Long Short-Term Memory for Bitcoin Price Prediction

Jordan Jones
Florida Polytechnic University
Jordanjones2078@floridapoly.edu

Doga Demirel
Florida Polytechnic University, Corresponding Author
ddemirel@floridapoly.edu

ABSTRACT

With time-series data being prevalent everywhere, there is a need to predict this data accurately. This kind of data includes weather data, financial data such as stock price, and cryptocurrency price. Most of the trades in the stock market in this day and age are being made using artificial intelligence. An estimated 50% of trades were done using an algorithm, which increased to 60% in 2020 [1]. This highlights the demand for reliable and accurate predictions. The prediction of the price is very challenging. Some success has been seen when predicting stock prices, but not many studies have been done on cryptocurrency. Cryptocurrency, specifically Bitcoin, has seen a substantial increase in popularity, and the price has reflected this popularity. The price also follows patterns specifically when reaching new all-time highs. In this work, an Artificial intelligence is created and trained on the previous data to observe these patterns and predict the next price. The artificial intelligence chosen for this subject is Long short-term memory (LSTM). LSTMs are capable of finding patterns in time series data. LSTM solves the vanishing gradient problem present in the RNN (Recurrent Neural Network). The Market Price of Bitcoin is used as input here. The data values for input range from 20,000 up to 65,000 in testing. Once an optimal starting point is found, there is an 80/20 split of data, 80 percent of the data is used for training and 20 is used for testing. With the data being split, one of the most important jobs is figuring out the optimal lags (how far back into the past) when used to predict values. This range for this experiment is set to ten previous price days. Epochs (number of iterations) and Batch size (how much of the training data is used per epoch) are tested at different values to find optimal solutions. With batch size values such that *batchSize* $\in \{2^0, 2^1 \ldots 2^6\}$ and epochs such that *epochs* $\in \{10, 20 \ldots 70\}$. Overfitting is hard to detect and thus can be an issue with too many epochs and smaller batch sizes (smaller means more of the training data is used). Too little and the LSTM will not learn the data patterns and thus will not have good accuracy. This is why different configurations are used in the experiment to maximize accuracy. This LSTM was used to achieve a Mean Absolute Percentage Error score of 3.23% and a Root Mean Squared Error score of 1892.87 when predicting next-day prices throughout 350.

CCS CONCEPTS

• **Computing methodologies** → Artificial intelligence.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICISDM 2022, May 27–29, 2022, Silicon Valley, CA, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9625-7/22/05…$15.00
https://doi.org/10.1145/3546157.3546162

KEYWORDS

Bitcoin, LSTM, Prediction

ACM Reference Format:
Jordan Jones and Doga Demirel. 2022. Long Short-Term Memory for Bitcoin Price Prediction. In *2022 the 6th International Conference on Information System and Data Mining (ICISDM 2022), May 27–29, 2022, Silicon Valley, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3546157.3546162

1 INTRODUCTION

Time-series is "a set of data collected sequentially, usually at fixed intervals of time" [2]. With how popular Bitcoin has gotten recently and many stores such as Starbucks and PayPal accepting as payment proves that we are moving towards a decentralized digital economy [3]. Time series data is present everywhere, from weather reports to the stock market. Time series data is just data collected at consecutive time intervals. Predicting this data is a challenging task because the data is observed and recorded. This data doesn't always have a clear pattern to observe. That is why a neural network is required to find the pattern. Most prediction models only attempt to predict the next day's price, with some only predicting if the price goes up or down the next day. In terms of the price, Bitcoin can be viewed as similar to a stock's price. Shares of stock are traded identical to how Bitcoins are traded. This allows us to use some of the research done on stock market analysis and prediction as a starting point. The previous research will not be a 1 to 1 copy but can help get us started. The reason for Bitcoin is the price has increased from $226 in April of 2015 to $65,000 on Nov. 15, 2021. This price increase is a 28,000% increase in 6 years [4]! With this popularity and meteoric price rise, Bitcoin is traded like most stocks. Cryptocurrencies are a relatively new concept; there isn't many historical data to go off.

