Predicting digital asset market based on blockchain activity data

Author: Zvezdin Besarabov
National School of Mathematics and Natural Sciences
me@zvezd.in

Supervisor: Todor Kolev
Chairman of the Comrade Cooperative
todor@comradecoop.com

February 16, 2018

Abstract

Blockchain technology shows significant results and huge potential for serving as an interweaving fabric that goes through every industry and market, allowing decentralized and secure value exchange, thus connecting our civilization like never before.

In our paper we explore how modern Deep Learning techniques can be applied to predict future facts about the Ethereum blockchain. Specifically, we are interested if blockchain's public raw data, such as the transaction count and the account balance distribution, can be used to predict other measures like the number of new accounts created and the market price per ETH token.

During a series of experiments, we achieved 330% lower error scores with blockchain data than an LSTM approach with trade volume data. By utilizing blockchain account distribution histograms, stacked dataset modeling, and a Convolutional architecture we reduced the error further by 35%.

Moreover, we have developed a reusable framework providing data gathering, processing, and storing functionality for performing Deep Learning experiments over blockchain data. Our future plans are towards automating neural network architecture and meta-parameter optimization tasks through training controller Machine Learning models on these tasks. Since the Ethereum network already utilizes a great amount of GPU processing power, which might become obsolete due to the adoption of a Proof-of-Stake algorithm, we believe that 2018 presents a unique opportunity for achieving an exascale decentralized supercomputer, dedicated to training AI.

1 Introduction

Crypto-currencies, the decentralized new ways of exchanging value, are gaining a lot of interest recently with the emergence of Bitcoin, Ethereum, and hundreds of other crypto-assets with total value in excess of $500 billion. Unlike the traditional markets, where all the data about the market and the trading itself are centralized and controlled by gatekeeper organizations, the crypto-asset markets are public, decentralized and transparent by definition. This new kind of data, available on the blockchain, provides an unique opportunity for developing more robust trading algorithms.

Most stock market predictions are based solely on previous price points. However, such prediction strategies perform very poorly in the stochastic and volatile cryptocurrency markets. To combat that we can analyze the public blockchain data, in addition to financial data, in an attempt to predict the future values.

Projects about crypto-asset market predictions have emerged in the past - a paper by a Stanford University team [ISA14] on the topic of Bitcoin price estimations has been published in 2014. They are also working on a dataset with data from the Bitcoin blockchain, but without employing more sophisticated data representation techniques like account distribution histograms and rather focusing on primitive measures such as the last block transactions count. Most of the errors in their results are false positives, signaling low generalization of their neural model. Nevertheless, they did not disclose the details needed to reproduce their results. Two more recent projects on Github also explore Bitcoin predictions [Byn15] [Ré17], however, both of them focus primarily on historical price data and did not reveal their data processing algorithms and neural architectures.

1.1 Project Goals

Our goal is to explore how modern Deep Learning techniques can be applied in estimating future facts about crypto-assets. More specifically we are interested in whether we can utilize the abundant blockchain data to improve our estimates.

We aim to create a flexible and open crypto-currency prediction framework. The framework allows the collection of blockchain and financial data, and provides a way for a quick implementation of data processing and feature extraction algorithms. The extracted features (or properties) are compiled into a dataset using different modeling strategies. Custom neural architectures are built, trained, evaluated and compared on the output dataset. The prediction target can be any feature of the crypto-currency, not just price movements.

We have concluded that the Ethereum blockchain (ETH) [Eth17a] is the most difficult to predict, due to its large and versatile user base. Some of its use cases include financial transactions, secure voting, autonomous organizations, company management, freedom of speech networks, online games, crowdfunding, speculation, and more. This stochastic environment will be the best to show the true performance of our prediction strategies, which can later be migrated to less noisy blockchains, like Bitcoin.

We want to create a reference implementation of our framework for Ethereum. We need to study the network's history, users, and behavior, observe the factors for its volatility, and propose and test strategies to overcome that. We will focus primarily, but not exclusively, on value predictions. Changes in value usually are followed by respective changes in other blockchain features.

1.2 Blockchain and Ethereum

Satoshi Nakamoto's introduction of Bitcoin in November 2008 has often been hailed as a radical development in money and currency as it's the first example of a digital asset which simultaneously has no backing or "intrinsic value" and no centralized issuer or controller. However, another arguably more important part of the Bitcoin experiment is the underlying blockchain technology as a tool of distributed consensus. The most important aspect of such technology is the absence of an intermediary (centralized server, bank, company, etc.) between the originator and the recipient, as any changes to the data on this chain are made by consensus among all members of a decentralized network. Thus, avoiding the need to trust third parties.

Ethereum (ETH) is a newer cryptocurrency [Eth17a], based on the blockchain technology. The blockchain can be thought of as a growing distributed public database with records for each transaction in history.

1.3 Autonomous decentralized applications

The intent of Ethereum is to create an alternative protocol for building autonomous decentralized applications. Due to the blockchain consensus, no one has control over the code execution or storage of these application. A decentralized application can not be stopped, deleted or attacked in any way, unless its logic allows it. Because such an immutable application resembles contractual relations, autonomous decentralized applications are often referred to as "cryptocontracts" or simply contracts.

1.4 Factors for digital asset value change

In basic economics, the correlation of supply and demand [Hay17] determines the price of an asset. Higher demand or lower supply relates to higher price. All the crypto-asset's operations and activity are contained within the blockchain transaction data.

Due to small market capitalization, digital currencies are very unstable and volatile. Based on our observations of Ethereum, we have found the following factors to have a significant influence on the supply and demand of the currency.

1.4.1 Speculative investment

Speculative investment is buying and holding a certain asset, speculating that its value will rise in the future. This is an issue in the case of Ethereum's low overall supply, due to investors increasing the demand for the asset and decreasing its supply by holding their tokens.

In our research, we investigate methods that track the influence of this method to the end value.

