Predicting Cryptocurrency Price Movement with LSTM Modeling Across Time Steps

Spencer King*
sdk81722@uga.edu
University of Georgia
Athens, Georgia, USA

Preston Jamieson*
preston.jamieson@uga.edu
University of Georgia
Athens, Georgia, USA

ABSTRACT

As the adoption of cryptocurrency has grown in recent years, the desire to effectively predict cryptocurrency price movement has grown alongside it. This desire stems from a universal need by all investors to assess the expected investment return and its associated risk. This study aims to apply Long Short-Term Memory (LSTM) neural network modeling to cryptocurrency value assessment in hopes of finding a reliable way to predict crypto price movement.

LSTMs have proven effective for modeling time series data in other areas and industries [5], however, not much work has been done linking LSTMs to time series data regarding cryptocurrencies. This study aims to shed light on the LSTM's effectiveness on time series linked to cryptocurrency and will also compare the model's effectiveness on data collected over various time intervals: daily, hourly, and minutely.

Spencer King and Preston Jamieson. 2021. Predicting Cryptocurrency Price Movement with LSTM Modeling Across Time Steps.

1 INTRODUCTION

Cryptocurrencies have come a long way from their relatively obscure origins. While the mainstream financial world once rebuffed digital currencies as tools for criminals and speculators, the cryptocurrency industry has made significant progress in establishing itself as a potentially legitimate world-changing space. Over the past ten years, cryptocurrencies like Bitcoin (BTC) and Ethereum (ETH) have seen massive growth in user base and price. While they continue to grow larger and larger so does the fundamental question: what are cryptos actually worth?

To begin, it is necessary to have a basic understanding of what cryptocurrencies are. Cryptocurrencies are digital currencies that can be stored and transmitted as a collection of bytes. They are secured by cryptography and transacted on decentralized networks called blockchains. Blockchains are distributed ledgers (collections of financial accounting information) enforced by a disparate network of computers. Not only are the networks that cryptos transact on distributed, but the coins themselves are not generally issued by any central authority. In theory, this makes them impervious to manipulation and government interference.

In addition, cryptos are unlike most other assets in the fact that they have few metrics which can be reliably used to determine their monetary value. For instance, stock prices can be justified with balance sheets and cash flows from the underlying business. A home can be evaluated based on square footage, geographic location, and prices of comparable homes in the area. However, in the case of cryptos few of these reliable metrics exist. This ambiguity makes the intrinsic value of a particular cryptocurrency extremely difficult to calculate and often leads to dramatic volatility as price is left solely up to the whims of supply and demand. To make matters worse, cryptocurrencies are also heavily dependent on investor sentiment which is affected dramatically by things such as changes in current technology, institutional adoption, and government regulation. From the standpoint of the investor, this unpredictability is quite disagreeable.

2 PROBLEM ADDRESSED

As discussed above, cryptocurrencies are extremely difficult to evaluate. This is a problem for investors everywhere as investors need a level of certainty that the asset in which they are purchasing has a known or estimable probability of return. Maintaining confidence in asset costs allow investors to better reduce investment risks and more intelligently select portfolio allocations [3]. To solve this problem, an efficient methodology for evaluating cryptocurrency must be developed and to that end, this paper will focus on solving the aforementioned problem and a several others that stem from it.

3 SOLUTION

Throughout history when the underlying reason for an event or phenomenon was not understood, scientists and engineers have taken an empirical approach to let the data explain the reasoning. For example, when the Wright brothers were developing the first airplane in 1903, they did not understand the underlying science behind the concept of lift in aerodynamics. Nonetheless, they were able to quantify it by using a device called an airfoil to take empirical measurements that allowed them to quantify the concept. These measurements were critical to the first successful mechanized human flight. Similarly, the underlying reason for the movement in crypto asset prices is often unknown, however, by modeling the data associated with crypto assets, it may be possible to quantify their movements and predict their future price even if the reason for such movement is unclear.

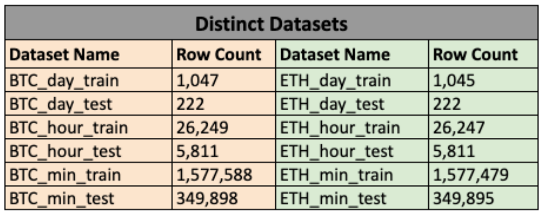
In the study, the modeling agent selected was the Long Short-Term Memory neural network. This particular neutral network has been deemed effective in modeling time series data which is the exact kind of data that crypto assets produce as they move through time [5]. This study aims to assess the effectiveness of LSTM modeling on time series data as it pertains to cryptocurrency. It seeks to determine this effectiveness by examining time series data of various time intervals (day, hour, and minute) across multiple crypto assets (Bitcoin and Ethereum).