The volatility of Bitcoin cannot be understated. The price can rise or drop based on public news and Twitter like any stock. When Elon Musk tweeted that Tesla would not accept Bitcoin for transactions anymore, the price of Bitcoin dropped by 15% [5]. With the concept of Bitcoin being very new, stability in some countries is hard to come by. China recently banned all crypto-related activities, and this includes mining. China was one of the biggest countries for mining Bitcoin. This announcement caused a drop of 5% in the total price of Bitcoin [6]. With the ban from China, other countries such as India are proposing a ban [7]. The approach developed here provides accurate results when predicting something as volatile as Bitcoin. This is done by changing hyperparameters and only focusing on the more recent data points. These current data points more accurately reflect the price and trends of Bitcoin. The LSTMs generated here should perform differently based on the hyperparameters each of them receives before training and the data each of them receives.

Many different papers were reviewed that carry out an analysis of the stock market. This analysis is primarily a price prediction but can also develop a neural network-assisted trading strategy. These papers highlight the importance of good data gathering and preprocessing before being fed into the network. These data preprocessing methods include normalization, attaching weights to the different time series data points, or even transforming the data entirely into something like binary values. Once preprocessing is done, most of the common approaches feed the values into an LSTM. Most results showed the power of LSTM by outperforming the popular ARIMA model. The ARIMA model is a more traditional algorithm used to predict time series data. Convolutional networks showed promise here as well, so a hybrid approach is also valid.

Some models don't do data preparation and preprocessing but instead, let the network do most of the work. This paper by Murtaza Roondiwala et al. [8] grabs the relevant technical indicators such as volume, open, close, high, and low. After gathering this data, they preprocess it by discretizing and normalizing it. Once the data is in the correct form, it is fed into an LSTM. The results show that the model predicts the overall structure of the data but appears to be a couple of time steps off. The results also show that the model is missing some rapid changes but instead averages those changes out.

ARIMA is one of the first attempts at analyzing time series data. The work done by S. Siami-Namini et al. [9] compares this model to the recently created LSTM at time series forecasting. Each model was fed the adjusted close value from the same stocks. Once evaluated, the results showed that the error rate decreased by 87% why using the LSTM. ARIMA was also compared against LSTM, RNN, CNN, and MLP models in this study by S. Selvin et al. [10]. CNN (Convolutional Neural Network) is a type of neural network that uses convolutions to analyze the input. A MLP or multilayer perceptron is a simple feed forward network consisting of an input layer, output layer and a hidden layer. It was proved that LSTM RNN CNN outperformed ARIMA. Among those three, CNN outperformed LSTM and RNN. The LSTM and RNN models both had points in each of the tests where their predictions had a significant divergence from the trend of the real data. This shows that CNNs can have a place in time series forecasting, and perhaps a convolution layer should be applied to the LSTM.

The work done by J. Choi et al. [11] attempts to combine ten different LSTMs. Each LSTM has a different sequence length of data. Each LSTM makes its prediction and is combined to create one combined prediction. The weights here are developed using their own algorithm, which accounts for the different delays in the time. The proposed method of an ensemble of varying LSTM networks proved to have the lowest error among the other ensemble methods. These include least square regression, averaging, neural network-based linear ensemble, average sample weights, and median.

An RNN was created to predict apple stock price upon opening [12]. The RNN created achieved an accuracy of 95% with an error of .1%. There were multiple tests run, and each RNN created has two layers with either 50 or 100 units. The RNN was trained with either 5 or 10 timesteps of data. The lower timesteps 5 performed better than the higher 10. This shows that you don't need to look far into the past to get good predictions. This indicates that different lags are required to find the optimal.

