High-frequency Trend Prediction of Bitcoin Exchange Rates Using Technical Indicators and Deep Learning Architectures

Hector Villafuerte
hvillafuerte3@gatech.edu

Thomas Passaro
tpassaro3@gatech.edu

Yichao Zhang
yzhang3414@gatech.edu

Zach Reeve
zreeve3@gatech.edu

Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332

Abstract

We explored the application of four different deep learning architectures: Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM), Convolution Neural Network (CNN), and a CNN-LSTM combination to predict the price trend direction of the next minute's closing price using 18 technical indicators represented as a 2D image-like matrix. These indicators were calculated using pricing data over a High-Frequency time frame of one-minute intervals.

The MLP, LSTM, and CNN-LSTM models when tuned and trained, resulted in a better prediction accuracy versus 2 out of 3 of our reference papers. Our results demonstrated the effectiveness of LSTM archictectures on financial prediction tasks as well as the relative effectiveness of a simple Deep MLP architecture. These results might be applicable to a short term time frame trading-algorithm considering the high volatility of the cryptocurrency exchanges in intraday time-frames.

1. Introduction/Background/Motivation

Predictive models in finance is a continuously improving field as people and companies try to incorporate the relentless advancements in AI and Machine Learning into their models. This field presents several challenges particularly in the areas of noisy time-series data and the inherent unpredictability of the market.

Predictive models in cryptocurrency is a newer and growing field; most research in this field has been conducted since 2018. To the best of our knowledge, Lahmiri and Bekiros [11] published the first predictive price model for cryptocurrencies in January 2018. They compared several models for different cryptocurrencies and found that the LSTM was the most accurate model. This was also the first publication that we have found which applies LSTM's to cryptocurrency prediction. Also in 2018, a model was proposed using a 7-layer fully-connected neural network which found to outperform a simple "buy-and-hold" strategy [13]. In September 2019, high-dimensional technical indicators were used successfully to categorize bitcoin price movements into percent-change ranges [5]. The challenges in predicting cryptocurrency price movements are the similar to the challenges posed by stock price prediction algorithms, however cryptocurrencies' can be plagued with more noise, drastic price movements, and inconsistent volumes.

The scope of this project is to build and evaluate various models to predict the direction of the next minute's closing price in a high frequency environment. Our objective is to get a good classification measure comparable to the most recent papers published about Bitcoin trend prediction. This research also focuses on the implementation and comparison of neural network architectures to better understand and evaluate their performances compared to each other in predictive tasks on cryptocurrency pricing data. Our work is based off of and expands upon the work of Alonso-Monsalve *et al*. (2020) [1]. We compare and evaluate their Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN) and CLSTM (which we called CNN-LSTM for clarity) architectures. We have modified the architectures set forth by Alonso-Monsalve *et al*. to obtain higher performance in our experiments and additionally implemented and evaluated a vanilla LSTM for comparison.

If this project succeeds, providing reliable predictions, it can open the possibility of implementation with a complete trading algorithm of buy and sell signals including integration into a trading system and money management

strategies.

The market cap of cryptocurrencies exceeds $3 trillion and cryptocurrencies are a rapidly growing asset with many technological and financial applications. High-Frequency Trading (HFT) can play an important part in regulating these markets by reducing volatility and increasing liquidity [7]. Additionally, there are much broader implications to any architectural discoveries made in this research. CNN-LSTM's have been used for a variety of different applications from Image Captioning [2] to Covid X-Ray diagnostics [6] and breakthroughs in one area can potentially be applied to any other.

2. Data Processing

2.1. Data Pre-Processing

The time series pricing data was acquired from Bitstamp, which collects cryptocurrencies pricing data from the exchanges. The OHLC (Open/High/Low/Close) pricing data used in this project uses 1-minute bar data. We selected Bitcoin as most popular cryptocurrency for this project. The data sampled ranges from 1st of January 2020 to September 30th 2020 with a total of 391,681 records. This sample is enough to represent differen different upward and downward trends that will help to train a model to be applicable in a short-term.

Using the raw pricing data, for each minute interval, we compute various technical indicators, shown in Fig 1. These are commonly used in High-Frequency trading [9]. We used the Technical Analysis Library in Python to generate the technical indicator features. All the data was normalized using z-score normalization after the indicators were calculated.

