Transaction Confirmation Time Estimation in the Bitcoin Blockchain

Limeng Zhang[1,2], Rui Zhou[1(✉)], Qing Liu[2], Jiajie Xu[3], and Chengfei Liu[1]

[1] Swinburne University of Technology, Melbourne, Australia
{limengzhang,rzhou,cliu}@swin.edu.au
[2] Data61, CSIRO, Hobart, Australia
Q.Liu@data61.csiro.au
[3] Soochow University, Suzhou, China
xujj@suda.edu.cn

Abstract. As Bitcoin is universally recognized as the most popular cryptocurrency, more and more Bitcoin transactions are expected to be populated to the Bitcoin blockchain system. However, transactions cannot be confirmed altogether into the next block due to the limited block capacity. One of the most demanding requirements for users to use Bitcoin is to estimate the confirmation time of a newly submitted transaction. In this paper, we propose two approaches for estimating the confirmation time for a single transaction. The first approach DcyMean makes the estimation based on the historical confirmation time of transactions included in the blockchain. The second approach CTEN is built on neural networks to estimate based on a variety of factors including the transaction itself, block states and mempool states. Finally, we conduct experiments on real Bitcoin blockchain datasets to demonstrate the effectiveness and efficiency of our proposed approaches. Each of our approaches can finish training and estimation within one block interval, demonstrating that our approaches can process real-time cases.

Keywords: Transaction confirmation time estimation · Bitcoin · Blockchain

1 Introduction

Nowadays, cryptocurrency has become a buzzword in both industry and academia [26]. Blockchain works as one of the popular systems to manage cryptocurrency transactions. Compared with traditional ledger systems, blockchain is decentralized, immutable and transparent. Bitcoin, one of the most popular cryptocurrencies, has achieved huge success with its capital market reaching 600

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).

W. Zhang et al. (Eds.): WISE 2021, LNCS 13080, pp. 30–45, 2021.
https://doi.org/10.1007/978-3-030-90888-1_3

billion dollars in 2021 as shown in coindesk[1]. Meanwhile, PayPal accountholders in the United States have been able to buy, hold, and sell cryptocurrencies directly through PayPal. As a result, more Bitcoin transactions are expected to be populated into the Bitcoin blockchain system. The bulk of new transactions, however, cannot be included altogether in the next block due to the restricted capacity of a block. After a transaction is submitted, the transaction is then verified and put into a miner's mempool where the transaction competes with other unconfirmed transactions to be included/confirmed into future blocks. As a result, a submitted transaction may suffer from confirmation delay as the number of submitted transactions grows. 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.

Existing research has shed some light on transaction confirmation time estimation in the Bitcoin blockchain [3,12,14,19–22,45]. However, they are not practical enough due to the following four types of drawbacks: (1) Estimation is not tailored for an individual transaction. The existing works [19,20,45] evaluate the average confirmation time of two classes of transactions [19,20] or of all the unconfirmed transactions [45], instead of the confirmation time of a single transaction. (2) Some works [12,21] only predict whether a transaction can be confirmed in the next block, i.e. treat the problem as a binary classification problem. They could not provide accurate confirmation time, and thus may be insufficient in practice. (3) Some assumptions are not realistic. The works [3,14,22] assume that mempool receives transactions falling into each feerate class at a specific fixed speed. However, the speed of inflow transactions in a feerate class often varies, such as sensitive to peak/off-peak trades, government policies, finance news, etc. In addition, they assume that every block contains a fixed number of transactions. This is often not the case. (4) Transaction features and block features are not sufficiently utilized. To the best of our knowledge, only the works [12,21] consider transaction metadata, such as transaction weight, number of inputs, number of outputs, etc. No prior work has considered block features which may give some hints on future block sizes and future block generation speeds that are likely to affect when a new transaction can be confirmed.

To address these limitations, we propose two transaction confirmation time estimation approaches, DcyMean and CTEN. DcyMean is designed based on decayed mean of historical transaction confirmation time in the blockchain. It estimates by aggregating the confirmation time of historical transactions in different blocks with feerates falling into the same feerate interval as the given transaction. A decay parameter is used to lower down the importance of earlier blocks. We also propose an advanced approach CTEN (short for Confirmation Time Estimation Network) to make the estimation based on neural networks. It works by predicting the characteristics of future blocks and simulating the mining behavior of miners. Specifically, it considers three types of information: the transaction itself, block states in the blockchain and mempool competition states. Historical block sequence is used to predict the properties of future blocks.

