World Wide Web (2023) 26:4173–4191
https://doi.org/10.1007/s11280-023-01212-9

Enhancing bitcoin transaction confirmation prediction: a hybrid model combining neural networks and XGBoost

Limeng Zhang[1] · Rui Zhou[1] · Qing Liu[2] · Jiajie Xu[3] · Chengfei Liu[1] · Muhammad Ali Babar[4]

Received: 30 April 2023 / Revised: 12 September 2023 / Accepted: 14 September 2023 /
Published online: 26 December 2023

Abstract
With Bitcoin being universally recognized as the most popular cryptocurrency, more Bitcoin transactions are expected to be populated to the Bitcoin blockchain system. As a result, many transactions can encounter different confirmation delays. Concerned about this, it becomes vital to help a user understand (if possible) how long it may take for a transaction to be confirmed in the Bitcoin blockchain. In this work, we address the issue of predicting confirmation time within a block interval rather than pinpointing a specific timestamp. After dividing the future into a set of block intervals (i.e., classes), the prediction of a transaction's confirmation is treated as a classification problem. To solve it, we propose a framework, Hybrid Confirmation Time Estimation Network (**Hybrid-CTEN**), based on neural networks and XGBoost to predict transaction confirmation time in the Bitcoin blockchain system using three different sources of information: historical transactions in the blockchain, unconfirmed transactions in the mempool, as well as the estimated transaction itself. Finally, experiments on real-world blockchain data demonstrate that, other than XGBoost excelling in the binary classification case (to predict whether a transaction will be confirmed in the next generated block), our proposed framework Hybrid-CTEN outperforms state-of-the-art methods on precision, recall and f1-score on all the multiclass classification cases (4-class, 6-class and 8-class) to predict in which future block interval a transaction will be confirmed.

Keywords Transaction confirmation time · Bitcoin · Blockchain · XGBoost · Neural network

1 Introduction

As Bitcoin is universally recognized by more organisations, institutes and governments, it is booming in an increasing number of areas [1]. Currently, many businesses, such as PayPal,

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2022*
Guest editors: Richard Chbeir, Helen Huang, Yannis Manolopoulos and Fabrizio Silvestri.

✉ Rui Zhou
rzhou@swin.edu.au

Extended author information available on the last page of the article

Microsoft, and Overstock, have embraced Bitcoin as a method of payment. Meanwhile, various online cryptocurrency trading platforms, such as Coinbase, Gemini[1], and PayPal, have enabled users to purchase, sell, store, and transfer Bitcoins. As a result, more Bitcoin exchanges are expected to be populated into the Bitcoin blockchain. Unfortunately, due to the confirmation mechanism in the system, only a limited number of transactions (restricted to the capacity of a block) can be confirmed at a time. Therefore, many transactions cannot be immediately confirmed, and confirmation delays commonly occur in the Bitcoin system. Concerned about this, it becomes vital to help a user to understand (if possible) how long it may take for a transaction to be confirmed in the Bitcoin blockchain.

Most previous attempts at estimating the confirmation time for a transaction focus on predicting a specific timestamp or predicting the number of blocks a transaction needs to wait for before it is confirmed [2–9]. However, it is usually more practical to predict the confirmation time as falling into the corresponding predefined time intervals (e.g., within 1 hour, between 1 hour and 4 hours, and more than 4 hours). It is motivated by the following considerations: On one hand, when attempting to estimate a specific timestamp, one issue is that the estimation performance can be affected by the submission time, especially for transactions that are scheduled for confirmation in the subsequent block. The confirmation time for these transactions is influenced by the remaining time before the next block is produced. Consequently, this can lead to a situation where, as a result of delayed submission, a transaction with a significantly higher fee can experience a longer delay than a transaction with a lower fee if the higher fee one is submitted later than the lower-fee one. The second issue arises from the unpredictable nature of block generation time, which can span from mere seconds to several hundred seconds. As a result, the confirmation time for two transactions submitted at different block heights but confirmed within the same block interval can exhibit unpredictable differences, which may undermine users' satisfaction when using a client-side transaction system.

On the other hand, by utilizing the block as the unit of measurement for confirmation time, the variance in confirmation time can be significantly diminished. However, a challenge arises as the estimation result can be heavily influenced by a small proportion of transactions, especially when there is a scarcity of historical transactions for that interval. In such cases, the estimation result may become highly dependent on a single or a few transactions. Moreover, when the estimated confirmation time (in terms of both a specific time and a block interval) exceeds a certain level, users tend to pay a higher transaction fee to prioritize the confirmation process. In conclusion, we suggest that as long as the confirmation time falls within an acceptable range, it may be more practical and reasonable to estimate a confirmation time range rather than a confirmation time stamp to system users. Under such background, if we divide the future into a number of block intervals (representing a number of classes), the confirmation time prediction problem can be considered as a classification problem.

The accuracy of transaction confirmation time estimation is crucial for blockchain-based applications. However, existing efforts suffer from four key drawbacks in their frameworks: (1) The existing methods for transaction confirmation estimation do not provide tailored estimates for individual transactions. Instead, most of them estimate the confirmation time for a group of transactions. For example, some works such as [5, 6] estimate the average confirmation time of high-feerate class transactions and low-feerate transactions, while others like [8] estimate the average confirmation time of all the unconfirmed transactions. (2) Models proposed in [3, 10] predict only whether a transaction can be confirmed in the next block, treating the problem as a binary classification task. However, such models may not be

[1] https://www.gemini.com

sufficient in practice as they do not provide more detailed confirmation information beyond a simple yes or no. (3) Some of the assumptions made in existing approaches [2, 4–8] are not realistic. For example, the confirmation process in blockchain systems is often modeled as a steady-state queueing system [11]. They assume that transaction submission is slower than transaction confirmation, and a fixed number of transactions can be confirmed each time. However, in reality, the number of transactions in a block can vary, and the rate of transaction submissions can exceed the rate of confirmations. (4) There is insufficient utilisation of information on transactions, blocks, and mempool, which can provide further information on the current blockchain system. For example, information in the block sequence can signal the size and generation rate of future blocks, which can help improve the estimation accuracy.

To address the limitations in the previous works, we propose a framework based on neural networks and XGBoost (Extreme Gradient Boosting) [12] to estimate transaction confirmation time for a specific transaction. This framework draws upon three different sources of information: historical transactions within the blockchain, unconfirmed transactions in the mempool, and the estimated transaction itself. In this framework, neural networks are applied to identify complex structures and generate high-level concepts from these inputs. Subsequently, XGBoost is employed to perform classification based on the hidden patterns derived from the neural networks.