Indicator	Formula
A/D	$\frac{H_t - C_{t-1}}{H_t - L_t}$
CCI	$\frac{M_t - SM_t}{0.015 D_t}$
LWR	$\frac{H_n - C_t}{H_n - L_n} \times 100$
MACD	$MACD(n)_{t-1} + 2/n + 1 \times (DF_t - MACD(n)_{t-1})$
Momentum	$C_t - C_{t-n}$
RSI	$100 - \frac{100}{1+(\sum_{i=0}^{n-1} Up_{t-i}/n)/(\sum_{i=0}^{n-1} Dw_{t-i}/n)}$
SMA (5,10,20,30,60)	$\frac{C_t + C_{t-1} + \ldots + C_{t-n+1}}{n}$
SD	$\frac{\sum_{i=0}^{n-1} K_{t-i}\%}{n}$
SK	$\frac{C_t - LL_{t-n}}{HH_{t-n} - LL_{t-n}} \times 100$
WMA (5,10,20,30,60)	$\frac{n \times C_t + (n-1) \times C_{t-1} + \ldots + C_{t-n+1}}{n + (n-1) + \ldots + 1}$

C_t: closing price; L_t: lowest price; H_t: highest price at time t; DF: $EMA(12)_t - EMA(26)_t$; EMA: Exponential moving average; $EMA(k)_t$: $EMA(k)_{t-1} + \alpha \times (c_t - EMA(k)_{t-1})$; α: smoothing factor: $2/1+k$; k: time period of k minute exponential moving average; LL_t and HH_t: mean lowest low and highest high in the last t minutes; $M_t : H_t + L_t + C_t/3$; $SM_t : \sum_{i=1}^{n} M_{t-i+1})/n$; $D_t : (\sum_{i=1}^{n} |M_{t-i+1} - SM_t|)/n$; Up_t: upward price change; Dw_t: downward price change at time t.

Figure 1. Technical Indicators used as features to generate image-like data. Formulas as reported in Kara *et al*. [9]

The minute level data is converted to a 2-dimensional matrix where each row is a point in time and each column is a technical indicator. We use a time window of 15 minutes and a total of 18 indicators per sample. The figure 2 shows a sample of the image-like data for sequential images. The label or "Trend" is calculated based on the price in the next minute after last time step within the image. If the price for the next minute goes up, then the value is 1. Otherwise, the trend label value is 0.

	I1	I2	I3	I4	I5	I6	Trend
T1							1
T2							0
T3							1
T4							1
T5							0
T6							1
T7							0
T8							0
T9							1
T10							0

	I1	I2	I3	I4	I5	I6	Trend
T1							1
T2							0
T3							1
T4							1
T5							0
T6							1
T7							0
T8							0
T9							1
T10							0

Figure 2. Example of sequential window-like 2D images. This sample uses a sequence of 7 minutes x 6 features or technical indicators.

2.2. Dataset Separation

After generating the 2-D image-like dataset, we separated it into three sets: Training (70%), Validation (15%) and Testing (15%) for independent cross validation [4]. In order to achieve more representative datasets [15], we split the data into 2 windows, and each window was split into training, validation, and testing. Figure 3 shows the cross validation dataset splits into 3 windows for illustration purposes, however it is important to note that our dataset was split into 2 not 3 windows.

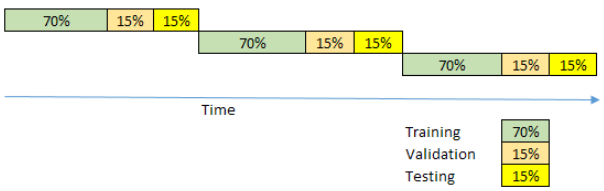

Figure 3. Data partition of the sample into training, validation and testing.

We checked the distribution of the calculated labels and

it is well balanced as shown in the table 1.

Dataset	Label	Bitcoin Baseline
Train	1	48.64%
	0	51.36%
Validation	1	48.95%
	0	51.05%
Test	1	50.0%
	0	50.0%

Table 1. Breakdown of 2-D image-like patterns classes for training, validation and testing set. The patterns computed at 1-minute interval for the period Q1 2020 to Q3 2020.

3. Approach

3.1. Approach Overview

What did you do exactly? How did you solve the problem? Why did you think it would be successful? Is anything new in your approach?