[1] https://www.coindesk.com/price/bitcoin.

CTEN also derives the mining behavior from the confirmation of transactions in the mempool at different block heights. Both block state sequence and mempool state sequence are learnt using Long Short-Term Memory (LSTM) or Attention mechanisms, which have demonstrated outstanding performance in time series data processing [13,24,33,42].

Our contributions in this paper are summarized as follows:

- We propose a feerate-based confirmation time estimation approach DcyMean, which infers the confirmation time for a transaction by aggregating the confirmation time of historical transactions in different blocks with feerates falling into the same feerate interval as the given transaction.
- We propose an advanced approach CTEN, which works on leveraging the power of neural networks for time series data processing techniques to learn future block characteristics and simulate miners' mining policy to help with transaction confirmation time estimation.
- Extensive experiments on real Bitcoin blockchain datasets have been conducted to demonstrate the effectiveness and efficiency of our proposed approaches.

2 Background and Related Work

In this section, we first introduce some recent attempts on managing blockchain systems, and then introduce the transaction confirmation process in Bitcoin and related works on transaction confirmation time estimation in the Bitcoin Blockchain.

2.1 Blockchain Management

Recently, extensive research has been made on the technologies underpinning the deployment of blockchain systems in different aspects, such as scalability, query evaluation, security and privacy, applications and etc. Scalability focuses on increasing transaction confirmation throughout in blockchain systems through, e.g., consensus protocols [4,15], concurrency control [7,10,31] and data management techniques [5,8,9,27,28,30,32,34,39]. Query evaluation tackles the problem of querying data stored on blockchains [36,38,43,46,47]. Another focus of the community is system maintainability and data privacy [40,41]. There are also works incorporating the blockchain technology into different fields, such as crowd-sourcing [16,17,23], disease control [29] and classification [44].

2.2 Transaction Confirmation in the Bitcoin Blockchain

In the Bitcoin blockchain system, transactions record the transferring of Bitcoin currencies between participants. Transaction are confirmed when they are included in a block of the blockchain. As a block's capacity is restricted, unconfirmed transactions implicitly compete to get confirmed. Transaction fee can be

set to prioritize transaction processing. *Transaction feerate* is a term used to describe the fee density of a transaction, which is calculated by dividing the transaction fee by the transaction size. Once transactions are submitted to the blockchain system, they are broadcast across the nodes in the system. If they meet the transactions' validity criteria [1], Bitcoin nodes will add them to their mempools (memory pools), where transactions wait until they can be included (mined) into a block. Miners are the nodes who need to conduct this verification process with their computing power. They play a critical part in the transaction confirmation process, ensuring that transactions are included on the blockchain. In the confirmation, they first select some unconfirmed transactions in the mempool and construct a candidate block. Then they compete to figure out the computational solution (referred to proof-of-work) for the next block. Finally, the miner who first works out this computational problem is awarded, which consists of a fixed Bitcoin reward and transaction fees included in the block. The competition process among miners is called *mining*. Once a new block is linked to the blockchain, miners update their local copy of blockchain data and start mining the next block. The confirmation of transactions in the new block is complete and these transactions will be removed from the miner's mempool.

2.3 Related Works on Transaction Confirmation Time Estimation

Some works have been done on transaction confirmation time estimation in the Bitcoin blockchain. In [12,21], the authors treated the confirmation time estimation problem as a binary classification problem to predict whether a transaction can be confirmed in the next block. They use supervised learning models like SVM, random forest, AdaBoost, and etc. The authors in works [3,19,20] introduced different queueing models to estimate the confirmation time of transactions with different feerates. Feerates are categorized into several classes in these works, and the works conduct the estimation for each transaction class. To be specific, it assumes that transactions in different feerate classes arrive in the queue according to an independent Poisson process with a specific fixed rate. To fill in each block, a fixed number of transactions (batch) are removed from the queue. Transactions with greater fees are placed in the front part of the queue, allowing them to be processed earlier than those with lower fees. The number of blocks required for the confirmation of a transaction can be thus calculated by its position in the queue. Works in [19,20] adopt the $M/G^B/1$ queueing model with batch B [6]. Balsamo et al. [3] considered a queuing model $M/M^B/1$ with batch B [25], and the works [14,22] model the confirmation process as Cramér-Lundberg process. In addition, Zhao et al. [45] established a type of non-exhaustive queueing model with a limited batch service and a possible zero-transaction service to study the average confirmation time. It assumes that the number of transactions in the system at a given observation point form a Markov chain, the elapsed time of block generation follows a certain density function in stochastic processes, and transaction arrivals follow a Poisson distribution. Finally, based on Little's law [37], the confirmation time is estimated.

