Bitcoin Price Forecast using Deep Learning

Georg TOROPOV
Tsinghua University
2020280260

Aurélien VU NGOC
Tsinghua University
2020280219

Saran ZEB
Tsinghua University
2019380050

Abstract

Cryptocurrency markets have established themselves as the first new asset class in 400 years which has been acknowledged by several major global banks. While quantitative investing methods have been used on traditional financial assets for decades, the scope of this project is to apply LSTM and GRU neural networks to Bitcoin and attempt to build a price forecasting model. For this we have used three time series datasets with price data and a myriad of technical indicators with each time series focused on either 15 minute, 4 hour or daily timeframes. Our best results using the GRU was achieved using the 4 hours dataset with an RMSE (Root-mean squared error) of 1249 while the best LSTM architecture has yielded an RMSE of 491.38 on the 15 minute dataset. We conclude that both the LSTM and GRU provide powerful frameworks for timeseries forecasting as results are strong, while also having some limitations which we address in the final chapter.

1. Introduction

Quantitative trading models were first popularized by hedge funds like Renaissance Technologies and DE.Shaw and predictions for all kinds of financial assets have been an area of active research for a long time. The invention of Bitcoin as the first digital decentralized cryptocurrency has also drawn a lot of interest from the quantitative investment community and especially the use of neural networks for price prediction models has been popular in the last few years.

In this project we use a time series model leveraging deep learning techniques to predict the price of Bitcoin utilising GRU and LSTM architectures. The reason we chose Bitcoin is that cryptocurrencies have established themselves as a new and exciting asset class in the last few years, which has drawn a lot of interest from both traditional financial institutions and retail money. We chose to focus on Bitcoin as it is the most important cryptocurrency and has the highest amount of liquidity or trading volume. The goal of this project is to firstly find a suitable GRU or LSTM architecture and secondly find a feature or combination of features such that for example taking the values of this feature from $t_{-10} \dots t_{-1}$ will predict the price in t_1.

2. Related Work

There is broad body of research that focuses on price forecasting models for financial assets in general and bitcoin [15] and cryptocurrencies in particular. As the methodology for forecasting bitcoin prices and prices of other financial assets are effectively the same in the course of the project we analysed both types of papers and not only papers exclusively focused on bitcoin. There have been many attempts to review progress in stock market forecasting, such as [9] where authors focus not only on stock markets but also exchange rate, trading, banking default risk, portfolio management and other financial applications. Bustos and Pomares-Quimbaya [3] on the other hand focus particularly on stock market predictions and review new Machine Learning techniques, Di Persio and Honchar [6] review the use of Multil-layer Perceptron (MLP), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) techniques. Similarly, Andrade de Oliveira et al. [1] also use a generic approach analysing a variety of neural networks for stock price prediction. Looking at bitcoin price forecasting specifically Cohen [5] is using a relatively simple approach with linear regressions whereas Huang et al. [10] are using a classification tree-based model using a large amount of technical indicators examining the predictability of bitcoin using technical analysis. In parallel to the LSTM approach on the traditional market, McNally et al. [14] compare an LSTM bitcoin price forecasting model to another approach using autoregressive integrated moving averages (ARIMA) where the LSTM clearly outperforms.

Another idea is for instance to use NLP and sentiment classification signals as opposed to regressions, for instance the attempts in [4, 7] have yielded very interesting results. Other interesting papers are the work of Jang and Lee [11] who used Bayesian neural networks for bitcoin price forecasting and the work done by Guo et al. [8] looking at bitcoin volatility and order book data.

In summary, our work is a combination of some of the

above approaches using LSTM models while as the comparison benchmark we chose a GRU model. Additionally, as opposed to most of the above work in our project we used data based on multiple timeframes instead of just one as well as technical indicators.

3. Methodology

The final objective of this work is to accurately predict Bitcoin market price changes. The scope of this study is more about solving a real world problem than improving crypto-currency price forecast state-of-the-art models. Still, we hope this study will be of interest to the research community, and that future work will be able to get some insights from our approach. To achieve the final goal, we proceed in 3 main steps: (i) collect price data, (ii) propose a model and finally (iii) test the robustness of our model.

