Bitcoin High-Frequency Trend Prediction with Convolutional and Recurrent Neural Networks

Zihan Qiang
Stanford University
zqiang@stanford.edu

Jingyu Shen
Stanford University
jingyu.shen@stanford.edu

March 17, 2021

Abstract

Cryptocurrencies, with Bitcoins created in 2009 as the forerunner, can be exchanged, with the state-of-the-art block-chain technology providing protections for transactions. In this work, the deep learning techniques of convolutional neural networks (CNN) and long short-term memory (LSTM) networks are used as classification algorithms to perform one-step ahead high-frequency trend predictions of Bitcoin prices using minute level technical indicators. A hybrid CNN-LSTM model has been developed after many rounds of hyperparameter tuning. Our work has shown promising results that the developed model can, to a certain degree, extract useful signals from technical indicators and generate trading strategies that outperform the benchmark strategy of passively holding Bitcoins in terms of net asset value and Sharpe ratio in the backtesting period. A future plan of applying rolling window cross validation is also proposed.

I. Introduction

In recent years, cryptocurrencies, especially Bitcoins, have been gathering widespread public interest since its price spike in 2017 to $19,783 USD and a recent surge to around $40,000 USD in January 2021. Due to the extremely volatile and manipulation-prone nature, there have been ongoing controversies around whether the cryptocurrency market is a good place for investment. In this project, we aim to forecast the minute level price changes of Bitcoins traded on Binance using convolutional and recurrent neural networks to generate investment strategies that can outperform the traditional strategy of passively holding assets. The inputs to our algorithms are the minute level open, close, high, and low prices, all represented as exchange rates with $\$USD$, as well as the volume traded at Binance Bitcoin exchange and the overall volume traded in the market. Our algorithm will then predict the direction of Bitcoin price change the next minute using the trained algorithms.

II. Literature Review

Deep learning algorithms have been applied and tested in various research, some of which have shown accurate yet robust results in Bitcoin's high-frequency trend prediction. Ji, Kim, & Im, have tried deep learning algorithms of DNN, LSTM, CNN, and RNN to predict Bitcoin returns. The accuracy of these methods is around 50% to 60% and it is concluded that no methods would significantly outperform others. In another research, by using 18 technical indicators calculated from minute-level Bitcoin prices, Alonso-Monsalve, Suarez-Cetrulo, Cervantes, & Quintana, have used CNN, hybrid CNN-LSTM network, MLP, and RBF neural network to predict the price changes of the six most popular cryptocurrencies - Bitcoin, Dash, Ether, Litecoin, Monero and Ripple, and have concluded that the hybrid CNN-LSTM network performs the best, with test accuracies reaching as high as 80% and 74% in predicting price changes of Monero and Dash respectively. Bergmeir & Benítez, has tested the method of blocked cross-validation on one-step ahead time series forecasting, which we call rolling windows cross-validation in this report, to make use of the most recent information for each round of validation process, and has yielded robust error estimates. We plan to re-calibrate and improve the CNN and CNN-LSTM hybrid model by using additional features and including the most recent roller-coaster Bitcoin price fluctuations, which may provide more meaningful implications going forward.

III. Dataset and Features

The dollar-denominated minute level Bitcoin prices and volumes from July 8, 2020 to Feb 11, 2021 are collected with a total of 313,327 data points, and 313,327 labels of price change are generated, where 0 and 1 denote a decrease and an increase in price respectively. The entire dataset is split into three parts, where the first 80% of the data are used as the training set, and the rest are equally

divided into dev and test sets. The following table shows the summary and balance of each dataset with respect to the labels.

	Labels	Samples	Percentages
Train	0	125,444	49.18%
	1	129,621	50.82%
Dev	0	14,330	50.55%
	1	14,017	49.45%
Test	0	14,899	49.77%
	1	15,016	50.23%

Table 1: *Dataset Summary*

Table 1 has shown that the labelling classes of three datasets are well balanced, with both labels occupying 50% in each set, and do not have class imbalance issues. The inputs are then used to calculate 30 technical indicators, similar to the ones in Alonso-Monsalve et al., with some additions, as shown in Table 2.