Using the same dataset, we trained four separate models: An MLP, LSTM, CNN, and CNN-LSTM. The variety of architectures enabled us to evaluate multiple different models for effectiveness, helping ensure that at least one achieved satisfactory results. Additionally, this approach gave us significantly more insight into the architectures themselves and the benefits and drawbacks of each. As mentioned previously, this work was based off of and inspired by the work of Alonso-Monsalve *et al*. (2020) [1]. To expand upon their work we implemented an LSTM, which was a critical addition as this model actually performed the best. We also made several changes to the CNN-LSTM architecture, which is elaborated on in **Section 4.4**.

All models end in a Sigmoid activation function and return a scalar output between 0 and 1. This number represents the probability that the "Trend" is 1, meaning a price movement upwards next minute. The output is rounded and the model is correct if the rounded output equals the "Trend" for the next minute.

All models used a Binary Cross-Entropy Loss Function and an Adam optimizer. We chose to use an Adam optimizer because we anticipated there to be some noise in the data. Adam maintains a decaying average of the previous gradients, which factors into the update step. This helps smooth the updates and reduce the effect of noisy gradients. Additionally, we were able to utilize the weight-decay parameter for the Adam optimizer, which is similar to L2 regularization in Stochastic Gradient Descent. This significantly reduced over-fitting in the CNN-LSTM and LSTM models.

The architecture-specific and implementation details of each model are further described in **Section 4**.

3.2. Challenges

There were several challenges that we had to overcome in order to successfully train these models. One problem that we anticipated was noise in the data. The cryptocurrency market is prone to high volatility and sharp price movements. Therefore, we decided to use technical indicators instead of raw pricing data as the inputs to our model to reduce noise and smooth price curves [14]. Additionally, a relatively unique challenge to cryptocurrency pricing data is that the price increases/decreases over time can be very large. This may present challenges when normalizing data over long periods of time. In our case, mostly all price-related data, such as moving averages, had negative z-scores at the beginning of the time frame, while the data at the latter end of the time frame had positive z-scores. We debated and experimented with sigmoid, tanh estimator, and min/max scaling normalization methods in order to determine the most suitable method for this project. These all had equivalent to or worse performance than z-score normalization. Additionally, much of the research that we conducted used z-scores. Since the model should mainly be focused on the change in time-series data rather than the number itself, we concluded that it would not impact the performance of the model to use z-score normalization.

Another issue that we encountered was that training these models was quite difficult. Small changes to hyper parameters could have large impacts in the results, including the model getting stuck in local minima where it chooses either "1" or "0" for all values. This is something that had to be trained around using carefully chosen learning rates and regularization. An example of this was the weight-decay in the Adam optimizer. While this reduced over-fitting in our model, larger values resulted in the model predicting all zeros. Other papers which ran into over-fitting issues like [10] and [1], used higher dropout rates and other techniques, which severely impacted our model.

4. Experiments and Results:

All experiments were performed using the Pytorch Framework in Google Colab Notebooks running a GPU and High RAM configurations.

We used testing accuracy average to quantitatively measure the results of our experimentation. We additionally calculated the loss and the Confusion matrix for each model to ensure that the model was training properly. We assessed the Loss and Accuracy curves throughout the training process to tune the hyperparameters, looking for issues such as over-fitting or the model not learning quickly enough. The Loss and Accuracy curves of each model can be found in **Appendix A** and the Confusion matrix for each model can be found in **Appendix C**.

All models used a batch size of 100, were trained for

40 epochs, and had the following hyperparameters tuned: learning rate, dropout, and Adam weight-decay. The CNN-LSTM was also tuned for the kernel size, number of convolutions, and number of fully connected layers. All models used a Binary Cross-Entropy Loss Function and an Adam optimizer.

The hyperparameters and other variables mentioned above are not learnable parameters and were tuned before the start of each trial run. The learnable parameters for the models were the bias units, weights in a fully-connected layer, the weight matrix in an LSTM, and the kernel weights in each convolution.

4.1. Experiments and Results: MLP

We trained an MLP model with 2 hidden layers, where each hidden layer has 64 nodes, followed by a Relu activation and a dropout layer with rate 0.2. Then it pass through the output layer with 1 output dimension and followed by a sigmoid layer. As shown in Figure 4.

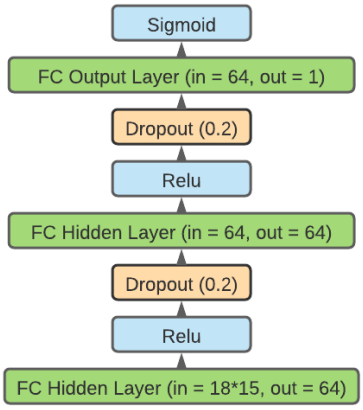

Figure 4. MLP architecture diagram.