Table 1. Notations

Notation	Explanation
tx	A transaction
$\hat{tx}$	The submitted transaction to be estimated
r	The feerate interval that the feerate of $\hat{tx}$ falls in
$w(tx)$	The weight of the transaction tx
$g(tx)$	The confirmation time of the transaction tx, i.e. the gap between The submission and confirmation timestamps of tx
$\hat{g}$	The estimated confirmation time for transaction $\hat{tx}$
dcy	A decay parameter reflecting the fading effect in DcyMean
h	A block height
h_{cur}	The height of the current (most recently generated) block
T_h	The confirmed transactions in the block with height h
T_h^r	The confirmed transactions with feerate interval r in T_h
Mem_h	The unconfirmed transactions in the mempool at block height h
b_h	The block state of the block with height h in CTEN
mem_h	The mempool state of the block with height h in CTEN

3 Problem Definition

Given a newly submitted transaction $\hat{tx}$, the studied problem is to predict the confirmation time $\hat{g}(\hat{tx})$ for $\hat{tx}$. We consider that three types of information are useful to help with the prediction/estimation: (1) the transaction itself, whose features are denoted as $FeaInfo$; (2) historical confirmed transactions stored in blocks, denoted as $ChainInfo$; (3) unconfirmed transactions constituting the mempool, denoted as $MemInfo$. We model estimating transaction confirmation time as looking for a function $\mathcal{F}$ taking the above three types of information as input and giving the estimation as output.

$$\hat{g} = \mathcal{F}(FeaInfo, ChainInfo, MemInfo)$$

The rationale to select the above three types of features is as follows: (1) $FeaInfo$ contains the unique features of a particular transaction, e.g., feerate, weight, validation complexity, submitted time, etc., which could differentiate one transaction from other transactions; (2) $ChainInfo$ has historical transaction confirmations, which may give some hints on the new transaction estimation. Meanwhile, the block sequence may contain block size and block generation trends, implicitly reflecting the volume and speed of confirming multiple transactions as a whole; (3) $MemInfo$ has all the unconfirmed transaction, i.e. the competitiveness of the submitted transaction can be evaluated against all the waiting transactions. As a result, we propose to learn an estimation function from these three types of information. We list some frequently used notations in Table 1.

4 Methods for Confirmation Time Estimation

In this section, we present the confirmation time estimation approaches.

4.1 Decayed Mean (DcyMean)

The first approach we propose is based on decayed mean of historical transactions (named DcyMean), which estimates the confirmation time for a new transaction, given a small feerate interval that the new transaction falls in. It infers the confirmation time from historical transactions that fall into the same feerate interval. More importantly, DcyMean considers that confirmation time of transactions confirmed in recent blocks are more helpful than those confirmed in earlier blocks when computes the estimation. To aggregate the effect of records from different blocks, it introduces a decay parameter $dcy \in [0, 1]$. A smaller dcy means DcyMean relies more on the results from recent blocks than earlier blocks. When a new block is mined, the impact of confirmation time of transactions in prior blocks will fade by a factor of dcy. When $dcy = 0$, it means only the current block (most recently generated block) is used to compute the estimation. When $dcy = 1$, it means all the historical blocks have equivalent importance. The estimated confirmation time is given in Equation (1):

$$\hat{g} = \frac{\sum\limits_{h \le h_{cur}} \sum\limits_{tx \in T_h^r} g(tx) * dcy^{h_{cur}-h}}{\sum\limits_{h \le h_{cur}} \sum\limits_{tx \in T_h^r} dcy^{h_{cur}-h}} \tag{1}$$

where h denotes a historical block height, h_{cur} denotes the current block height, r denotes the feerate interval $\hat{tx}$ falls in and T_h^r denotes the transactions confirmed in block h within the feerate interval r.

By collecting historical confirmation time related to a given small feerate interval in the blockchain, DcyMean obtains the aggregated confirmation time of that feerate interval to predict for a new transaction. In fact, a transaction's confirmation is involved in a more complicated block mining process. which may be affected by not only the transaction itself, but also the hidden mining policies, blockchain states, and possibly what other unconfirmed transactions are. In Sect. 4.2, we propose a more powerful neural network based approach to solve the transaction confirmation time estimation problem.