1.4.2 Initial Coin Offerings

Intial Coin Offerings (ICOs) [Inv17] are a form of fundraising for starting projects. The team creates a new digital coin and people invest by buying it. The coin's value is proportional to the project's success. Recently, the total cumulative ICO funding has increased by more than 80 times [Coi17a]. Unfortunately, investing in an ICO does not bear any of the legal protections that regular investments provide, which makes them a risky investment.

Without a doubt, the excitements and disappointments around ICOs have left a major mark on digital currencies as a whole. Due to Ethereum's smart contract framework, it has become the most targeted platform for ICO fund-raising. This is believed to have been a major reason for Ethereum's value spike during June 2017.

We track ICO activities on the blockchain in attempts to observe their impact on value changes.

1.4.3 Altcoin interference

After Bitcoin, hundreds of other cryptocurrencies have emerged. Despite the fact that they are completely unrelated and independent from each other, major events in one cryptocurrency can cause unexpected fluctuations in others. Major ICO events in Ethereum have caused demand peaks in numerous other currencies, like Litecoin [Coi17b]. Our explanation is that most cryptocurrency users still do not fully understand what makes cryptocurrencies independent - the blockchain concept.

The interference of other alternative coins to Ethereum leads to a whole research topic on its own. We have plans to include historical data and events from other currencies to improve our predictions.

1.4.4 Media influence

Crypto market's value can also be easily influenced by media publications. A prime example of this is a case of widespread hoax about an incident with Ethereum's founder Vitalik Buterin [Rob17]. This led to temporary plummet and loss of 4$ billion from Ethereum's market capitalization.

We will experiment with cases where we track the news stories about Ethereum and similar cryptocurrencies via sentiment analysis.

1.5 Use of Deep Learning in asset value predictions

Deep Learning [YYG15] is the act of designing a multi-layered neural network architecture and training it with large amounts of data. This technique is especially useful when there is a need to find a correlation, dependence, or patterns in a large sequence of data. That makes it ideal for our needs, as we are trying to find if, given data based on the blockchain transactions, we can find a pattern with future facts.

2 Data sources

We gather two kinds of data: historical financial tick data and raw Ethereum blockchain data. The financial data is aggregated from multiple exchanges to achieve an exchange independent global view of the financial state. We also process all data in the Ethereum blockchain, which includes every value transaction, crypto-contract execution, and blockchain event throughout the existence of the cryptocurrency. The blockchain is hundreds of gigabytes in size, growing each day [Eth17b]. Due to the large size, we filter out most of the irrelevant to our research raw data.

Our blockchain data spans from the week after the creation of Ethereum - 8-08-2015 and covers every event in the existence of Ethereum. For each block in the chain, we hold the following data points: creation timestamp, number (chain index), miner (block creator), list of confirmed transactions, size in bytes, creation difficulty, and computational resource usage (Gas limit and Gas used).

For each transaction in a block, we save the following data points: address of the initiator and receiver, transferred value, used resource units (Gas), and amount paid per resource unit (Gas price).

Our financial data also covers the same period, starting from 8-08-2015. The size of each tick is one hour. This is the most frequent and highest quality data that we found as publicly available. For each tick we have information about the *open*, *close*, *low*, and *high* prices, as well as the trade volume to and from the currency.

The total size of our filtered raw blockchain data is around 60GB, containing $4,300,000$ blocks with a total of $60,331,058$ transactions. It took 14 consecutive days to download and another 3 to process, filter, format and save to a database.

3 Generation of data properties

In order to optimize the learning process and minimize overfit on our neural models, we extract the most important properties (or features) from our raw and basic data. We have created a set of properties that we have picked based on their importance and significance to the problem. For each financial tick in our historical data, we calculate the value of each property in our property set. Generated properties are absolute values. Property names with suffix "*_rel*" denotes the same property, converted to relative values. Please refer to Figure 1 as an example.

The basic (single value) properties that we generate are denoted in Table 1. The following subsections explain more advanced concepts to expand our set of properties.

Property	Description
openPrice	The value of ETH at the start of the tick
closePrice	The value of ETH at the end of the tick
stickPrice	The difference between the close and open prices
volumeTo	Exchange volume to that cryptocurrency
volumeFrom from	Exchange volume from that cryptocurrency
transactionCount	Amount of transactions
dappOperations	Amount of transactions to crypto contracts
blockSize	The average size of a block in bytes
difficulty	The average difficulty for block mining
uniqueAccounts	Number of accounts in existence
gasLimit	Average limit on computational resource usage
gasPrice	Average price per gas unit
gasUsed	Average amount of gas used
networkHashrate	Combined hashrate of every miner in the network
ETHSupply	Total amount of Ether in circulation
pendingTransactions	How many transactions per minute are pending for inclusion in the next block
blockchainGrowth	Growth of the blockchain size in gigabytes

Table 1: List and description of the properties compiled from the raw data. The properties relate to a time period of one financial tick.

3.1 Financial indicators

Our property values include significant amount of statistical noise which can hamper the training process. We smoothen all of our property values through a set of statistical functions, which are successfully applied in numerous of prediction strategies on the stock market. The functions include Simple Moving Average (SMA) and Exponential Moving Average (EMA). All of our functions use 10 tick intervals. Property names with suffixes "*_sma*" and "*_ema*" denote that the property values are modified by the respective function.

