Predicting Transaction Latency with Deep Learning in Proof-of-Work Blockchains

Enrico Tedeschi*, Tor-Arne S. Nordmo*, Dag Johansen* and Håvard D. Johansen*
* UIT: The Arctic University of Norway
Emails: (enrico.tedeschi, tor-arne.s.nordmo, dag.johansen, havard.johansen)@uit.no

***Abstract*—Proof-of-work based cryptocurrencies, like Bitcoin, have a fee market where transactions are included in the blockchain according to a first-price auction for block space. Many attempts have been made to adjust and predict the fee volatility, but even well-formed transactions sometimes experience delays and evictions unless an enormous fee is paid. In this paper, we present a novel machine-learning model, solving a binary classification problem, that can predict transaction fee volatility in the Bitcoin network so that users can optimize their fees expenses and the approval time for their transactions. The model's output will give a confidence score whether a new incoming transaction will be included in the next mined block. The model is trained on data from a longitudinal study of the Bitcoin blockchain, containing more than 10 million transactions. New features that we generate include information on how many bytes were already occupied by other transactions in the mempool, assuming they are ordered by fee density in each mining pool. The collected dataset allows to generate a model for transaction inclusion pattern prediction in the Bitcoin network, hence telling whether a transaction is well formed or not, according to the previous transactions analyzed. With this, we obtain a prediction score for up to 86%.**

***Index Terms*—Bitcoin, blockchain, longitudinal study, performance, transaction latency, machine learning, neural networks.**

I. Introduction

Bitcoin was intended to provide its users with a low-cost payment scheme, with transaction fees close or equal to zero [1]. In a tragedy of the commons, the cryptocurrency's rising popularity made the inherent throughput limitations of the underlying Proof-of-Work (PoW) scheme for establishing consensus a key scalability bottleneck [2, 3]. The high cost of mining has led to an increased usage of transaction fees as a means for miners to make a profit. Users are experiencing this as delays and performance issues, and the cost of transactions have become nonzero and volatile. Transactions offering low fees are now typically experiencing higher transaction latency and, after 2016, zero-fee transactions were evicted from most of Bitcoin's miners [4]. For Bitcoin, transaction fees are intended to replace miner's minting reward in the long run [5].

As a consequence, a transaction fee market has emerged where transactions are included according to a first-price auction for block space. All bidders (users) submit sealed bids (transaction fee) [6] and the highest bids will have the highest chances of being included in the next block. Although new consensus techniques have been proposed with an indirect scope to also improve scalability, such as Proof-of-Stake (PoS) [7, 8] or Proof-of-Storage (PoSt) [9], Bitcoin cannot easily accommodate for such changes and its users must expect to bid and pay transaction fees to have their transactions included in the blockchain.

This first-price auction for Bitcoin block space is ultimately bad for most users, and more so as the number of daily transactions increases [4]. However, the alternatives of deciding on a common static fee is not possible [5, 10] and providing instant confirmation remains a key challenge [11]. Users are instead left to chose an appropriate payment dynamically when submitting their transactions with no exact formula to optimize expenditure or to control the time it takes for a transaction to be confirmed.

Current transaction fee estimators, such as the one implemented by Bitcoin Core, primarily use historical fee data. These estimators are susceptible to manipulation since miners can arbitrary add transactions to themselves with a high carried fees. Applications following these estimators observe that the fee to be paid is higher than what actually is needed. Other fee estimation algorithms interact with each other in unpredictable ways, which can lead to oscillating fee estimates. Many users end up using their own estimator based on their intuition [6]. This turns out to be a difficult task in practice and in most cases becomes expensive for the users [12] as they often end up with estimators that have the notorious problem of aggregate overpaying. In 2017, poorly designed fee estimators contributed to driving up average Bitcoin fees to over $20 per transaction [4, 6].

In this paper, we propose a transaction inclusion model based on data of historic blocks. Our model is constructed using deep-learning methods such as Neural Networks (NNs) [13], Deep Neural Networks (DNNs), and Residual Neural Networks (ResNets). The aim is to predict whether a certain transaction has a good chance of being included immediately ($\sim$ 10 min) in the next block. Note that we want to provide information on transaction inclusion and not the actual approval time ($\sim$ 6 blocks). This because our assumption is that if a transaction is included in a block, it will be approved regardless. We assume that because, after the inclusion, the waiting time for each additional confirmation is completely independent of the transaction fee, since the transaction has already been included in the blockchain. We develop our model by using data collected daily from the Bitcoin blockchain. We collect and gather longitudinal data, batch them, and for each one we create a prediction model which will be used for the subsequent batch.

II. Background

Due to the cost of mining blocks, most miners are assumed to behave rationally [14], but with individualized policies for inclusion that maximize own profit. These inclusion policies are not publicly known, and they might differ from one miner to the next. We conjecture that miners will depend more on users' fee to keep the network and mining alive, and that they are not only relying on transaction fees to include new transactions. From that, we argue that with the right selection of features, we can generate a model which can predict whether a transaction will be included in the next block. This idea relies on *fee rates* or *fee density* (ρ), and the limitation that miners have with respect to the block size Q, considering the *offset* (δ) of the block space already occupied by previous not-yet-approved transactions.

A. Models

According to the work of Rizun (2015) [2], each mined block has an expected profit $\langle \Pi \rangle$, of

$$\langle \Pi \rangle = \langle V \rangle - \langle C \rangle \tag{1}$$

where $\langle V \rangle$ is the expected revenue for mining and having your block approved in the network, and $\langle C \rangle$ is the expected hashing cost.