4 DATA SETS

The data used in the study comes from an online data sharing platform called Kaggle [1]. The data is a collection of time series taken every minute for 14 different crypto assets that spans over 3 years with more than 24.2 million rows. The assets contained are Bitcoin Cash, Binance Coin, Bitcoin, EOS IO, Ethereum Classic, Ethereum, Litecoin, Monero, TRON, Stellar, Cardano, IOTA, Maker, and Dodgcoin. The two assets chosen for this study were Bitcoin and Etheruem as they are the two most well established coins by a significant margin at 9T and 5T market caps respectively.

Each of the chosen asset's data was then further partitioned into 3 distinct data sets: daily, hourly, and minutely. The daily and hourly data sets were not given. Rather, they were calculated based on the minute data and aggregated with respect to hours and days. For example, there are 60 minutes in an hour so 1 hour value would contain the sum of the 60 minutes before it.

Next, each asset and time interval pairing was divided into a training and test set. The training sets were approximately 3 years worth of data starting from midnight of 2018-01-01 and spanning until midnight of 2021-01-01. The training sets contained approximately 9 months worth of data starting from the midnight of 2021-01-01 and going until midnight of 2021-09-01. The 12 distinct data sets are shown below in Table 1.



Distinct Datasets			
Dataset Name	Row Count	Dataset Name	Row Count
BTC_day_train	1,047	ETH_day_train	1,045
BTC_day_test	222	ETH_day_test	222
BTC_hour_train	26,249	ETH_hour_train	26,247
BTC_hour_test	5,811	ETH_hour_test	5,811
BTC_min_train	1,577,588	ETH_min_train	1,577,479
BTC_min_test	349,898	ETH_min_test	349,895

Table 1) A list of all data sets and their row counts.

Finally, each of the 12 data sets above were processed in the following way. The original columns present were "timestamp", "Asset_ID", "Count", "Open", "High", "Low", "Close", "Volume", "VWAP", and "Target". "Asset_ID" is a foreign key designed to link the data to the asset name it corresponds to. "Count" refers to the number of trades made on the network of a particular asset. "Open" refers to the price of the first trade executed in a given time period. By the same logic, the "Close" price will be the price of the last trade executed in a given time period. One thing to note is that close and open prices do not have to be the same and often are not the same even at the minute interval because the prices are based on the underlying trade that is executed. Prices are discrete rather than continuous and based on the trades in the time interval, not the exact time itself in which it was executed. For example, if the minute time interval is examined, the price of the last trade in one minute might be \$5 while the price of the first trade in the next minute might be \$10. In this way, the close price of one interval does not necessarily equal the open price of the next. "High" and "Low" are the minimum and maximum prices that a particular asset experiences in a given time interval. "Volume" in this case refers to the quantity of the asset that is traded displayed in the equivalent amount of USD transacted.

$$\text{Volume (\$)} = \sum_{period}(traded\_volume \times price)$$

"VWAP" refers to volume weighted average price over the time interval. Finally, "Target" is a value unique to the original data set and will not be considered.

In this work, the original 10 columns have been reduced to the following 6: Count, Close, Volume, VWAP, DeltaPrice, and Zscore. Count, Close, Volume, and VWAP are the same as in their original form. DeltaPrice, or change in price, is a calculated value that is obtained by finding the difference between a particular closing price and the closing price directly preceding it. Finally, Zscore was calculated by finding the moving average from the close prices of 14 consecutive time periods, then determining the number of standard deviations the current closing price was away from that average. All other columns from the original data set were discarded.

At this point, it makes sense to discuss why this data was chosen. For the most part, traders (institutional and retail) evaluate most of these charting metrics when making decisions about investing in assets so it makes sense that we include some of the same metrics in our evaluation as well. This will allow our model to understand what other traders may be looking at and what exactly might be causing the price movement within a particular asset. First, many traders rely heavily on volume based metrics. For this reason, we have included Count and Volume as they are both directly related to the amount that an asset is traded although both measure slightly different things. Similarly, VWAP and Close price have been included. The VWAP is a volume related metric that is heavily used by institutional traders when trying to figure out entrance and exit strategies for a particular asset. For example, if the close price for a period was less than the VWAP then that might be a signal for an institutional trader to start to accumulate that asset. Next, many traders use some form of mean reversion strategy. A very common implementation of this is an indicator called Bollinger Bands. Bollinger Bands are a measure of standard deviation typically of the closing price with respect to a 14 day moving average of closing prices. The Zscore value above is a way to quantify the mean reversion strategy and allows our model to factor in this metric when training. Finally, change in price across periods is another big indicator for many traders. This is difficult to capture in a single price and instead is much more useful when viewed as change in price across time periods. For this reason, DeltaPrice has been included as input to the model.

Finally, before the data was input to the model, it was normalized. The normalization in this case is just a simple min-max scaling applied to each of the columns of data. Several normalization ranges were tried, but the range that gave the best results was from 0 to 1.

