Bitcoin Price Prediction with Neural Networks

Kejsi Struga
kejsi.struga@fshnstudent.info

Olti Qirici
olti.qirici@fshn.edu.al

University of Tirana

Abstract

In this work, we use the LSTM version of Recurrent Neural Networks, to predict the price of Bitcoin. In order to develop a better understanding on its price influencers and the general vision of this brilliant innovation, we first give a brief perspective on Bitcoin and its economics. Then we describe the dataset, which is comprised of data from stock market indices, sentiment, blockchain and Coinmarketcap[1]. Further on this investigation, we show the usage of LSTM architecture with the aforementioned time series. To conclude, we outline the results of predicting Bitcoin price for 30 and 60 days ahead.

1 Introduction

With the advent of Bitcoin 10 years ago the world of economics, albeit in small scale, has and is experiencing a revolution. Bitcoin introduced itself as the system that solved the Double Spend problem [Nakamoto2008], a prevalent issue with inherent Digital Cash systems. Nevertheless, the impact during coming years was greater. Distributed Ledger Technologies (DLT), Smart Contracts, Cryptocurrencies, etc. all stemmed from this very "Bitcoin idea". This is attributed, to the unique decentralization mixed with intuitive incentive. On the other end of the spectrum, with data being regarded as the oil of nowadays, along with the tremendous increases in hardware efficiency, Machine Learning is increasingly being utilized. As a result, we are inclined in attempting to predict the Bitcoin price, despite of the dynamic nature present not only in Bitcoin exchanges (Fig. 1), but in financial markets in general.

[1]Coinmarketcap takes the average of every exchange that trades Bitcoin

2 Bitcoin Price

Many endorse Bitcoin, while other are sceptic. Regardless, the price of Bitcoin is a topic discussed from an economic angle, computer science, financial, and psychological perspective. In the time of writing this article, the price seems to be getting stable. In some way, this may be normal given that many investors are waiting to see regulations, the scaling problems is not seeing any improvement, the fact that the first hype around 2017 is now over and it is normal for a market to get stable for some time. Nevertheless, given the features of Bitcoin like:

- Tax free
- Unforgeable
- Bordless and unbound from distance
- Decentralized
- Verifiable and secure
- Normally, negligible transaction fees
- Cannot be counterfeited

It is a plausible solution for countries with developing economies and financial systems to improve their economical position, while struggling to access the best technology of its time. However, for a country to have an aptitude towards this monetary system, a tech-savvy population should be present to adopt the system, along with regulations from financial institutions that support it. Both of these are absent for the moment, but the future prospect is very potent. Another factor why we believe Bitcoin should be studied, is the fast paced advancements in technology (Fig. 1), that favour Bitcoin, considering its qualities as a software and a decentralized system, which can not be beaten by banking systems, or a likes.

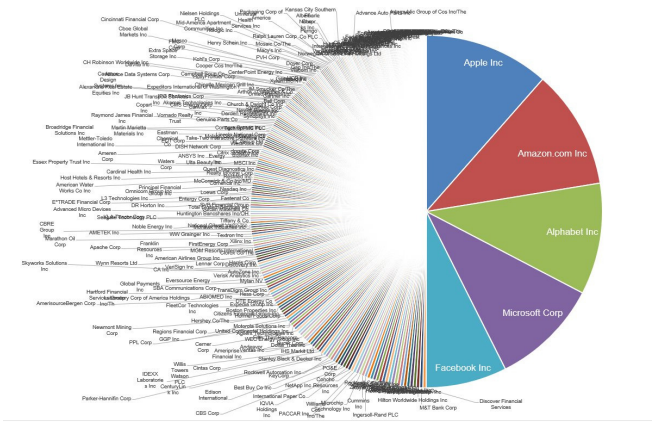

Figure 1: 5 tech companies together are worth more than 282 other companies, by market capitalization. *Source:* [Batnick2018]

2.1 Bitcoin Deflation

Bitcoin's supply is predetermined *by design*, and represented by this geometric series:

$$S_n = \frac{a\,(1-r^n)}{1-r} = 210000 \times \frac{50\,(1-0.5)}{1-0.5} \approx 21 \times 10^6. \tag{1}$$

As it is clearly stated in Bitcoin's wiki [BitcoinWiki2017], the decrease of supply, resembles the rate at which commodities like gold are mined. This makes many consider Bitcoin as deflationary, but this currency is infinitely divisible, not only because 1 BTC = 10^8 *satoshis*[2], and in turn, no one would run out that fast of every satoshi, but also because the protocol could be updated allowing for satoshis to be more divisible (have more zeros). As a result, deflation, does not have to occur.