3.1. Data & Features

We have extracted our datasets from *Tradingview*[1], which is one of the most popular charting tools and price data aggregators. In total we have used three datasets for running our experiments on different timeframes in order to see whether different resolutions would have an impact on the prediction quality. We have extracted 15 minute, 4 hour and daily resolution data. We included price data in the form of opening, closing, high and low prices during each time period. In addition to that, we have extracted eight other technical indicators: the 20-, 50- and 100-day simple moving averages, the hull moving average which reduces the time lag, the trading volume data, the funding rates[2] for bitcoin futures from the Bitmex cryptocurrency exchange as well as the Moving Average Convergence Divergence (MACD) indicator and the relative strength index (RSI). Generally speaking a technical indicator can be understood as a heuristic which is derived from price or volume data or a combination of the two that is then used by traders to aid with decision making.

The data is represented as follows: the rows are the timestamps i.e. for example in the daily dataset each row represents the price metric or value of the respective technical indicator during that particular timestamp – in this case it would relate to one day. The columns represent the price data or technical indicators and thus are the features of this dataset. In Appendix A we have outlined the formulae used for the calculation of the technical indicators we chose. Our features are solely of continuous nature.

The daily dataset has 3932 datapoints in total going back all the way to July 2010 whilst both the 15 minute and 4 hour dataset have $11,895$ datapoints going back to January 2021 and January 2016, respectively.