4.2 Confirmation Time Estimation Network (CTEN)

The second approach we propose is based on neural networks to estimate the confirmation time for a new transaction (named CTEN). Before introducing CTEN, we summarize three groups of features that may influence the confirmation time:

- **Transaction Features**, which describes the submitted transaction.
- **Block States**, which reflects the characteristics of the mined blocks including block size, block generation speed, etc.

– **Mempool States**, which records the distribution of feerates of unconfirmed transactions in the mempool, implicitly modelling the competition among unconfirmed transactions.

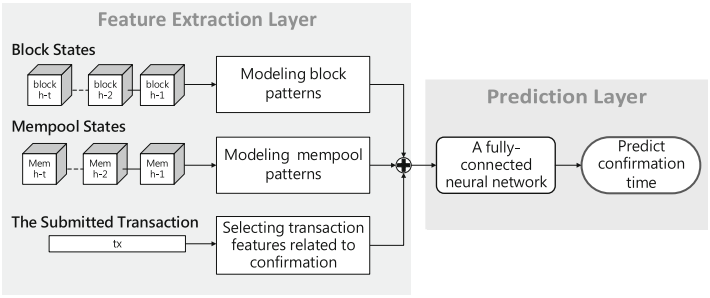

Fig. 1. The framework of CTEN.

These three groups of features correspond to the three types of information fed to the estimation function $\mathcal{F}$ in Sect. 3. Although transaction features are already available in the submitted transaction, future block states and mempool states are not known. However, such features are desirable, because if we had known how many transactions would be contained in future blocks, how fast future blocks would be generated, how competitive the submitted transaction would be in future mempools, we would increase the chance to predict the confirmation time more accurately. Consequently, in CTEN, our main idea is to predict future block states and mempool states from historical state sequences by utilizing sequence learning models. Finally, we combine the three groups of features to do the estimation.

Estimation Framework. The estimation framework can be divided into two layers, one *feature extraction layer* to extract patterns from block states, mempool states and transaction features of the submitted transaction, and one *prediction layer* to analyze the relationship between transaction confirmation time and the extracted features. Figure 1 shows the framework.

– **Feature Extraction Layer.** It includes three parts. Other than modelling the submitted transaction itself, the feature extraction layer also predicts the future characteristics of block states and model the miners' mining behavior from mempool competition states of the unconfirmed transactions.
  1. *Transaction Features* describe the details of the submitted transaction waiting for confirmation. We select the features that we believe may affect a transaction's validation and confirmation. In addition, we add an additional feature to represent its current competition position in the mempool for modelling the mining priority.

- *transaction feerate* Generally, transactions with higher feerates are likely to be more competitive than those with lower feerates.
- *number of inputs* and *number of outputs* Miners need to verify the validity of the satoshis claimed in transaction inputs by tracking historical transactions which transferred satoshis to those addresses.
- *transaction weight* It is a parameter measuring the transaction size. It is related to the blockchain protocol Segwit, and a transaction containing Segwit inputs has additional script data. In order to represent the size of transactions under different protocols, transaction virtual size (weight) is used in our work.
- *transaction first-seen time* The first-seen time refers to the time that a transaction is first observed. It is difficult to find out the exact submission time of a historical transaction, so its publicly recorded first-seen time is treated as its submission time in our work.
- *time interval to last block* It is the interval between *transaction first-seen time* and the timestamp of the last generated block. This feature is added to address the disparity in transaction confirmation time in the scenario when two transactions are submitted at the same block height but at different timestamps, since the later one may experience a shorter confirmation delay due to its proximity to the timestamp of the next block's confirmation time.
- *position in the current mempool* It aggregates the weight of unconfirmed transactions in the mempool with higher feerates than the submitted transaction. Strictly speaking, it is a feature built from both the transaction itself and the mempool.

2. *Block States* are the predicted features of future blocks, which are expected to encode future block size and generation speed, which can affect a transaction's confirmation time. Historical block states are learned as a sequence to predict future block states.
   - *block weight* and *the number of confirmed transactions* They represent the size of a block. We evaluate from two aspects, the overall weight of transactions in a block and the number of transactions in this block.
   - *difficulty* It reflects the difficulty of the mathematical problem in mining a block. In the Bitcoin system, *difficulty* is tuned to control that blocks can be generated on an average 10-minute basis.
   - *block interval* It is constructed based on the time gap between a block and its precedent block. It assesses the pace with which blocks are generated.
   - *average feerate in the block* This is the average feerate of the transactions in one block. We aim to capture the feerate trend in continuous blocks by implementing this feature.
   - *feerate distribution in the block* The feerate distribution is extracted by aggregating the weight of the transactions confirmed in one block within each feerate interval.
3. *Mempool States* Transactions in the mempool compete to be included in the next block. We model transaction competition in the mempool at a

certain block height h by computing the overall weight of transactions in the specific feerate interval r.

$$mem_h^r = \sum_{tx \in Mem_h^r} w(tx) \tag{2}$$

In this way, before each block is generated, we have a mempool, whose feature is modelled as a concatenated weight-histograms of different feerate intervals. Similar to block states, historical mempool states are learned to predict future mempool states.

