficient and accurate performance optimization mechanisms and models.

Acknowledgment

The authors would like to express their utmost appreciation to the anonymous reviewers for their invaluable and insightful comments, which have significantly contributed to the refinement and enhancement of the quality of this article. They also express their appreciation to the Editor-in-Chief, Associate Editor, and all potential participants for their valuable contributions in advancing the article review process.

References

[1] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021.

[2] Z. Lv, L. Qiao, M. S. Hossain, and B. J. Choi, "Analysis of using blockchain to protect the privacy of drone big data," *IEEE Netw.*, vol. 35, no. 1, pp. 44–49, Jan./Feb. 2021.

[3] S. R. Niya, D. Dordevic, A. G. Nabi, T. Mann, and B. Stiller, "A platform-independent, generic-purpose, and blockchain-based supply chain tracking," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2019, pp. 11–12.

[4] L. Tan, K. Yu, N. Shi, C. Yang, W. Wei, and H. Lu, "Towards secure and privacy-preserving data sharing for COVID-19 medical records: A blockchain-empowered approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 271–281, Jan./Feb. 2022.

[5] L. Xue, H. Huang, F. Xiao, and W. Wang, "A cross-domain authentication scheme based on cooperative blockchains functioning with revocation for medical consortiums," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 2409–2420, Sep. 2022.

[6] M. Du, Q. Chen, J. Chen, and X. Ma, "An optimized consortium blockchain for medical information sharing," *IEEE Trans. Eng. Manag.*, vol. 68, no. 6, pp. 1677–1689, Dec. 2021.

[7] J. Cui, F. Ouyang, Z. Ying, L. Wei, and H. Zhong, "Secure and efficient data sharing among vehicles based on consortium blockchain," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8857–8867, Jul. 2022.

[8] J. Wang, R. Zhu, T. Li, F. Gao, Q. Wang, and Q. Xiao, "ETC-oriented efficient and secure blockchain: Credit-based mechanism and evidence framework for vehicle management," *IEEE Trans. Veh. Technol.*, vol. 70, no. 11, pp. 11324–11337, Nov. 2021.

[9] Z. Zhou, M. Wang, J. Huang, S. Lin, and Z. Lv, "Blockchain in big data security for intelligent transportation with 6G," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9736–9746, Jul. 2022.

[10] M. U. Hassan, M. H. Rehmani, and J. Chen, "Optimizing blockchain based smart grid auctions: A green revolution," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 462–471, Mar. 2022.

[11] T. V. Le, C. L. Hsu, and W. X. Chen, "A hybrid blockchain-based log management scheme with nonrepudiation for smart grids," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 5771–5782, Sep. 2022.

[12] K. Zhou, J. Chong, X. Lu, and S. Yang, "Credit-based peer-to-peer electricity trading in energy blockchain environment," *IEEE Trans. Smart Grid*, vol. 13, no. 1, pp. 678–687, Jan. 2022.

[13] B. Wang, Z. Chang, S. Li, and T. Hämäläinen, "An efficient and privacy-preserving blockchain-based authentication scheme for low earth orbit satellite-assisted Internet of Things," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 6, pp. 5153–5164, Dec. 2022.

[14] C. Qiu, H. Yao, C. Jiang, S. Guo, and F. Xu, "Cloud computing assisted blockchain-enabled Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 247–257, Jan.–Mar. 2022.

[15] A. Ceccarelli, M. Cinque, C. Esposito, L. Foschini, C. Giannelli, and P. Lollini, "FUSION—Fog computing and blockchain for trusted Industrial Internet of Things," *IEEE Trans. Eng. Manag.*, vol. 69, no. 6, pp. 2944–2958, Dec. 2022.

[16] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "BAFL: A blockchain-based asynchronous federated learning framework," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1092–1103, May 2022.

[17] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[18] N. Gao, R. Huo, S. Wang, T. Huang, and Y. Liu, "Sharding-hashgraph: A high-performance blockchain-based framework for Industrial Internet of Things with hashgraph mechanism," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17070–17079, Sep. 2022.

[19] W. Zhang et al., "A trustworthy safety inspection framework using performance-security balanced blockchain," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8178–8190, Jun. 2022.

[20] J. Xu, Q. Xie, S. Peng, C. Wang, and X. Jia, "Adaptchain: Adaptive scaling blockchain with transaction deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 6, pp. 1909–1922, Jun. 2023.

[21] Z. Cai et al., "Benzene: Scaling blockchain with cooperation-based sharding," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 2, pp. 639–654, Feb. 2023.

[22] M. Li, W. Wang, and J. Zhang, "LB-Chain: Load-balanced and low-latency blockchain sharding via account migration," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 10, pp. 2797–2810, Oct. 2023.