To summarize, we have made the following contributions:

- We comprehensively examine the features of historical transactions in the blockchain, unconfirmed transactions in the mempool, as well as the estimated transaction itself features related to transaction confirmation.
- We design a strategy to discretize confirmation time into non-overlapping intervals based on transaction distribution.
- We develop a transaction confirmation time estimation framework Hybrid-CTEN based on neural networks and XGBoost to analyze transaction confirmation time in the Bitcoin blockchain.
- We demonstrate the efficiency and effectiveness of the proposed framework Hybrid-CTEN in handling complex estimation tasks using real-world blockchain data.

This work builds upon our prior work [13] by proposing a new transaction confirmation time framework based on neural networks and XGBoost. In this work, we also incorporate transformer [14] as an alternative technique for feature extraction, leveraging its well-established efficiency in handling sequential data. Furthermore, during the feature construction process, we have optimized our approach for modeling transaction distribution to better align with the characteristics of real-world blockchain data in both block and mempool analyses. Moreover, we have adopted the strategy of training a model for the entire testing dataset, instead of retraining at each block height, and we have meticulously fine-tuned hyperparameters to achieve improved performance.

The rest of this paper is organized as follows: Section 2 provides a review of the related work and Section 3 defines the transaction confirmation time estimation problem. Section 4 presents our proposed framework Hybrid-CTEN for transaction time estimation. Section 5 details our experimental setup and evaluation. Finally, Section 6 concludes this paper.

2 Related work

Many approaches have been proposed to estimate the transaction confirmation time in the Bitcoin blockchain. In [3, 10], the authors approached the confirmation time estimation problem as a binary classification task, focusing on predicting whether a transaction can be confirmed in the upcoming block. They employed supervised learning models such as SVM, random forest, and AdaBoost to estimate the confirmation delay. This estimation was based on two key factors: the characteristics of the transaction itself and the characteristics of the unconfirmed transactions in the mempool. The latter was described through metrics such as transaction count and transaction feerate distribution.

Other studies such as [2, 4–8] have adopted a different approach by analyzing the distribution of transaction submission and confirmation. Among these, [5, 6] approach the estimation problem by modeling it as a bulk service queueing system denoted as $M/G^B/1$, where transaction arrivals follow the Poisson distribution and batches of transactions (B) are confirmed at a rate with a specified distribution. Balsamo et al. [2] describe it as another type of bulk service queueing systems, $M/M^B/1$, with transaction arrival following the Poisson distribution and the confirmation of batches following an exponential distribution. Zhao et al. [8] introduce the concept of a zero-transaction service within the traditional bulk queueing system, accounting for the possibility of a block containing zero transactions. It assumes that transaction arrivals follow a Poisson distribution, and batch confirmations adhere to a stochastic density function. Apart from these queueing system-based solutions, [4, 7] model the confirmation process as a Cramér-Lundberg process with a fixed rate of transaction arrivals and an exponential distribution for the confirmation of a predetermined number of transactions. In the work [13], transaction confirmation time was approached as a classification problem. They conducted an extensive comparison of various models, including neural networks, ensemble learning models, and a method that solely considers transaction fees.

3 Problem definition

Previous studies have demonstrated that transaction confirmation is a complex process that depends on various factors, such as the characteristics of the transaction, the competition among unconfirmed transactions in the mempool, mining policy, and system resources.

Given a newly submitted transaction $\hat{tx}$, the goal is to predict its confirmation time (interval) denoted as $y \in \{y_1, \cdots, y_n\}$, where $\{y_1, \cdots, y_n\}$ constitute a collection of non-overlapping confirmation time intervals, collectively representing the future timeline. $\mathcal{F}$ is the designed function to estimate the confirmation time within which the submitted transaction will likely be confirmed, relying on various sources of information.

$$y = \mathcal{F}(\text{TxInf}(\hat{tx}), \text{BlockInfo}, \text{MemInfo})$$

where,

- **TxInf**($\hat{tx}$) contains the information of transactions that is related to transaction validation and confirmation in the network, such as transaction feerate, transaction weight, transaction inputs, and submission time.
- **BlockInfo** provides information on the characteristics of mined blocks, such as historical transaction feerate, block size, and transaction distribution. These characteristics can implicitly reflect the volume and mining preference of mining activity, which can be helpful in predicting future confirmation time.

- **MemInfo** provides information about unconfirmed transactions in the mempool, which is a temporary storage area for transactions that have been broadcasted to the Bitcoin network but are yet to be included in a block. As the capacity of a block is limited in the Bitcoin system, submitted transactions compete with each other to be confirmed in the next block.

4 Methodology

Hybrid-CTEN is a framework based on neural networks and XGboost to estimate the transaction confirmation time in the Bitcoin blockchain. It operates by taking the estimated transaction itself, confirmed transactions in the blockchain, and unconfirmed transactions in the mempool as inputs, and generates the estimated confirmation time for this transaction.

Figure 1 presents the overview workflow of Hybrid-CTEN. It consists of three components: a Data Preprocessing Module, a Feature Extraction Module, and a Confirmation Time Estimation Module. The *Data Preprocessing Module* encompasses the construction of features derived from the blockchain, which holds records of confirmed transactions in the blocks, mempool information representing unconfirmed transactions, and transaction data encompassing transaction-specific characteristics. Meanwhile, within this module, the entire confirmation time spectrum is discretized into distinct block intervals. In the subsequent *Feature Extraction Module*, the constructed features from blocks, mempool, and transactions are amalgamated to unveil underlying patterns. These extracted patterns are then relayed to the *Confirmation Time Estimation Module* for the final estimation process.

4.1 Data preprocessing module

This module is employed for constructing features from transactions, blocks and mempool. It also outlines the procedure of discretizing continuous confirmation time into intervals.

4.1.1 Feature construction

By analyzing the confirmation process in the Bitcoin blockchain, we identified three factors that contribute to transaction confirmation, namely, transaction features, block states, and mempool states.

- **Transaction features** describe the unique details of a submitted transaction. We select the features that we believe may affect a transaction's validation and confirmation.
  - *transaction weight:* measures transaction size[2].
  - *transaction feerate:* refers to the transaction fee per size unit, with each unit being approximately equivalent to a quarter of a transaction weight unit. Generally, transactions with higher feerates are considered confirmed earlier than those with lower feerates.
  - *number of inputs* and *number of outputs:* mainly contribute to the validation cost of miners, as they need to check the legitimacy of the assets stated in each transaction input by tracking the previous transactions in the blockchain.