If we ignore factors that are not directly limited by the network, such as the monetary hashing cost $\langle C \rangle$, which depend on the individual hashing rate; the block creation time, which is normally distributed with a fixed known mean; and if we also ignore the probability of block orphaning $\mathbb{P}_{orphan}$ due to block propagation delays, which is shown to be mostly dependent on the block size [15], the total expected revenue can be expressed as

$$\langle V \rangle = (R + M)\frac{h}{H} \tag{2}$$

Here h is the individual hashing rate, H is the total hashing power of the network, R is the reward for mining a block, for Bitcoin currently at $12.5\,\text{฿}$. M is the sum of all the transaction fees t_f in a block including N transactions and is formalized as:

$$M = \sum_{i=1}^{N} t_f^{(i)} \tag{3}$$

Based on the Bitcoin Unspent Transaction Outputs (UTXO) model, if n is the number of transaction inputs in a certain block, and m is the number of transaction outputs, a transaction fee t_f is calculated

$$t_f = \sum_{i=1}^{n} t_{in}^{(i)} - \sum_{j=1}^{m} t_{ou}^{(j)} \tag{4}$$

We can then state, that the expected profit $\langle \Pi \rangle$ gets lower when H increases. Furthermore, R is halved every 210 thousands blocks, making transaction fees, M, the only source of profit for miners.

Permissionless blockchains using PoW are hardly efficient in energy usage, causing miners to be more dependent on t_f as the network scales. If the number of miners scales, then the difficulty raises in order to keep up with the block creation time, having then more energy consumption, hence the fees are higher in order to pay back the miners. On the other hand, if the number of daily transactions scales, there is more competition to be included immediately in the next block, and a rational miner would prioritize higher fee transactions. Increasing the hashing power does not add any better performance related to transaction throughput, while it only increases the overall energy consumption. This is due to the network difficulty d (Equation 5), which allows to mine new blocks, and which is normalized according to $\mathcal{T}' \simeq 2016\,\mathcal{T}$, where $\mathcal{T} = 10\,\text{min}$ is the fixed block creation time. It does not matter then how powerful the network is, and as long as the block size is fixed, it will be impossible to gain any performance. Difficulty d at time t is defined as

$$d_t = \begin{cases} 1, & \text{if } t = 0 \\ d_{t-1}\frac{2016\mathcal{T}}{\mathcal{T}'}, & \text{if } t > 0 \end{cases} \tag{5}$$

where $\mathcal{T}'$ represents the calculated actual time in minutes spent mining the previous 2016 blocks. Consequently, t_f is expected to grow with the network size, making a system based on permissionless PoW blockchain difficult to maintain without a substantial revenue for the miners (i.e., money put into the system by its users). Our work is fundamental to define how a well-formed transaction should look like, so that miners will have their fair revenue, but still such that users have acceptable fee-performance trade-off.

B. Prediction Methods

Machine-learning models have previously been shown effective at finding patterns in data streams in various fields and sectors, ranging from education [16, 17], business and marketing [18, 19], healthcare [20, 21, 22], financial services [23, 24], and transportation [25, 26]. Deep learning is part of a broader family of machine learning methods based on artificial NNs [27, 28, 29]. ResNet models are a variation of deep neural networks, and are implemented with double (or triple) layer skips that contain nonlinearities Rectified Linear Unit (ReLU) and batch normalization in between [30]. These skip connections help with gradient flow in deeper neural networks, which aims to reduce training and test error [31, 32]. The versatility and efficiency of these method has motivated us to use them in this work.

III. Methodology

To perform a prediction based on historic blocks, we need a local instance of the portion of the Bitcoin blockchain that needs to be analyzed. We fetch data from the Bitcoin blockchain and generate a dataset D, containing the features relevant for our model. We wanted to have our instance of the blockchain locally in order to save up space on disk while keeping only the information we needed for the evaluation. We collected data using a third-party APIs,[1] also gathering

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

information about money exchange price with libraries such as *forex-python*.[2]

With our dataset D, we build a prediction model for transaction inclusion pattern using machine-learning models. Having knowledge of PoW-based blockchains, we can then make assumptions on which features might be relevant for the model. From D we derive the feature set ($\mathbb{F}$) for the prediction, and then perform a supervised classification using different deep learning approaches to find the one that performs the best.

Not all the features we wanted were in the information available directly on the Bitcoin blockchain. Some features were generated by gathering information from multiple sources or from multiple features. One example of this typology is Δt_{ep}, which gives information for each transaction t, on how much time is elapsed from the previous block creation time to t's timestamp, t_{ep}. For t_{ep} we refer to the transaction time registered by the API's node, so it does not include information about the real transaction's timestamp, since the latter is not available in Bitcoin.