The attention model will assign weights to the time series data to help improve accuracy. The models use four weeks of information to predict the next day's closing price. After the weights are assigned to the time series data, it is fed into the LSTM. The weights act as ways for the model to value specific inputs as more critical and some data points as less critical. This helps the LSTM learn because it knows which values are important and which aren't. Mean absolute percentage error is used. The AT-LSTM is compared with LSTM and ARIMA. Both LSTMs outperform the ARIMA easily. The AT-LSTM has an improvement over strictly the LSTM. The AT-LSTM still has trouble during some extreme conditions [13]. In the work done by A. H. Manurung et al.[14], the models use the recent price data from 1, 3, and 5 years to predict the next day's value. This study shows the increase in accuracy when increasing the epochs from 5 to 20. However, the best results are with 100 epochs. The Arima model's accuracy is only 56% compared to the 94.59% achieved by the LSTM model. This study is more proof that the ARIMA model, while effective, is getting outshined by newer LSTM models.

The inputs used in this model are "trading volume", "adjustment close price", "profit margin", "diluted earnings per" share, "company beta", "return of equity", "debt-to-equity ratio". the data was scaled to values between 0 and 1 using the min-max scaler. ARIMA, LSTM, stacked LSTM, and Attention LSTM are created in this study done by Z. Zou et al [15]. The ARIMA model was used using the "adjusted close price." The LSTM model is created with one layer, while the stacked model is more complicated. The Attention LSTM is an LSTM where the inputs have weights attached to them. This will allow the model to value data differently based on its importance before the LSTM is run. The models here predict the value on the next day. On top of the models created to predict the price, two different trading strategies are created: a long-short strategy and a long-only strategy. The long-only strategy means if the stock is projected to increase in price, it is bought when the market opens and sold at close. If the projection is the price will decrease, no buying or selling is done. For the long-short strategy, the same is done if the price is projected to increase, but if the price is projected to decrease, the stock is short sold upon the opening of the market, and when the market closes out, the short-sell is then closed out. The TANH function is used for activation, and the activation function sigmoid is used in the output layers. Smaller lookback days were proven to be more effective than longer lookback days, so 20 is used here.

To increase the model's efficiency, a batch size of 30 is used. The Stacked LSTM model did not outperform the LSTM model. This shows that you do not need to add layers to make a neural network more effective. The LSTM and attention-based LSTM were used in the trading strategy and compared vs. the SAP 500 annual return. Both LSTM networks did substantially better. The Attention LSTM was the most superior out of the LSTM models, proving that the attention portion helps the LSTM model capture more drastic changes [15].

The RNN was created to predict the opening price using the previous 12 days. The RNN was trained on a small sample size of only 156 points. The prediction accuracy of the network has a percent error of 5%. This work shows that you do not need a large dataset to predict trends in a time series dataset [16].

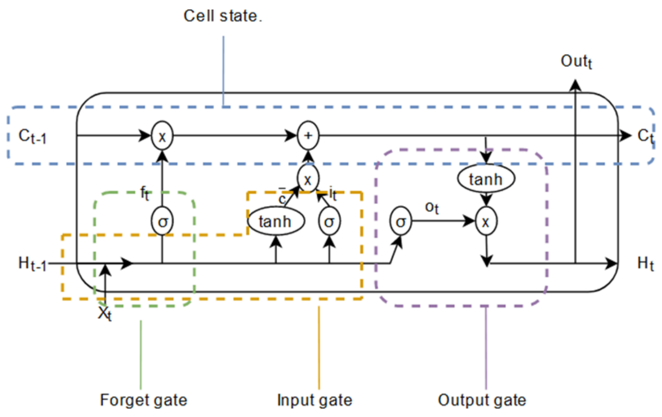

Figure 1: Diagram of LSTM

2 METHODOLOGY

The approach for predicting time series data here can be broken down into three different steps. The three steps are data collection, data preparation, and network training. 1) Data collection. An ample amount of data is required for an LSTM to be effective at time series prediction. Therefore, LSTM can have enough data to test on and pick up the patterns in the time-series data. 2) Data preparation. The data should be prepared in batches of input and output. Along with the preparation, the data is scaled down to 0-1s so the LSTM can accurately train on them. 3) Creation and training of the LSTM. An LSTM is chosen because it's a type of RNN that solves the vanishing gradient problem by adding a forget gate. The LSTM is composed of three different gates, an input gate, an output gate, and a forget gate as shown in Figure 1. Each of the three different gates is responsible for specific tasks, and each gate has a formula for how the values are updated [17] [18].