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

[2] https://www.bitmex.com/app/perpetualContractsGuide#Funding

3.2. Proposed solution

Given the Bitcoin price data, we propose the following model to solve the prediction problem with the architecture described in Figure 1. It has 5 hidden layers composed of a 50 units LSTM cell with a Dropout layer to avoid overfitting. The biggest advantage of this particular architecture is its simplicity: both in terms of architectural design since there is no additional Deep Learning technique such as attention, and in terms of complexity since it only has $\sim 100k$ parameters (this is low compared to other architectures we tried with millions of trainable parameters).

```
Model: "CryptoRegressor"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 90, 50)            10400
_________________________________________________________________
dropout (Dropout)            (None, 90, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 90, 50)            20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 90, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 90, 50)            20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 90, 50)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 90, 50)            20200
_________________________________________________________________
dropout_3 (Dropout)          (None, 90, 50)            0
_________________________________________________________________
lstm_4 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_4 (Dropout)          (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 91,251
Trainable params: 91,251
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Model architecture

Optimal results are reported in Table 1 and obtained through training our model on the daily dataset using the following hyperparameter configuration: learning rate of 0.001, dropout rate of 0.2, batch size of 32, 90 time frames, `close` price as only feature and predicting 1 step ahead in the future. We report various metrics including the Root Mean Squared Error (RMSE) loss, the Mean Absolute Error (MAE) as well as the unscaled RMSE. The unscaled RMSE indicates how much error can be expected when predicting Bitcoin price in terms of actual US dollars and as long as this range of error stays within the actual range of prices throughout the day (i.e. open price compared to close price) we consider the model good.

	daily	**4hours**	**15min**
RMSE	0.000727	0.000632	0.001099
MAE	0.025801	0.022617	0.026513
RMSE (unscaled)	1450	2916	501

Table 1: Our model performance on each dataset

Results are to be qualified with their respective dataset because they all have their own specificies. The daily dataset is extremely wide, ranging from 0 from over $40k$, thus data transition is often not smooth. Even though a normalizing function is applied to help the model understand the data it is seeing, scales are often skewed during the process thus loosing precious information. As a result, this architecture struggles to model sharp transitions in data. On the other hand, the 15-min dataset is much more focused on recent fluctuations of the Bitcoin price, hence the better evaluation scores. It is worth mentioning that 501 error score largely falls between the open price and the close price. Therefore, although it is not suitable for forecasting prices with high fluctuations (see experiments in section 4), it can accurately do so if input data points are already close to each other, i.e. correlated.