5 IMPLEMENTATION

As mentioned earlier, one of the goals of this study was to implement an LSTM model. Before diving into the implementation details, it is worth reviewing the inner workings of an LSTM. Shown below is a basic diagram of an LSTM.

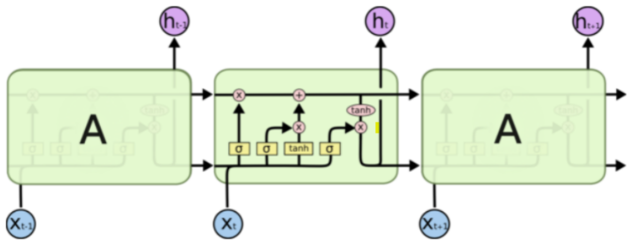


Figure 1) The inner workings of the LSTM.

LSTM, as previously mentioned, stands for Long Short-Term Memory. In Figure 1 above, the top horizontal line running through the diagram can be thought of as the long term memory or cell state [4]. The bottom part of the diagram can be thought of as the short term memory storing the input from the previous cell.

To start, the cell must decide what information to throw away. This decision is made by the sigmoid layer called the forget gate. Since the sigmoid function outputs either a 0 or 1, either the information is completely remembered or completely forgotten [4].

Next, the cell must decide what new information to store. This has two parts. First a sigmoid layer called the input gate decides which values will be updated. Then a tanh layer creates a vector of new candidate values that could be added to the state.

Now that the cell has decided what to do, it must actually implement it. This is done by multiplying by the old state then adding the product of the previous 2 steps. This is basically just a new scaled candidate value.

Finally, the cell needs to decide what information it is going to be output. This will be a filtered version of the cell state. First, the information will run through a sigmoid layer that will decide what parts of the cell state to output [4]. Then, similarly to before, the cell state will be run through a tanh function to produce candidates [4]. Then in taking the product of both of these, only the chosen candidates are output to the next cell.

Now that it is clear how the internal structure of an LSTM works, it makes sense to discuss the model architecture. The model consists of the following layers. Four stacked LSTM layers followed by a dense layer to output a vector of predicted values. It has been shown that 2 LSTM layers are sufficient to capture more complex features [2]. However, in this case since the data is very volatile and complex, the decision was made to include 2 extra layers (beyond the standard 2 layer model) to try to capture the complex nature of the data. The architecture is shown in Figure 2 below.

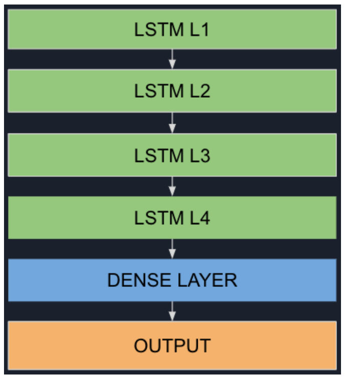


Figure 2) The structure of the model implemented in this study.

Now the model must be trained. For this study, there were 6 different training sets: BTC Day, BTC Hour, BTC Min, ETH Day, ETH Hour, and ETH Min. These sets correspond to a particular asset and a specified time interval as discussed previously in the Data Sets section. Additionally, for every training set, several hyperparameters were adjusted to optimize the performance and runtime of the model.

The first hyperparameter considered was window size. This is the amount of past data considered by the model. For example, if the window size was set to 30 for a particular training session, the model would only take into consideration the current and previous 30 time intervals in making predictions. For each time interval, the window sizes were selected based on what time intervals are customarily used by traders. For example, common time delineations for the daily time interval used by traders are 3 days, 7 days, 14 days, 30 days, and 60 days.

Next, the epochs and batch sizes were adjusted to try and strike the right balance between performance and runtime. Epochs are defined as the number of times the data is passed back and forth through the LSTM, and batch size is the total number of training examples run through a model at once. Both have a significant impact on the model, however, after extensive testing it was determined that 10 epochs and a batch size that separates the data into roughly 100 batches would be standard in this study.

Finally, each of the LSTM layers were optimized as well by adjusting the units and dropout. The units correspond to the number of neurons in each layer, and dropout corresponds to the percentage of neurons randomly ignored during training to help avoid overfitting. Again after thorough testing, 2 different configurations seemed to produce the best results and will be shown in the next section.

After each training session, the model was run on the associated test set. The resulting output (shown in orange in Figures 2 and 3) is a vector containing all predicted values. The predicted values in this case are a min-max scaled change in price. Before the predicted values are used they must be descaled and converted back into a price value. This process is shown below.

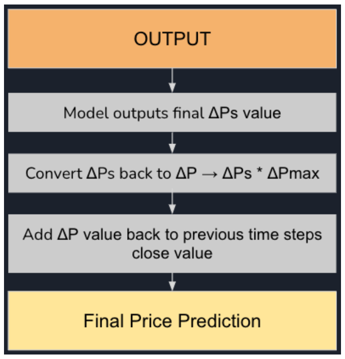

OUTPUT
Model outputs final ΔPs value
Convert ΔPs back to ΔP → ΔPs * ΔPmax
Add ΔP value back to previous time steps close value
Final Price Prediction

Figure 3) The conversion of scaled output into usable price values.