The forget gate's main responsibility is making sure old information isn't lost over time by deciding which information should be taken from the previous runs. The forget gate takes both the current input and the value from the previous hidden state. The hidden state can be viewed as the short-term memory of the LSTM. The forget gate takes four inputs and applies a sigmoid activation function. The weight associated with the forget gate U_f, the current input at time t (X_t), The hidden state at t-1 (H_{t-1}), and lastly, the weights associated with the hidden state for the forget gate W_f.

$$Ft = \sigma(Wf * Ht - 1 + Uf * Xt)$$

The input gate is responsible for handling the input from the dataset. The input gate's main task is to decide how to update the values. The input gate has four inputs and a sigmoid activation function. The input gate takes the current input at time t (Xt), the weights associated with the input Ui, and the ones dealing with the hidden state Wi. lastly, the value at the previous time step Ht-1. The full function for the input gate is:

$$It = \sigma(Wi * Ht - 1 + Ui * Xt)$$

The value $\boldsymbol{Ct}$ is calculated here. This value is used when updating the cell state. The cell state can be thought of as the long-term memory of the LSTM. The value is calculated by:

$$\bar{C}_t = tanh(Wc * Ht - 1 + Uc * Xt)$$

The following function calculates the value of the cell state.

$$Ct = Ct - 1 * ft + \bar{C}_t * it$$

Here the forget gate performs its task and is responsible for how much of the previous state we should remember. The input gate I affect how much of C should affect the current state. The output gate is responsible for calculating the value of the hidden state at the next time step. The output gate takes the hidden state H_{t-1} and current input X_t. The equation for the output gate is:

$$Ot = \sigma(Wo * Ht - 1 + Uo * Xt)$$

The new hidden state value uses this new O_t value and C_t's current state value.

$$Ht = tanh\,(Ct) * Ot$$

The output can be received at the given timestep by applying a softmax function to this hidden state value

$$Outt = softmax(Ht)$$

The system we used was Windows 10 version 20H2 64-bit machine. AMD Ryzen 7 3700x 8-core processor (16 CPUs ~3.6ghz minimum speed). 16gbs of RAM. Nvidia GeForce RTX 3080 TI for the graphics card.

2.1 Model Evaluation

The accuracy here is evaluated in 2 different ways, Mean Absolute Percentage Error (MAPE) and RMSE. Both MAPE and RMSE are common formulas for assessing the efficiency of a forecasting model. MAPE stands for mean squared percent error. The MAPE provides the prediction error as a percentage, thus allowing you to compare models with different data sets. The percentage from the MAPE equation is how far off your prediction is from the real value. The value for MAPE is calculated by the below formula [19].

$$\frac{1}{n}\sum_{x=1}^{n}\left|\frac{real_x - predicted_x}{real_x}\right|$$

Where real is the real value, predicted is the predicted value. The formula goes from 1 to n, where n is the length of the test set. This value provides the error in a percentage form

The RMSE Root Mean Squared Error is done by taking the sum of the squared difference between the real value and predicted value. This will produce the MSE. To get the RMSE you take the square root of this. [20].

$$\sqrt{\sum_{x=1}^{n}\frac{(real_x - predicted_x)^2}{n}}$$

2.1.1 *Step 1: Data Collection.*

Bitcoin market price is the primary data used for this LSTM. The market price is rapidly changing, so values are acquired daily. The website data.nasdaq.com has a collection of various datasets on there published by different teams from different datasets. For this study, the Bitcoin dataset from blockchain.com was used. This dataset was published by Quandl, a team of data scientists that scrape numerous data sets and allows the public to use them for free. There are multiple ways to access Quandl datasets, and the Python API was used for this project.