On Figure 2, we can observe the predictions of our model on each dataset compared to the real values. Overall, it achieves decent performance: it always follows global trends very well, but lags a little bit behind when changes are very sharp. On the other hand, predicted values often do not match the real value exactly which seems intuitive and might be due to the min-max scaling we are applying during the training. Having said that, the forecasting of the general price direction is good. As a result, our system cannot be used as is for forecasting the exact price of Bitcoin, but rather, would be more useful in a trend-prediction scenario in which it works in tandem with other predicting systems. Finally figures in Appendix B show how the model performs on the same period but with different sample rates, i.e. the same period taken from different dataset. It is clear that when using coarsed-grained data (e.g. daily), this architecture is better suited to predict the general trend rather than the exact Bitcoin price in USD.

4. Experimental Results

4.1. Preliminary experiment

Before diving into the model performance evalution, and even before choosing the model architecture, we explore different popular Deep Learning techniques: bi-direction and attention. Many recent works highlight the necessity of using attention mechanism [17] or bidirectional RNN structures [20] to solve the price prediction problem. Therefore we give it a try and investigate how much they improve performance. The basic architecture consists of a stack of RNN cells with a single fully-connected layer on top to do the regression. We use PyTorch's simple RNN cells for this because they are the simple, fast to train, and because it helps get better insights about the best configuration since we do not care about the type of recurrent unit used for now.

Preliminary experiments are then conducted consisting of 4 parts, each dedicated to a special architecture design: (i) the first controls other experiments by giving the results for a simple architecture which does not use any additional mechanism, (ii) the second focuses on attention and adds an attention layer to the previous experiment, (iii) the third uses the bidirectional counterpart of each component from the first experiment, and lastly, (iv) the fourth features both mechanism and creates a bi-directional attention model. Each architecture is then optimized using Optuna: after a brief search to determine which hyperparameter set works best for a given architecture, we fix them. After that, we let the framework optimize the key parameters for the model structure: number of layers and hidden dimension size. In total, after the hyperparameter pre-search, we run 50 trials for each part of the experiment, train the models for 30 epochs each on the hourly dataset. We use a Mean Squared Error (MSE) loss function and accuracy as validation metric which is mostly useful for pruning unpromising trials.

Out of all trials in each experiment, we draw the 20% best attempts and plot their distribution using boxplots in Figure 3, the y-axis represents the validation loss. This way, results are made comparable to one another so that we can determine which architecture is best. We make the following 3 observations.

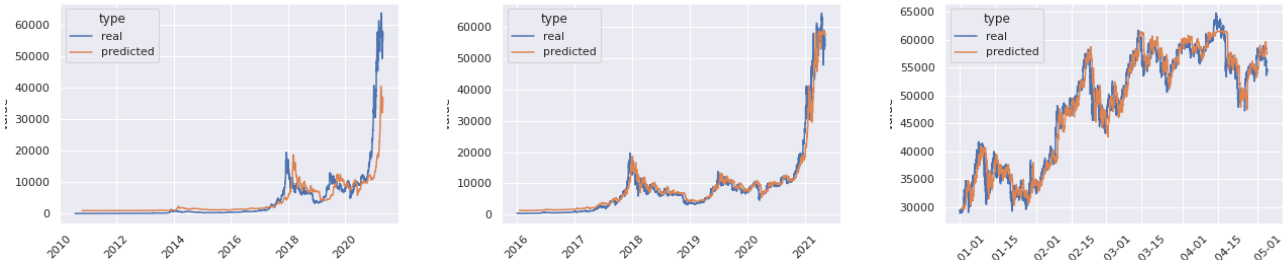
First, the *simple* architecture actually yields the best performance to our surprise, beating the others in pure performance (lowest mean) as well as in robustness (lowest variance). Even though we can safely say that it has the best results, the robustness part may be due to pure luck, better optimisation or even statistical bias. Still, it definitely seems like trials using the *simple* architecture do perform better than others.

Second, the attention mechanism appears to be harmful to the performance, see the *attn* column. We believe that the model focuses too much on small details, because there are no latent features involved that can mitigate this effect by embodying a larger range of causes to a price change. These small changes harm the prediction by hidding the general trend from the model. This phenomenon can also be observed in the fourth experiment *attn+bi* where the information from the future (thanks to the bidirectional connections) helps address the issue, but still is not enough to compensate. Different attention mechanisms should be investigated, such as [2, 13].