A. Data Acquisition and Feature Selection

Our feature set $\mathbb{F}$ is formed by both, fetched (Φ) and derived ($\mathbb{D}$) features,

$$\mathbb{F} = \Phi \cup \mathbb{D},$$

where

$$\Phi = \{t_q,\, t_{in},\, t_{ou},\, B_{mi}\},$$

and

$$\mathbb{D} = \{t_f,\, \rho,\, t_{\%},\, \Delta t_{ep},\, \delta\}.$$

For our model, we select the features that have most impact on transaction (t) latency. These are as follows:

- t_q Transaction size, in bytes. The block size (Q) restrains the number of transactions that are included in a certain block B, limiting the throughput (γ) and indirectly also the revenue (M). As long as $Q = 1\,\text{MB}$, t_q will always play a role in transaction selection by miners.
- t_f The transaction fee directly affects the revenue M of miners. Rational entities then would choose transactions with higher fees when the systems scales.
- ρ The fee density is a derived features using the previous two. It will enhance the correlation and the importance of the transaction size and the fee. We also assume that a rational miner is ordering the incoming unapproved transactions by fee density.
- $t_{\%}$ The percentage paid in fee matters since it is important to contextualize the fee with the total transaction's amount. If this feature would not matter, it will be impossible to execute small transactions since they will be surmounted by big carrying-fee ones.
- Δt_{ep} Waiting time for a transaction t. This is the time elapsed from the transaction timestamp (t_{ep}), until the latest B_{ep}. The time elapsed up to the latest block is relevant since miners can prioritize older transactions over ρ. By intuition, we assume that a transaction can not wait forever for approval if it has paid a fair amount of t_f. This will also avoid that well formed transactions will not lose their space in the block just because some newer transaction with higher fee came right before a newly mined block.
- δ Represents the offset in bytes. It defines for each transaction, the amount of bytes already occupied by better unapproved transactions in terms of ρ, in an hypothetical next block. The offset is relevant to give each transaction a place in the future block, assuming that miners are rational and are ordering the incoming transactions by ρ, then they are limited by the block size Q. Greater is the offset fewer are the chances that a transaction is included in the next block.

The features in Φ are directly available on the blockchain. Those in $\mathbb{D}$ are derived. For instance, t_f is calculated as showed in Equation 4, while the fee density ρ of a transaction t is obtained from the following equation:

$$\rho_t = \frac{t_f}{t_q}. \tag{6}$$

Other features were more elaborate and difficult to obtain, like Δt_{ep} or δ. During model training we need to define new concepts in order to contextualize every single transaction analyzed to the moment of their inception. The feature Δt_{ep} is calculated as showed in (7), where $Bs_t^{(i)}$, represents successor/ predecessor of a transaction t. The successor is the first block epoch (B_{ep}) mined after t's inception time (t_{ep}) if $i = 0$, while the predecessor is the first previous B_{ep} of t_{ep} if $i = -1$.

$$\begin{aligned} &\Delta t_{ep} = t_{ep} - Bs_t^{(-1)}, \\ \text{where}\quad &Bs_t^{(-1)} \le t_{ep} < Bs_t^{(0)}. \end{aligned} \tag{7}$$

To explain the offset δ, we define the set of transactions S, showed in (8). Then each offset δ_S is a growing number, starting from 0, in a particular set S, where the number of sets goes from 0 to $\max(B_{he})$[3]. In the set S are contained all the transactions whose their t_{ep} is included in a certain timespan, ΔBs_t. We order these transactions in S by their fee density, ρ, and define the offset, δ, as showed in (9) and (10).

$$\begin{aligned} &S = (t^{(0)},\, t^{(1)},\, \ldots,\, t^{(n)}), \\ \text{where}\quad &\rho^{(0)} < \rho^{(1)} < \cdots < \rho^{(n)}, \\ \text{and}\quad &Bs_{tj}^{(-1)} \le t_{ep}^{(j)} < Bs_{tj}^{(0)}, \quad \text{for } j = 1, 2, \ldots, n \end{aligned} \tag{8}$$

In Equation 8, for readability, $t^{(j)} \equiv tj$, and $B_{tj}^{(i)}$ represents the successor or predecessor for transaction $t^{(j)}$. Equation 9 represents the offset for a transaction $t^{(n)}$ included in a set S.

$$\delta_S^{(n)} = \begin{cases} t_q^{(j)}, & \text{if } j = 0 \\ \delta_S^{(j-1)} + t_q^{(j)}, & \text{for } j = 1 \text{ to } n \end{cases} \tag{9}$$

[2]https://pypi.python.org/pypi/forex-python

[3]B_{he} represents the block height.

We can then simplify (9) as showed in (10).

$$\delta_S^{(n)} = \sum_{j=0}^{n} t_q^{(j)} \qquad \forall\ S_k, \\ \text{where} \quad 0 \leq k \leq max(B_{he}). \tag{10}$$

B. Prediction Model Hyperparameters and Parameters

Two different models, solving a binary classification problem, are generated and evaluated: DNN and ResNet. The latter, contains skip connections between more distant layers. These skip connections help with gradient flow in deeper neural networks, which aims to reduce training and test error [31, 32]. The output of both models represent the probability for a transaction to be immediately included or not in the next block. The model's output is an array represented with $\boldsymbol{\theta}$, where for each transaction t, $\boldsymbol{\theta}^{(t)}$ is:

$$\boldsymbol{\theta}^{(t)} = [\, P_t(\upsilon_0),\, P_t(\upsilon_1)\,] \tag{11}$$

and $P_t(\upsilon_i)$ indicates the probability, or confidence, for a transaction t to be labeled in class υ_i. With υ_i we refer to the class in which a transaction t belongs. Our task is a binary classification problem, hence $i \in \{0, 1\}$ and the class indicates whether a transaction t is well formed or not, in order to be approved in the next block. The class υ_1 contains all the transactions that according to the model will be included in the next block, while υ_0 represents the class for transactions which are not good enough to be included in the next block.