2.2 Bitcoin Inflation

Bitcoin is not debt based, and no artificial money can be issued. Additionally, because of the fixed supply mentioned above no more Bitcoins than predicted can be created, unlike the economy system of today. Bitcoin's deflationary attribute, stems from imitating gold, in that a currency must be scarce (or with a finite supply in case of BTC), consequently, no one can increase the supply and inflate the value of goods.

Nonetheless, the bitcoin has also its dark side which sometimes makes users quite sceptical on its posession and usage. Several of these problems have been reported (and may not be limited to) by Kaspersky Lab [Kaspersky2017]. We may include the facts that: blockchain nodes do exactly the same thing (no paralleling, no synergy, no mutual assitance), growing size of storage used (currently 100 GB for the Bitcoin for each high-grade Bitcoin network client), transaction confirmation needs 10 50 minutes, etc.

While we will try to build a predictive model for the Bitcoin prospect value calculation, we are aware in advance that price may differ greatly because of internal and external factors to Bitcoin. By internal factors we are presuming factors inside the Bitcoin security (some breach). By external we are referring to agents which influence indirectly the price of Bitcoin (exchange closures, replacing cryptocurrencies, speculation markets, the fact that as its believed widely over 80% of Bitcoins in circulation is concentrated in a limited number of investors etc.)

Anyway, we shall compare our results to other models built for cryptocurrency prediction. Lets not forget that in the first month of 2018 there were models which predicted that Bitcoin would surpass the 100,000.00 USD per Bitcoin till the end of the year, while we are barely reaching the 7,000.00 USD value just 2 months before the end of the year.

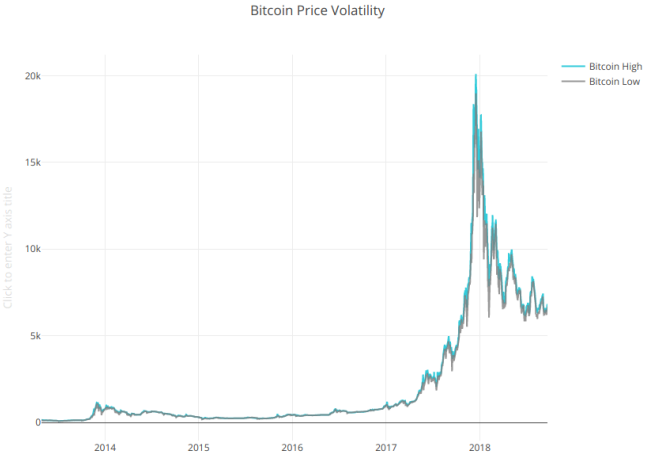

Figure 2: Bitcoin's steep price movements.

3 Data preprocessing

3.1 Data gathering

Daily data of four channels are considered since 2013. First, the Bitcoin price history, which is extracted from Coinmarketcap through its open API. Secondly, data from Blockchain is gathered, in particular we choose the average block size, the number of user addresses, number of transactions, and the miners revenue. We found it counter intuitive to have some Blockchain data, given the incessant scaling problem, on the other hand, the number of accounts, by definition is related to the price movements, since an increase in the number of accounts, either means more transactions oc-

[2]Satoshi is the smallest unit of Bitcoin currency

curring (presumably for exchanging with different parties and not just transferring Bitcoins to another address), or it is a sign of more users joining the network. Thirdly, for the sentiment data we obtain the *Interest over time* for the word 'Bitcoin' using PyTrends library. Lastly, two indices are considered, that of S&P 500 and Dow and Jones. Both are retrieved through Yahoo Finance API. All in all, these make for 12 features. The Pearson correlation between the attributes is shown in Figure 2. Clearly, some attributes are not too correlated, for example, the financial indices are relevant with each other, but not with any of bitcoin-related attributes. Also, we see how Google Trends are related to Bitcoin transactions.