We used cross entropy as our loss function, and used Adam optimizer to optimize the MLP model. We used accuracy and confusion matrix to evaluate the model's performance.

During the training, the batch size is 100, and the learning rate is initialised as 0.0001, with 0.001 weight decay. We trained the model on the training dataset for 40 epochs. In each epoch, we evaluated the model on the validation dataset. Then we save the model with the best validation performance.

Figure 10 in **Appendix A** shows the learning curves of the MLP model. Both training curves look good, but the validation curves are oscillated. The highest accuracy is 58.38% on the validation dataset, and is 57.69% on the testing dataset. It is much higher than the MLP model in [1], and is competitive with their LSTM and CLSTM models.

Table 4 in **Appendix C** shows the normalized confusion matrix on the testing dataset. We can see that our MLP mode correctly predict 62.16% of negative samples, and 53.14% of positive samples. Our MLP works better on the falling market.

4.2. Experiments and Results: LSTM

We obtained an average testing accuracy of 57.46%, which is a measure of success compared with the accuracy obtained in [8] for similar architectures. It also surpasses the baseline accuracy given in Table 1.

Experimentation begun with a simple LSTM architecture to start different hyper-parameters and architectures. Batch normalization layers were added to the model, even though it did not improve the accuracy in the validation nor testing sets. Another experiment included additional LSTM layers, the best results were given when adding two LSTM layers. The optimization method selected is Adam with a learning rate of 0.001 and weight decay of 0.001.
The height of the image was another parameter that we could change. The height or time length of 15 was used for the training and higher values of 30 or 60 did not improve the measures. The figure 5 shows the final network architecture configured for the maximum average testing accuracy.

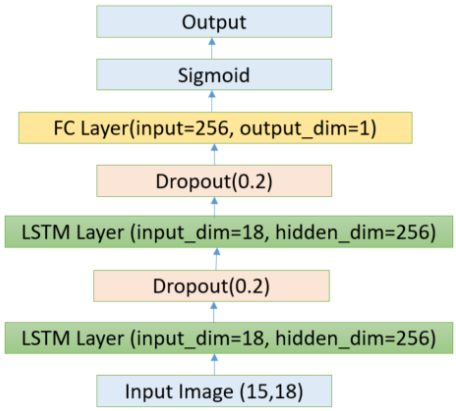

Figure 5. Selected LSTM architecture to predict Bitcoin Trends.

Figure 11 in **Appendix A** show the loss and accuracy curves generated by the training with the best hyperparameter configuration.

4.3. Experiments and Results: CNN

Using CNN layers with no memory or recurrent layers and only considering the current time's financial indicators did not yield results above the benchmark. The CNN model was only able to achieve an average testing accuracy of 51.14%. Qualitatively, the Loss and Accuracy curves 12

show that the model was not learning very well. Additionally, the accuracy of the validation set was decreasing while the training accuracy was increasing, which suggests that this model was over-fitting.

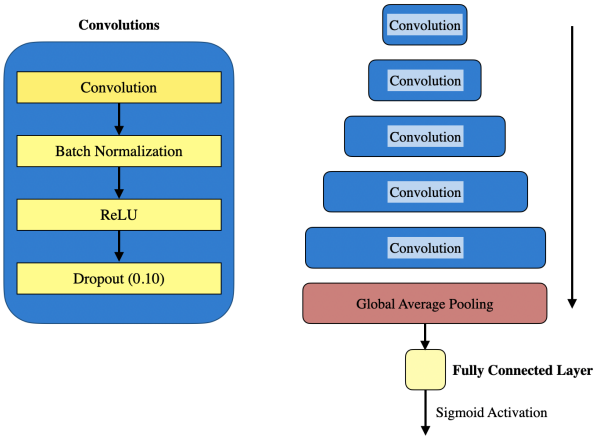

Figure 6. CNN Architecture Diagram.

Kernel Size	Channels
1 x 4	16
4 x 1	16
1 x 4	32
4 x 1	64
1x4	80

Figure 7. CNN Dimensions and Kernel Sizes.