As shown above in Figure 3, scaled change in price, ΔP_s, is multiplied back by the absolute value of the maximum change in price observed to descale the output. Once the output is descaled, it is added back to the close price of the previous time step to get a final prediction price.

6 RESULTS

Before diving into results of the study, it is import to understand the evaluation metrics that are used. The model was evaluated on 3 different criteria: Root Mean Squared Error (RMSE), Mean Absolute Percent Error (MAPE), and directional accuracy. RMSE and MAPE are used to gauge how accurately the model can predict the magnitude of the price movement of an asset for the time interval it has been trained on [6]. Directional accuracy is a measure of how often the model correctly predicted the direction of the price movement. This was calculated by summing the total number of sign matches between the predicted values and the actual values, then dividing by the total number of predictions. The most important metric for this study will be directional accuracy, followed by the error metrics (RMSE and MAPE). This is because even if a prediction is wrong or significantly off in magnitude, as long as the predicted direction is correct the investor will still increase returns just not by the amount predicted.

Additionally, two main model configurations were used below. The first (denoted by the orange color in Tables 2-7 below) used fewer units in the first layer and employed a gradual increase in dropout percentage over the consecutive layers. The second (denoted by the green color in Table 2-7 below) used more units in the first layer and employed a consistent dropout percentage of 0.2 for each consecutive layer. These two configurations were chosen empirically as they produced the best and most consistent results.

Finally, the only hyperparameters adjusted besides the model configuration discussed above, were window size and batch size. Both were adjusted per the specific time interval.

6.1 Bitcoin

BTC DAY												
Test #	**Window Size**	**R1 Units**	**R1 Dropout**	**R2 Units**	**R2 Dropout**	**R3 Units**	**R3 Dropout**	**R4 Units**	**R4 Dropout**	**RMSE**	**MAPE**	**Directional Accuracy**
1	3	50	0.2	60	0.3	80	0.4	120	0.5	2024	3.354%	76.818%
2	7	50	0.2	60	0.3	80	0.4	120	0.5	1733	2.757%	79.545%
3	14	50	0.2	60	0.3	80	0.4	120	0.5	3014	4.791%	78.636%
4	30	50	0.2	60	0.3	80	0.4	120	0.5	2875	5.060%	77.727%
5	60	50	0.2	60	0.3	80	0.4	120	0.5	3178	5.859%	79.091%
6	3	60	0.2	60	0.2	80	0.2	120	0.2	2334	3.968%	77.273%
7	7	60	0.2	60	0.2	80	0.2	120	0.2	2087	3.512%	80.455%
8	14	60	0.2	60	0.2	80	0.2	120	0.2	2709	4.737%	79.091%
9	30	60	0.2	60	0.2	80	0.2	120	0.2	2625	4.597%	78.636%
10	60	60	0.2	60	0.2	80	0.2	120	0.2	2710	4.935%	78.636%

Table 2) This table shows the best test results for Bitcoin at the daily time interval. The following hyperparameters were used: Normalization Range: (0, 1), Epochs: 10, Batch Size: 10, Activation Function: Relu

BTC HOUR												
Test #	**Window size**	**R1 units**	**R1 dropout**	**R2 units**	**R2 dropout**	**R3 units**	**R3 dropout**	**R4 units**	**R4 dropout**	**RMSE**	**MAPE**	**Directional Accuracy**
1	4	50	0.2	60	0.3	80	0.4	120	0.5	369	0.537%	78.103%
2	9	50	0.2	60	0.3	80	0.4	120	0.5	367	0.471%	79.704%
3	12	50	0.2	60	0.3	80	0.4	120	0.5	385	0.572%	79.566%
4	**24**	**50**	**0.2**	**60**	**0.3**	**80**	**0.4**	**120**	**0.5**	**388**	**0.524%**	**80.737%**
5	48	50	0.2	60	0.3	80	0.4	120	0.5	447	0.576%	80.685%
6	4	60	0.2	60	0.2	80	0.2	120	0.2	353	0.489%	77.441%
7	9	60	0.2	60	0.2	80	0.2	120	0.2	365	0.516%	79.067%
8	12	60	0.2	60	0.2	80	0.2	120	0.2	372	0.534%	79.256%
9	24	60	0.2	60	0.2	80	0.2	120	0.2	436	0.734%	80.220%
10	48	60	0.2	60	0.2	80	0.2	120	0.2	398	0.571%	80.737%

Table 3) This table shows the best test results for Bitcoin at the hourly time interval. The following hyperparameters were used: Normalization Range: (0, 1), Epochs: 10, Batch Size: 260, Activation Function: Relu