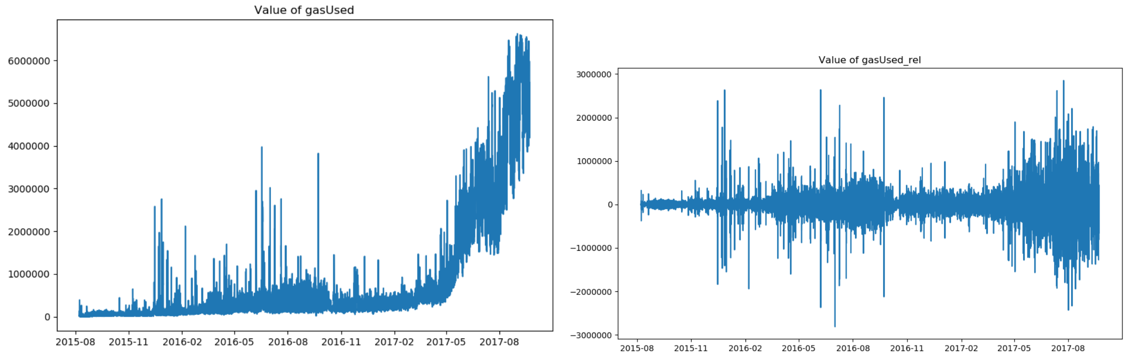

Figure 1: The absolute (left) and relative (right) values of property $gasUsed$ over time.

3.2 Account balance distribution

We take all active Ethereum accounts within a period of one price tick and assign a certain group number to each account based on its balance category:

$$groupN = \lfloor \log_b max \rfloor; \;\; group = \min(\lfloor \log_b bal \rfloor, groupN - 1)$$

Where bal is the balance of that account and max is a theoretical maximum balance chosen based on Ethereum's constant supply increase. We use a log scale because there are balances are higher than 100σ from the mean balance. The value of The b correlates to $groupN$ and therefore the resolution of the distribution. We recommend values $\in [1.2; 2]$

For each group, we calculate the total volume of ETH transferred to, transferred from, and total number of transactions initiated by any account from the group for the certain financial tick. We arrange these values in a $3 * groupN$ matrix, representing a snapshot of the summarized activity in the network for that period of time. These values are spatially-related and have a strict ordering. Figure 2 is a visual example of the distribution.

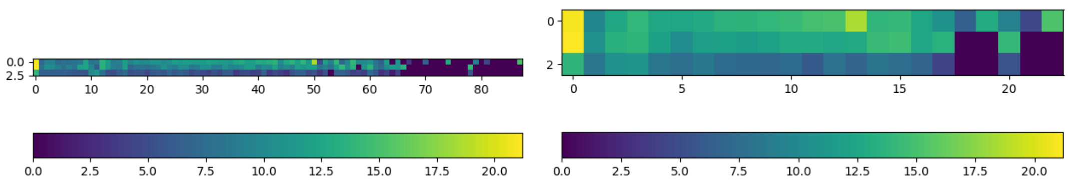

Figure 2: Examples with b 1.2 (left) and 2 (right). X axis represents the group number. Values at $Y = 0$ are volume to, $Y = 1$ are volume from and $Y = 2$ are amount of transactions. All values are additionally scaled with $\log_2$.

3.3 Account number distributions

In order to increase the spatial value in our data, we hereby increase the dimensions of our properties. We also process changes outside of just one financial tick, making a whole network state snapshot. For every financial tick, we take all Ethereum accounts in existence at that point and assign two group numbers to each one of them:

$$groupN_{1,2} = \lfloor scl_{1,2}(max_{1,2}) \rfloor; \;\; group_{1,2} = \min(\lfloor scl_{1,2}(feat_{1,2}) \rfloor, groupN_{1,2} - 1)$$

The features which we track for the accounts are $feat_{1,2}$. Examples of a feature include seconds since last activity, balance, total ETH volume received in the last 30 days, and average transaction value. The value of max is the theoretical maximum value for that feature. The best scl function for our features is a logarithm ($y = \log_{base} x$). Because we are assigning two group numbers for each account, we have two sets of independent groups (denoted by indices 1 and 2). Each set of groups have different values for $feat$, scl, and $groupN$. We create a matrix, where the value at $[i, j]$

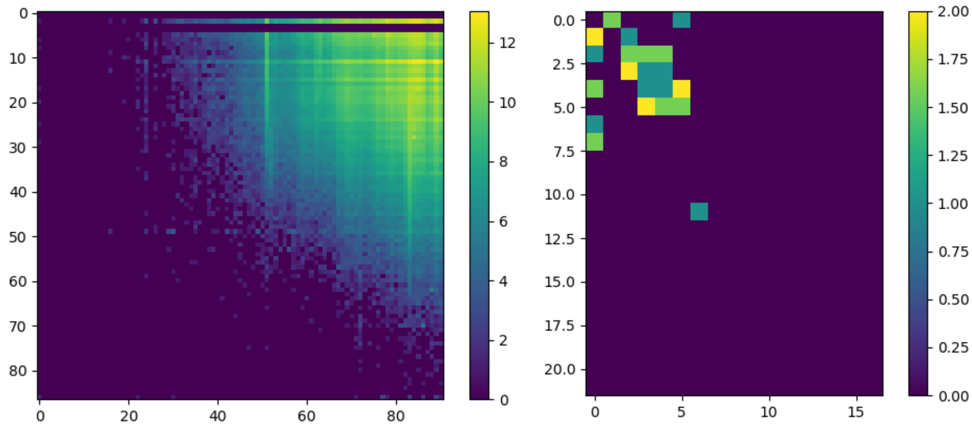

Figure 3: **balanceLastSeenDistribution (left)**: distribution of all existing accounts based on their balance (Y) and seconds since last activity (X), both with scl $\log_{1.2}$. A time-lapse video of the distribution values can be seen here: youtu.be/czfBa75HuWs. **contractVolumeInERC20Distribution (right)**: distribution of recently active ICO crypto contracts based on volume in (Y) and amount of operations with them (X), with $\log_2$ scl.

is the $\log_2$ of the amount of accounts assigned to feature groups i and j. Example configurations are in Figure 3.

Another configuration is the **contractBalanceLastSeenDistribution** - a distribution of smart contract accounts with the same $feat$ and scl as the balanceLastSeenDistribution.