An example of DNN representing one of our models is showed in Fig. 1. We changed the number of hidden layers during test, but the activation function used was a ReLU for each node in the network, except for the output layer, where we used a Normalized Exponential Function (or softmax). The weights were initialized with He normalization, which takes into account ReLU and it makes it easier for deep models to converge [33]. The tested models were implemented using Keras[4] with Tensorflow backend.[5]

Parameters that cannot be estimated from data, known as hyperparameters, are set manually by trial and error. This includes the number of hidden layers, number of skip connections, the batch size, and the epoch for each model. The batch size controls the granularity or precision of gradient descent, meaning that the optimization function, so the model internal parameters, are optimized every *batch size* of tuples. The epoch instead, represents the number of times that the learning algorithm will work through the entire training dataset, ideally, getting closer to the optimal solution at every iteration. The model's hyperparameters are configured to optimize the model performance and accuracy.

Our models, using the set of features $\mathbb{F}^6$ (12), aim to take into account how trends and policies for transaction inclusion and eviction change over time. Therefore, if our assumptions on which are the most relevant factors influencing transaction inclusion are correct, we can build accurate models despite the system's dynamicity.

[4]https://keras.io/

[5]https://www.tensorflow.org/api_docs

IV. OBSERVATIONS

The purpose of our observations is to make sure that our models can give an accurate prediction on what the transaction inclusion pattern will be on newer transactions. This will help users in making decisions regarding the transaction fee to pay, knowing how likely that transaction will be included in the blockchain. The metrics used for the test include accuracy, precision, and recall. We perform our tests on transactions registered on the Bitcoin blockchain from 6^{th} April 2019 until 23^{rd} May 2019. We take into account almost 20 million of transactions over ~ 7 thousand blocks.

For testing, DNN and ResNet models are used. Tests also include different set of features, which in our case are $\mathbb{F}^5$ and $\mathbb{F}^6$, defined as:

$$\mathbb{F}^5 = \{t_q,\, t_f,\, \rho,\, t_{\%},\, \Delta t_{ep}\}, \\ \mathbb{F}^6 = \mathbb{F}^5 \cup \{\delta\}. \tag{12}$$

Depending on the cardinality of the feature set, $|\mathbb{F}^c| = c$, our model names will identify the model's type and the feature set used, for instance, DNN6 identifies the deep neural network model having $\mathbb{F}^6$ as feature set.

Data from diverse type measurements can have different scale and the biggest can dominate the model's algorithm. Therefore, during pre-processing phase we normalize our training and test dataset ($\boldsymbol{X}$ and $\boldsymbol{Xte}$) to have an homogeneous dataset, based on the mean and variance of the features rather than single values (Algorithm 1). We also noticed that the number of transactions belonging to the two classes, υ_0 and υ_1, is always unbalanced, with an average score of 35% for class υ_0 and 65% for class υ_1. Because of that, we decided to test our models with both, balanced, and unbalanced but weighted classes. We refer to the balanced models with (*), for instance, DNN* or ResNet*.

Algorithm 1 Normalization of $\boldsymbol{X}$

procedure NORMALIZATION($\boldsymbol{X}$, $\boldsymbol{Xte}$)
- $\mu \leftarrow$ Mean($\boldsymbol{X}$) ▷ expected value
- $\sigma \leftarrow$ Std($\boldsymbol{X}$) ▷ standard deviation
- $\boldsymbol{X}_{norm} \leftarrow {}^{(\boldsymbol{X}-\mu)}/_{\sigma}$ ▷ normalizing training set
- $\boldsymbol{Xte}_{norm} \leftarrow {}^{(\boldsymbol{Xte}-\mu)}/_{\sigma}$ ▷ normalizing testing set
- **return** $\boldsymbol{X}_{norm}, \boldsymbol{Xte}_{norm}$

For performance evaluations we use the holdout method, which belongs to cross validation [34, 35, 36] class. Holdout method results optimal when the dataset contains n elements, where $n \to \infty$. Plus, the holdout method has a lower computational overhead if compared with leave-one-out and k-fold cross validation methods, and it still keeps the training and test set independent, unlike happens in residual methods [35]. We then train a model with 85% of the total training set, we compute the metrics on the remaining 15% (validation set), where the model is adjusted over all training data, and then we compute the metrics on test set, which is a complete new set, with newer transactions.