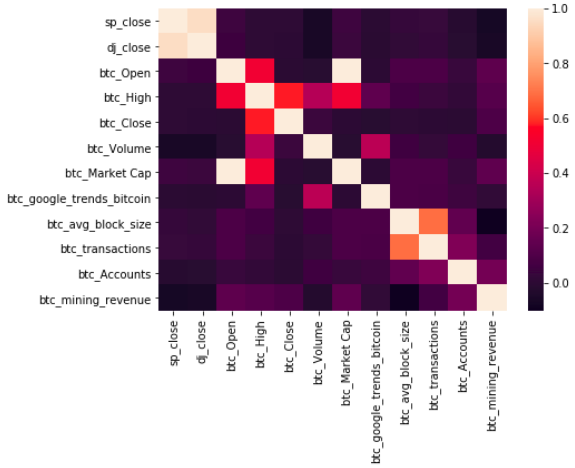

Figure 3: Pearson correlation, 1.0 means the highest correlation

3.2 Data cleansing

From exchange data we consider relevant only the Volume, Close, Open, High prices and Market capitalization. For all data sets if *NaN* values are found to be existent, they are replaced with the mean of the respective attribute. After this, all datasets are merged into one, along the time dimension. Judging from Bitcoin price movements during the period from 2013 until 2014, we considered best to get rid of data points before 2014, hence the data which will be passed to the neural network lies from 2014 until September 2018.

3.3 Data normalization

Deciding on the method for normalizing a time series, especially financial ones is never easy. What's more, as a rule of thumb a neural network should load data that take relatively large values, or data that is heterogeneous (referring to time-series that have different scales, like exchange price, with Google Trends). Doing so can trigger large gradient updates that will prevent the network from converging. To make learning easier for the network, data should have the following characteristics [Geron2017]:

- Take small values - Typically, most values should be in the range 0-1 range
- Be homogeneous - That is, all features should take values at roughly the same range.

The most common normalization methods used during data transformation include:

- ***Min-Max Scaling***, where the data inputs are mapped on a number from 0 to 1:

$$x' = \frac{x - min(X)}{max(X) - min(X)} \tag{2}$$

- ***Mean Normalization***, which makes data to have a values between -1 and 1 with a mean of 0:

$$x' = \frac{x - mean(X)}{max(X) - min(X)} \tag{3}$$

- ***Z-Score (Standardization)***, where the features are redistributed with their mean of 0 and standard deviation of 1:

$$x' = \frac{x - mean(x)}{\rho} \tag{4}$$

For our problem, we use Min-Max Scaling and adjust features on a scale from 0 to 1 given that most of our time-series have a peek, therefore we might argue we know the maximum of the series, in which case Min-Max Scaling does a good job.

4 Machine Learning Pipeline

In this section we describe how to make time series data adaptable for supervised machine learning problems. The price prediction is treated as regression rather than classification, and we show how LSTM can be used in such cases. We then, discuss hyperparameters.

4.1 Software used

For Deep Learning backend system we choose Tensorflow, and Keras as the front-end layer of building neural networks fast. Pandas is used extensively for data related tasks, Numpy is utilized for matrix/vector operations and for storing training and test data sets, Scikit-learn (also known as: sklearn) is used for performing the min-max normalization. Lastly, Plotly is used for displaying the charts.

4.2 Time series data

Normally a time series is a sequence of numbers along time. LSTM for sequence prediction acts as a supervised algorithm unlike its autoencoder version. As such, the overall dataset should be split into inputs and outputs. Moreover, LSTM is great in comparison with classic statistics linear models, since it can easier handle multiple input forecasting problems. In our approach, the LSTM will use previous data to predict 30 days ahead of closing price. First, we should decide on how many previous days one forecast will have access to. This number we refer as the window size. We have opted for 35 days in case of monthly prediction, and 65 days in that of 2 months prediction, therefore the input data set will be a tensor comprising of matrices with dimension 35x12/65x12 respectively, such that we have 12 features, and 35 rows in each window. So the first window will consist of 0 to the 34 row (python is zero indexed), the second from 1 to 35 and so on. Another reason for choosing this window length is that a small window leaves out patterns which may appear in a longer sequence. The output data takes into account not only the window size but also the prediction range which in our case is 30 days. The output dataset starts from row 35 up until the end, and is made of chunks of length 30. The prediction range also determines the output size for the LSTM network.

4.3 Split into training and test data

This step is one of the most important, especially in the case of Bitcoin. We first wanted to predict the year ahead, but this would mean, that data from 1 Jan 2018 until September 2018 would be used for testing, the downside of this, is of course the steep slope in 2017, which would make the neural network learn this pattern as by the last input, and the prediction of year 2018 would not be very logical. Thus we go for training data from 2014-01-01 until 2018-07-05, this leaves us with approximately 2 months for prediction, while we predict for two months, the data set is split a bit earlier to leave room for 2 months: 2018-06-01. Each training set and test set is composed of input and output features.