Third, bidirectional systems perform a little bit worse than the *simple* ones, but they also come with a larger amount of parameters: they have twice as much trainable parameters as their uni-directional counterpart. Overall, all systems are beaten by the *simple* architecture, therefore we stick to this design choice for future experiments and inference.

4.2. Structural study

Although the previously described model already performs decently at forecasting Bitcoin prices, we want to



(a) Daily dataset (b) 4 hours dataset (c) 15 minutes dataset

Figure 2: Predicted values vs real values on each dataset using our proposed model

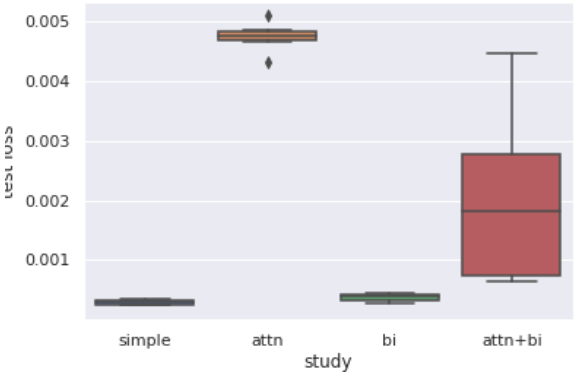


Figure 3: How relevant are attention and bi-directional structures for Bitcoin price prediction? A boxplot of the 20% best runs for each study, compared in terms of validation loss.

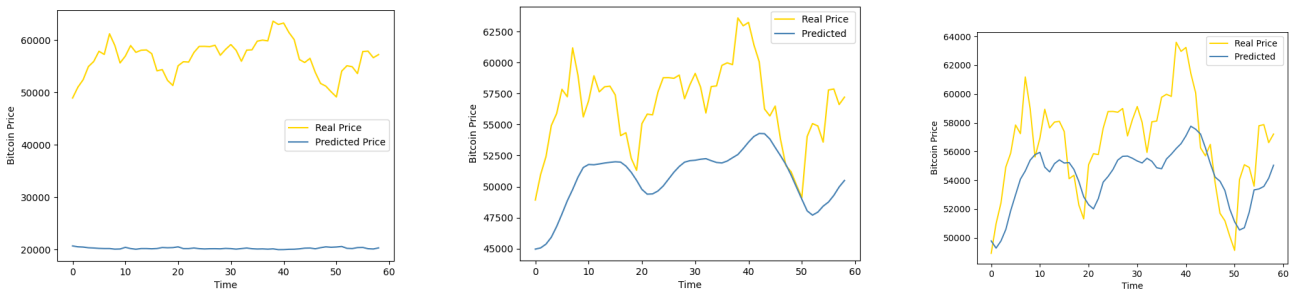
make sure it is robust. By investigating how architectural design choices influence the performance of our model, we can determine its robustness to structural changes. It is also a good opportunity to determine which parameters makes our proposed model the best (out of all architectures we tried).
More precisely, we experiment upon 4 different architectural aspects:

- the number of time frames used for prediction,
- the type of RNN cell used: comparing GRU with LSTM,
- the number of recurrent layers: since the model consists of stacked RNN cells, it is important to determing how many is really necessary,
- the hidden dimension within each RNN cell: increasing the hidden dimension size basically increase the number of parameters of the model and thus, its representational power.

Similar to preliminary experiments, we let the Optuna framework optimize numerical parameters (number of layers and hidden dimension) while keeping all other parameters constant for comparison, including hyperparameters. This allows for a relevant comparison of architecture performances, and thus, practical reasons to justify our final proposed model.

Time Frames We experiment with 3 different values for time frame usage: 90, 10 and 3 timeframes. Clearly, the less frames are used for prediction, the more accurate the results are (see Figure 4). We suspect that too much decorrelated data at once fools the regressor into wrong predictions because it contains fluctations having nothing in common with each other. For instance, early days of Bitcoin have seen variations of extremely low magnitude ($10 at most) while recently, prices have been taking big jumps from $40k to $30k in few hours. Trying to make predictions based on such price changes which have apparently nothing in common is prone to errors. However this particular behaviour can actually be of interest when predicting general trends or price directions (such as in 4b where prediction loosely follow the overall tendency).
Moreover figures in Appendix B illustrate the ability of our model to predict trends when given more data. In these experiments the number of timeframes used is identical but what changes is the sample rate. As a result, the model sees more data, thus more fluctuations, in the daily data than in the more fine grained 15-min data.

Number of layers The influence of changing the number of stacked layers of RNN cells can be observed on Figure 5a. Both figures in Figure 5 represent boxplots created out of the results distribution from all trials which y-axis is a logarithmic loss, thus the lower the better. At first, we can argue that careful training configuration through hyperparameters is required when changing the model architecture: many outliers can be detected within the graph that are a direct consequence of a sub-optimal configuration. For



(a) 90 timeframes (b) 10 timeframes (c) 3 timeframes

Figure 4: Influence of the number of time frames on a LSTM architecture trained on the daily dataset