BTC MINUTE												
Test #	**Window size**	**R1 units**	**R1 dropout**	**R2 units**	**R2 dropout**	**R3 units**	**R3 dropout**	**R4 units**	**R4 dropout**	**RMSE**	**MAPE**	**Directional Accuracy**
1	5	50	0.2	60	0.3	80	0.4	120	0.5	73	0.108%	73.436%
2	10	50	0.2	60	0.3	80	0.4	120	0.5	72	0.096%	74.561%
3	15	50	0.2	60	0.3	80	0.4	120	0.5	83	0.148%	75.058%
4	30	50	0.2	60	0.3	80	0.4	120	0.5	72	0.083%	75.879%
5	60	50	0.2	60	0.3	80	0.4	120	0.5	74	0.092%	74.311%
6	5	60	0.2	60	0.2	80	0.2	120	0.2	83	0.138%	72.060%
7	10	60	0.2	60	0.2	80	0.2	120	0.2	101	0.200%	73.203%
8	15	60	0.2	60	0.2	80	0.2	120	0.2	73	0.112%	73.290%
9	30	60	0.2	60	0.2	80	0.2	120	0.2	73	0.105%	74.713%
10	60	60	0.2	60	0.2	80	0.2	120	0.2	71	0.091%	75.081%

Table 4) This table shows the best test results for Bitcoin at the minutely time interval. The following hyperparameters were used: Normalization Range: (0, 1), Epochs: 10, Batch Size: 1577, Activation Function: Relu

Tables 2-4 above show the results pertaining to Bitcoin. Directional accuracy for BTC was best at the hourly time interval and worst at the minutely time interval. There was a consistent theme that as the time internal decreased, so did RMSE and MAPE. Finally, the difference in the two model configurations seemed to have little effect on the overall prediction results. The best run was Test 4 in Table 3. This run had the highest directional accuracy and relatively low RMSE and MAPE scores. The results of this run are shown graphically in Figures 4 and 5.

6.2 Ethereum

ETH DAY												
Test #	**Window Size**	**R1 Units**	**R1 Dropout**	**R2 Units**	**R2 Dropout**	**R3 Units**	**R3 Dropout**	**R4 Units**	**R4 Dropout**	**RMSE**	**MAPE**	**Directional Accuracy**
1	3	50	0.2	60	0.3	80	0.4	120	0.5	2061	3.400%	75.357%
2	7	50	0.2	60	0.3	80	0.4	120	0.5	1751	2.807%	78.289%
3	14	50	0.2	60	0.3	80	0.4	120	0.5	3047	4.874%	77.239%
4	30	50	0.2	60	0.3	80	0.4	120	0.5	2911	5.123%	76.627%
5	60	50	0.2	60	0.3	80	0.4	120	0.5	3223	5.963%	77.644%
6	3	60	0.2	60	0.2	80	0.2	120	0.2	2361	4.029%	75.956%
7	7	60	0.2	60	0.2	80	0.2	120	0.2	2119	3.549%	79.282%
8	14	60	0.2	60	0.2	80	0.2	120	0.2	2749	4.805%	77.645%
9	30	60	0.2	60	0.2	80	0.2	120	0.2	2668	4.667%	77.702%
10	60	60	0.2	60	0.2	80	0.2	120	0.2	2754	4.989%	77.813%

Table 5) This table shows the best test results for Ethereum at the daily time interval. The following hyperparameters were used: Normalization Range: (0, 1), Epochs: 10, Batch Size: 10, Activation Function: Relu

ETH HOUR												
Test #	**Window size**	**R1 units**	**R1 dropout**	**R2 units**	**R2 dropout**	**R3 units**	**R3 dropout**	**R4 units**	**R4 dropout**	**RMSE**	**MAPE**	**Directional Accuracy**
1	4	50	0.2	60	0.3	80	0.4	120	0.5	373	0.545%	76.897%
2	9	50	0.2	60	0.3	80	0.4	120	0.5	371	0.477%	78.365%
3	12	50	0.2	60	0.3	80	0.4	120	0.5	392	0.578%	78.593%
4	**24**	**50**	**0.2**	**60**	**0.3**	**80**	**0.4**	**120**	**0.5**	**392**	**0.531%**	**79.839%**
5	48	50	0.2	60	0.3	80	0.4	120	0.5	454	0.587%	79.325%
6	4	60	0.2	60	0.2	80	0.2	120	0.2	358	0.494%	76.044%
7	9	60	0.2	60	0.2	80	0.2	120	0.2	371	0.525%	77.996%
8	12	60	0.2	60	0.2	80	0.2	120	0.2	377	0.540%	78.222%
9	24	60	0.2	60	0.2	80	0.2	120	0.2	442	0.748%	78.758%
10	48	60	0.2	60	0.2	80	0.2	120	0.2	403	0.579%	79.831%

Table 6) This table shows the best test results for Etheruem at the hourly time interval. The following hyperparameters were used: Normalization Range: (0, 1), Epochs: 10, Batch Size: 260, Activation Function: Relu