4.4 Turn data into tensors

LSTM expects that the input is given in the form of a 3 dimensional vector of float values. A key feature of tensors is their shape, which in Python is a tuple of integers representing the dimensions of it along the 3 axis. For instance, in our testing data of Bitcoin, the shape of training inputs is: $(1611, 35, 12)$, so we have 1611 samples, a window size (timestep) of 35 values, and 12 features. In overall the idea is simple, in that we separate the data into chunks of 35, and push these small windows of data into a numpy array. Each window is a $35x12$ matrix, so all windows will create the tensor. Furthermore, in LSTM the input layer is by design, specified from the *input_shape* argument on the first hidden, the these three dimensions of input shape:

- Samples
- Window size
- Number of features

4.5 LSTM implementation

4.5.1 LSTM internals

A chief feature of feedforward Networks, is that they don't retain any memory. So each input is processed independently, with no state being saved between inputs. Given that we are dealing with time series where information from previous Bitcoin price are needed, we should maintain some information to predict the future. An architecture providing this is the Recurrent neural network (RNN) which along with the output has a self-directing loop. So the window we provide as input gets processed in a sequence rather than in a single step. However, when the time step (size of window) is large (which is often the case) the gradient gets too small/large, which leads to the phenomenon known as vanishing/exploding gradient respectively [Chollet2017]. This problem occurs while the optimizer backpropagates, and will make the algorithm run, while the weights almost do not change at all. RNN variations mitigate the problem, namely LSTM and GRU (Fig. 2).

The LSTM layer adds some cells that carry information across many timesteps (Fig. 2). The cell state is the horizontal line from C_{t-1} to C_t, and its importance lies in holding the long-term or short term memory. The output of LSTM is modulated by the state of these cells. And this is important when it comes to predict based on historic context, rather than only the last input. LSTM networks manage to remember inputs by making use of a loop. These loops are absent in RNN. On the other hand, as more time passes, the less likely it becomes that the next output depends on a very old input, therefore forgetting is necessary. LSTM achieves this by learning when to remember and when to forget, through their forget-gates. We mention them shortly to not consider LSTM just as a black box model [Olah2015].

- Forget gate: $f_t = \sigma(W_f S_{t-1} + W_f S_t)$
- Input gate: $i_t = \sigma(W_i S_{t-1} + W_i S_t)$
- Output gate: $o_t = \sigma(W_o S_{t-1} + W_o S_t)$

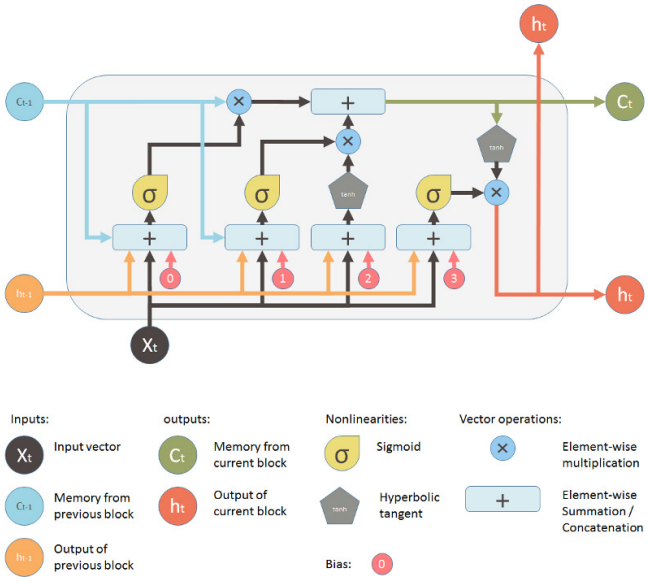

Figure 4: LSTM cell

- Intermediate Cell State: $\tilde{C} = tanh(W_c S_{t-1} + W_c X_t)$
- Cell state (memory for next input): $c_t = (i_t * \tilde{C}_t) + (f_t * c_{t-1})$
- Calculating new state: $h_t = o_t * tanh(c_t)$

As it can be seen from the equations, each gate has different sets of weights. In the last equation, the input gate and intermediate cell state are added with the old cell state and the forget gate. Output of this operation is then used to calculate the new state. So, this advanced cell with four interacting layers instead of just one *tanh* layer in RNN, make LSTM perfect for sequence prediction.

4.6 Hyperparameters

4.6.1 Optimizer

While Stochastic Gradient Descent is used in many Neural Network problems, it has the problem of converging to a local minimum. This of course presents a problem considering Bitcoin price. Some other nice optimizers are variations of adaptive learning algorithms, like Adam, Adagrad, and RMSProp. Adam was found to work slightly better than the rest, and that's why we go for it. (All of these come packed with Keras.)