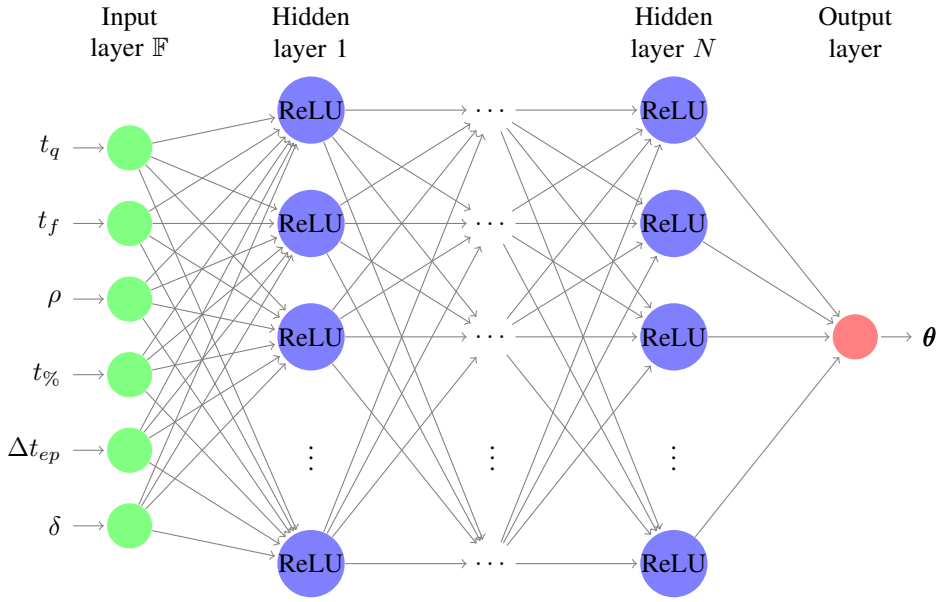


Fig. 1: Neural Network representing our DNN6 model. For each hidden layer a ReLU function is used, the number of hidden layers varies between different tests.

The metrics used to test our models, over the binary classification problem (υ_0, υ_1), are based on the confusion matrix ($\boldsymbol{A}$) defined as:

$$\boldsymbol{A} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}, \tag{13}$$

where a_{ij} is the number of elements which truly belong to i but were classified in j. The metrics obtained from $\boldsymbol{A}$, are represented in Equation 14 and defined as follows:

R_i Recall for class υ_i. Tells the number of $t \in \upsilon_i$ which were correctly classified in class υ_i.

P_i Precision for class υ_i. Tells the number of data points classified in υ_i which actually belongs to υ_i.

A Overall accuracy. How many elements were rightly classified.

$$\begin{aligned} R_i &= \frac{a_{ii}}{\sum_{j=0}^{1} a_{ij}}, \quad P_i = \frac{a_{ii}}{\sum_{j=0}^{1} a_{ji}}, \\ A &= \frac{\sum_{i=0}^{1} a_{ii}}{\sum_{i=0}^{1}\sum_{j=0}^{1} a_{ij}} \end{aligned} \tag{14}$$

For clarity, we will represents these metrics with their value in percentage score in relation with the total transactions analyzed.

We train our models by batching the same time frame into twenty or five days of transactions, tested over the next fifteen or five days. Our purpose other than having an high score in accuracy, is (1) to test different feature sets ($\mathbb{F}^5$ and $\mathbb{F}^6$), (2) enhance differences using both models (DNN and ResNet), and (3) highlight how the class balancing can affect our results. The next subsections describes two different analysis and it underlines model's performance following our metrics and points (1) (2) (3).

A. *Twenty Days Analysis*

The first analysis is done over a training time period of twenty days and a test time of fifteen days. Data are trained considering transactions from 6^{th} of April 2019 to 26^{th} of April 2019, and tested for the next following days until 11^{th} May 2019. In Fig. 2a the scores of our metrics for two different models, ResNet and DNN, are represented respectively with dashed and continuous lines, while two different feature set, $\mathbb{F}^5$ and $\mathbb{F}^6$, are listed with light blue and red colors. From Fig. 2a we can see that the two different models have almost an identical trend if the same input space is considered.

Analyzing point (1), when the input space does not include δ ($\mathbb{F}^5$), both models have an accuracy score which is $\simeq 14\%$ lower than $\mathbb{F}^6$ models. We see that ResNet5* has more difficulties in classifying transactions which the real class is υ_1, having $R_1 \simeq 65\%$, while it performs better in classifying transactions when their real label is υ_0, with $R_0 \simeq 75\%$. Without δ the model has no information on the block size limitation, so the reason of overpopulating υ_0 might be that it classifies in υ_1 only extremely profitable transactions, ignoring the fact that is better to have lower fee transactions rather than no transaction at all. By adding δ, having $\mathbb{F}^6$ as input space, accuracy is higher and the biggest improvement in the model is related to R_1, that goes from $\sim 65\%$ to $\sim 85\%$. This means that the model significantly improved in assigning transactions $t \in \upsilon_1$ to the their right class υ_1. Finally, looking at Fig. 2b, the increment from a two dimensional feature space, $\mathbb{F}^2 = \{t_q, t_f\}$, to a six dimensional one, $\mathbb{F}^6$, is significant in terms of accuracy. A big increment comes with Δt_{ep}, where both models go from an accuracy of 67% to 70%. However, the biggest increment comes for both models when the feature δ is added, with an increment of $\simeq 10\%$. The scores for both

models are constant to $\sim 66\%$ with $\mathbb{F}^3 = \mathbb{F}^2 \cup \{\rho\}$ and $\mathbb{F}^4 = \mathbb{F}^3 \cup \{t_{\%}\}$.

Referring to point (2), we notice a small boost in accuracy if ResNet is used, the difference is less than 1%, but since there is no computational ovrehead in running ResNet instead of DNN, we preferred the first to the latter. In all the metrics, ResNet6* scores above 80%, with an overall accuracy of 83.32%, while DNN6* is less precise in assigning $t \in \upsilon_0$ to their right class υ_0, having a total accuracy of 83.13%.

In this analysis, we show the importance for both models to have knowledge of the block space Q, thanks to δ. The training set entropy with such new information is significantly higher.

Both models weakness is R_0, which means that it is harder to correctly classify transactions which should not be immediately included. However, the R_0 score for both models is $\sim 80\%$, which means that only less than 20% of transactions $t \in \upsilon_0$ are not correctly classified.