LSTM, it is best to have 2 layers because too much layers (> 2) actually harms performance, whereas 1 layer is enough for the GRU-based models. Less observable than its LSTM counterpart, the 2 layers version of GRU is slightly better but not worth the extra complexity that results from the added layer.

Hidden dimension Interpretation of Figure 5b is more straightforward: in general, increasing the hidden dimension size gradually improves performance. Augmented complexity also seems to help robustness to hyperparameter changes when paying attention to the variance of each boxplot: it is lower at higher levels of hidden dimension size. Although it is ovious for LSTM that 1024 dimensions are necessary to obtain the best outcome, it is not so evident for GRU for which 512 dimensions is also a good candidate (also see Figure 7).

Cell type Long Short Term Memory (LSTM) cells have long been utilised for sequence prediction, particularly in the financial industry as in [12, 16, 18, 19], and are known to yield the best results compared to all other types of RNN cells. As expected, Figure 5 demonstrates that LSTM produces better results than GRU. But there is more to it: Table 2 describes the very best run for each cell type. Interestingly, the LSTM-based model has almost twice as much parameters as its GRU counterpart, but outperforms it by 49% in RMSE loss. Since it represents the best run out of all trials, the GRU architecture using 1024 hidden dimensions and only 1 layer is undeniably the best choice. Common sense encourages adding more parameters to improve performance, but the contratry happens: 512x3 is a 5 millions parameters model but only performs a 0.001 loss, so as the 1024x2 which has 8m parameters. Surprisingly, it indicates that the GRU cell has less potential than LSTM as it achieves its top performance with 4m parameters and does not seem to be able to go higher. Although the number of parameters is critical in such models, the GRU architecture has already reached its limit.

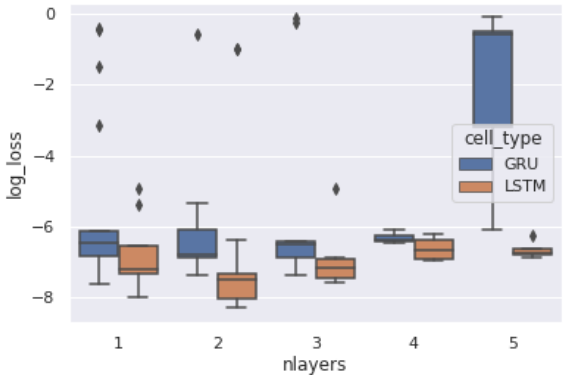
	Loss	Params	Training Time	Hidden	Layers
GRU	0.000491	4,207,617	220 sec	1024	1
LSTM	0.000246	9,453,569	380 sec	1024	2

Table 2: Influence of the cell type: LSTM versus GRU on the 4 hours dataset

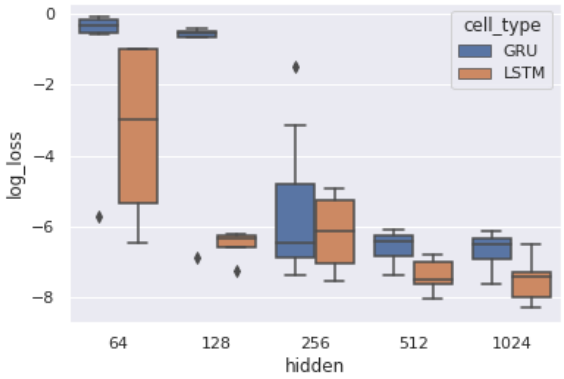
4.3. Design choice

After such an analysis of the structure, we wish to explain our design decisions for the final model. One important issue that needs to be clarified is: does the number of parameters have such influence over the model performance? In Figure 6 we try to demonstrate the correlation between these two aspects. LSTM and GRU have different behaviour, while LSTM-based models' loss gradually decrease with the increasing number of parameters, GRU has more sharp turns, especially a very good performance around $2e^5$ parameters but then goes up again. In general, adding parameters to the model helps achieve good results, but models have a limit to how much they can improve. We observe that best results are reached with millions of parameters ($\sim 1e^6$), and there is a significant difference compared to $1e^5$. Passed this limit, there is no point in adding more parameters because it only increases the complexity, thus the training time.

Finally, Figure 7 can be interpreted as a summary of all our experiments. The diagonal reveals how much of each configuration has been tested, while the lower corner indicates the estimated kernel density (KDE) for some magnitudes of interest, grouped by pairs. In fact, only the last line of graphs is effectively interesting as they show what performance we should expect for a given configuration. We can employ these estimations to define the best architecture: it seems a number of layers as low as 2 (respectively 1 for GRU) is expected to yield the best performance for LSTM (resp. GRU). On the other hand, 2 clusters appear when it comes to the hidden dimension size: 1024 appears to have the best results but 512 is a very interesting compromise. We definitely want to go for the best performing model in