Multiple different data sets were imported, and as a future work of this project, various data combinations can be used. The data is imported in either a pandas Dataframe format or a Numpy format. Both different forms were used in this approach.

2.1.2 Step 2: Data Preparation. We only use the most accurate and relevant data points for the data to be prepped effectively. Once the data is imported from the Quandl library, several different data preparations and methods are used. The main problem with the dataset acquired from Quandl is the price is minuscule at the beginning, and there is a period where the price is so low it only registers 0 on the dataset. A simple method is created to pull only the relevant data. This method iterates through the data and figures out the first non-zero entry (around 590 data points). This means the first few data points can confidently be ignored as they will provide no use to the network. Another thing is because of the nature of Bitcoin and its rise in popularity, the trends, and patterns Bitcoin follows will change drastically over time. For this reason, datasets of varying lengths are used. The optimal range of data is ignoring the first 2500+ data points. Different configurations of dataset lengths are tested to determine the correct starting point. The data is normalized here to values between 0 and 1. This is to increase the accuracy of the LSTM. After the length of the dataset is determined, the data is then split into a training set and a testing set. This ensures that the LSTM has not been trained on the same data it is being tested on. This is because the LSTM would already have the answer in this case. An 80/20 split is done here to ensure enough data to train on and enough to get an accurate prediction. The number of lags (days in the past) is set at 10. This means the LSTM will look ten days into the past when making a prediction.

The *trainx* values consist of the price at the previous days such that $trainx_t \in \{price_{t-1}, price_{t-2} \ldots price_{t-timePeriod}\}$. Where *timePeriod* is 10. These *trainy* value is simply the price at the current day such that $trainy_t = price_t$. This method is run twice, once for the training dataset and a second time for the testing dataset. Once the data has been preprocessed, it is fed into the LSTM.

2.1.3 LSTM.. The LSTM is created with a single dense layer at the end. The LSTM has 100 neurons. The loss is calculated with mean_squared_error, and ADAM is used as the optimizer. The input for the LSTM is a tuple of shape (1, *timePeriod*, *nFeatures*). This iteration only uses one feature, so *nFeatures* is one, and *timePeriod* is set to 10. The LSTM has different hyperparameters such as epochs and batch size. Epochs are how many times the LSTM is run. The batch size is how much data will be used in training. A batch size of 1 will use all of the data. Different configurations of these hyperparameters are tested. At intervals such that $epochs \in \{10, 20 \ldots 70\}$ epochs are used as well as batch sizes of powers of 2. $batchSize \in \{2^0, 2^1 \ldots 2^6\}$. This, along with different amounts of data, is used to determine the effect each has on the network and figure out which is the most accurate. Data length ranges varied from 1882-1658, decreasing by a value of 25 each iteration of testing. The LSTM will look at the previous *timePeroid* (10) values and use these values to predict the price of Bitcoin on the following day. This is run, and the model then predicts the testing dataset 1 data point at a time. The results are stored and compared against the actual results. The Root Mean Squared Error is recorded for each run and stored inside a CSV file. The models are all saved. Thus, the results can be reproduced. The goal is to find the models with the highest accuracy. The configurations of these models are what we are most interested in. The models with very low accuracy can also be observed and figuring out a trend of configuring hyperparameters and datapoints causes a model to fail. These combinations can be observed and used to see what hyperparameters to avoid.

3 RESULTS

The model was run 490 times with ten different data configurations, seven different batch sizes, and seven different number of epochs. The results showed promise with a batch size of 16. This provided the models with good stability and Efficiency. It appeared that the less amount of data used provided more accurate results. 6 results are shown in Table 1 to analyze the different effects of Batch Size, Epochs, and data lengths. The best result can be seen in Figure 2 (a), with a configuration of 16 for the batch size, 50 for epochs, a training length of 1432, and a testing length of 351. A few other runs of interest were included in the table showing the effect of different configurations. The high values for RMSE are because RMSE is relative, and the scale of data used here is \$20,000-\$65,000.