B. Five Days Analysis

In the five days analysis we run the same models over a batched dataset, for a total time frame of one month. Each batch identifies transactions occurred in five days, the training is performed for each batch, and the test is done over the following five days. The method is straightforward, if the training set is represented by $\boldsymbol{X}$, and the dataset of the labels as $\boldsymbol{Y}$, then we train and test each batch as showed in Algorithm 2. For each batch we build a model, then we run it with data from the next batch $\boldsymbol{X}_{i+1}$, obtaining in this way the predicted values $\hat{\boldsymbol{Y}}$ for the i^{th} batch, $\hat{\boldsymbol{Y}}_i$.

Algorithm 2 Training Batched Dataset $\boldsymbol{X}$

```
procedure TrainBatch(X, Y n_batches)
    for 0 ≤ i < n_batches do
        ResNet_i ← Train(X_i, Y_i)
        Ŷ_i ← ResNet_i(X_{i+1})
```

If we observe Fig. 3, accuracy (A) varies from 80% til 86%, and both models have a similar trend. However, to focus on (3), we notice that ResNet6* ability to correctly classify positive samples is same as its ability to correctly classify negative samples, which makes ResNet6* a more balanced model than ResNet6. $P_{0(1)}$[6] for both models have a similar trend, which means that, independently from the class balancing, ResNet has good precision for classes, since most of the classified transactions in $\upsilon_{0(1)}$ actually belong to $\upsilon_{0(1)}$. Even if the precision is high for ResNet6, with scores above 80%, the recall $R_{0(1)}$ show that in ResNet6 the good $P_{0(1)}$ is caused by the significantly higher number of $t \in \upsilon_1$ over $t \in \upsilon_0$. The misclassification mostly occurs for transactions which belong to υ_0 but are classified in υ_1 instead, therefore, rightly classified transactions in υ_1, showed by P_1, outnumber misclassified transactions, resulting in good precision score. We can tell by observing Fig. 3 that ResNet6 has trouble in classifying $t \in \upsilon_0$, since R_0 has a score that goes from 61% to 71%, while in ResNet6* it goes from 73% to 83%.

The precision is in line with the accuracy, between 80% and 90%, which means that, despite having the same days for training and test, which might drag accuracy down, the model is still confident on rightly classifying most of the transactions. We also tested DNN6 and DNN6* and they resulted to have a lower accuracy in all tests by at least 1%.

[6]with the nomenclature $P_{0(1)}$ we refer to P_0 and P_1

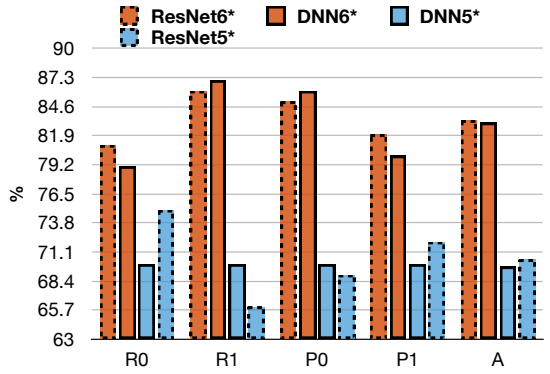
V. DISCUSSIONS

This paper wants to be the beginning of a broader project, a project which aims to educate users on how they can spend their fee in a PoW-based blockchain, in a way to optimize their expenses while miners or workers, are guaranteed a fair revenue for their work. The first step was then to create a prediction model based on historic blocks, which enables the implementation of future systems who wants to include such model in order to educate users on transaction fee.

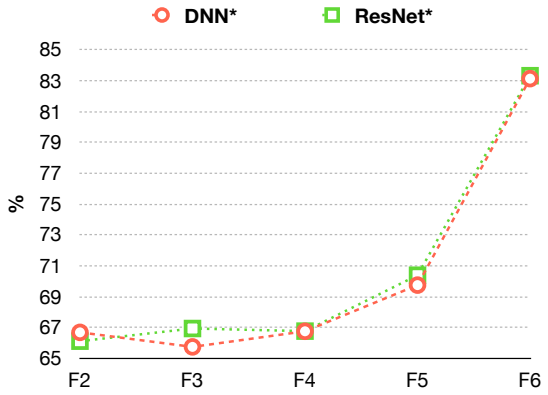
The inclusion pattern estimation problem is proved to be a not trivial one, since it depends on transaction fee estimation. the latter involves some unpredictable factors, such as supply unpredictability, demand unpredictability and different requirements from users. The supply is predictable only in the long run, having approximately 2 MB of space every 10 minutes, while is unpredictable over shorter time. Having a Poisson distribution over space supply means that we might have one block over a hundred discovered within 7 seconds from the previous, and one in a hundred discovered after 45 minutes the previous [37]. Also the demand is more predictable in the long run, since there is some cyclicity in transaction flow. However, it is hard to predict the demand on a shorter period, and since transaction fees will change following the demand, fee estimators will suffer of that.

Our model, since it includes the offset δ, aims to learn the demand unpredictability over time, and it tries to give an accurate confidence based on the network demand. We do not consider in our model the network supply, since the model predicts whether a transaction has probability of being inserted in the next block, independently on when the next block will be mined.

We gather data and build our dataset with the assumption to have a complete blockchain, including every transaction present in it for a particular time frame. Because of that it will be possible to generate the expected δ for every transaction. Due to this assumption, the cost of calculating δ is computationally high in a considerable large dataset, having for twenty days analysis a total training time of 6 hours, where 3 hours are used for calculating δ. The process of fetching δ over a five days dataset takes from 1 to 2 hours, while the total time of training goes up to 2 hours and 40 minutes. However, the offset gives a boost in the model accuracy of $\sim 15\%$, thus it will not be possible to exclude it from the analysis, even if the model will train faster. Furthermore, our local instance of the blockchain saves 44% of the space on disk, making it a valuable dataset even for other purposes.