(a) Number of layers



(b) Hidden dimension

Figure 5: Evaluating the influence of various parameters over the model performance using boxplots (diamond points are considered outliers)

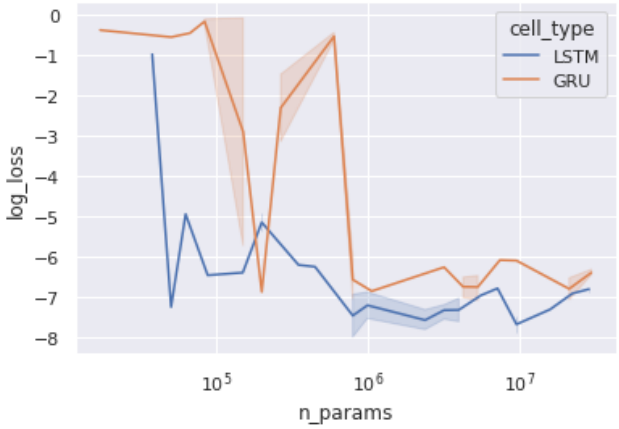


Figure 6: Investigating a possible correlation between the number of parameters and the model performance. Log loss (RMSE) versus the n_{params} using all our trials of LSTM and GRU.

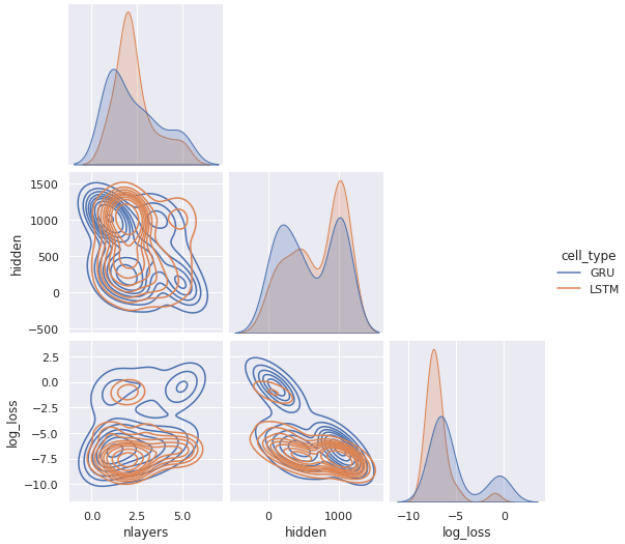


Figure 7: Summary of all trials indicating the best model configuration: a 2-layers LSTM model with 1024 hidden dimensions supplemented with a single Linear layer as regressor.

preparation of a future implementation within an end-to-end Bitcoin trading system.

5. Conclusion

Overall, as we have seen the results by both the GRU and the LSTM approach are quite good and could be fine-tuned even more by using different sets of hyperparameters, features and the number of periods used to forecast the next price. With regards to practical applicability if we wanted to build a trading system around this model even the level of precision we achieved would not be enough as especially during periods of volatility such as sharp market drops for instance the RMSE jumps up.

However, a case could be made for using the results of this model as an input for a trading algorithm for example as a general momentum indicator.

Another conclusion we came to is that there is a trade-off to be made between model complexity and precision although generally relatively simple LSTM architectures with less than 100,000 trainable parameters have achieved almost the same results as more complicated architectures with up to 1,000,000 trainable parameters. Again looking at the applicability in real life probably a simpler model

would be more advantageous even at the expense of slightly more imprecise results as it would be much quicker to run and could produce actionable results on time. For instance a model creating predictions for prices in 15 minutes that needs to run for several hours would only be of theoretical interest.

Lastly, the questions arises of how the results could be improved further and as initially certainly other combinations of hyperparameters than the ones tested could potentially be found. Additionally, Fourier series could be applied to the price data for example to de-noise it and use it as another feature. Moreover, classification features could be also added to the dataset for example mining social sentiment from platforms such as Twitter or Reddit with NLP methods.

References

[1] F. Andrade de Oliveira, L. Enrique Zárate, M. de Azevedo Reis, and C. Neri Nobre. The use of artificial neural networks in the analysis and prediction of stock prices. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2151–2155, Oct. 2011. doi: 10.1109/ICSMC.2011.6083990. ISSN: 1062-922X. 1

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, May 2016. URL http://arxiv.org/abs/1409.0473. arXiv: 1409.0473. 3

[3] O. Bustos and A. Pomares-Quimbaya. Stock market movement forecast: A Systematic review. *Expert Systems with Applications*, 156:113464, Oct. 2020. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.113464. URL https://www.sciencedirect.com/science/article/pii/S0957417420302888. 1

[4] R. Chen and M. Lazer. Sentiment analysis of twitter feeds for the prediction of stock market movement. *stanford edu Retrieved January*, 25:2013, 2013. 1

[5] G. Cohen. Forecasting Bitcoin Trends Using Algorithmic Learning Systems. *Entropy*, 22(8):838, Aug. 2020. doi: 10.3390/e22080838. URL https://www.mdpi.com/1099-4300/22/8/838. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute. 1

[6] L. Di Persio and O. Honchar. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing*, 10(2016):403–413, 2016. 1