ETH MINUTE												
Test #	**Window size**	**R1 units**	**R1 dropout**	**R2 units**	**R2 dropout**	**R3 units**	**R3 dropout**	**R4 units**	**R4 dropout**	**RMSE**	**MAPE**	**Directional Accuracy**
1	5	50	0.2	60	0.3	80	0.4	120	0.5	74	0.110%	72.605%
2	10	50	0.2	60	0.3	80	0.4	120	0.5	73	0.098%	73.751%
3	15	50	0.2	60	0.3	80	0.4	120	0.5	84	0.150%	73.672%
4	30	50	0.2	60	0.3	80	0.4	120	0.5	73	0.084%	74.829%
5	60	50	0.2	60	0.3	80	0.4	120	0.5	75	0.093%	73.275%
6	5	60	0.2	60	0.2	80	0.2	120	0.2	84	0.140%	71.103%
7	10	60	0.2	60	0.2	80	0.2	120	0.2	103	0.204%	72.041%
8	15	60	0.2	60	0.2	80	0.2	120	0.2	74	0.113%	71.956%
9	30	60	0.2	60	0.2	80	0.2	120	0.2	73	0.106%	73.608%
10	60	60	0.2	60	0.2	80	0.2	120	0.2	72	0.093%	74.012%

Table 7) This table shows the best test results for Ethereum at the minutely time interval. The following hyperparameters were used: Normalization Range: (0, 1), Epochs: 10, Batch Size: 1577, Activation Function: Relu

Tables 5-7 above show the results pertaining to Ethereum. Directional accuracy for ETH was also best at the hourly time interval and worst at the minutely time interval. There was a consistent theme that as the time interval decreased, RMSE and MAPE decreased as well. Finally, as with BTC, the difference in the two model configurations seemed to have little effect on the overall prediction results. The best run was Test 4 in Table 6. This run had the highest directional accuracy and relatively low RMSE and MAPE scores. The results of this run are shown graphically in Figures 6 and 7.

7 CONCLUSION & REFLECTION

This paper studied the application of LSTMs to crytocurrency time series data over various time intervals. The goal was to decide if

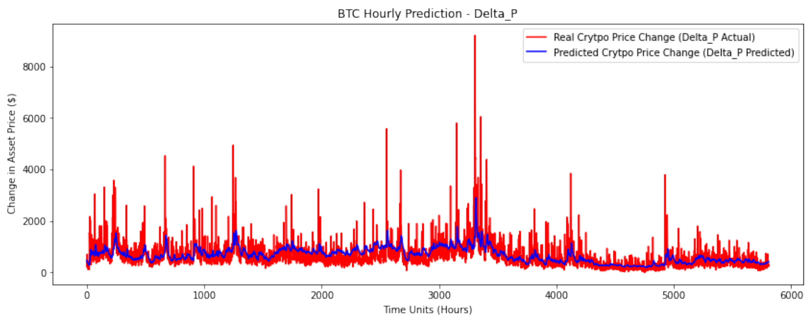


Figure 4) The predicted change in price and actual change in price of Bitcoin are shown above over the hourly time interval. This chart corresponds to the red bolded metrics in Table 3.