4.6.2 Loss function

The performance measure for regression problems, will typically be either RMSE (Root Mean Square Error) or MAE (Mean Absolute Error).

- RMSE(X, h) $= \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(h(\mathrm{x}^i) - y^i\Big)^2}$
- MAE(X, h) $= \frac{1}{n}\Sigma_{i=1}^{n}\Big|\Big(h(\mathrm{x}^i) - y^i\Big)\Big|$

RMSE is generally used when distribution resembles a bell-shaped curve, but given the Bitcoin price spikes we chose to go MAE, since it deals better with outliers.

4.6.3 Activation function

The choice for activation function was not very difficult. The most popular are *sigmoid*, *tanh*, and *ReLu*. *Sigmoid* suffers from vanishing gradient, therefore almost no signal flows from the neuron to its weight, moreover it is not centered around zero, as a result the gradient might be to high or to low a number. By contrast, *tanh* makes the output zero centered, and in practice is almost always preferred to *sigmoid*. *ReLu* is also widely used, and since it was invented later, it should be better. Nevertheless, for predicting Bitcoin price that was not the case, and we chose *tanh* due to better results.

4.6.4 Dropout Rate

Regularization is the technique for constraining the weights of the network. While in simple neural networks, l_1 and l_2 regularization is used, in multi layer networks, drop out regularization takes place. It randomly sets some input units to 0 in order to prevent overfitting. Hence, its value represents the percentage of disabled neurons in the preceding layer and ranges from 0 to 1. We have tried 0.25 and 0.3 and lastly we decided for 0.3.

4.6.5 Number of Neurons in hidden layers

We opted for 10 neurons in the hidden layers, it actually costs a lot to have more neurons, as the training process will last longer. Also, trying a larger number did not give improved results.

4.6.6 Epochs

Rather arbitrarily, we decided for 100 epochs, after trying other values, like 50, or 20. As with the number of hidden layer neurons, the more epochs, the more time it takes for training to finish, since one epoch is a full iteration over the training data. Also, it may overfit the model.

4.6.7 Batch Size

We decided to feed the network, with batches of 120 data (again this number is a guess).

4.6.8 Architecture of Network

We used the Sequential API of Keras, rather than the functional one. The overall architecture is as follows:

- **1 LSTM Layer**: The LSTM layer is the inner one, and all the gates, mentioned at the very beginning are already implemented by Keras, with a default activation of *hard-sigmoid* [Keras2015]. The LSTM parameters are the number of neurons, and the input shape as discussed above.
- **1 Dropout Layer**: Typically this is used before the Dense layer. As for Keras, a dropout can be added after any hidden layer, in our case it is after the LSTM.
- **1 Dense Layer**: This is the regular fully connected layer.
- **1 Activation Layer**: Because we are solving a regression problem, the last layer should give the linear combination of the activations of the previous layer with the weight vectors. Therefore, this activation is a linear one. Alternatively, it could be passed as a parameter to the previous Dense layer.

5 Results and Analysis

In this section we show the results of our LSTM model. It was noted during training that the higher the batch size (200) (Fig. 7, 8) the worst the prediction on the test set. Of course this is no wonder, since the more training, the more prone to overfitting the model becomes . While it is difficult to predict the price of Bitcoin, we see that features are critical to the algorithm, future work includes trying out the Gated Recurrent Unit version of RNN, as well as tuning, on existing hyper-parameters. Below we show the loss from the Mean Absolute Error function, when using the model to predict the training and test data.

6 Conclusions

All in all, predicting a price-related variable is difficult given the multitude of forces impacting the market. Add to that, the fact that prices are by a large extent depended on future prospect rather than historic data. However, using deep neural networks, has provided us with a better understanding of Bitcoin, and LSTM architecture. The work in progress, includes implementing hyperparameter tuning, in order to get a more accurate network architecture. Also, other features can be considered (although from our experiments with Bitcoin, more features have not always led to better results). Microeconomic factors might be included in the model for a better predictive result. Anyway, maybe the data we gathered for Bitcoin, even though has been collected through years, might have become interesting, producing historic interpretation only in the last couple of years. Furthermore, a breakthtrough evolution in peer-to-peer transactions is ongoing and transforming the landscape of payment services. While it seems all doubtes have not been settled, time might be perfect to act. We think its difficult to give a mature thought on Bitcoin for the future