Figure 2 (a) shows the model with the highest accuracy. The RMSE was 1892.87. This high RMSE number is due to the data in the test set ranging from \$20,000 up to \$65,000s. The model here had the Batch Size = 16, Epochs = 50, Training length = 1432, Test length = 351. The model can follow the trends but has trouble when the values fluctuate at a high rate. This volatility is a quality of Bitcoin that must be accounted for. In Figure 2 (b), the effect of different numbers of epochs can be observed. This model has the same hyperparameters as Figure 2 (a) but had epochs = 10. This caused the model to pick up the general trend but not accurately predict the sudden changes. The number of epochs means that the model only trained for ten iterations. This limited training time caused the model to understand how the price would move in a general sense but couldn't pick up on the volatility of bitcoin at price points.

A model with a batch size of 1 was found to have the most fluctuations in the results. As seen in Figure 2 (c), the model has accurate results with an RMSE of 2055. The model was able to follow the trends and only had some problems when dealing with data peaks. These results show a good model, but the model runs into some problems when increasing the epochs to 70. The results when the hyperparameters were set as: Batch Size = 1, Epochs = 50, Training length = 1492, Test length = 366. The model in Figure 3 (a) had the same hyperparameters as Figure 2 (c), but the epochs were increased to 70. This shows the volatility with a batch size of 1. Again, this model can follow some of the trends, but when reaching data peaks, the model overestimates the price. This can be very bad as if you were an investor, you would see these price points and think to invest beforehand.

The data fed into the network can be observed with these two different examples. The hyperparameters are Batch Size = 16, Epochs = 50, Training length = 1511, Test length = 371. The difference in the amount of data being fed to each network is 224, but the results change the RMSE from 2797.98 in Figure 3 (b) to 1965.3 in Figure 3 (c). Figure 3 (c) has, Training length = 1332, Test length = 326. This shows that more data does not always mean a more accurate result.

Table 1: List of Results

Batch Size	Epochs	Train Length	Test Length	RMSE	MAPE	Corresponding Figure
16	50	1432	351	1892.87	3.23%	Figure 2 (a)
16	10	1432	351	2653.6	4.78%	Figure 2 (b)
1	50	1492	366	2055.22	3.47%	Figure 2 (c)
1	70	1492	366	6205.67	7.9%	Figure 3 (a)
16	50	1511	371	2797.98	4.88%	Figure 3 (b)
16	50	1332	326	1965.31	3.31%	Figure 3 (c)

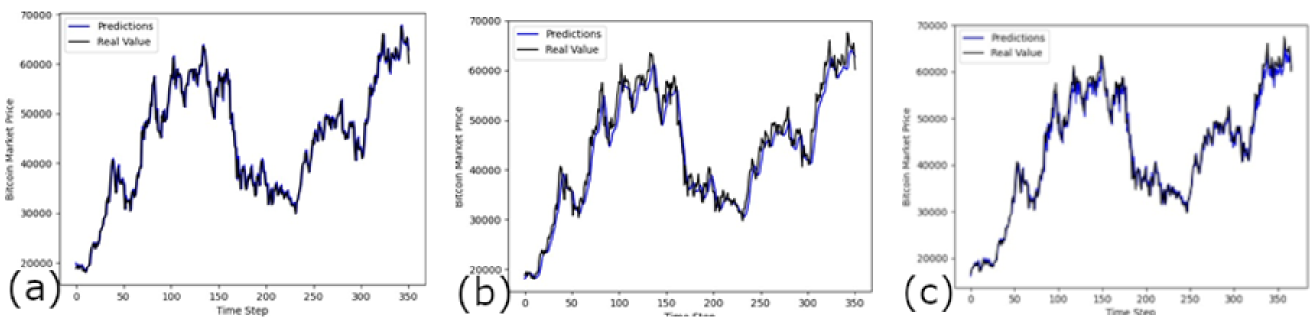

Figure 2: (a) Model with Batch Size = 16, Epochs = 50, Training length = 1432, Test length = 366, (b): Model with Batch Size = 16, Epochs = 10, Training length = 1432, Test length = 351, (c): Model with Batch Size = 1, Epochs = 50, Training length = 1492, Test length = 366.