While different hyper-parameters led to different loss and accuracy curves the accuracy never eclipsed the benchmark. Most experiments yielded the best validation loss very early in the training, specifically before the 6th epoch, after which the training and validation loss would diverge. This over-fitting was very difficult to overcome. Different dropout strategies, learning rates, and regularizations would help with the problem of over-fitting, however they did not help with the overall performance of the models accuracy. From this we can derive the outcome that this CNN model with no recurrence was not a good model for Bitcoin price prediction. The shortcomings of our CNN implementation are further elaborated upon in the **Conclusions**.

4.4. Experiments and Results: CNN-LSTM

While this model was inspired by the CNN-LSTM model described by Alonso-Monsalvo *et al*. [1], we made several modifications which we found to increase its performance over our implementation of their original model. First, we used vertical and horizontal kernels instead of the 1x1 kernels, used in their paper. Alonso-Monsalvo *et al*. used these filters for their vanilla CNN implementation, however due to resource restrictions, did not use it in their hybrid model. We thought these kernels would add more utility to the CNN portion of the architecture and that the horizontal kernels would capture multi-indicator relationships that the LSTM would not be as adept at identifying. Secondly, we used a 2-layer LSTM, similar to [15], which we found to be more effective than the original 1-layer LSTM. Using two layers allows the model to more accurately identify temporal features. Lastly, we reduced the number of fully-connected layers from 8 (excluding the output layer) to 3. This was done to help combat over-fitting in the model. The architecture used for the final results of the CNN-LSTM is shown in Figure 8 and the configuration is shown in Figure 9.

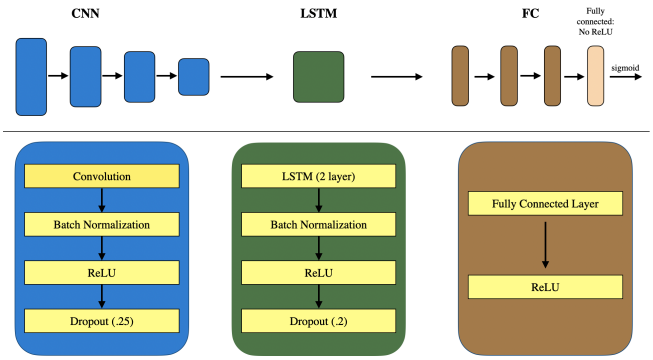

Figure 8. CNN-LSTM Architecture Diagram.

		CNN		FC
		Kernel Size	Channels	Nodes
Image Size	15 x 18	1 x 4	64	200
		4 x 1	32	100
LSTM Hidden Size	150	1 x 4	8	100
		1 x 1	1	1

Figure 9. CNN-LSTM Dimensions and Kernel Sizes.

Quantitatively, it returned a test accuracy of 57.3% which is quite similar to the accuracy of the vanilla LSTM. We found the accuracy of the CNN-LSTM to be comparable to other results conducted on these architectures for cryptocurrency prediction. Our results were even able to outperform most studies in this field. The three studies we found with similar experimentation reported accuracies of 61%, 55%, and 51% by Alonso-Monsalvo *et al*. [1], Livieris *et al*. [12], and Qiang and Shen [15], respectively. While these accuracies may seem low compared to the accuracy of deep neural networks in other tasks, these accuracies are expected and reasonable for predicative

	Average Testing Accuracy (Bitcoin)
MLP	57.84%
LSTM	57.55%
CNN	51.14%
CNN-LSTM	57.29%

Table 2. Average Testing Accuracy comparison for the best models trained for different deep learning architectures.

price tasks. The market is extremely unpredictable and a trader making the correct decision 60% of the time will be very profitable long-term. Qiang and Shen, whose model had a 51% accuracy still found that their model outperformed passive investing [15].

This model was prone to over-fitting and although we achieved high-training accuracy early in our experiments, the model required a lot of regularization testing to achieve a satisfactory validation and test accuracy. Batch normalization, dropout, and Adam weight-decay were used to discourage over-fitting. Separate optimizers were used for each piece of this architecture in order to finer tune hyperparameters. Another method to reduce over-fitting which was out of the scope of this project but could yield interesting results is data augmentation. As far as we know, this has not been utilized in any price prediction models. We believe this could be beneficial as data augmentation may force the model to learn different features in the model and allow the model to generalize better to the unforeseen market scenarios of the future. Data augmentations that came to mind were cutout and uniform price adjustments (similar to color-jitter in an image). These specifically may be applicable to market data as cutout would encourage the model to train on more parts of the image (and therefore more indicators), and price adjustments could make the model more robust to potentially unseen prices of the future, such as record highs or record lows. These of course would need to be tested, along with other augmentations, to determine the validity of this strategy.