Indicator	Formula
$MOM(n,t)$	$C_t - C_{t-n}$
$MOM_{ret}(n,t)$	$log(\frac{C_t}{C_{t-n}})$
$SMA_{ret}(n,t)$	$\frac{\sum_{i=0}^{n-1} log(C_{t-i}/C_{t-i-1})}{n}$
$WMA_{ret}(n,t)$	$\frac{\sum_{i=0}^{n-1}(n-i)\times log(C_{t-i}/C_{t-i-1})}{n}$
$RSI(n,t)$	$100 - (\frac{100}{1+\frac{n_{up}}{n_{down}}})$
$SK(n,t)$	$(\frac{C_t - Low_n}{High_n - Low_n})$
$SD_{10}(n,t)$	$\sum_{i=0}^{10-1} SK(n,t-i)$
$LWR(n,t)$	$(\frac{High_n - C_t}{High_n - Low_n})$
$CCI(n,t)$	$\frac{1/3\times(High_n+Low_n+C_t)-SMA(n,t)}{0.015\times MD(n,t)}$
$MACD(t)$	$EWMA^{12}_{close}(t) - EWMA^{26}_{close}(t)$
$ADOSC_{Binance}(t)$	$EWMA^{3}_{V_{Bingnce}}(t) - EWMA^{10}_{V_{Binance}}(t)$
$ADOSC_{all}(t)$	$EWMA^{3}_{V_{all}}(t) - EWMA^{10}_{V_{all}}(t)$

$n \in \{5, 10, 30\}$; C_t: close price at time t; n_{up}, n_{down}: averages of n-day up and down closes; $High_n$, Low_n: highest and lowest price from last n periods; The subscript of EWMA denotes data and the superscript denotes the lookback period.

Table 2: *Technical Indicators*

The indicators of simple moving average and weighted moving average in the original paper are changed to be SMA and WMA of Bitcoin returns to ensure stationarity. The calculation of technical indicators can be viewed as a transfer learning process that extracts useful features of price dynamics, including momentum, relationships among recent price levels, overbought and oversold conditions, and the overall accumulation and distribution. All the above indicators in the training set are normalized, and the corresponding parameters of mean and variance are used to normalize the indicators in the validation and test sets.

IV. Methods

The model implemented in this paper includes two major components: CNN and LSTM.

i. Convolutional Neural Networks

The architecture of convolutional neural networks (CNN) is used in our project to utilize its ability to exploit relationships among adjacent data and to speed up the processing of the massive amount of high frequency data.

To implement CNN, our minute level data points are first converted to a set of 2 dimensional "images" shaded in blue as shown in Figure 1, with the 2 dimensions being time lags (height) and technical indicators (width). Every input image covers all of the 30 indicators and a certain number of time lags for a given time stamp in our data. The number of technical indicators are fixed throughout the project whereas the number of time lags are fine tuned as hyperparameters (the number of lags is shown as 15 in Figure 1 and 2 for the purpose of illustration). The direction of the next minute price change, shaded in orange, is used as the label of a specific input "image".

	I1	I2	I3	···	I29	I30	Label
T1	T1I1	T1I2	T1I3	···	T1I29	T1I30	L1
T2	T2I1	T2I2	T2I3	···	T2I29	T2I30	L2
T3	T3I1	T3I2	T3I3	···	T3I29	T3I30	L3
···	···	···	···	···	···	···	···
T15	T15I1	T15I2	T15I3	···	T15I29	T15I30	L15
T16	T16I1	T16I2	T16I3	···	T16I29	T16I30	L16
T17	T17I1	T17I2	T17I3	···	T17I29	T17I30	L17
	I1	I2	I3	···	I29	I30	Label
T1	T1I1	T1I2	T1I3	···	T1I29	T1I30	L1
T2	T2I1	T2I2	T2I3	···	T2I29	T2I30	L2
T3	T3I1	T3I2	T3I3	···	T3I29	T3I30	L3
···	···	···	···	···	···	···	···
T15	T15I1	T15I2	T15I3	···	T15I29	T15I30	L15
T16	T16I1	T16I2	T16I3	···	T16I29	T16I30	L16
T17	T17I1	T17I2	T17I3	···	T17I29	T17I30	L17

Figure 1: *Two Examples of Data Points*

During the construction of our convolutional neural network algorithm, we choose to use 1-dimensional filters that encode information of time series of a given indicator (vertical filters) or of a set of indicators at a given time (horizontal filters) as in Figure 2. The filter sizes are also tuned as hyperparameters throughout the course of the project. 1-D filters are been applied instead of 2-D filters, which are widely used in computer vision, because as a 2-D filter slides through our input "images", the nature of the information that a given filter encodes is constantly changing for each iteration (i.e. the filter looks at different indicators at different time stamps).