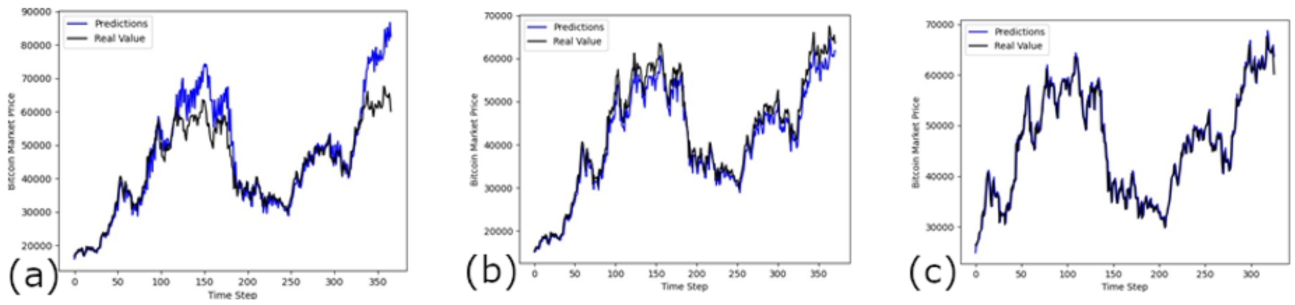

Figure 3: (a) Model with Batch Size = 1, Epochs = 70, Training length = 1492, Test length = 366, (b) Model with Batch size = 16, Epochs = 50, Training length = 1511, Test length = 371, (c): Model with Batch size = 16, Epochs = 50, Training length = 1332, Test length = 326.

With trends changing all the time, the most recent data will be the most accurate. This will require users to be more diligent when selecting the amount of data to use. The results here showed that it is difficult to train a model to predict the price accurately with bitcoin's volatility. The models trained here perform differently when presented with different data and hyperparameters. An optimal batch size of 16 was observed to have the most accurate and most stable results. Fifty epochs were observed to be the optimal number for the models. Too little, and the model wouldn't have enough time to train and thus wouldn't be as accurate. Too many and the model would overfit and not produce accurate results when presented with the test data. As initially thought, less data is better for this type of experiment. With bitcoin reaching new territory and all-time high price points, it is important to only train on the most recent data. The trends of previous data points are not the same as the most recent ones.

4 CONCLUSION

The world of predicting the stock market has been researched heavily with a vast array of different approaches. This research has not been translated to the cryptocurrency market yet. The LSTMs generate models with high accuracy (RMSE of 1900 with data ranging from \$20,000 to \$60,000). These models are trained only on the previous price of Bitcoin, with only looking at the previous ten prices. The models generated have a wide variety of hyperparameters used. These different hypermeters show which variables affect accuracy the most and introduce variance into the models. Different ranges of data also showed how those could affect the results. The approach here showed transferability between the stock market and Bitcoin when it comes to price prediction. With the effects of different hyperparameters and data lengths explored, they offer a good foundation for the future development of LSTMs.