Other definitions and examples are available on the project's Wiki page: goo.gl/zDDxxA

4 Generation of a dataset

4.1 Prediction target

Our prediction target is the value of one or multiple properties in the financial tick that follows the current moment. Other than price movements, our strategies can be applied to predicting amount of new accounts, trade volume, transaction count and even network cloggings (a serious issue in the light of recent events [Mad17]).

Our experiments will be mainly focused on predicting price movements and new accounts.

4.2 Normalization

Due to numerical instability caused by large floating point operations in computers, every value in our dataset needs to be normalized. Some algorithms fit within a small interval like $\in [0; 1]$, while others are unbounded and output sequences with zero mean and equal variance. We propose the following normalization strategies depending on the situation:

$$basic_i = \frac{x_i - min(x)}{(max(x) - min(x))}$$

We use this basic min-max scale to map independent sequences with the same sign to the interval $\in [0; 1]$.

$$around\_zero_i = \frac{x_i + max(|max(x)|, |min(x)|)}{2 * max(|max(x)|, |min(x)|)}$$

A bounded scale like *basic*, but which maps positive and negative inputs to $[0.5; 1]$ and $\in [0; 0.5$ respectively. We use this scale when normalizing sequences with values of varying sign.

$$image_i = (x_i - \frac{1}{n}\sum_{t=1}^{n} x_t) * \frac{1}{std(x)}$$

An unbounded strategy, producing zero mean and equal variance time series. We use it when normalizing spatial time series like the distributions in Section 3.3.

Every property in our dataset is a different time series and is normalized separately with a chosen algorithm. A method to automatically determine the algorithm for each property is called *prop*. If the values of the certain property are absolute, *basic* scale is used. Otherwise, *around_zero* scale is used.

Sequences with singular values over 10σ from the mean are best to be first scaled by a logarithm before normalization. Most of the aforementioned strategies scale based on the min and max values of the series, and future values that surpass those bounds will not fit the initial scale. The problem is mitigated if we use only relative values. In many cases of our experiments, relative values also lower overfit and increase generalization in our deep networks. We also propose a technique called normalization space overprovisioning, which increases the initial min and max values by a given percentage so that future values are more likely to fit. All of the aforementioned methods to deal with the problem can be used in conjunction with each other.

4.3 Dataset models

A dataset is a set samples, containing inputs (property data) and expected outputs (predictions). In order to train a neural network for value predictions, we have created models that arrange normalized property values into dataset samples. All of our models have the following in common. To generate the dataset samples, we create a sliding window with a configurable size of n and step increments of 1 over a configurable set of property values:

$$x \in [0, n-1]; \;\; y \in [0, propN - 1]$$

$$win_{x,y} = propY_{x+step}$$

$$tar = propT_{n+step}$$

Where $prop0, 1, 2, .., y$ is a the list of chosen properties with length $propN$. For each sample, $step$ is incremented by 1. The prediction target property is $propT$. The value of that property for that dataset sample is tar.

How the window of values are structured to a dataset sample depends on the chosen dataset model. We propose the following dataset models:

4.3.1 Matrix model

The matrix model representation matches the LSTM network's 2D input shape. A single dataset sample is created as follows:

$$sam_{x,y} = norm_y(win_{x,y})$$

Where $norm_y$ is the chosen normalization strategy for the certain property.

Table 2 visualizes the structure of a dataset sample.

Property	Value 0	Value 1	... Value $n-1$
Close value	349$	358$	
Number of TXs	2876	1583	
Operations with Dapps	459	508	

Table 2: Example structure of a matrix model dataset sample.

4.3.2 Stacked layers model

The stacked layers model (Figure 4) matches the 3D input shape for the Convolutional Neural Network and its perception of space. We take all property values for one time step in the window and arrange them in 2D space, forming a layer. The length of the window n denotes the number of layers, which stacked together form an image with n color channels. The shape of a dataset sample is ($height$, $width$, n). $width$ and $height$ are configurable, as is the property ordering. This model can most effectively compile advanced spatial properties, like the discussed distributions in Sections 3.2 and 3.3.

5 Generated datasets

We have generated multiple variants of a dataset, on the base of which we are running our experiments. We have chosen the most appropriate time interval for our datasets to be from 2017-03-01 to 2017-11-01. This range contains the largest fluctuations in Ethereum's history, the widespread media hoax about Ethereum [Rob17] and the rise of ICOs [Inv17].

Each complete dataset is split on *warmup*, *train* and *test* datasets respectively. The first month of the range is reserved for the *warmup* dataset. This part is only utilized in our stateful LSTM experiments and is used to warm the network up to a certain state before training or evaluation. The following 6 months build our *train* dataset, on the basis of which the neural network is trained. The last month is utilized by our *test* dataset, used for evaluation of the already trained network.

Datasets 1 to 4 do not include blockchain data and are a baseline for comparison. Window size (win), prediction target and the normalization ($Norm$) are specified later in the experiments.

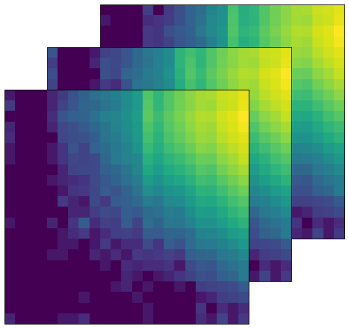

Figure 4: A visualization of the stacked layers model.

setN	Properties in dataset	Model
1	volumeFrom, volumeTo	matrix
2	volumeFrom_rel, volumeTo_rel	matrix
3	highPrice, volumeFrom, volumeTo	matrix
4	highPrice_rel, volumeFrom_rel, volumeTo_rel	matrix
5	accountBalanceDistribution	matrix
6	balanceLastSeenDistribution	stacked
7	contractBalanceLastSeenDistribution	stacked
8	balanceLastSeenDistribution, contractBalanceLastSeenDistribution, contractVolumeInERC20Distribution, accountBalanceDistribution	stacked

Table 3: Our dataset definitions. Properties with suffix "_rel" are relative values.