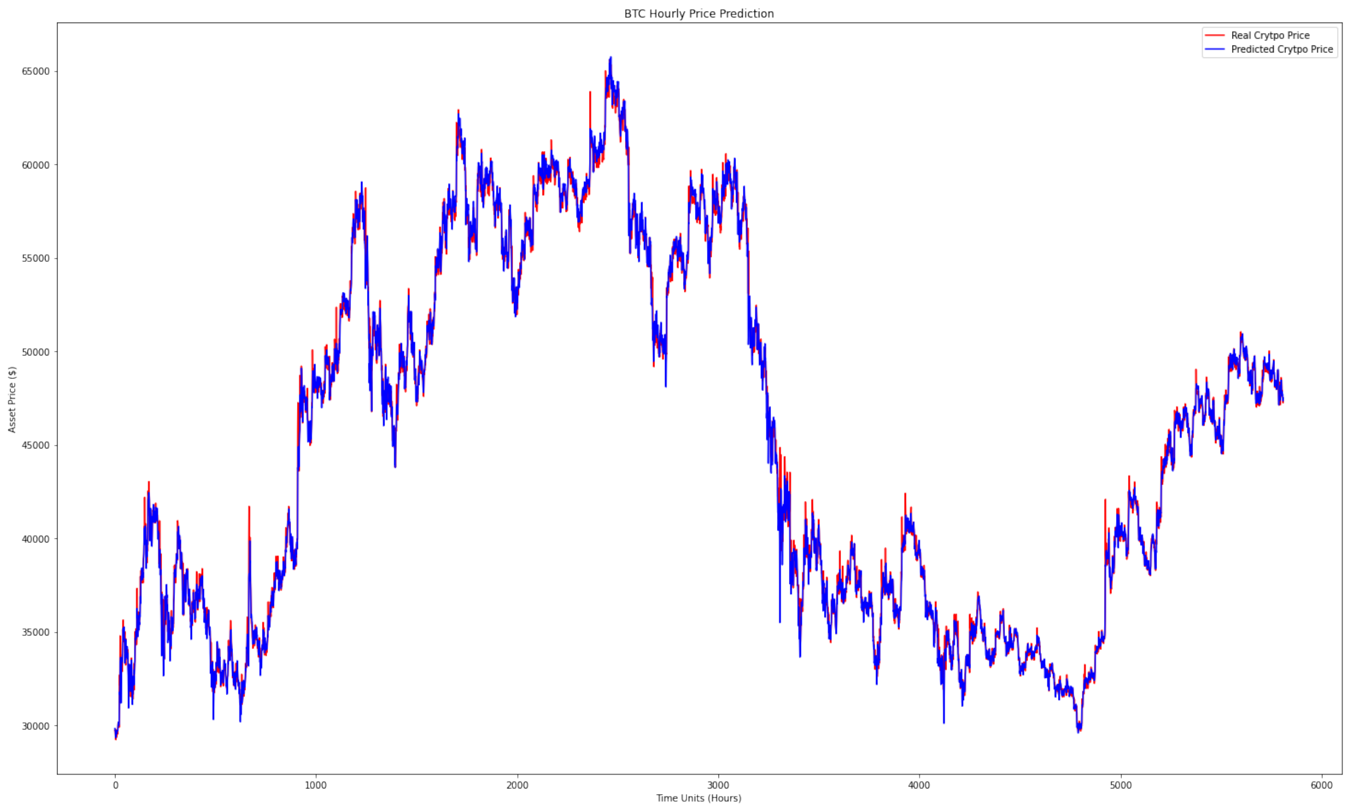


Figure 5) The predicted price and actual price of Bitcoin are shown above over the hourly time interval. This chart corresponds to the red bolded metrics in Table 3.

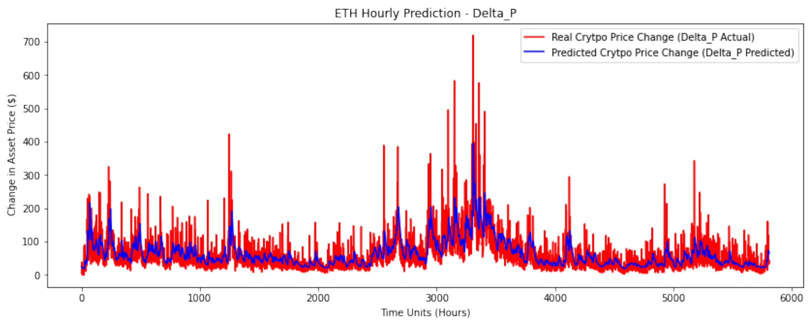


Figure 6) The predicted change in price and actual change in price of Ethereum are shown above over the hourly time interval. This chart corresponds to the red bolded metrics in Table 6.