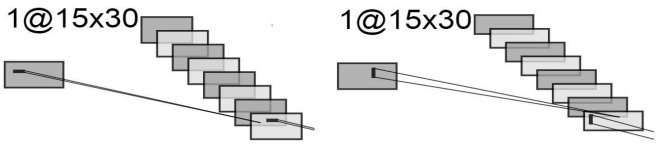

Figure 2: *1-D Filters*

After tuning hyperparameters, we achieved the optimal structure of our CNN model shown below in Figure 3. Batch normalization layers are used in each convolutoinal layer to speed up training. Pooling layers are not used in our CNN architecture because our input 'images' are relatively small compared to the more commonly seen RGB images which often have significantly larger sizes, and the use of pooling layers can summarize the features too early.

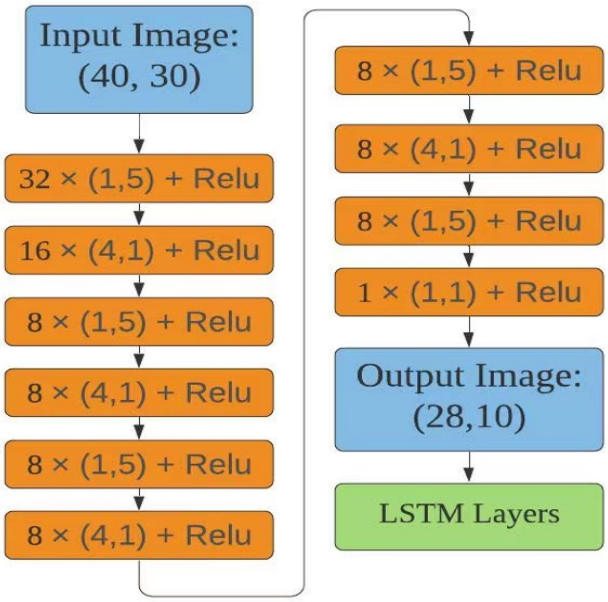

Figure 3: *CNN Layers*

The optimal lag for the input 'images' is 40, and so the CNN layers take the input of size 40 by 30 and return the output of size 28 by 10 after going through the filters. The output 'images' are inputted into the LSTM networks as described in Section ii.

ii. Long Short-Term Memory Networks

The long short-term memory (LSTM) networks are used in our project to further exploit the sequential relationships among Bitcoin data, which, in our case, are time series data of technical indicators. It is an ideal tool for time-series forecasting given its capability of dealing with the entire sequence of data. The two deep LSTM layers are added as additional layers to our already established CNN architecture, and will take the output of the CNN layers as inputs. The optimal architecture of our LSTM network is illustrated in Figure 4.

Our LSTM architecture consists of a 2-layer deep version of LSTM. The output units of each layer is 100. It means that the input 28 timestep by 10 features matrix will been fed into this model and transformed into 28 1 by 100 vectors first and eventually into one 1 by 100 vector. This layer is then followed by a dense layer to generate the label output with sigmoid activation function, which takes on the value of 0 or 1. Dropout factors are added in the recurrent layers as well as in the dense layer as regularization to our architecture.

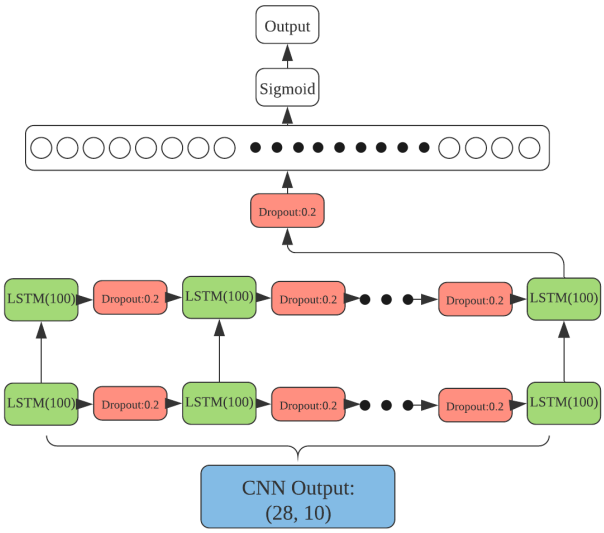

Figure 4: *LSTM Layers*

iii. Temporal Cross Validation

As our project is dealing with a time series data of Bitcoins that do not have a well-defined nature, unlike other popular asset classes such as stocks, bonds, and futures whose values are based on the corresponding companies, governments, and the underlying assets respectively, the distribution of Bitcoin's historical data are constantly experiencing imperceptible shifts. Therefore, the methods of temporal cross validation is a preferable choice to validate our models by keeping up-to-date with the market conditions.