[7] D. Garcia and F. Schweitzer. Social signals and algorithmic trading of Bitcoin. *Royal Society open science*, 2(9):150288, 2015. Publisher: The Royal Society Publishing. 1

[8] T. Guo, A. Bifet, and N. Antulov-Fantulin. Bitcoin Volatility Forecasting with a Glimpse into Buy and Sell Orders. *2018 IEEE International Conference on Data Mining (ICDM)*, pages 989–994, Nov. 2018. doi: 10.1109/ICDM.2018.00123. URL http://arxiv.org/abs/1802.04065. arXiv: 1802.04065. 1

[9] J. Huang, J. Chai, and S. Cho. Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, 14:1–24, 2020. Publisher: Springer. 1

[10] J.-Z. Huang, W. Huang, and J. Ni. Predicting bitcoin returns using high-dimensional technical indicators. *The Journal of Finance and Data Science*, 5(3):140–155, Sept. 2019. ISSN 2405-9188. doi: 10.1016/j.jfds.2018.10.001. URL https://www.sciencedirect.com/science/article/pii/S2405918818300928. 1

[11] H. Jang and J. Lee. An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information. *IEEE Access*, 6:5427–5437, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2779181. Conference Name: IEEE Access. 1

[12] S. Lahmiri and S. Bekiros. Deep Learning Forecasting in Cryptocurrency High-Frequency Trading. *Cognitive Computation*, 13(2):485–487, Mar. 2021. ISSN 1866-9964. doi: 10.1007/s12559-021-09841-w. URL https://doi.org/10.1007/s12559-021-09841-w. 5

[13] M.-T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [cs]*, Sept. 2015. URL http://arxiv.org/abs/1508.04025. arXiv: 1508.04025. 3

[14] S. McNally, J. Roche, and S. Caton. Predicting the Price of Bitcoin Using Machine Learning. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 339–343, Mar. 2018. doi: 10.1109/PDP2018.2018.00060. ISSN: 2377-5750. 1

[15] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Technical report, Manubot, Nov. 2019. URL https://git.dhimmel.com/bitcoin-whitepaper/. Publication Title: Manubot. 1

[16] E. G. Pintelas. DEEP NEURAL NETWORKS FOR BITCOIN PRICE PREDICTION. 2020. 5

[17] J. Qiu, B. Wang, and C. Zhou. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PLOS ONE*, 15:e0227222, Jan. 2020. doi: 10.1371/journal.pone.0227222. 3

[18] B. Spilak. Deep neural networks for cryptocurrencies price prediction. Master's thesis, Humboldt-Universität zu Berlin, 2018. 5

[19] K. Struga and O. Qirici. Bitcoin Price Prediction with Neural Networks. In *RTA-CSIT*, pages 41–49, 2018. 5

[20] M. Sunny, M. M. S. Maswood, and A. Alharbi. Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model. pages 87–92, Oct. 2020. doi: 10.1109/NILES50944.2020.9257950. 3

A. Technical indicators

Simple Moving Average (MA)

$$SMA_n = \frac{P_1 + \dots P_n}{n} \tag{1}$$

Exponential Moving Average (EMA)

$$EMA_n = (C \times K) + (EMA_{n-1} \times (1 - K))$$
$$K = \frac{2}{N+1}$$

where EMA_n is the value of the current exponential moving average and EMA_{n-1} is the value for the previous period, while N is the length of the EMA.

Relative Strenght Index (RSI)

$$RSI_{14} = 100 - \frac{100}{1 + \frac{x}{y}} \tag{2}$$

where x is the average gain of up closes in the last 14 periods and y is the average loss of down closes in the last 14 periods.

Weigthed Moving Average

$$WMA_n = P_1 W_1 + \dots + P_n W_n \tag{3}$$

where P_1 is the first price to be considered and W_1 is the weight of the first price.

Hull Moving Average

$$HMA = WMA_{raw}(\sqrt{n}) \tag{4}$$

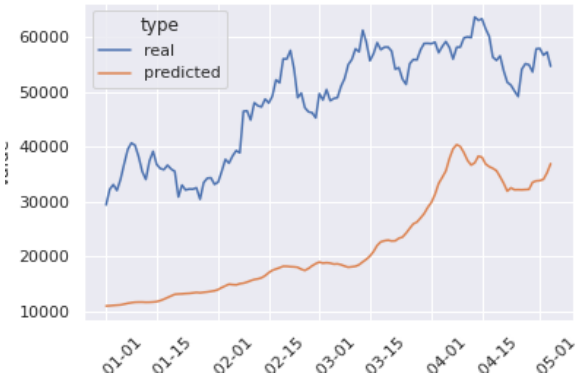
where $WMA_{raw} = WMA_n(2 \cdot WMA_{n/2}) - WMA_n)$ [3]

Moving Average Convergence Divergence

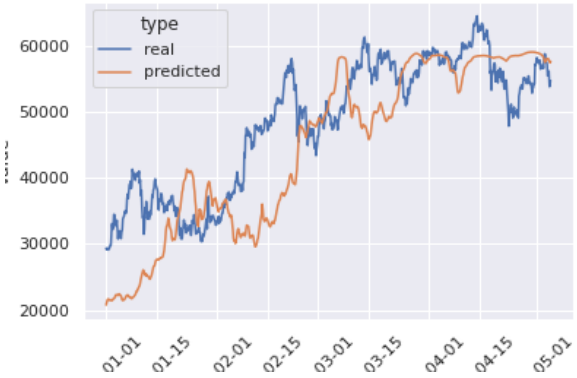
$$MACD = EMA_{26} - EMA_{12} \tag{5}$$

B. Model performance

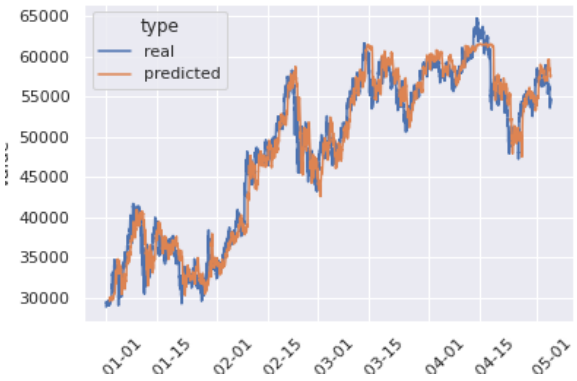
[3] https://school.stockcharts.com/doku.php?id=technical_indicators:hull_moving_average



(a) Daily dataset



(b) 4 hours dataset



(c) 15 minutes dataset

Figure 8: Model predictions on the same period taken from each dataset (thus with different sample rates)