Model	Predicted	Distribution
MPL	1	45.42%
	0	54.58%
LSTM	1	43.57%
	0	56.43%
CNN	1	33.91%
	0	66.09%
CNN-LSTM	1	45.75%
	0	54.25%

Table 3. Distribution of model predictions applied to testing samples.

5. Conclusions

Our results show that maintaining a lookback period of time-series data and generation of features using technical indicators can lead to positive results. We were able to train multiple models which outperformed models from similar studies by using technical indicators, purposefully selected training procedures and tools, and careful hyperparameter tuning. The small percentages (2+%) in accuracy can represent the difference between a successful and unsuccessful trading strategy.

The concept of treating this time series data like images opens the door for a lot of image-related techniques which can be applied to time-series data. Examples of this being image-related architecture or data augmentation techniques. We believe this overlap between data types and techniques can have many novel applications in a number of different time-series related tasks, one of which is financial predictions.

All models, except the CNN, were able to learn independent of the label distribution. The Confusion Matrices in **Appendix C** (Tables 4, 5, 6, and 5) show the distribution of labels predicted by our models when applying on testing samples. We can verify that the successfully trained models are not predicting zero-only or one-only values. Instead, they are able to recognize patterns and predict different trends. Given the distribution of these predictions, it gives us the certainty that the models are identifying patterns in the testing data.

We think the LSTM performed well due to its temporal memory and its uses as a predictive model in finance and other fields [3]. While the CNN-LSTM architecture performed similarly to the LSTM, we believe that it has the potential to outperform a vanilla LSTM but those performance improvements would require much more research.

We do not believe we have utilized the CNN to its fullest potential yet. The ordering of columns and data augmentation are two areas that we feel can be further studied and optimized to maximize the efficacy of the CNN portion. We believe that the CNN may have performed poorly due to non-optimal configurations or not enough temporal focus, which can be corrected via different architectures. This is an area that may require extensive experimentation but a CNN may be a beneficial piece to a robust predicative neural network architecture. That being said, we do not believe a CNN by itself would outperform an LSTM, even with the proper configurations. All models were trained using a short-term date range to predict the next minute price movement. The dynamics, volatility, and liquidity of the cryptocurrency markets can limit the implementation or set of production of similar models.

References

[1] Saúl Alonso-Monsalve, Andrés L. Suárez-Cetrulo, Alejandro Cervantes, and David Quintana. Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems With Applications*, 149:1–15, 2020. 1, 3, 4, 5

[2] Jiuxiang Gu et al. An empirical study of language cnn for image captioning. pages 1222–1231. IEEE International Conference on Computer Vision, 2017. 2

[3] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018. 6

[4] Helder Sebastião Pedro Godinho. Forecasting and trading cryptocurrencies with machine learning under changing market conditions. *SpringerOpen - Open Access*, 7(3):12–14, 2021. 2

[5] Jing-Zhi Huang, William Huang, and Jun Ni. Predicting bitcoin returns using high-dimensional technical indicators. *The Journal of Finance and Data Science*, 5(3):140–155, 2019. 1

[6] Zabirul Islam, Milon Islam, and Amanullah Asraf. A combined deep cnn-lstm network for the detection of novel coronavirus (covid-19) using x-ray images. *Elsevier Public Health Emergency Collection*, 20, 2020. 2

[7] Archana Jain, Chinmay Jain, and Christine X. Jiang. Active trading in etfs: The role of high-frequency algorithmic trading. *Financial Analysts Journal*, 77(3):66–82, 2021. 2

[8] Patrick Jaquart, David Dann, and Christof Weinhardt. Short-term bitcoin market prediction via machine learning. *ScienceDirect - The Journal of Finance and Data Science*, 7:45–66, 2021. 4

[9] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert Systems With Applications*, 38:5311–5319, 2011. 2

[10] Christopher Krauss, Xuan Anh Do, and Nicolas Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the sp 500. *European Journal of Operational Research*, 259(2):689–702, 2017. 3

[11] Salim Lahmiri and Stelios Bekiros. Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons, and Fractals*, 118:35–40, 2019. 1