One method of rolling windows cross validation is shown in Figure 5 below. For this method, there are a number of rolling windows (5 in the figure for illustration), with each consisting of a training and validation set and the size of each set is kept the same for each rolling window. In each window, a training-validation process is performed, after which the window is moved in time by the amount of data points in the validation set, so that the

next training set gets updated by adding the previous validation set and discarding the data points that are further in the past.

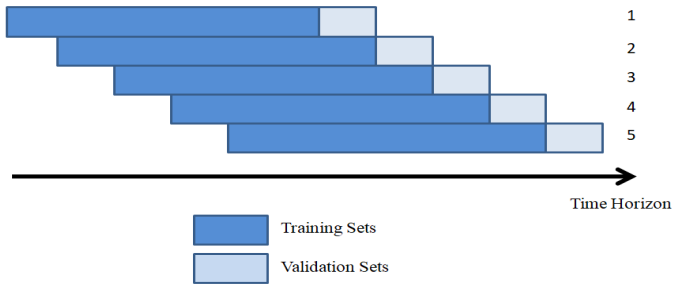

Figure 5: *Rolling Windows Cross Validation*

However, the mentioned validation method has high computational cost and is not feasible for the scope of this project. Instead, we choose to select K independent blocks of data, as shown in Figure 6 below, and break them down into training and validation sets to evaluate the algorithm's performance throughout different historical periods. The data used in the training-validation can shrink for large K's, which may solve the distributional shift. For the evaluation of both methods, we use the average of our algorithm's performances on each validation sets.

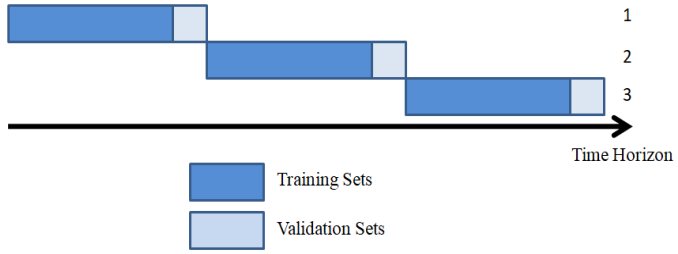

Figure 6: *Independent Cross Validation*

V. Experiments

The hyperparameters of the aforementioned model have been fine tuned in great efforts. Note that the hyperparameters of CNN layers are tuned together with the LSTM layers. We start off the tuning process by trying to overfit the training set, and then add dropout and regularization to achieve the balance between bias and variance.

In our CNN architecture, we experimented the hyperparameters that affect the complexity of our model - number of layers, number and size of filters at each layer. The training loss did not decrease dramatically in the beginning, and thus we need to tune the above hyperparameters to increase the model complexity in order to capture more signals. The number and size of filters remain relatively unchanged in the tuning process, as one of the literature implemented similar size and number of filters and have achieved reasonable results. We make the vertical filter size smaller than horizontal filter so the LSTM can capture more time-series signals. The most important hyperparameters we tuned is the number of layers, where we tried the number of layers all the way from 4 to 10 and ended up with 5 horizontal and 4 vertical layers. The reason why we have one less vertical filter is that, again, we prefer to use LSTM, instead of CNN, to capture time-series relationship.

The hyperparamters to tune in the LSTM layers are the output units and number of LSTM layers. We mainly tuned the output units of LSTM, all the way from 30 to 150 units, and choose to fix the LSTM layers to 2 as the LSTM layers are typically no more than 3. We ended up with 100 units to increase the model complexity while keeping the algorithm computationally feasible.

The hyperparameters tuned in the overall hybrid model are batch size, epochs and dropout rate. We chose the batch size from 64 to 256 and the epochs from 80 to 120 for different cases to ensure convergence while save training time. We tried dropout factors of 0.2 to 0.4 to add regularization effect, and ended up with 0.2 dropout rate as 0.3 and 0.4 would result in non-converging training loss.

VI. Results & Discussion

We trained the aforementioned model with Adam as the optimizer, the binary cross-entropy function as the loss function, and hyperparameters mentioned before. As suggested in the section Temporal Cross Validation, we have trained the model on K = 1,2,3 for testing purposes. As this project is intended to generate optimal Bitcoin trading strategies for investors by capturing signals from historical data, the metrics of net asset value (NAV) and Sharpe ratio (SR), which are commonly used in the financial industry, are used in addition to predictive accuracy. NAV measures the value of an investor's asset at the end of the investment horizon assuming 1 dollar is invested at the beginning. The results for K = 1 are shown below. Note that for each case, we trained for 3 times and average the results to ensure robustness.