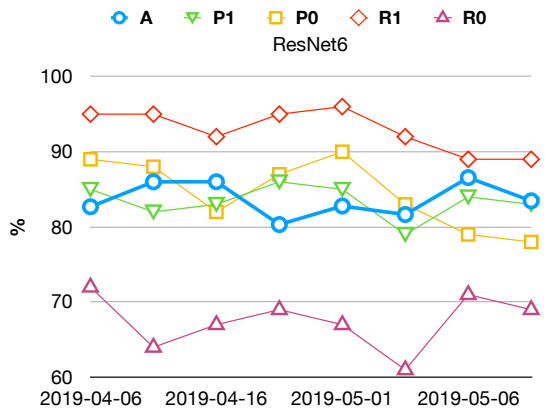


(a) Recall, precision and accuracy of four different models (DNN5*, ResNet5*, DNN6*, ResNet6*) trained from 6^{th} April 2019 to 26^{th} April 2019 and tested on transactions until 11^{th} May 2019. The red bars indicate $\mathbb{F}^6$ models while the blue ones $\mathbb{F}^5$. The dashed bars represent ResNet while the continuous DNN.

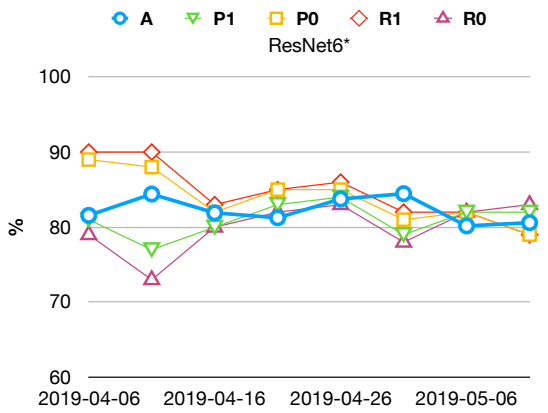


(b) Graph showing the improvement in accuracy for different models, DNN and ResNet, when the feature set is incremented from $\mathbb{F}^2$ to $\mathbb{F}^6$.

Fig. 2: Twenty days (2a) and features analysis (2b).



(a) Without class balancing



(b) Class balancing

Fig. 3: ResNet6 and ResNet6* models over one month, tested in batches of five days training and five days test. Precision, recall and accuracy are represented for each batch, from 6^{th} April 2019 til 6^{th} May 2019. The left figure represents the score using the ResNet6 model with no class balancing, hence, classes are only weighted. The figure on the right instead, represents the score for a balanced model, ResNet6*.

The two models, DNN and ResNet, got very similar results; we assume that the cause of it is the simplicity of the data distribution, which means that the loss function surface remains smooth, therefore with our data, it is way more important to select the right feature set rather than a particular model. The reason for the accuracy ceiling, not higher than $\sim 86\%$, might be caused by data noise and related to the unpredictability factors mentioned above.

We also need to consider that the accuracy is calculated by rounding off the probability output value to either 0 or 1, thus a transaction might get classified in υ_1 with a confidence of only 51%. An user will get the confidence instead, so with 51% confidence, the user might reconsider its fee. In our analysis, misclassified transactions got a quite low confidence if compared to the ones which were rightly classified.

Future improvements of the model have been thought. First of all, we assume that miners have different policies for transaction inclusion, then we want to include another feature, B_{mi}, that indicates which miner mined a certain block. In this way an user can have different confidences, each one representing the probability of inclusion if the next block is mined by B_{mi}. The output confidences will be paired together with the probability $P(B_{mi})$, which indicates the probability for B_{mi} to mine the next block, given the historical record for instance, of the last two months of miner's activity.

Even if we do something slightly different than purely estimating transaction fee, it would not be difficult to get the expected approval time of a transaction from our model, since

it depends directly on the inclusion pattern. Because of that, it will be possible to benchmark the system with a simple fee and volume strategy currently in use, such the one implemented in Bitcoin Core.

VI. CONCLUSIONS

The current Bitcoin transaction fee market is based on a first-price auction principle, which is not an optimal solution for PoW-based blockchain systems. Such fee markeds make transaction inclusion for users a complex task that often ends up disadvantaging them by paying an unfair amount of fee. However, transaction fees are still the primary motivation for encouraging mining. Without it, soon the miners will not have any other revenue, It is therefore important to balance miners profit with a fair, but not too high, fee paid by users.

Our proposed model does not want to be a fee estimator that impose a fee that will be paid to miners. Instead, our goal is to learn from previous blocks in order to give users a confidence on how much well formed their transactions are. In a way that, users can use this information to trade their fee with better latency in the network. Our assumption is that miner does not select a transaction only by its carried fee. We select several other features we believe that have impact on how miners include transactions. We also assume that transactions are ordered based on their fee density in a miner's mempool. To each transaction we assign a successor and a predecessor as explained in Section III, then from the retrieved dataset we derive our new features, *offset* and *delta*, and finally add those to our training dataset.

We propose two machine-learning models, DNN6*, and ResNet6*, both having as input space $\mathbb{F}^6$ set, with the purpose of helping users in understanding the transaction fee trend related to the auction fee market. By studying the transaction inclusion pattern, users can optimize their expenses with no loss in transaction latency, and still miners can get their expected revenue without big losses.