[23] F. Jamil, S. Ahmad, T. K. Whangbo, A. Muthanna, and D.-H. Kim, "Improving blockchain performance in clinical trials using intelligent optimal transaction traffic control mechanism in smart healthcare applications," *Comput. Ind. Eng.*, vol. 170, Aug. 2022, Art. no. 108327.
[24] X. Chen, K. Nguyen, and H. Sekiya, "On the latency performance in private blockchain networks," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19246–19259, Oct. 2022.
[25] F. Wilhelmi, S. Barrachina-Muñoz, and P. Dini, "End-to-end latency analysis and optimal block size of proof-of-work blockchain applications," *IEEE Wireless Commun. Lett.*, vol. 26, no. 10, pp. 2332–2335, Oct. 2022.
[26] J. Wang et al., "BPR: Blockchain-enabled efficient and secure parking reservation framework with block size dynamic adjustment method," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3555–3570, Mar. 2023.
[27] Z. H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.
[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, Aug. 2017, pp. 1126–1135.
[29] M. Chen, Y. Wang, and X. Zhu, "Few-shot website fingerprinting attack with meta-bias learning," *Pattern Recognit.*, vol. 130, Oct. 2022, Art. no. 108739.
[30] M. Cheng, H. Wang, and Y. Long, "Meta-learning-based incremental few-shot object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 2158–2169, Apr. 2022.
[31] Y. Hu, R. Liu, X. Li, D. Chen, and Q. Hu, "Task-sequencing meta learning for intelligent few-shot fault diagnosis with limited data," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 3894–3904, Jun. 2022.
[32] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arxiv.abs/1803.01271*.
[33] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat. (AISTATS)*, Apr. 2011, p. 4.
[34] L. M. Zintgraf, K. Shiarlis, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, vol. 97, Jun. 2019, pp. 7693–7702.

Jishu Wang (Member, IEEE) received the B.E. degree in computer science and technology from Kunming University in 2019, and the M.E. degree in software engineering from Yunnan University, Kunming, China, in 2022, where he is currently pursuing the Ph.D. degree with the School of Information Science and Engineering. He has authored peer-reviewed papers in IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. His current research interests include blockchain and intelligent transportation systems.

Yaowei Wang received the B.E. degree in network engineering from the Nanyang Institute of Technology, Nanyang, China, in 2019, and the M.E. degree in software engineering from Yunnan University, Kunming, China, in 2022, where he is currently pursuing the Ph.D. degree with the School of Information Science and Engineering. He has authored peer-reviewed papers in IEEE JOURNAL OF TRANSLATIONAL ENGINEERING IN HEALTH AND MEDICINE and *Neural Computing and Applications*. His current research interests include deep learning, multimodal learning, and computer vision.

Xuan Zhang (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in system analysis and integration from Yunnan University, Kunming, China, where she is a Professor with the School of Software. She is a Core Scientist of the Yunnan Key Laboratory of Software Engineering and Yunnan Software Engineering Academic Team. She has been a principal investigator for more than 30 national, provincial, and private grants and contracts. She is the author of three books and more than 90 articles. Her research interests include blockchain, knowledge graph, natural language processing, recommendation system, and computer vision.

Zhi Jin (Fellow, IEEE) received the B.S. degree in computer science from Zhejiang University in 1984, the M.S. degree in computer science from the Changsha Institute of Technology in 1987, and the Ph.D. degree in computer science from the Changsha Institute of Technology in 1992. She is currently a Professor of Computer Science with Peking University, where she is the Deputy Director of the Key Laboratory of High Confidence Software Technologies (Ministry of Education). She is interested in software engineering, requirements engineering, knowledge engineering, and machine learning. She has received the IEEE TCSVC Distinguished Leadership Award, the ACM Distinguished Paper Awards (four times), and published three monographs. She serves as an Associate Editor for IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON RELIABILITY, *ACM Transactions on Autonomous and Adaptive Systems*, *Empirical Software Engineering*, and *Requirements Engineering*. She is also a Fellow of CCF and AAIA.

Chao Zhu (Member, IEEE) received the B.E. degree in new energy science and engineering from Hohai University, Nanjing, China, in 2017, and the M.E. degree in software engineering from Yunnan University, Kunming, China, in 2023. He is currently working with BYD Company, Shenzhen, China. He has authored peer review paper in IEEE SANER. His current research interests include spatiotemporal data mining and traffic prediction.

Linyu Li (Student Member, IEEE) received the B.E. degree in software engineering from Yunnan University, Kunming, China, in 2024. He is currently pursuing the Ph.D. degree with the School of Computer Science, Peking University, Beijing, China. He has authored peer-reviewed papers in *Information Sciences* and *Knowledge-Based Systems*. His current research interests include using deep learning to solve knowledge graph-related problems and requirements engineering.

Rui Zhu (Member, IEEE) received the Ph.D. degree in software engineering from Yunnan University, Kunming, China, in 2016, where he is currently the Head and an Associate Professor of Artificial Intelligence with the School of Software. His current research interests include intelligent transportation systems, blockchain, and deep learning.

Shenglong Lv (Student Member, IEEE) received the B.E. degree in network engineering from Qufu Normal University, Qufu, China, in 2022. He is currently pursuing the M.E. degree with the School of Software, Yunnan University, Kunming, China. His current research interests include blockchain, smart grid, and cyberspace security.