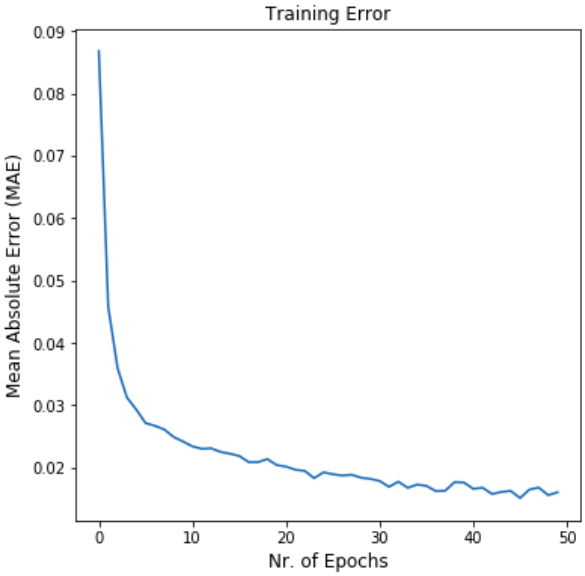

Figure 5: Error loss during training

Figure 6: Bitcoin Prediction on Training Set

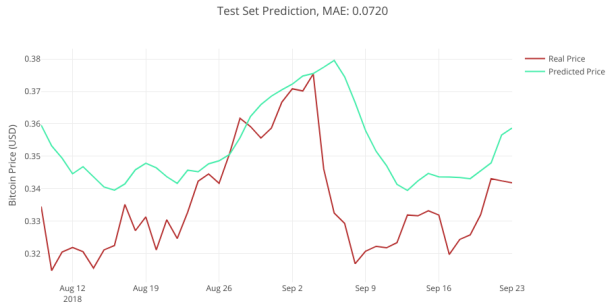

Figure 7: Bitcoin Prediction on Test Set, Batch Size 100

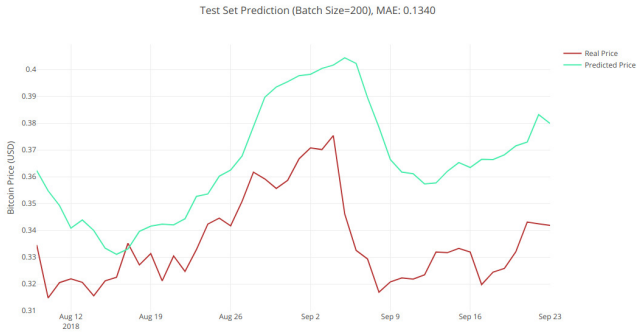

Figure 8: Bitcoin Prediction on Test Set, with a batch size of 200. Loss is greater than with batch 50

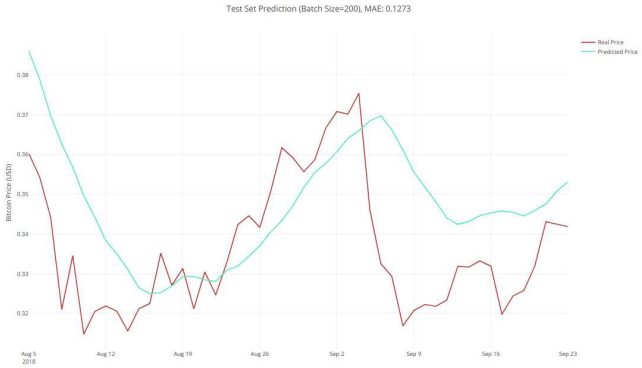

Figure 9: Bitcoin Prediction on Test Set, 60 days prediction

References

[Nakamoto2008] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System

[BitcoinWiki2017] Bitcoin Wiki. *Deep Learning with Python.* https://en.bitcoin.it/wiki/Controlled_supply

[Chollet2017] *Deep Learning with Python.* https://www.manning.com/books/deep-learning-with-python

[Batnick2018] M. Batnick. *The market cap of the top 5 S&P 500 companies.* https://theirrelevantinvestor.com/2018/07/19/pareto/

[Geron2017] Bitcoin Wiki. *Hands on Machine Learning with scikit-learn and tensorflow* https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/

[Olah2015] *Understanding LSTMs* https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[Kaspersky2017] , K1 Kaspersky Lab Daily, Six Myths about blockchain and Bitcoin: Debunking the effectiveness of the technology https://www.kaspersky.com/blog/bitcoin-blockchain-issues/18019/

[Keras2015] Keras https://keras.io/getting-started