[2] Segwit transactions relocate the unlocking script (witness) from within the transaction to an external data structure, resulting in a smaller size in terms of its raw data

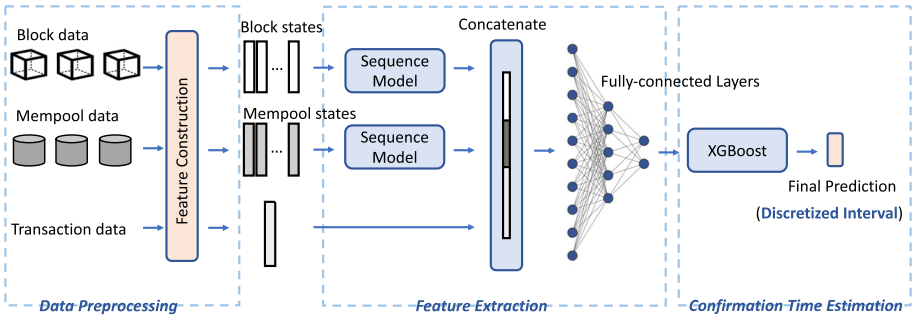


Figure 1 An overview of Hybrid-CTEN

- *transaction first-seen time:* refers to the moment when a Bitcoin transaction is initially observed by a node in the Bitcoin network. In this work, it is treated as an approximation of the transaction submission time. This is because it can be challenging to accurately determine the exact time when a transaction was submitted to the network, as well as the transaction propagation time in the network before the Bitcoin system observes it.
- *mempool position:* indicates the competition among unconfirmed transactions in the system. As higher fee transactions are typically expected to be processed earlier than this one, we computed it by summing the weights of all unconfirmed transactions with higher feerates and subsequently dividing the sum by the maximum block size.

- **Block states** encompasses attributes associated with the blocks that have been mined, such as block size and transaction confirmation distribution. These characteristics are used to infer future transaction confirmation and mining preference based on historical transactions confirmed in each block.

Miners typically prefer to include transactions with higher fees to maximize their profits. Therefore, we construct the block states by characterizing the transaction distribution within a block based on their respective feerates. During the distribution construction process, it becomes infeasible to represent every individual feerate value due to the continuous nature of feerate values and the uneven distribution of transactions across different feerates. Hence, we made a trade-off between the precision of the scale and the scale dimension, assigning smaller scales to the lower feerate zones and larger scales to the higher feerate zones, based on the distribution characteristics. Figure 2 illustrates the confirmed transaction distribution across different feerates in the blockchain, revealing that 94% of transactions have a feerate lower than 84 and transactions are more densely packed at lower feerate levels than at higher levels.

Specifically, we partition the complete feerate spectrum into 36 non-overlapping intervals with exponentially increasing interval sizes. The first interval is defined with a feerate range of 0-3, and for each subsequent interval, the maximum feerate is expanded by 10% of the prior interval's maximum feerate. This progression continues until the final interval, which encompasses all transactions with feerates surpassing the maximum feerate (approximately 84). The division criteria for every three intervals are depicted by the dotted line in Figure 2, using smaller scales for the lower feerate range and larger scales for the higher feerate range.

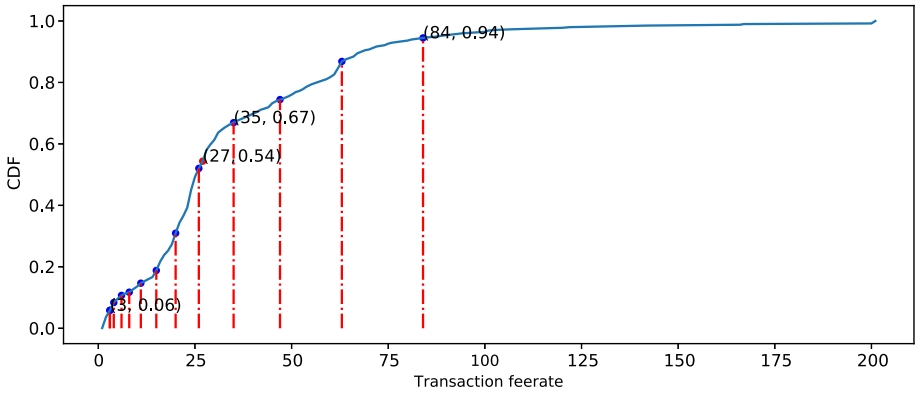


Figure 2 Cumulative distribution function (CDF) of transaction across different feerates

Finally, the block states of a block, denoted as $dist\_b$, are modeled by the distribution of transactions in this block b across each feerate interval u. It is calculated by summing the weight w of every transaction tx in this block with its feerate r falling into that interval according to (1):

$$dist\_b(u) = \sum_{tx \in b, r \in u} w(tx) \tag{1}$$

- **Mempool states** indicates the competition among the unconfirmed transactions in the mempool Mem, where transactions compete to be included in the next block. Similar to the block states, we model the mempool states mem as the distribution of unconfirmed

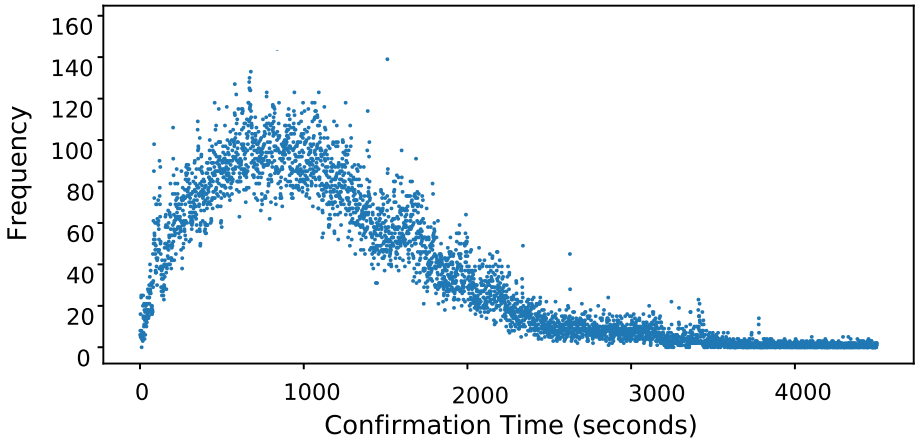


Figure 3 The distribution of transactions confirmation within a 2-block interval

transactions across each feerate interval u. It is calculated by summing the weight of each transaction tx with its feerate r falling into that interval, as defined in (2):

$$mem(u) = \sum_{tx \in Mem, r \in u} w(tx) \tag{2}$$

4.1.2 Confirmation time discretization

This work discretizes future time into block intervals rather than time intervals. This choice is mainly owing to the unpredictability of block generation time, which can introduce significant variance in the estimation outcomes. Figure 3 illustrates the confirmation time of transactions with a 2-block confirmation interval. We can see that the duration of the 2-block interval can vary from a few seconds to hundreds of seconds. Therefore, a fixed-time range could represent different block intervals, with some cases having a one-block interval and others having 2-block intervals, leading to inevitable classification errors. Fortunately, these variances can be addressed by discretizing future time into block intervals, which mitigates classification errors that may occur due to the variation in confirmation time.