REFERENCES

[1] "Algorithmic Trading Market | 2021 - 26 | Industry Share, Size, Growth - Mordor Intelligence." https://www.mordorintelligence.com/industry-reports/algorithmic-trading- market (accessed Nov. 22, 2021).
[2] "Definition of TIME SERIES." https://www.merriam-webster.com/dictionary/time+series (accessed Nov. 22, 2021).
[3] "7 Companies Where You Can Pay With Crypto," The Motley Fool, Oct. 05, 2021. https://www.fool.com/the-ascent/cryptocurrency/articles/7-companies-where-you-can-pay- with-crypto/ (accessed Nov. 22, 2021).
[4] "Bitcoin price history 2013-2021," Statista. https://www.statista.com/statistics/326707/bitcoin-price-index/ (accessed Nov. 22, 2021).
[5] R. Molla, "Elon Musk says Tesla will once again accept bitcoin," Vox, May 18, 2021. https://www.vox.com/recode/2021/5/18/22441831/elon-musk-bitcoin-dogecoin-crypto- prices-tesla (accessed Nov. 24, 2021).
[6] R. Browne, "Bitcoin and ether slide as China intensifies crackdown on cryptocurrencies," CNBC, Sep. 24, 2021. https://www.cnbc.com/2021/09/24/bitcoin-ethereum-sink-as-china- intensifies-crypto-crackdown.html (accessed Nov. 24, 2021).
[7] "Indian government set to ban cryptocurrencies," BBC News, Nov. 24, 2021. Accessed: Nov. 24, 2021. [Online]. Available: https://www.bbc.com/news/technology-59402310
[8] M. Roondiwala, H. Patel, and S. Varma, "Predicting Stock Prices Using LSTM," vol. 6, no. 4, p. 4, 2015.
[9] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, Dec. 2018, pp. 1394–1401. doi: 10.1109/ICMLA.2018.00227.
[10] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, Sep. 2017, pp. 1643–1647. doi: 10.1109/ICACCI.2017.8126078.
[11] J. Choi and B. Lee, "Combining LSTM Network Ensemble via Adaptive Weighting for Improved Time Series Forecasting," Mathematical Problems in Engineering, vol. 2018, pp. 1–8, Aug. 2018, doi: 10.1155/2018/2470171.
[12] Y. Zhu, "Stock price prediction using the RNN model," J. Phys.: Conf. Ser., vol. 1650, p. 032103, Oct. 2020, doi: 10.1088/1742-6596/1650/3/032103.
[13] X. Zhang, X. Liang, A. Zhiyuli, S. Zhang, R. Xu, and B. Wu, "AT-LSTM: An Attention- based LSTM Model for Financial Time Series Prediction," IOP Conf. Ser.: Mater. Sci. Eng., vol. 569, no. 5, p. 052037, Jul. 2019, doi: 10.1088/1757-899X/569/5/052037.
[14] A. H. Manurung, W. Budiharto, and H. Prabowo, "Algorithm and Modeling of Stock Prices Forecasting Based on Long Short-Term Memory (LSTM)." ICIC International 学会, 2018. Accessed: Nov. 22, 2021. [Online]. Available: https://doi.org/10.24507/icicel.12.12.1277
[15] Z. Zou and Z. Qu, "Using LSTM in Stock prediction and Quantitative Trading," p. 6.
[16] I. Jahan and S. Sajal, "Stock Price Prediction using Recurrent Neural Network (RNN) Algorithm on Time-Series Data," p. 6.
[17] "LSTM | Introduction to LSTM | Long Short Term Memor," Analytics Vidhya, Mar. 16, 2021. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term- memory-lstm/ (accessed Nov. 22, 2021).
[18] M. S. (Mady), "Chapter 10.1: DeepNLP — LSTM (Long Short Term Memory) Networks with Math.," Deep Math Machine learning.ai, Jan. 21, 2018. https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short- term-memory-networks-with-math-21477f8e4235 (accessed Nov. 22, 2021).
[19] M. Riva, "Understanding Forecast Accuracy: MAPE, WAPE, WMAPE | Baeldung on Computer Science," Sep. 12, 2020. https://www.baeldung.com/cs/mape-vs-wape-vs-wmape (accessed Nov. 28, 2021).
[20] "RMSE: Root Mean Square Error," Statistics How To. https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root- mean-square-error/ (accessed Nov. 28, 2021).