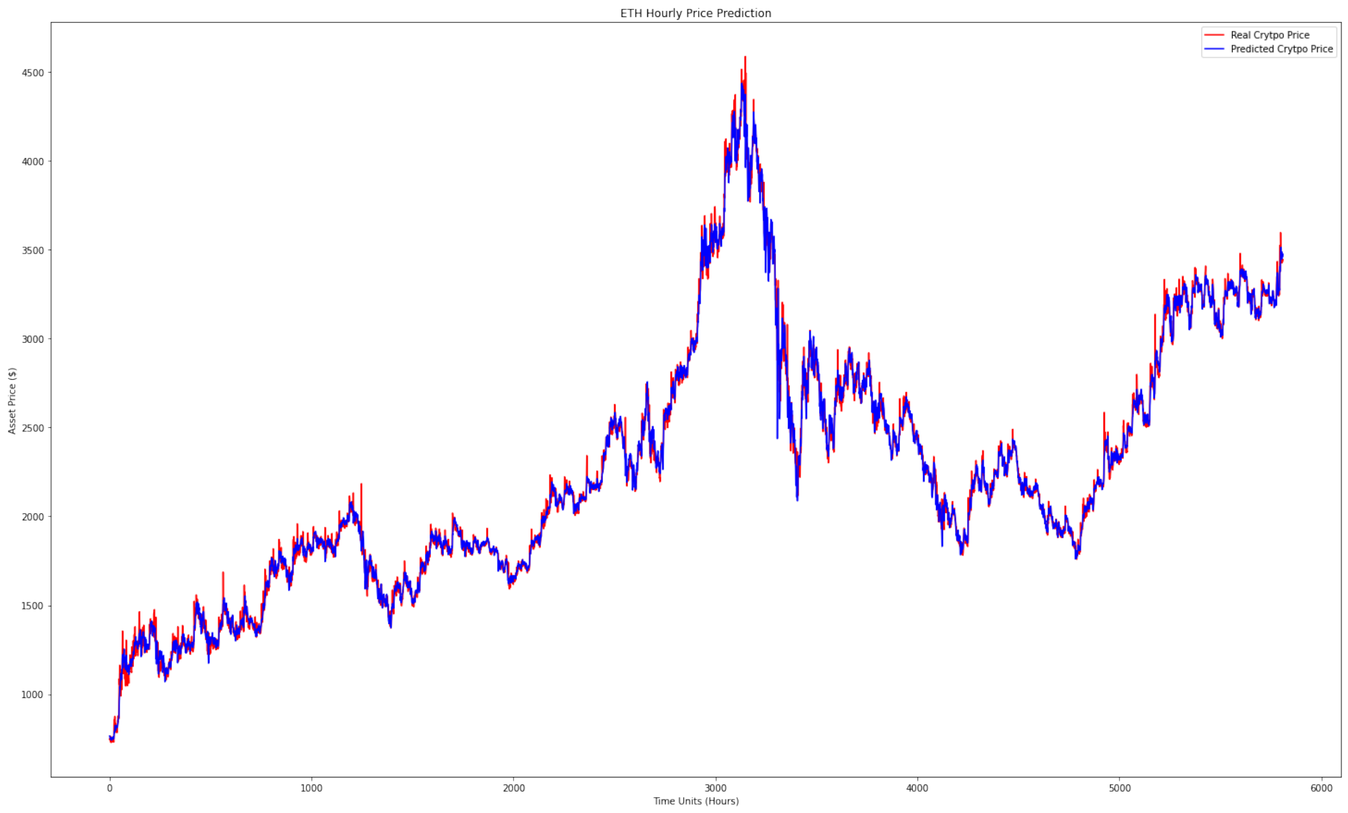


Figure 7) The predicted price and actual price of Ethereum are shown above over the hourly time interval. This chart corresponds to the red bolded metrics in Table 6.

the LSTM is a good choice for the modeling and prediction of price moments in the crypto industry, as well as pinpointing what time intervals best served by this model.

The model performed best at the hourly time interval for both Bitcoin and Ethereum. It performed slightly better with Bitcoin (average directional accuracy of 77.43%) than Ethereum (average directional accuracy of 76.63%). One theory for this is the fact that Bitcoin is slightly less volatile than Etheruem. The LSTM model also had significantly less error at the lower time intervals. However, this is most likely due to the fact that at the smaller time intervals significantly more data was available. Also, volatility is normally lower over smaller time intervals.

After the full evaluation, it is fair to say that the LSTM model works well in predicting price for two of the largest cryptocurrencies (Bitcoin and Ethereum) across multiple time intervals. The average directional accuracy across all tests was 77.03%. This is extremely good as most traders would be happy to get accuracy percentages over 50%. Finally, even in the worst case actual predicted values only had an error of 5.96%.

Even though the results are promising and the study has proven that LSTMs can be effective in predicting crypto price movement across multiple time intervals, there is still a long way to go before this could actually be implemented in real life. The model must be generalized further so that it can be applied to all cryptocurrencies and avoid overfitting. It would also need further optimization with extensive testing, thorough back testing, and implementation on a real-time basis. Finally, since the price movement of cryptocurrencies is heavily dependent on investor sentiment, adding in features to allow the model to account for sentiment would likely improve the overall model accuracy [3].

REFERENCES

[1] 2021. *G-Research Crypto Forecasting | Kaggle.* https://www.kaggle.com/c/g-research-crypto-forecasting
[2] Karsten Eckhardt. 2018. Choosing the right Hyperparameters for a simple LSTM using Keras. https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046#:~:text=Generally,2layershaveshown,tofindreasonablycomplexfeatures.
[3] Ayman E Khedr, Nagwa Yaseen, et al. 2017. Predicting stock market behavior using data mining technique and news sentiment analysis. *International Journal of Intelligent Systems and Applications* 9, 7 (2017), 22.
[4] Adil Moghar and Mhamed Hamiche. 2020. Stock Market Prediction Using LSTM Recurrent Neural Network. *Procedia Computer Science* 170 (2020), 1168–1173. https://doi.org/10.1016/j.procs.2020.03.049 The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.
[5] Pham Ngoc Hai, Nguyen Manh Tien, Hoang Trung Hieu, Pham Quoc Chung, Nguyen Thanh Son, Pham Ngoc Ha, and Ngo Tung Son. 2020. An Empirical Research on the Effectiveness of Different LSTM Architectures on Vietnamese Stock Market. In *2020 International Conference on Control, Robotics and Intelligent System.* 144–149.
[6] Murtaza Roondiwala, Harshal Patel, and Shraddha Varma. 2017. Predicting stock prices using LSTM. *International Journal of Science and Research (IJSR)* 6, 4 (2017), 1754–1756.