In CTEN, block states and mempool states are learnt by sequence models. We implement two alternatives, LSTM and Attention. In both models, we use x_i to represent the input feature. To be specific, for block states, x_i is block characteristics b_h; for mempool states, x_i is mempool vector mem_h. t is the length of the sequence.

Approach 1: LSTM [18] aggregates information on a token-by-token basis in sequential order. Compared to the standard RNN, LSTM addresses the vanishing gradient problem by incorporating three gating functions into state dynamics. At each time step, LSTM maintains a hidden vector h and a memory vector c responsible for controlling state updates and outputs.

$$\begin{aligned} i_t &= \sigma(W^i x_t + M^i n_{t-1}) \\ f_t &= \sigma(W^f x_t + M^f n_{t-1}) \\ o_t &= \sigma(W^o x_t + M^o n_{t-1}) \\ \tilde{c}_t &= tanh(W^c x_t + M^c n_{t-1}) \\ c_t &= i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \\ n_t &= o_t \odot tanh(c_t) \end{aligned} \tag{3}$$

where $[W^i, W^f, W^o, W^c, M^i, M^f, M^o, M^c]$ are weight matrices, x_t is the vector input at the time step t, n_t is the current exposed hidden state, c_t is the memory cell state, and $\odot$ is element-wise multiplication.

Approach 2: Attention is another popular strategy for dealing with sequential data. It attempts to capture the relationships between different positions of a single sequence in order to generate a representation for the sequence. Unlike LSTM which returns the final hidden state as the extracted feature, the attention mechanism returns a new representation based on the importance in different positions in a sequence. In CTEN, the additive attention architecture [2] is used to replace the LSTM module as an alternate feature extraction module:

$$\begin{aligned} n_{t,t'} &= tanh(W^t x_t + W^x x_{t'}) \\ e_{t,t'} &= \sigma(W^a n_{t,t'}) \\ a_t &= softmax(e_t) \\ l_t &= \sum_{t'} a_{t,t'} x_{t'} \end{aligned} \tag{4}$$

where $[W^t, W^x, W^a]$ are weight matrices, x_t is the vector input (block states b_h or mempool states mem_h in this work) at time step t, n_t is the current exposed hidden state.
We also study the performance of self attention [35] and weighted attention [11]. Results are reported in the experiment section.

– **Prediction Layer** is designed to generate estimation based on the concatenated features from transaction features, block states and mempool states, which is fed to a fully-connected neural network for confirmation time estimation.

5 Experiments

We conduct experiments on real-world blockchain data to evaluate the performance of our proposed approaches DcyMean and CTEN. Experiment settings are detailed in Sect. 5.1. The performance of the approaches is reported in Sect. 5.2. In addition, we also demonstrate the efficiency of our approaches.

5.1 Experiment Settings

Datasets and Data Processing. We constructed two datasets by randomly selecting two different block intervals (S1: block range 621057–621281 and S2: block range 622057–622281) via Blockchain Explorer[2]. Each dataset has 225 blocks, the first 180 blocks are used for training (about 400,000 transaction instances) and the last 45 blocks for testing. At each block height, transactions whose first-seen time fall in between the confirmation time of the current block and the next block are considered as training instances (for the first 180 blocks) or testing instances (for the last 45 blocks).

Considering that transaction feerate in the Bitcoin blockchain system is a continuous value, we discretize it into different small intervals. In DcyMean and the mempool states in CTEN, transaction feerates are divided into 1001 intervals, with 1 being the first and increased by 1 in every step. Feerates more than 1000 are put to in the last interval 1001. Our interval division is based on a statistical study of all the transactions confirmed in the block range 621001–621500 (about 1.18 million), which shows that 99.98% transactions are with feerates under 1000.