Specifically, we adhere to two discretization rules when dividing the confirmation time (block intervals) into multiple classes: The first rule is that *transactions with the same confirmation block interval will be grouped in the same class*. The second rule is *to try to seek a balance in the number of transaction samples in each class*. According to Figure 4, the Bitcoin blockchain system exhibits a long tail in the distribution of transaction confirmation time. Transaction confirmation time ranges from a few blocks to over a hundred blocks, with the majority of transactions being confirmed within 10 blocks. When the confirmation duration exceeds 10 blocks, the distribution becomes sparser across each confirmation block interval.

The discretization process, guided by the two rules, is outlined in Algorithm 1. It encompasses the following steps: (1) Set the lower bound e^{low} of a new decentralized interval, with the smallest block elapsed time (starting from a 1-block interval) in the unclassified time range. (2) Determine the split ratio p_{split} to discretize the unclassified confirmation time range, aiming for an equal division of the remaining block elapsed time range. (lines

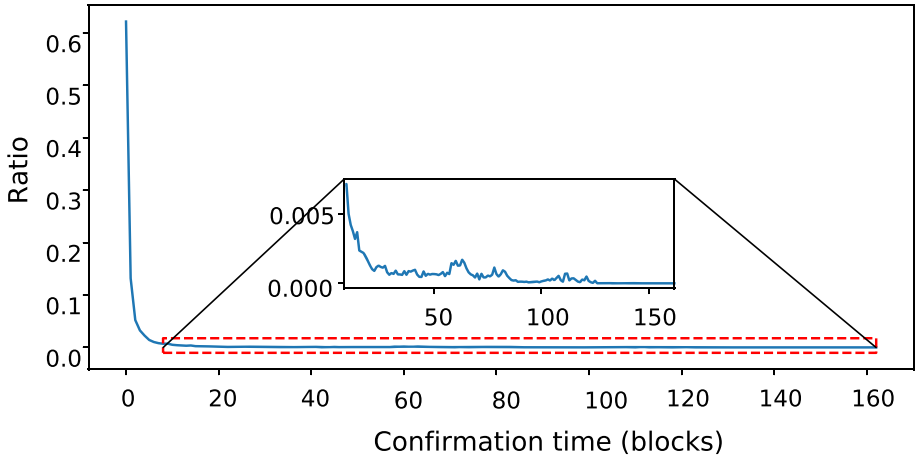


Figure 4 The distribution of transaction confirmation in the Bitcoin blockchain

4–7). (3) Determine the upper bound e^{upp} of the new decentralized interval by accumulating higher block elapses until the split ratio is reached (lines 9-11), yielding the new decentralized interval with the time range $[e^{low}, e^{upp}]$. (4) Repeat (1)–(4) iteratively until k intervals (E) have been defined.

Algorithm 1 Confirmation Time Discretization Algorithm

Input: confirmation time (block elapse e) ratio, $P = \{p_1, p_2, \cdots, p_{e_{max}}\}$; the classification size, k.
Output: The discretized block intervals, $E = \{E_1, E_2, \cdots, E_k\}$.

1: $c\_num \leftarrow 1$; ▷ construct a new block interval
2: $e^{low} \leftarrow 1$; ▷ the minimum confirmation time in each class
3: **while** $c\_num \leq k$ **do**
4: **if** $c\_num == 0$ **then**
5: $p_{split} \leftarrow \frac{1}{k}$
6: **else**
7: $p_{split} \leftarrow \frac{1-\sum_1^{e^{low}-1} p_i}{k-c\_num}$;
8: **end if**
9: $e^{upp} \leftarrow argmin(p_{split} \leq \sum_{e^{low}}^{e^{upp}} r_i)$ ▷ the maximum confirmation time in each class
10: $c\_num \leftarrow c\_num + 1$
11: $E_{c\_num} \leftarrow \{e^{low}, \cdots, e^{upp}\}$
12: $e^{low} \leftarrow e^{upp} + 1$
13: **end while**
14: **return** $E = \{E_1, E_2, \cdots, E_k\}$;

4.2 Feature extraction module

After embedding these three sources of information, the Feature Extraction Module is applied to generate high-level concepts from these inputs. Specifically, it takes inputs from sequences of block states and mempool states, as well as the transaction features and then outputs the extracted patterns. In this module, the incorporation of block states is intended to assist in inferring transaction confirmation by considering the volume of future blocks and mining preferences based on confirmed transactions in each block. The sequence of mempool states is utilized to infer the competition among unconfirmed transactions in the future.

As shown in Figure 1, sequence models are first applied to the block sequence and mempool sequence to extract trend information. The generated sequential trend patterns are then combined with the transaction features and passed through a series of stacked fully-connected layers for further pattern extraction. Finally, a softmax algorithm is applied to produce a classification result, which is then used to train the feature extraction module. To mitigate overfitting, each fully-connected layer is trailed by a dropout operation and activated using a ReLU function.

In addition to the Long Short-Term Memory (LSTM) sequence model [15], which aggregates information on a token-by-token basis in sequential order, we have also implemented alternative sequence models that attempt to capture the relationships between different positions of a single sequence to generate a representation for the sequence. These models encompass additive attention [16], self-attention [14], weighted attention [17], and transformer encoder [14]. We compare their performance in processing sequential data, particularly given their significant achievements in handling time series data [18–21].

Table 1 Experimental dataset for transaction confirmation time estimation

Dataset	Training	Testing
S1	621005–621184	621185–621229
S2	621255–621434	621435–621479
S3	621505–621684	621685–621729
S4	621755–621934	621935–621979
S5	622005–622184	622185–622229
S6	622255–622434	622435–622479

4.3 Confirmation time estimation module

In this module, the output of the chosen fully-connected layer in the Feature Extraction Module (the third layer in Figure 1), encompassing both specific and conceptual attributes of the transaction features, block states, and mempool states, will be obtained. Subsequently, XGBoost is employed for the final classification, drawing from its well-established effectiveness and efficiency within the machine learning community.

5 Experiments

5.1 Datasets

We collected transaction data from the block range 621001–622500 via Blockchain.com[3]. Each dataset consists of 225 continuous blocks selected from every 250 blocks. The first 80% of blocks in each dataset (approximately 400000 transactions) are used for training, while the remaining 20% (approximately 100000 transactions) are used for testing. Only newly submitted transactions are selected when selecting instances for training and testing. Once a model is trained, we evaluate its performance on the testing dataset relative to the training dataset, as shown in Table 1.