6 Measures of error

We have chosen to use the following measurement factors to evaluate the network's performance:

$$MSE = \frac{1}{n}\sum_{t=1}^{n} err_t^2; \quad RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} err_t^2}; \quad R^2 = 1 - \frac{\sum_{t=1}^{n} err_t^2}{\sum_{t=1}^{n} null_t^2} \quad sign = \frac{t}{t+f}$$

$$err_t = pred_t - true_t; \quad null_t = 0.5 - true_t$$

Where n denotes the number of samples in our dataset, $pred$ and $true$ are lists of our predicted and expected values respectively. The amount of correctly and incorrectly predicted signs are t and f respectively.

The metrics R^2 and $sign$ measure accuracy and our aim is to maximize their value, which is ≤ 1. In R^2 the performance of our model is compared to that of the $null$ model - one returning the same value (0.5 in this case) regardless of the input. MSE and $RMSE$ are measures of error which we need to minimize. We use MSE as the training loss function for all of our networks. The other measures are used for evaluation of the trained network.

7 Neural Network architectures

We have created a Long Short-Term Memory (LSTM) model [Hoc97] and a Convolutional model (CNN). Their network architectures are in Figure 5. LSTM networks are well known for their performance in time series data and it would be interesting to see how they interpret our time series. LSTMs can be stateful or stateless. Stateless networks get their memory (or state) reset after each batch of data, while stateful ones keep it. The LSTM model, trained with financial data, is used as a baseline for comparison of our blockchain data approach. Because we experiment with distributions (Sections 3.2 and 3.3) that have strict ordering space, we also investigate Convolutional networks, known to perform well in finding dependencies or patterns in spatial data.

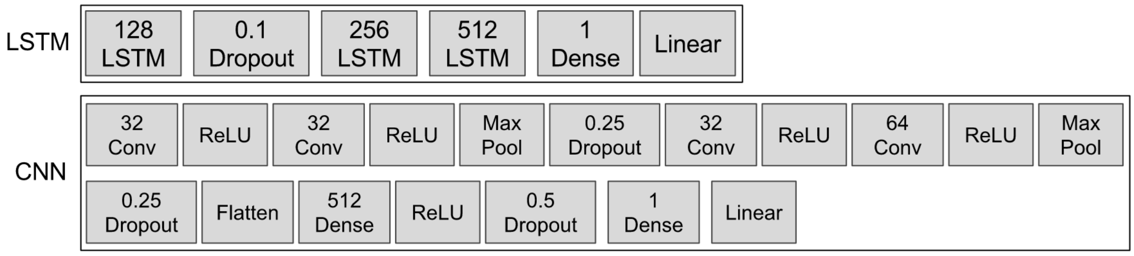

Figure 5: The architecture of our LSTM and Convoluitonal models respectively

8 Hyperparameter optimization

Deep models usually have hundreds of hyper- and meta-parameters that have to be tuned in order to achieve low error rates. Other than our supervised experiments and manual fine-tuning of hyper-parameters, our prediction framework allows for automatic parameter tuning by repetitive training, evaluation and respective parameter adjustments. This process has much in common with genetic algorithms, but because parameter changes usually lead to random effects, we do not have parameter inheritance from parent to child.

9 Experiments

We have conducted experiments on the base of our defined datasets (Section 5). Additional experiments can be found at the project's Wiki: https://goo.gl/WxnsG1. The experiments are grouped by prediction target. Our performance measurements are taken after inverse normalizing the network output and are therefore normalization independent. The following experiments are the best performants out of our thousand initial experiments. All measurements besides *sign* rely on the distance between the curves for prediction and actual values and are therefore directly incomparable between experiments with different targets.

We have found the optimal training parameters to be batch size of 16, $1e^{-5}$ learning rate and using the Adam optimizer [DPK14]. All of our networks are trained with these parameters.

All of our trained model weights, including detailed training histories and performance visualizations, are available on the project's Wiki: https://goo.gl/RzqEB9.

9.1 Predicting value ($highPrice$)

N	$setN$	win	$Norm$	$Network$	$RMSE$	R^2	$sign$
1	2	24	image	LSTM	90.418527	0.913118	0.525815
2	3	8	prop	LSTM	5.106049	0.999723	0.509511
3	5	24	image	LSTM	21.930655	0.994889	0.505435
4	6	104	image	CNN	16.216766	0.997204	0.511502
5	6	8	image	CNN	21.284594	0.995178	0.503329
6	7	104	prop	CNN	83.103040	0.926567	0.529093
7	7	24	prop	CNN	82.219956	0.928068	0.536716

Table 4: The results of experiments on value predictions. The measurements are the best ones from all training epochs.

Experiment 2 predicts on the basis of trade volume and price data, while experiment 1 predicts only on trade volume. Error scores for exp 2 are drastically lower. The network does not actually predict, but returns the input price data as predictions (upper Figure 6). That is an easy way to lower its error. In exp 1 the network has no access to price data and is forced to find an actual solution. All of our CNN experiments do not include price data as input.