Evaluation Metrics. In order to evaluate the accuracy on the confirmation time, we employ two measures, Mean Squared Error (MSE) and Mean Absolute Error (MAE), to report the performance. MSE and MAE are defined as follows, where the estimated result is $\hat{y}_i$ and the ground truth is y_i.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

[2] https://www.blockchain.com/explorer.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6}$$

Compared Methods. We compare the performance of our proposed DcyMean and CTEN variants.

- **DcyMean**: *dcy* is set to 0.96.
- **CTEN_Lstm** employs LSTM to extract patterns in block state sequence and mempool states sequence.
- **CTEN_Wht**, **CTEN_Self** and **CTEN_Adv** correspond to use different attention techniques in the CTEN approach for extracting features from block and mempool sequences: weighted attention [11], self attention [35] and additive attention [2].
- **Adv_Tx**, **Adv_MemTx** and **Adv_BloTx** They correspond to different feature combinations in CTEN_Adv for ablation study.
  - **Adv_Tx:** Transaction feature only
  - **Adv_MemTx:** Mempool states and transaction features
  - **Adv_BloTx:** Block states and transaction features

Implementation Details. The hidden units in sequence processing is set to 64 and the prediction layer is made up of a fully-connected three-layer neural network with hidden units 64, 8 and 1, respectively. The length of sequence is set to 3. When training models, parameters are optimized by stochastic gradient descent (SGD) with the Adam optimizer, and the objective function is set to the standard mean squared error with batch size set to 1000. All the methods are implemented in the TensorFlow framework, and all the experiments are run on one NVIDIA P100 12 GB PCIe GPU.

5.2 Result Analysis

In this part, we evaluate the performance of confirmation time estimation methods, DcyMean and CTEN.

Evaluation of DcyMean and CTEN. The estimation performance is presented in Table 2. According to Table 2, Adv_Tx outperforms DcyMean on both datasets, which reveals the potential of incorporating transaction features and employing neural networks. Meanwhile, all the CTEN models are super to DcyMean, which reinforces this conclusion. Among all the CTEN models, additive attention CTEN_Adv performs the best.

Table 2. Evaluation of methods under MSE and MAE

Model		MSE		MAE (e7)	
		S1	S2	S1	S2
CTEN	CTEN_Lstm	2532.4	2513.7	2.4	3.7
	CTEN_Wht	2611.7	2445.6	2.6	3.2
	CTEN_Self	2637.7	2491.8	3.0	3.8
	CTEN_Adv	**2070.5**	**2179.1**	**1.7**	**3.1**
Adv_Tx		2481.3	4331.4	3.2	7.4
DcyMean		3197.4	8023.1	4.8	18.8

Impact of Different Features in CTEN_Adv. We test four different feature compositions (Adv_Tx, Adv_MemTx, Adv_BloTx and CTEN_Adv) to study the importance of the features. The performance is shown in Table 3. CTEN_Adv which combines three different features, stands out among these four variants. Meanwhile, after comparing Adv_Tx and Adv_MemTx, we can see that adding mempool states to transaction features can improve accuracy significantly. Although the accuracy of Adv_BloTx on dataset S1 is lower than that of Adv_Tx, the accuracy is further improved when mempool states are added, which turns into CTEN_Adv. In conclusion, transaction confirmation time estimation can benefit from both block states and mempool states. Moreover, mempool states contribute more significantly, which sheds light on the significance of mempool competition state research.

Table 3. Impact of different features in CTEN

Model	MSE		MAE (e7)	
	S1	S2	S1	S2
CTEN_Adv	**2070.5**	**2179.1**	**1.7**	**3.1**
Adv_MemTx	2168.5	2330.1	2.1	3.2
Adv_BloTx	3819.2	2904.3	5.7	5.2
Adv_Tx	2481.3	4331.4	3.2	7.4

Time Efficiency of CTEN. We conduct another set of experiments to study the training time of the models under 100 training epochs, with 8G Memory and 1000 instances as a batch. The training time of $DcyMean$ can be ignored since the results can be computed directly by updating the estimation result at the last time step with the newly mined block. From Table 4, we can find that all the attention variants in CTEN can finish training within 10 min (roughly one block interval), which means that our approaches can serve for real-time Bitcoin

blockchain data. Moreover, compared to CTEN_Lstm, the training time of the attention variants can reduce by almost 50%.