[12] Ioannis E. Livieris, Niki Kiriakidou, Stavros Stavroyiannis, and Panagiotis Pintelas. An advanced cnn-lstm model for cryptocurrency forecasting. *Electronics*, 10(287):11–12, 2021. 5

[13] Masafumi Nakano, Akihiko Takahashi, and Soichiro Takahashi. Bitcoin technical trading with artificial neural network. *Physica A: Statistical Mechanics and its Applications*, 510:587–609, 2018. 1

[14] S.C. Nayak, B.B. Misra, and H.S. Behera. Impact of data normalization on stock index forecasting. *International Journal of Computer Information Systems and Industrial Management Applications*, 6:257–269, 2014. 3

[15] Zihan Qiang and Jingyu Shen. Bitcoin high-frequency trend prediction with convolutional and recurrent neural networks, 2021. 2, 5, 6

Appendices

A. Loss and Accuracy training curves

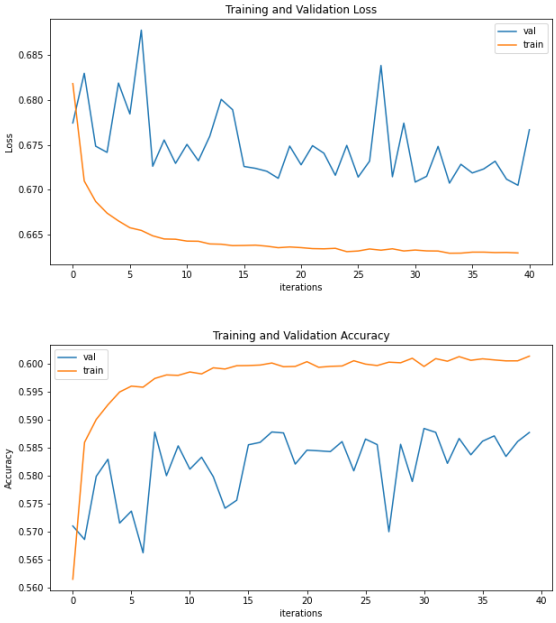

Figure 10. Loss and Accuracy Curves for MLP model.

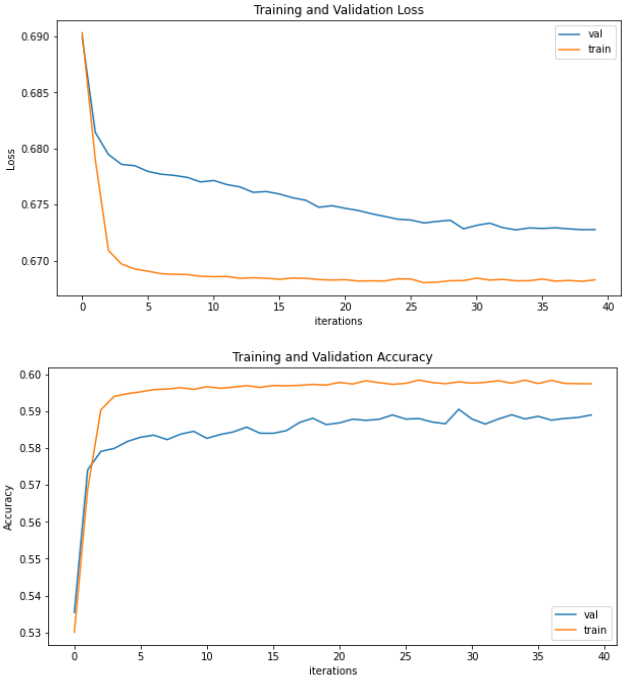

Figure 11. Loss and Accuracy Curves for LSTM model.

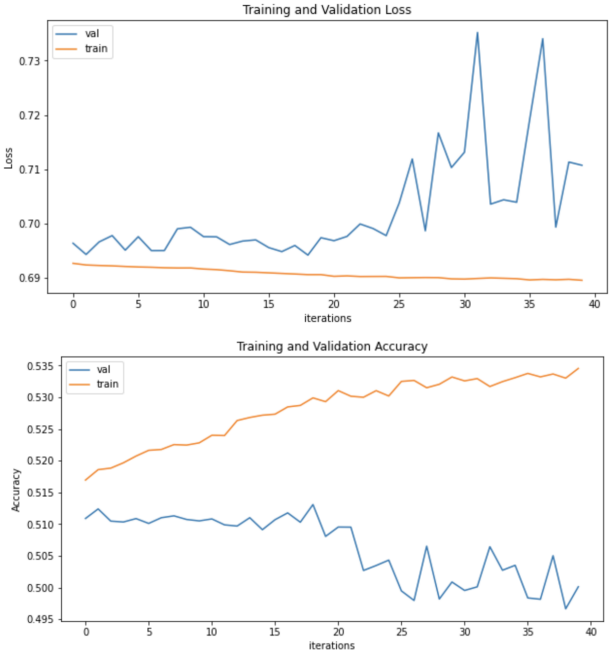

Figure 12. Loss and Accuracy Curves for CNN model.