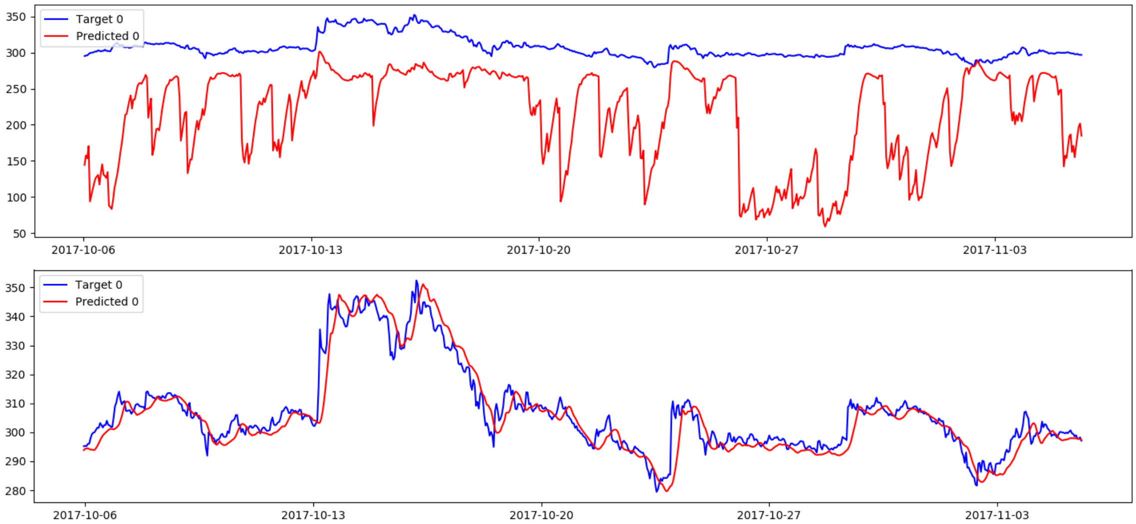

Figure 6: The test results of exp 2 with previous price data (top) and exp 1 without (bottom).

Exp 3 reaches low error despite not using any financial data. The network is mostly able to predict correct value movements (Figure 7). The added blockchain data reduces the error by 330% compared to exp 1.

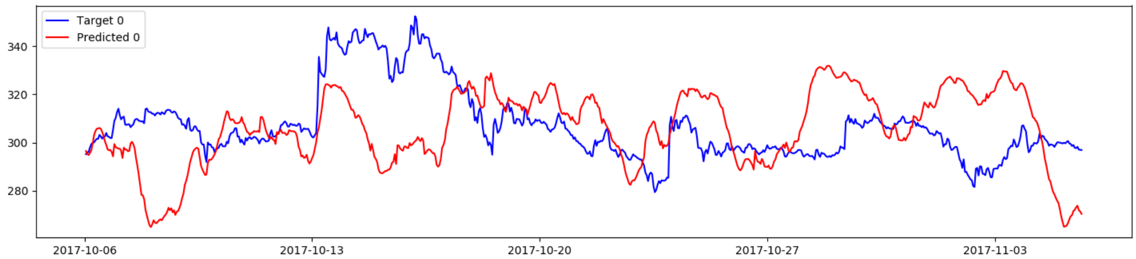

Figure 7: The test results of exp 3.

Exp 4 includes even more blockchain spatial value to our data and that combined with CNN results in our lowest error scores for this target (an additional improvement of 36% over exp 3). Despite the high error, exp 7 reaches our highest sign accuracy. These experiments prove that blockchain data improves predictions and place CNN as the best network to find patterns in them. The visualized activations of exp 4 are found at project's Wiki: goo.gl/etAaod

The errors on 6 and 7 demonstrate how inefficient is to normalize spatial distributions with *prop* scaling. We highly recommend image normalization with this data.

9.2 Predicting relative value ($highPrice\_rel$)

Relative value predictions force the network to find the correct solution, as it cannot cheat by returning input price data (as seen in exp 2 in section 9.1). Due to that, the error difference between exp 1 (volume data) and 2 (price & volume data) is not large.

Experiment 4 results in the lowest error and highest sign accuracy. Its *sign* is close to that of exp 1, but the CNN delivers higher R^2 accuracy. When observing the prediction graph (Figure 8), we can see that despite the scores, the network is not very confident in the value of the price change.

N	$setN$	win	$Norm$	$Network$	$RMSE$	R^2	$sign$
1	1	104	prop	CNN	2.365230	-0.003124	0.550272
2	4	104	image	CNN	2.301938	0.049843	0.493207
3	7	24	image	CNN	2.282585	0.051339	0.534045
4	8	8	image	CNN	2.287867	0.046130	0.551265
5	8	104	prop	CNN	9.195665	-14.206040	0.546685

Table 5: Results.

This suggests that there is need to further optimize the hyperparameters and network architecture to lower the error. The dataset of exp 4 increases the spatial value by having 4 distributions and this is seen in the increased *sign* value.

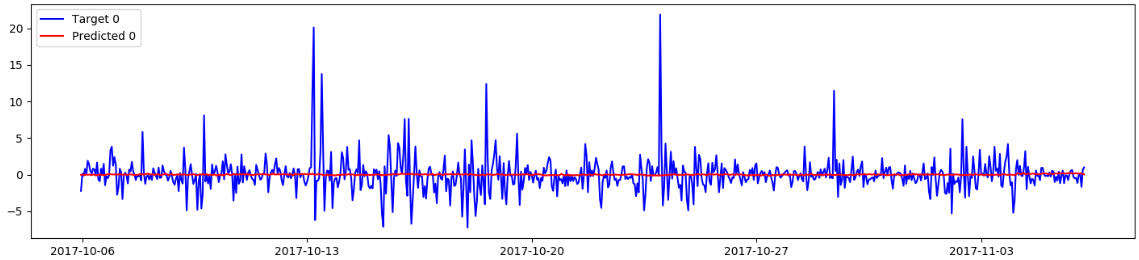

Figure 8: The test results of exp 4.

9.3 Predicting user growth (uniqueAccounts_rel)

N	$setN$	win	$Norm$	$Network$	$RMSE$	R^2
1	2	104	prop	LSTM	1139.972426	0.737456
2	1	104	prop	LSTM	718.971374	0.895567
3	7	8	image	CNN	812.303885	0.864637
4	6	24	image	CNN	639.336022	0.916289
5	8	8	image	CNN	622.716696	0.920449

Table 6: Results.

This set of experiments aims to demonstrate how blockchain data can be used to predict other facts about the blockchain, like how the public demand about it will change.

Sign accuracy is irrelevant in this case, as the number of *uniqueAccounts* always increases.