Considering the difficulty of having a pattern when the creation time is a randomized process, and the miner's policy of inclusions are unknown, we obtained significant results, thus confirming the importance of some features we believed to be relevant. While analyzing more than 10 million transactions for the twenty days analysis and batches of 1.5 million for the five days analysis, we obtained an accuracy between 80% and 90%, even though we did not use the confidence given by the model as users might do, but a rounded off value as explained in Section V, hence, the information an user can get by using the model it goes beyond model's accuracy score. Therefore, by following the confidence of these models, users can perform their transactions in order to have a trade off between the fee paid and the confidence of $P(\upsilon_1)$.

ACKNOWLEDGEMENT

This work is funded in part by Research Council of Norway project numbers 263248 and 275516.

REFERENCES

[1] S. Nakamoto *et al.*, "Bitcoin: a peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf, 2008.

[2] P. R. Rizun, "A transaction fee market exists without a block size limit," *Block Size Limit Debate Working Paper*, 2015.

[3] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*, pp. 1–10, Sept 2013.

[4] E. Tedeschi, H. D. Johansen, and D. Johansen, "Trading network performance for cash in the bitcoin blockchain," in *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018.*, pp. 643–650, 2018.

[5] M. Möser and R. Böhme, "Trends, tips, tolls: A longitudinal study of Bitcoin transaction fees," in *Financial Cryptography and Data Security: FC 2015.*, no. 8976 in LNCS, (Berlin, Heidelberg), pp. 19–33, Springer Berlin Heidelberg, 2015.

[6] H. Qureshi, "Blockchain fees are broken. here are 3 proposals to fix them.," Jun 2019.

[7] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, ACM, 2017.

[8] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[9] M. Samaniego and R. Deters, "Blockchain as a service for iot," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 433–436, IEEE, 2016.

[10] N. Houy, "The economics of Bitcoin transaction fees," Working Papers 1407, Groupe d'Analyse et de Théorie Economique (GATE), Université Lyon 2, 2014.

[11] Y. Zhu, R. Guo, G. Gan, and W. Tsai, "Interactive incontestable signature for transactions confirmation in bitcoin blockchain," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 443–448, June 2016.

[12] S. Basu, D. Easley, M. O'Hara, and E. Gün Sirer, "The old fee market is broken, long live the new fee market," Jan 2019.

[13] S. Haykin, *Neural networks*, vol. 2. Prentice hall New York, 1994.

[14] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar fault tolerance for cooperative services," in *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP '05, (New York, NY, USA), pp. 45–58, ACM, 2005.

[15] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized Blockchains," in *Financial Cryptography and Data Security. FC 2016.*, vol. 9604 of *LNCS*, (Berlin, Heidelberg), pp. 106–125, Springer Berlin Heidelberg, 2016.

[16] R. Baker *et al.*, "Data mining for education," *International encyclopedia of education*, vol. 7, no. 3, pp. 112–118, 2010.

[17] I. Lykourentzou, I. Giannoukos, V. Nikolopoulos, G. Mpardis, and V. Loumos, "Dropout prediction in e-learning courses through the combination of machine learning techniques," *Computers & Education*, vol. 53, no. 3, pp. 950 – 965, 2009.

[18] I. Bose and R. K. Mahapatra, "Business data mining—a machine learning perspective," *Information & management*, vol. 39, no. 3, pp. 211–225, 2001.

[19] J. Dean, *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. John Wiley & Sons, 2014.

[20] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, "Disease prediction by machine learning over big data from healthcare communities," *Ieee Access*, vol. 5, pp. 8869–8879, 2017.

[21] S. Dua, U. R. Acharya, and P. Dua, *Machine learning in healthcare informatics*, vol. 56. Springer, 2014.

[22] M. Riegler, M. Lux, C. Griwodz, C. Spampinato, T. de Lange, S. L. Eskeland, K. Pogorelov, W. Tavanapong, P. T. Schmidt, C. Gurrin, D. Johansen, H. Johansen, and P. Halvorsen, "Multimedia and medicine: Teammates for better disease detection and survival," in *Proc. of the the 2016 ACM on Multimedia Conference*, MM '16, pp. 968–977, ACM, 2016.

[23] S. Stolfo, D. W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Credit card fraud detection using meta-learning: Issues and initial results," in *AAAI-97 Workshop on Fraud Detection and Risk Management*, 1997.

[24] T. B. Trafalis and H. Ince, "Support vector machine for regression and applications to financial forecasting," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 6, pp. 348–353, IEEE, 2000.

[25] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in *Proceedings of the 17th international conference on World Wide Web*, pp. 247–256, ACM, 2008.

[26] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PloS one*, vol. 10, no. 3, p. e0119044, 2015.

[27] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[28] J. Schmidhuber, "Deep learning in neural networks: An overview," *CoRR*, vol. abs/1404.7828, 2014.

[29] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[32] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Advances in Neural Information Processing Systems*, pp. 6389–6399, 2018.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[34] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," pp. 1137–1143, Morgan Kaufmann, 1995.

[35] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*. Academic Press, 4th ed., 2008.

[36] S. Geisser, *Predictive inference: an introduction*. CRC Press, 2017.

[37] J. Newbery, "An introduction to bitcoin core fee estimation," *Bitcoin Tech Talk*, 2017.