5.2 Confirmation time discretization

In this work, we discretize the range of transaction confirmation time into four different class sizes, as indicated in Table 2: $k = 2$, $k = 4$, $k = 6$, and $k = 8$. For $k = 2$, the confirmation time range is categorized: '*confirmed within 1 block interval*' and '*confirmed within more than 1 block interval* ($\geq$ 2 blocks)'. In this case, this problem can be viewed as a binary classification problem of predicting whether a transaction will be confirmed in the next block as studied in the works [3, 10].

In this work, we choose not to further discretize the confirmation time into more than 8 classes, as transactions confirmed beyond 50 blocks are rare at each block interval, as depicted in Figure 4. Meanwhile, *Class 8* in the set of 8 classes (k = 8) already encompasses transactions confirmed beyond a 59-block interval, as shown in Table 2.

[3] https://www.blockchain.com/api/blockchain_api

5.3 Evaluation metrics

In this work, we have selected precision, recall, and f1-score as our primary metrics for evaluation, while accuracy is considered a secondary metric. This choice is due to the nature of the data and the distribution among the prediction classes. Given the uneven distribution of transactions confirmed with various block intervals, approximately 60% of them being confirmed within a 1-block interval and the rest representing a smaller fraction (as indicated in Table 2), an important consideration arises. This situation can lead to a classification model achieving a high accuracy score simply by predicting the majority class, even if its performance on the minority class is unsatisfactory.

- **Primary metric:**

$$\text{recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{3}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{4}$$

$$\text{f1-score} = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall+precision}} \tag{5}$$

- **Secondary metric:**

$$\text{accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{6}$$

where TP (true positive), FP (false positive), FN (false negative), and TN (true negative) are observed classification results.

5.3.1 Compared methods

We compare the estimation performance of neural network models, ensemble learning models, a baseline model as well as the proposed Hybrid-CTEN framework.

- **Neural Network (NN)** refers to neural network models, which have gained prominence in the machine learning community for their effectiveness in addressing classification tasks [22, 23].
  - **MLP**, Multi-Layer Perceptron, is composed of stacked fully-connected layers. In this work, it takes only transaction features as input.

Table 2 Transaction confirmation time discretization under class k=2, 4, 6, 8

Class	k=2		k=4		k=6		k=8	
	blocks	ratio	blocks	ratio	blocks	ratio	blocks	ratio
Class 1	1	62.2%	1	62.2%	1	62.2%	1	62.2%
Class 2	≥2	37.8%	2	13.2%	2	13.2%	2	13.2%
Class 3			3–7	13.2%	3–4	8.5%	3	5.2%
Class 4			≥8	11.5%	5–8	5.5%	4–5	5.5%
Class 5					9–28	5.3%	6–9	4.0%
Class 6					≥29	5.3%	10–18	3.4%
Class 7							19–58	3.4%
Class 8							≥59	3.2%

- **Lstm**, **Lstm**$^+$ and **Lstm_prev** utilise LSTM as a sequence model to process both block states and mempool states. The output of the LSTM is then combined with transaction features and passed through a fully-connected layer network. The difference lies in that **Lstm**$^+$ applies a deeper 7-layer fully-connected network, while **Lstm_pre** models the block states and mempool states applied in the work [13].
- **Adv**, **Wht**, **Self**, and **Transf** correspond to models that utilize different attention techniques to extract features from block states and mempool states. These techniques include additive attention [16], weighted attention [17], self-attention [14], as well as the transformer model [14]. The extracted sequential features are then combined with transaction features and fed into a fully-connected layers network.

In the neural network models, the sequence processing model is configured with 8 hidden units and, if applicable, a sequence length of 3. The fully connected layers consist of a three-layer neural network with hidden unit configurations of [64, 8]. A dropout rate of 0.2 is applied, followed by the specified class size. For the Transformer model, we used 2 attention heads and 32 units in its fully-connected layer. The batch size is set to 1000 when applicable, and the models are optimized using stochastic gradient descent with the Adam optimizer.

- **Ensemble Learning (EL) models** enhance prediction performance by training multiple estimators and integrating their predictions [12, 24–27]. In this work, we study the classification performance of four state-of-the-art ensemble approaches: **XGBoost** [12], **LightGBM** [25], Random Forest (**RF**) [26], and Rotation Forest (**RoF**) [27], all of which are well-known for their outstanding performance in handling classification tasks. XGBoost is a cutting-edge gradient boosting framework of decision trees that gained popularity in the 2015 Kaggle classification challenge. Compared to XGBoost, LightGBM employs histogram-based algorithms to reduce execution time and memory consumption. RF, ensembling decision trees based on the bagging technique, is popular owing to its generalized performance, high prediction accuracy, and quick operation speed. Meanwhile, RoF has been demonstrated to score much better on classification tests than other ensemble approaches such as Bagging, AdaBoost, and Random Forest [27].
  In addition, we also study the classification performance of deep forest (**DF**) [28], which maintains the layer structure of a neural network while replacing the neurons in the fully connected layers with base estimators (some ensemble learning models). In this work, the base estimators consist of two random forest models and two extremely randomized trees classifiers as introduced in [29]. We also investigate its variants with the introduction of a misclassification penalty, **DF_Cost** [30].
  In both the ensemble learning models and deep forest models, the number of estimators is set to 100 by default. However, for XGBoost, the number of estimators is set to 300, and the booster is set to gbtree. Additionally, the values for $\gamma = 0.1$, $max\_depth = 6$, $\lambda = 2$, and $colsample\_bytree = 0.7$ are set, with all other parameters left at their default values.
- **Hybrid-CTEN** is our proposed framework that combines neural network and XGBoost. We implemented two feature extraction frameworks, Lstm and Adv, and applied XGBoost to analyze the extracted features, resulting in two variants: **HybridLstm** and **HybridAdv**. Additionally, we introduced a variant **HybridLstm**$^+$ by stacking a seven-layer neural network with hidden unit configurations of [64, 48, 36, 24, 18, 12] for the first 6 layers, along with a final layer designed for the specified class size, as opposed to the 2-layer network used in HybridLstm.