Our experiments with financial data (1 and 2) result in the highest error and lowest accuracy. When predicting facts about the blockchain, much more meaning can be extracted from the blockchain itself, as seen with the CNN results. Experiment 5 (Figure 9) gives impressive results in all measurements, because of the high spatial value with 4 distributions in its dataset.

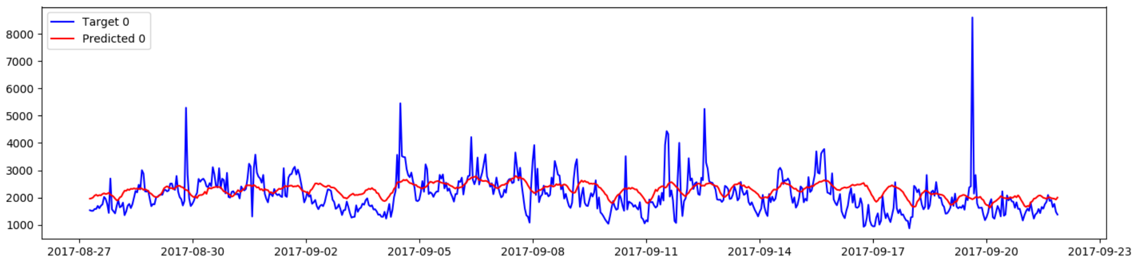

Figure 9: The test results of exp 5.

10 Results

Additional results can be found at the project's Wiki: https://goo.gl/WxnsG1.

Through our experiments, we have observed the problem from multiple different angles, by training multiple networks on multiple datasets and tracking multiple error and accuracy measurements.

Our results have shown that Ethereum is too volatile to be predicted based solely on financial data.

We have achieved 330% reduction in error scores when using only blockchain data compared to only trade volume data. With our Convolutional architecture, spatial account distributions, image normalization and spatial dataset modeling, we reduced the error by an additional 35% and reduced training times by a factor of 10.

Our best overall results are achieved using blockchain data, which confirms our initial hypothesis that this kind of data can assist predictions. Basic blockchain properties (like the defined in Table 1) are not sufficient for high prediction accuracy. These findings explain the low hourly prediction scores on the Stanford Bitcoin trading research [ISA14]. The best kind of blockchain data are our spatial distributions (Section 3.3). They are able to extract the most meaning out of the blockchain raw data. Our overall best dataset combines all 4 of our spatial distributions (Number 8 in Table 3), modeled by our Stacked layers model (Section 4.3.2). The best prediction target is relative high price, which forces the models to find the best possible strategy. The best neural network to find correct patterns in this dataset is our Convolutional model, trained with the Adam optimizer [DPK14]. We recommend batch size of 16, which provides balance between training times and network generalization. There are methods to increase accuracy with large batches [PG17], but a smaller batch is always better in terms of generalization.

Our best blockchain approach includes many more parameters than the standard financial data approach and therefore the address space for possible configurations is exponentially larger. We need to experiment with more hyperparameter tuning before being conclusive about the results.

11 Technical implementation

For our experiment, we have created and documented a flexible framework that allows predicting facts about any blockchain on the base of any data. Our properties, dataset models, normalization strategies and the aforementioned techniques are all in our implementation of the framework for Ethereum. The framework allows the implementation of any data processing algorithms and prediction strategies.

The framework is split into multiple tools, managing data download and storage, property generation, dataset generation, and neural network training and evaluation:

- **Dataset extraction module**: responsible for connection to data providers (like the discussed in Section 2), download, validation of integrity, and storage of the raw data.
- **Data storate module**: reads and post-processes the raw data by filtering out unimportant for our research data and saving the rest as a time series in a database. For fast retrieval by date interval, the data is split by a chunking algorithm.
- **Property generation module**: reads the produced raw time series data and generates and stores time series of property values (discussed in detail in Section 3).
- **Dataset generation module**: reads the generated properties, normalizes them, and applies a chosen dataset model to generate a dataset (the process is described in Section 4).
- **Training and evaluation module**: loads a created dataset and trains and evaluates a chosen neural architecture. Our proposed architectures are in Section 7 and our error measures are described in Section 6.

11.1 Automated tests

We have developed unit and integration tests for all of our tools, where applicable. This is done to ensure that the dataset generated by those tools is always with the expected and true values.

12 Technologies

1. **JavaScript** is used for the data extraction module;
2. **Web3 API** allows the extraction of blockchain data from a local running Ethereum node;
3. **Node.js** creates connections to other APIs for the rest of our raw data;
4. **Python** is used for all other modules;
5. **Pandas** provides high-performance data structures and data analysis tools for Python;
6. **MongoDB** is the database we use to store our data throughout all stages of data processing;
7. **Arctic** is a high performance datastore for numeric and time-series data used in hedge funds;
8. **Tensorflow** is the deep learning library most commonly used in deep learning research and development;
9. **Keras** is a frontend framework for a backend (like Tensorflow) that allows fast prototyping of neural network architectures.

13 Github repository

Our framework and Ethereum implementation are open-source unter MIT license. The GitHub repository of the project can be found at: https://github.com/Zvezdin/blockchain-predictor

The respository includes technical documentation regarding the project, including setup, usage instructions and steps to reproduce our findings.

The Wiki about the project, including additional information for the research, is available here: https://github.com/Zvezdin/blockchain-predictor/wiki

14 Future work

During our research, we have implemented not only a number of Deep Learning models but we have also built a reusable framework for data gathering, processing and storage of block-chain and price data. Once we had this framework, most of our actual research work was reduced to trying different network architectures and meta-parameters. While this process alone was challenging and demanding of both deep knowledge and creative thinking, we are excited by the possibilities to experiment with automating it.