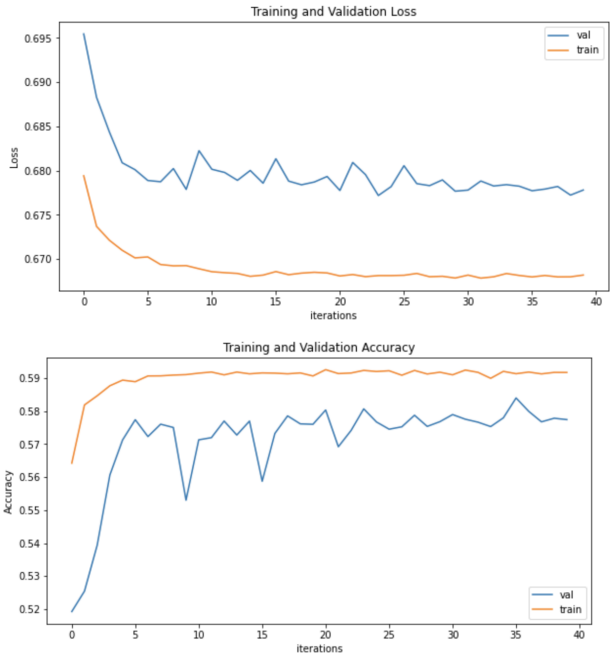

Figure 13. Loss and Accuracy Curves for CNN-LSTM model.

B. Raw Dataset

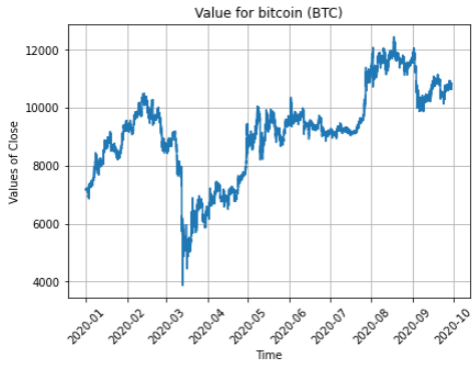

Figure 14. Bitcoin price from 1/1/2020-9/30/2020.

C. Confusion Matrices

	Predict 0	Predict 1
True 0	0.6215	0.3785
True 1	0.4686	0.5314

Table 4. Normalized Confusion Matrix for the MLP Model.

	Predict 0	Predict 1
True 0	0.6455	0.3544
True 1	0.5018	0.4981

Table 5. Normalized Confusion Matrix for the LSTM Model.

	Predict 0	Predict 1
True 0	0.6722	0.3277
True 1	0.6495	0.3504

Table 6. Normalized Confusion Matrix for the CNN Model.

	Predict 0	Predict 1
True 0	0.6154	0.3845
True 1	0.4695	0.5304

Table 7. Normalized Confusion Matrix for the CNN-LSTM Model.

D. Ethereum

We performed some experiments using a different cryptocurrency: Ethereum. However, the testing accuracy was not as good as the ones obtained for Bitcoin. This might be explained due to the illiquidity of the Ethereum exchange, since a third of the data at the minute level did not any volume, which produced duplicate data that affected the training of our models.

E. Work Division

Student Name	Contributed Aspects	Details
Hector Villafuerte	Research, Data Processing, Code Implementation, Report editing	Research Papers, Process the datasets, trained and tuned the LSTM model.
Thomas Passaro	Research, Data Processing, Code Implementation, Report editing	Research Papers, Process the datasets, trained and tuned the CNN-LSTM model.
Yichao Zhang	Research, Data Processing, Code Implementation, Report editing	Research Papers, Process the datasets, trained and tuned the MLP model.
Zach Reeve	Research, Data Processing and Code Implementation	Research Papers, Process the datasets, trained and tuned the CNN model.

Table 8. Contributions of team members.