Table 3 Comparison of model performance across different class sizes

(a) Prediction performance on class size k= 2 and k= 4

Model		k=2				k=4			
		precision	recall	f1-score	accuracy	precision	recall	f1-score	accuracy
NN	MLP	94.8%	91.8%	92.6%	93.3%	69.3%	62.0%	62.1%	82.8%
	Adv	95.0%	92.3%	93.1%	93.7%	70.9%	65.1%	65.6%	82.6%
	Wht	94.7%	91.6%	92.4%	93.1%	62.4%	60.6%	59.7%	82.1%
	Self	93.4%	91.2%	91.8%	92.4%	71.2%	59.6%	59.0%	79.1%
	Lstm	94.3%	92.3%	93.0%	93.6%	62.7%	63.8%	58.7%	81.4%
	Transf	94.0%	91.8%	92.5%	93.1%	72.7%	72.1%	69.2%	83.6%
EL	RF	96.9%	96.3%	96.5%	96.7%	78.5%	79.4%	78.0%	87.9%
	RoF	94.5%	94.4%	94.4%	94.7%	73.1%	74.1%	72.9%	84.5%
	DF	96.9%	96.3%	96.5%	96.7%	78.5%	79.4%	78.0%	88.0%
	DF_Cost	-	-	-	-	78.2%	78.7%	78.0%	88.0%
	LightGBM	97.0%	96.4%	96.6%	96.8%	78.6%	79.4%	78.0%	88.0%
	XGBoost	**97.1%**	**96.4%**	**96.7%**	*96.9%*	78.6%	79.5%	78.0%	*88.1%*
Hybrid	HybridAdv	90.5%	91.2%	90.1%	90.3%	78.0%	78.9%	76.7%	84.7%
-CTEN	HybridLstm	93.2%	94.0%	93.4%	93.7%	77.5%	**79.9%**	76.5%	84.8%
	HybridLstm^{+}	91.5%	92.3%	91.5%	91.8%	**78.6%**	79.6%	**78.5%**	87.7%
Baseline		96.6%	96.1%	60.9%	66.0%	77.6%	78.0%	30.9%	51.5%

(b) Prediction performance on class size k= 6 and k= 8

Model		k=6				k=8			
		precision	recall	f1-score	accuracy	precision	recall	f1-score	accuracy
NN	MLP	41.2%	41.2%	37.9%	74.7%	32.5%	35.0%	31.0%	73.0%
	Adv	43.2%	44.0%	40.7%	74.4%	40.8%	38.7%	37.3%	74.2%
	Wht	44.4%	46.3%	42.2%	75.1%	44.1%	41.3%	39.1%	74.0%
	Self	47.7%	41.9%	38.3%	72.9%	31.9%	34.4%	29.8%	70.9%
	Lstm	45.7%	45.3%	43.0%	74.9%	34.2%	34.5%	30.9%	72.6%
	Transf	41.8%	42.1%	38.8%	75.1%	40.2%	39.3%	36.8%	74.6%
EL	RF	66.4%	66.8%	65.1%	84.4%	58.0%	57.5%	56.7%	82.4%
	RoF	62.5%	63.0%	61.7%	81.1%	55.3%	54.9%	54.5%	79.2%
	DF	66.0%	66.3%	64.6%	84.1%	58.1%	58.2%	57.1%	82.4%
	DF_Cost	66.0%	66.3%	64.6%	84.1%	58.1%	58.2%	57.1%	82.4%
	lightGBM	66.3%	66.3%	65.0%	84.5%	58.1%	58.0%	57.1%	82.8%
	XGBoost	66.8%	66.4%	65.1%	84.9%	58.6%	58.3%	57.3%	83.1%
Hybrid	HybridAdv	67.4%	68.5%	64.9%	81.5%	58.7%	58.3%	55.8%	78.8%
-CTEN	HybridLstm	67.5%	69.2%	66.0%	81.4%	59.8%	59.4%	56.9%	80.2%
	HybridLstm^{+}	**68.5%**	**69.7%**	**66.9%**	81.6%	**60.6%**	**60.4%**	**59.4%**	81.7%
Baseline		66.4%	65.4%	21.1%	48.6%	59.6%	58.2%	15.8%	47.3%

The bold text is to highlight the best performance among all the methods

5.4 Result analysis

We compare the prediction performance of our proposed Hybrid-CTEN framework with the other models across different class sizes. Additionally, we conduct a further study on the effectiveness of the feature construction proposed in this work.

5.4.1 Evaluation of classification performance

Table 3 presents the overall performance of different classification models across different class sizes, obtained by averaging the performance across each dataset. We can observe that as the class size increases, the performance of each model decreases under every evaluation strategy. Meanwhile, we can find that both ensemble learning models and our proposed Hybrid-CTEN models outperform neural network models in handling this classification task.

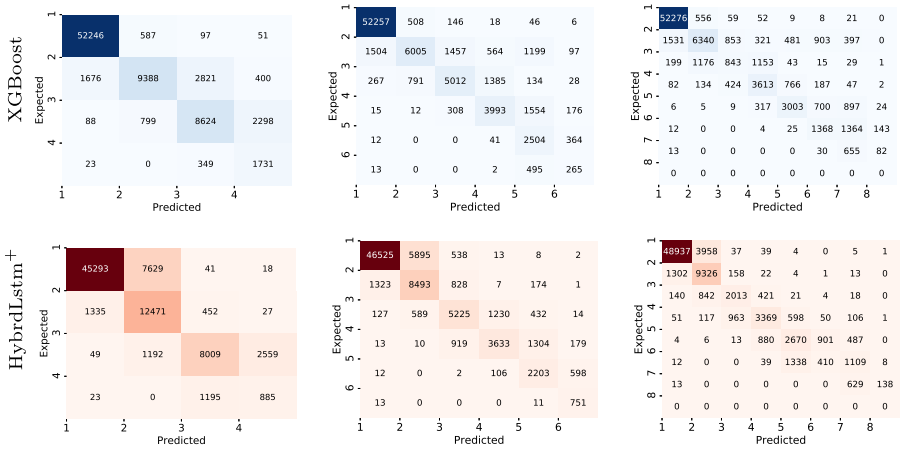
Among all the models, XGBoost achieves the most competitive performance in predicting whether a transaction can be classified into the next block (k = 2). However, when handling more complicated classification tasks, our proposed Hybrid-CTEN framework variants, HybridAdv, HybridLstm, and HybridLstm$^+$, which incorporate the features of block states and mempool states, gradually dominate the prediction performance on precision, recall, and f1-score, outperforming XGBoost. In the classification case k = 4, HybridLstm$^+$ achieves the best performance on precision and f1-score, while HybridLstm achieves the best performance on recall. When the class size increases to 4 and 6, both HybridLstm$^+$ and HybridLstm dominate the other models on all precision, recall, and f1-score. Furthermore, it is worth noting that HybridLstm$^+$ outperforms HybridLstm with more complex feature extraction work, highlighting the significance of block states and mempool states in handling complicated classification tasks. This conclusion is consistent with the observation in neural network models, where incorporating block states and mempool states yields better performance than solely relying on transaction features.

XGBoost dominates the other models in terms of accuracy. This is due to its strength in binary classification and the unbalanced nature of the classification task, which favors predicting the majority class in each task. Figure 5 presents the confusion matrices generated by XGBoost and HybridLstm$^+$ for different classes on dataset S3. It can be seen that for *Class 1*, where the majority of instances sit, XGBoost outperforms HybridLstm$^+$ with a significantly higher proportion of correct predictions. However, XGBoost loses its dominance in the other classes. The same conclusion can be drawn for the other datasets as well.