Two recent publications by Google Research [ER17] [ZL17] show that given enough computing power, we can create a "controller" algorithm which produces model designs for predicting the CIFAR10 dataset [AKH09], with performance on par with state-of-art models designed by humans. The authors emphasize the enormous computational power required for achieving this results which was in order of 10^{20} computation. For comparison, most estimations about the computational power of the human brain are for between petascale (10^{15}) and exascale (10^{18}) operations per second [NRBMJ12].

Our ambition is to design and implement a scalable decentralized architecture which allows remote nodes to provide computing power to "controller" algorithms for evaluating individual Deep Learning models. In such a system, Ethereum smart-contracts can be utilized to implement a decentralized marketplace for neural network model training and optimization.

We are following closely the developments around the Ethereum network's adoption of Proof-of-Stake algorithm which might render the crypto-currency mining with GPUs obsolete. Currently such "mining" computers are consuming enormous amounts of electricity power [Her17], wasted to produce "proofs" for the network. Soon, this job can be taken by the Proof-of-Stake algorithm which does not require any significant computing power and a huge number of GPU cards might be switched to do more reasonable work, like training Deep Learning models.

We are excited to work towards our new research goal during 2018, since this year might present a unique opportunity to build the first decentralized exascale supercomputer dedicated for training AI.

15 Conclusion

Blockchain technology enables a new class of tradeable crypto-token assets, thus creating new kinds of markets and numerous new use-cases. Due to their decentralized and open nature, public blockchains provide abundance of market-related data that has never been available before. Having every account balance and every transaction visible, encourages us to seek a holistic view on the blockchain dynamics.

In our research project, we have evaluated different Deep Learning methods at their ability to find patterns in the Ethereum blockchain data and provide estimations about future measurements of the blockchain assets, including their market price. To facilitate our experiments, we have implemented a data gathering, processing, and storing framework that enabled us to deal with the enormous amount of data on the blockchain (more than 300GB) and efficiently produce datasets for training.

We have tested both recurrent and convolutional models with different data sets and prediction targets. The most interesting results came from our convolutional models which were working on account distribution histograms with one, two and three dimensions. These histograms serve as "pictures" of the blockchain in certain period of time and when we observe a time-lapse video of such 2D histograms, we can clearly see patterns that correlate to major market moves. After experimenting with these novel blockchain data representations, we found that convolutional Deep Learning models are capable of providing estimations for the ETH token price based solely on such histogram data. Still, we believe that better results can be achieved by further optimizations of the model architectures and meta-parameters.

Since the search space for such optimizations is enormous, we are excited by the possibilities to automate these tasks. Thus, our future work will be towards designing and implementing a smart-contract based, decentralized network of computing nodes that provide processing power to "controller" algorithms which generate and evaluate Deep Learning model designs.

References

[AKH09] Vinod Nair Alex Krizhevsky and Geoffrey Hinton. The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html, 2009.

[Byn15] Christopher Bynum. Bitpredict. https://github.com/cbyn/bitpredict, 2015.

[Coi17a] Coindesk. Ico tracker. https://www.coindesk.com/ico-tracker/, 2017.

[Coi17b] CoinGecko. Litecoin price chart (ltc/usd. https://www.coingecko.com/en/price_charts/litecoin/usd#panel, 2017.

[DPK14] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980, 2014.

[ER17] Andrew Selle Saurabh Saxena Yutaka Leon Suematsu Jie Tan Quoc Le Alex Kurakin Esteban Real, Sherry Moore. Large-scale evolution of image classifiers. https://arxiv.org/abs/1703.01041, 2017.

[Eth17a] Ethereum. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper, March 2017.

[Eth17b] Etherscan. Ethereum blockchain size. https://etherscan.io/chart/chaindatasizefull, 2017.

[Hay17] Adam Hayes. Suppply and demand. http://www.investopedia.com/university/economics/economics3.asp, 2017.

[Her17] Alex Hern. Bitcoin mining consumes more electricity a year than ireland. https://www.theguardian.com/technology/2017/nov/27/bitcoin-mining-consumes-electricity-ireland, 2017.

[Hoc97] Sepp Hochreiter. Long short-term meory. http://www.bioinf.jku.at/publications/older/2604.pdf, 1997.

[Inv17] Investiopedia. Initial coin offering. https://www.investopedia.com/terms/t/trillion-dollar-coin.asp, 2017.

[ISA14] Madan Isaac, Saluja Shaurya, and Zhao Aojia. Automated bitcoin trading via machine learning algorithms. https://www.semanticscholar.org/paper/Automated-Bitcoin-Trading-via-Machine-Learning-Alg-Madan-Saluja/e0653631b4a476abf5276a264f6bbff40b132061, 2014.

[Mad17] Paul Madore. Status ico clogs up ethereum blockchain. https://www.cryptocoinsnews.com/status-ico-clogs-up-ethereum-blockchain/, 2017.

[NRBMJ12] WOLFRAM ERLHAGEN NUNO R. B. MARTINS and ROBERT A. FREITAS JR. Non-destructive whole-brain monitoring using nanorobots: Neural electrical data rate requirements. http://www.worldscientific.com/doi/abs/10.1142/S1793843012400069, 2012.

[PG17] Ross Girshick Pieter Noordhuis Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He Priya Goyal, Piotr Dollar. Accurate, large minibatch sgd: Training imagenet in 1 hour. https://arxiv.org/pdf/1706.02677.pdf, 2017.

[Rob17] Jeff John Roberts. Hoax over 'dead' ethereum founder. http://fortune.com/2017/06/26/vitalik-death/, 2017.

[Ré17] Philippe Rémy. Deep learning bitcoin. https://github.com/philipperemy/deep-learning-bitcoin, 2017.

[YYG15] LeCun Yann, Bengio Yoshua, and Hinton Geoffrey. Deep learning. http://pages.cs.wisc.edu/~dyer/cs540/handouts/deep-learning-nature2015.pdf, 2015.

[ZL17] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. https://arxiv.org/abs/1611.01578, 2017.