Table 4. Training time of estimation models (seconds)

	Adv_Tx	CTEN_Lstm	CTEN_Wht	CTEN_Self	CTEN_Adv
S1	137	653	357	317	383
S2	145	648	370	323	401

6 Conclusion

In this work, we propose two approaches on estimating the confirmation time of a submitted transaction in the Bitcoin blockchain system. The first approach DcyMean computes the estimation based on decayed mean of the confirmation time of historical transactions. The second approach CTEN works on learning the relationship between transaction confirmation time and a variety of factors including block states, mempool states, and the transaction itself. Our experimental results show that CTEN outperforms DcyMean in tackling this problem on the real datasets. Meanwhile, we demonstrate that the competition in mempool is not neglectable on estimating transaction confirmation time.

References

1. Antonopoulos, A.M.: Mastering Bitcoin: Programming the Open Blockchain. O'Reilly Media, Inc. (2017)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Balsamo, S., Marin, A., Mitrani, I., Rebagliati, N.: Prediction of the consolidation delay in blockchain-based applications. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering, pp. 81–92 (2021)
4. Buchnik, Y., Friedman, R.: FireLedger: a high throughput blockchain consensus protocol. Proc. VLDB Endow. **13**(9), 1525–1539 (2020)
5. Bui, H.T., Hussain, O.K., Saberi, M., Hussain, F.: Assessing the authenticity of subjective information in the blockchain: a survey and open issues. World Wide Web **24**(2), 483–509 (2021)
6. Chaudhry, M., Templeton, J.: The queuing system M/GB/1 and its ramifications. Eur. J. Oper. Res. **6**, 57–61 (1981)
7. Chen, Z., et al.: SChain: a scalable consortium blockchain exploiting intra-and inter-block concurrency. Proc. VLDB Endow. **14**(12), 2799–2802 (2021)
8. Dang, H., Dinh, T.T.A., Loghin, D., Chang, E.C., Lin, Q., Ooi, B.C.: Towards scaling blockchain systems via sharding. In: Proceedings of the 2019 International Conference on Management of Data, pp. 123–140 (2019)

9. El-Hindi, M., Heyden, M., Binnig, C., Ramamurthy, R., Arasu, A., Kossmann, D.: BlockchainDB-towards a shared database on blockchains. In: Proceedings of the 2019 International Conference on Management of Data, pp. 1905–1908 (2019)
10. Fang, M., Zhang, Z., Jin, C., Zhou, A.: High-performance smart contracts concurrent execution for permissioned blockchain using SGX. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 1907–1912. IEEE (2021)
11. Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., Lehmann, S.: Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and Sarcasm. arXiv preprint arXiv:1708.00524 (2017)
12. Fiz, B., Hommes, S., State, R.: Confirmation delay prediction of transactions in the bitcoin network. In: Park, J.J., Loia, V., Yi, G., Sung, Y. (eds.) CUTE/CSA -2017. LNEE, vol. 474, pp. 534–539. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7605-3_88
13. Fu, R., Zhang, Z., Li, L.: 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 (2016)
14. Gundlach, R., Gijsbers, M., Koops, D., Resing, J.: Predicting confirmation times of bitcoin transactions. ACM SIGMETRICS Perform. Eval. Rev. **48**(4), 16–19 (2021)
15. Gupta, S., Rahnama, S., Hellings, J., Sadoghi, M.: ResilientDB: global scale resilient blockchain fabric. Proc. VLDB Endow. **13**(6), 868–883 (2020)
16. Han, S., Xu, Z., Zeng, Y., Chen, L.: Fluid: a blockchain based framework for crowdsourcing. In: Proceedings of the 2019 International Conference on Management of Data, pp. 1921–1924 (2019)
17. Hao, K., Xin, J., Wang, Z., Wang, G.: Outsourced data integrity verification based on blockchain in untrusted environment. World Wide Web **23**(4), 2215–2238 (2020)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
19. Kasahara, S., Kawahara, J.: Effect of bitcoin fee on transaction-confirmation process. J. Ind. Manag. Optimiz. **15**(1), 365 (2019)
20. Kawase, Y., Kasahara, S.: Priority queueing analysis of transaction-confirmation time for bitcoin. J. Ind. Manag. Optimiz. **16**(3), 1077 (2020)
21. Ko, K., Jeong, T., Maharjan, S., Lee, C., Hong, J.W.-K.: Prediction of bitcoin transactions included in the next block. In: Zheng, Z., Dai, H.-N., Tang, M., Chen, X. (eds.) BlockSys 2019. CCIS, vol. 1156, pp. 591–597. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2777-7_48
22. Koops, D.: Predicting the confirmation time of bitcoin transactions. arXiv preprint arXiv:1809.10596 (2018)
23. Ma, Y., Sun, Y., Lei, Y., Qin, N., Lu, J.: A survey of blockchain technology on security, privacy, and trust in crowdsourcing services. World Wide Web **23**(1), 393–419 (2020)
24. McNally, S., Roche, J., Caton, S.: 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 (2018)
25. Miller, D.R.: Computation of steady-state probabilities for M/M/1 priority queues. Oper. Res. **29**(5), 945–958 (1981)
26. Mukhopadhyay, U., Skjellum, A., Hambolu, O., Oakley, J., Yu, L., Brooks, R.: A brief survey of cryptocurrency systems. In: 2016 14th annual conference on privacy, security and trust (PST), pp. 745–752. IEEE (2016)
27. Nathan, S., Govindarajan, C., Saraf, A., Sethi, M., Jayachandran, P.: Blockchain meets database: design and implementation of a blockchain relational database. arXiv preprint arXiv:1903.01919 (2019)