In the context of our Hybrid-CTEN framework, the superior performance of HybridLstm and HybridLstm$^+$ over HybridAdv suggests that LSTM-based models are more effective in estimating transaction confirmation time.

5.4.2 Evaluation of optimization on feature construction

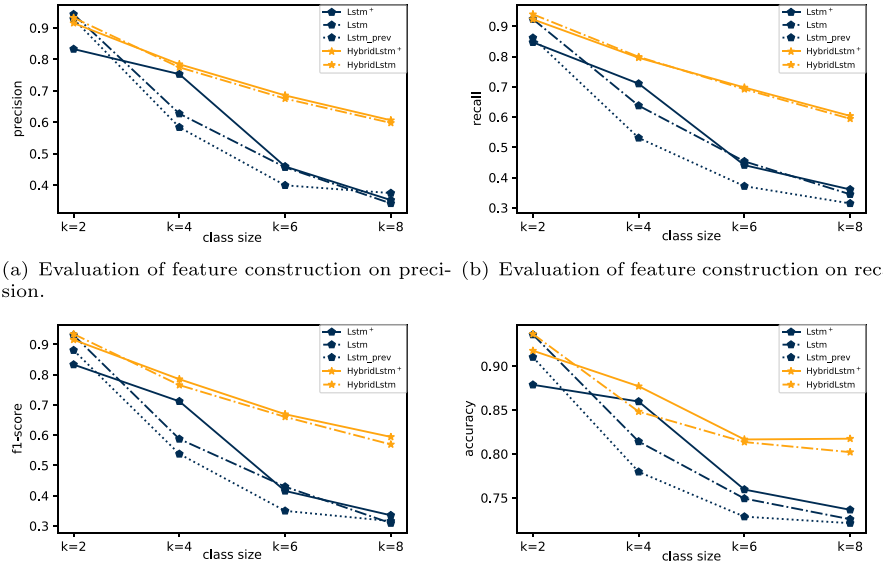
Compared to our previous work [13], we made several adjustments to the feature extraction framework: (1) We updated the transaction distribution modeling in block states and mempool states based on the feerate characteristics of the blockchain. This was achieved by assigning smaller scales to lower feerate zones and larger scales to higher feerate zones, with a 10% incremental increase in each subsequent interval. (2) We increased the number of fully-connected layers in the neural network solutions from 3 to 7 layers. Results on Figure 6 demonstrates the optimization. First, in terms of feature construction on block states and mempool states, we proved its effectiveness by comparing the performance of Lstm models



(a) Confusion matrix of 4-class. (b) Confusion matrix of 6-class. (c) Confusion matrix of 8-class.

Figure 5 Confusion matrix of XGBoost and HybridLstm^{+} on different classes in dataset S3

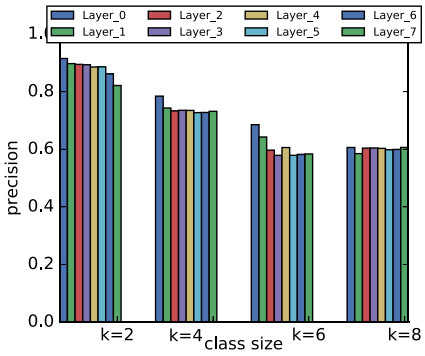
with different feature constructions. We observed that the model Lstm with updated features outperformed the previous one, Lstm_prev. Additionally, we further demonstrate the effectiveness of deeper feature extraction layers through the superior classification performance of HybridLstm^{+} over HybridLstm, especially when handling complex classification tasks with class size $k \in \{4, 6, 8\}$.



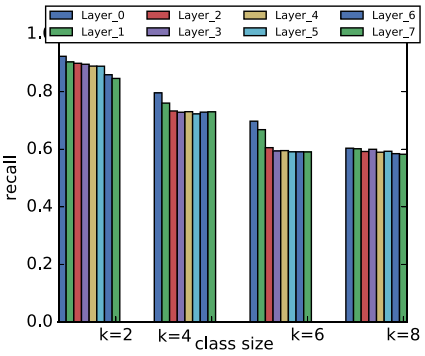
(a) Evaluation of feature construction on precision. (b) Evaluation of feature construction on recall.

(c) Evaluation of feature construction on f1-score. (d) Evaluation of feature construction on accuracy.

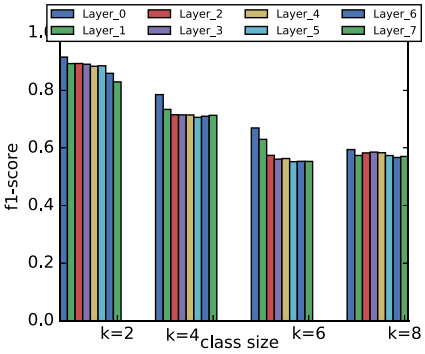
Figure 6 Prediction performance evaluation of feature construction



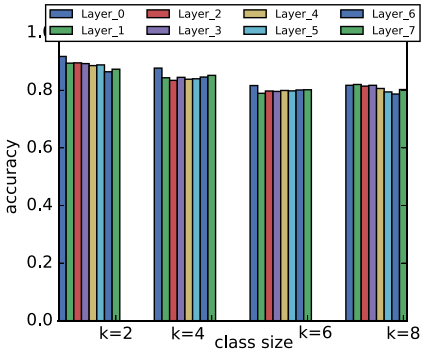
(a) Performance evaluation of different extraction layer on precision.



(b) Performance evaluation of different extraction layer on recall.



(c) Performance evaluation of different extraction layer on f1-score.



(d) Performance evaluation of different extraction layer on accuracy.

Figure 7 Prediction performance evaluation of HybridLstm^{+} with different extraction layers

Furthermore, we evaluated the classification performance of HybridLstm^{+} using features extracted from different layers. Starting from the output of Layer_0, which contains the original concatenated features from the outputs of the sequential model and transaction raw features, we compared its prediction performance with features extracted from each of the following 7-layer fully-connected layers. Our results, as shown in Figure 7, indicate that Laye_0 outperforms the other layers, suggesting that the extracted data from the subsequent layers may have missed essential information required by XGBoost.

In conclusion, these findings emphasize the importance of feature selection and extraction in developing effective classification models for transaction confirmation time estimation tasks.