28. Peng, Y., Du, M., Li, F., Cheng, R., Song, D.: FalconDB: blockchain-based collaborative database. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 637–652 (2020)
29. Peng, Z., Xu, C., Wang, H., Huang, J., Xu, J., Chu, X.: P2B-trace: privacy-preserving blockchain-based contact tracing to combat pandemics. In: Proceedings of the 2021 International Conference on Management of Data, pp. 2389–2393 (2021)
30. Qi, X., Zhang, Z., Jin, C., Zhou, A.: BFT-store: storage partition for permissioned blockchain via erasure coding. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1926–1929. IEEE (2020)
31. Ruan, P., Loghin, D., Ta, Q.T., Zhang, M., Chen, G., Ooi, B.C.: A transactional perspective on execute-order-validate blockchains. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 543–557 (2020)
32. Sharma, A., Schuhknecht, F.M., Agrawal, D., Dittrich, J.: Blurring the lines between blockchains and database systems: the case of hyperledger fabric. In: Proceedings of the 2019 International Conference on Management of Data, pp. 105–122 (2019)
33. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using LSTMs. In: International Conference on Machine Learning, pp. 843–852 (2015)
34. Tao, Y., Li, B., Jiang, J., Ng, H.C., Wang, C., Li, B.: On sharding open blockchains with smart contracts. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1357–1368. IEEE (2020)
35. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
36. Wang, H., Xu, C., Zhang, C., Xu, J.: vChain: a blockchain system ensuring query integrity. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2693–2696 (2020)
37. Wolff, R.W., Yao, Y.C.: Little's law when the average waiting time is infinite. Queueing Syst. **76**(3), 267–281 (2014)
38. Xu, C., Zhang, C., Xu, J.: vChain: enabling verifiable Boolean range queries over blockchain databases. In: Proceedings of the 2019 International Conference on Management of Data, pp. 141–158 (2019)
39. Xu, C., Zhang, C., Xu, J., Pei, J.: SlimChain: scaling blockchain transactions through off-chain storage and parallel processing. Proc. VLDB Endow. **14**(11), 2314–2326 (2021)
40. Xu, Z., Chen, L.: DIV: resolving the dynamic issues of zero-knowledge set membership proof in the blockchain. In: Proceedings of the 2021 International Conference on Management of Data, pp. 2036–2048 (2021)
41. Yan, Y., et al.: Confidentiality support over financial grade consortium blockchain. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2227–2240 (2020)
42. 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)
43. Zhang, C., Xu, C., Xu, J., Tang, Y., Choi, B.: GEM∧ 2-tree: a gas-efficient structure for authenticated range queries in blockchain. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 842–853. IEEE (2019)
44. Zhang, Z., et al.: Refiner: a reliable incentive-driven federated learning system powered by blockchain. Proc. VLDB Endow. **14**(12), 2659–2662 (2021)

45. Zhao, W., Jin, S., Yue, W.: Analysis of the average confirmation time of transactions in a blockchain system. In: Phung-Duc, T., Kasahara, S., Wittevrongel, S. (eds.) QTNA 2019. LNCS, vol. 11688, pp. 379–388. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27181-7_23
46. Zhu, Y., Zhang, Z., Jin, C., Zhou, A., Qin, G., Yang, Y.: Towards rich Qery blockchain database. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM 2020, pp. 3497–3500 (2020)
47. Zhu, Y., Zhang, Z., Jin, C., Zhou, A., Yan, Y.: SEBDB: semantics empowered blockchain database. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1820–1831. IEEE (2019)