6 Conclusion

This paper approaches the problem of transaction confirmation time estimation by framing it as a classification problem. Then a transaction confirmation time estimation framework, Hybrid-CTEN, is proposed to solve this problem. This framework combines historical con-

firmation time in the block, unconfirmed transactions in the mempool, and the estimated transaction itself to improve estimation performance over multiple classification tasks. The experiments on the real-world blockchain data demonstrate that other than XGBoost excelling in the binary classification case (to predict whether a transaction will be confirmed in the next generated block), Hybrid-CTEN surpasses state-of-the-art methods in terms of precision, recall, and f1-score across all multiclass classification scenarios (4-class, 6-class, and 8-class).

Our future research will concentrate on two main directions. Firstly, we will optimize our framework for the Bitcoin blockchain. This will involve enhancing the extraction of transaction features and addressing the issue of handling imbalanced data distribution in estimating transaction confirmation time, which is a common challenge encountered in real-world datasets. Secondly, we plan to adapt our proposed framework to different blockchain systems. For example, we will explore its applicability to predicting gas usage in the Ethereum network. This adjustment will involve considering the specific characteristics and requirements of other blockchain platforms to ensure the effectiveness of our framework across diverse contexts.

Author Contributions Limeng Zhang: Conceptualization, Methodology, Software, Writing - Original Draft. Rui Zhou: Project administration, Supervision, Writing - Review & Editing. Qing Liu: Supervision, Validation, Writing - Review & Editing. Jiajie Xu: Writing - Review & Editing. Chengfei Liu: Supervision, Writing - Review & Editing. Muhammad Ali Babar: Supervision, Writing - Review & Editing. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. This research is supported by Data61, Australian Research Council Discover (Grant No. DP170104747, No. DP180100212 and No. DP200103700) and National Natural Science Foundation of China (Grant No. 61872258).

Availability of data and materials The data used in this work is collected from the block range 621001–622500 in the Bitcoin blockchain via Blockchain.com[3].

References

1. Corbet, S., Meegan, A., Larkin, C., Lucey, B., Yarovaya, L.: Exploring the dynamic relationships between cryptocurrencies and other financial assets. Econ Lett **165**, 28–34 (2018)
2. Balsamo S, Marin A, Mitrani I, Rebagliati N (2021) Prediction of the consolidation delay in blockchain-based applications. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering, pp 81–92
3. Fiz, B., Hommes, S., et al (2017) Confirmation delay prediction of transactions in the bitcoin network. In: Advances in Computer Science and Ubiquitous Computing, pp 534–539. Springer
4. Gundlach, R., Gijsbers, M., Koops, D., Resing, J.: Predicting confirmation times of bitcoin transactions. ACM SIGMETRICS Perform Eval Rev **48**(4), 16–19 (2021)
5. Kasahara, S., Kawahara, J.: Effect of bitcoin fee on transaction-confirmation process. J Ind Manag Optim **15**(1), 365 (2019)
6. Kawase, Y., Kasahara, S.: Priority queueing analysis of transaction-confirmation time for bitcoin. J Ind Manag Optim **16**(3), 1077 (2020)

7. Koops D (2018) Predicting the confirmation time of bitcoin transactions. arXiv preprint arXiv:1809.10596
8. Zhao W, Jin S, Yue W (2019) Analysis of the average confirmation time of transactions in a blockchain system. In: International Conference on Queueing Theory and Network Applications, pp 379–388. Springer
9. Zhang L, Zhou R, Liu Q, Xu J, Liu C (2021) Transaction confirmation time estimation in the bitcoin blockchain. In: International Conference on Web Information Systems Engineering, pp 30–45. Springer
10. Ko K, Jeong T, Maharjan S, Lee C, Hong JW-K (2019) Prediction of bitcoin transactions included in the next block. In: International Conference on Blockchain and Trustworthy Systems, pp 591–597. Springer
11. Kleinrock, L.: Theory, vol. 1. Wiley-Interscience, Queueing systems (1975)
12. Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., et al.: Xgboost: extreme gradient boosting. R package version 0.4-2 **1**(4), 1–4 (2015)
13. Zhang L, Zhou R, Liu Q, Xu J, Liu C (2022) Bitcoin transaction confirmation time prediction: A classification view. In: Web Information Systems Engineering–WISE 2022: 23rd International Conference, Biarritz, France, 1–3-November-2022, Proceedings, pp 155–169. Springer
14. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł., Polosukhin I (2017) Attention is all you need. In: Advances in Neural Information Processing Systems, pp 5998–6008
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput **9**(8), 1735–1780 (1997)
16. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
17. Felbo B, Mislove ASA, Rahwan I, Lehmann S (2017) Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv preprint arXiv:1708.00524
18. Fu R, Zhang Z, Li L (2016) Using lstm and gru neural network methods for traffic flow prediction. In: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp 324–328. IEEE
19. McNally S, Roche J, Caton S (2018) Predicting the price of bitcoin using machine learning. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp 339–343. IEEE
20. Srivastava N, Mansimov E, Salakhudinov R (2015) Unsupervised learning of video representations using lstms. In: International Conference on Machine Learning, pp 843–852
21. Yin, J., Tang, M., Cao, J., Wang, H.: Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. Knowl-Based Syst **210**, 106529 (2020)
22. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
23. Zhang, G.P.: Neural networks for classification: a survey. IEEE Trans Syst Man Cybern Syst **30**(4), 451–462 (2000)
24. Sagi, O., Rokach, L.: Ensemble learning: A survey. Wiley Interdiscip Rev: Data Min Knowl **8**(4), 1249 (2018)
25. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y (2017) Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst **30**
26. Breiman, L.: Random forests. Mach learn **45**(1), 5–32 (2001)
27. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. IEEE Trans Pattern Anal Mach Intell **28**(10), 1619–1630 (2006)
28. Zhou Z-H, Feng J (2017) Deep forest: Towards an alternative to deep neural networks. In: IJCAI
29. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach learn **63**(1), 3–42 (2006)
30. Ma, C., Liu, Z., Cao, Z., Song, W., Zhang, J., Zeng, W.: Cost-sensitive deep forest for price prediction. Pattern Recognit **107**, 107499 (2020)

Authors and Affiliations

Limeng Zhang[1] · Rui Zhou[1] · Qing Liu[2] · Jiajie Xu[3] · Chengfei Liu[1] · Muhammad Ali Babar[4]

Limeng Zhang
limengzhang@swin.edu.au

Qing Liu
Q.Liu@data61.csiro.au

Jiajie Xu
xujj@suda.edu.cn

Chengfei Liu
cliu@swin.edu.au

Muhammad Ali Babar
ali.babar@adelaide.edu.au

1 Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia

2 Data61, CSIRO, Hobart, Australia

3 School of Computer Science and Technology, Soochow University, Jiangsu Province, China

4 Centre for Research on Engineering Software Technologies (CREST), The University of Adelaide, Adelaide, Australia