Valid	BTC NAV	NAV	BTC SR	SR	Acc
K = 1	1.194	1.613	0.003	0.0078	0.514
Test	BTC NAV	NAV	BTC SR	SR	Acc
K = 1	1.29	1.455	0.0052	0.0074	0.511

Table 3: *K = 1 Validation and Test Set Results*

We can observe that the valid and test accuracy are just above the 50% threshold, meaning that our model is not confident at predicting the next minute return. It is likely

to be caused by a large distributional shift between our training set and validation or test test. The fundamentals have changed over the period so that the signal we learnt from training set is not effective in validation set. However, we can see that the NAV and Sharpe ratio generated by our model outperforms the passive strategy. Our model can utilize past signal to generate trading strategies that can outperform the benchmark.

We then tested with the temporal cross validation. The results for K = 2 and K = 3 are shown below. Note that the results are the average over 2 or 3 trials.

	BTC NAV	NAV	BTC SR	SR	Acc
K = 2	1.23	1.34	0.013	0.018	0.53
K = 3	1.10	1.02	0.0082	0.0008	0.507

Table 4: *K = 2 and 3 Validation results*

We can observe that the accuracy when K = 2 is around 53%, which is much higher than K = 1. When K = 2, both the training and validation set have been shortened, which clearly offsets the distributional shift issue. It demonstrates that the model is relatively good at predicting the next minute return. Moreover, the NAV and Sharpe ratio have outperformed the passive investing strategy. The accuracy and NAV of one trial have been plotted below.

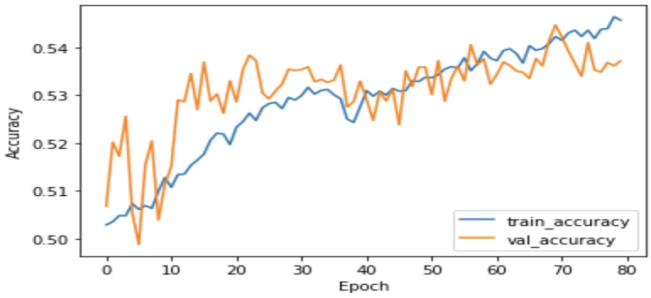

Figure 7: *K = 2 Accuracy*

Figure 8: *K = 2 Net Asset Value*

From Table 4, we can see that our strategy generated 34% return while Bitcoin has increased 23%. The profit from K = 2 is not as good as K = 1 as we are training on less data which may results in less signals being captured, however, it is more robust given the higher accuracy with less distributional shift.

For K = 3, we have observed that the accuracy is 0.507 and the validation loss is even increasing. It means that the we have learnt noises or overfitted given that the training set is too short. Moreover, the NAV and Sharpe Ratio have underperformed the passive investing strategy. It can be caused by the ineffective trained strategy, and the validation set may also be too short to realize the strategy.

In general, we have observed the distributional shift when K = 1. Then, we tried to reduce the training set by splitting the current dataset into 2 and 3 independent blocks (K = 2,3). We realized that K = 2 would be a good balance between the demand of short training set to offset distributional shift and the demand of long training set to offset overfitting. As suggested in section 4.3, we should further test the K = 2 strategy (i.e. the same length of training and validation) on the rolling basis to test out the stability of this strategy.

VII. Conclusion

In this paper, we have implemented a hybrid deep learning model that includes CNN and LSTM to predict the Bitcoin returns and generate investment strategy. 30 technical indicators have been computed as the features. We then generated the input images by cropping the dataset with time lag = 40. Next, we designed a CNN model that is comprised of 9 layers with 5 horizontal filters to encode relationships among features and 4 vertical layers to encode those among time steps. Then, the output image is fed into a 2 layer LSTM model with 100 units to further capture the time-series. Finally, a sigmoid dense layer is added to produce the binary result. Overall, the batch normalization and dropout layers are also added to ensure the regularization effect to avoid the overfitting. We first trained the entire dataset and get only 51% accuracy but a good NAV and Sharpe ratio, which outperforms passive investing in Bitcoin. Although it suggests that there is a distributional shift between train and validation which causes lower accuracy, signals from the past are captures to generate an outperforming investment strategy. We then implemented the temporal cross validation (split), and achieved 53% accuracy with better NAV and Sharpe ratio when K = 2. We realized that the shortened training set resolves the distributional shift issue, but is also sufficient to capture the signals from historical data. For K = 3, the accuracy and NAV is too low as the model learns only the noise. The next step would be to implement a rolling temporal CV with the same length of training and validation data as K = 2 to further test the strategy.

VIII. Contributions

Zihan Qiang and Jingyu Shen constantly and closely collaborated with each other. There is no clear line that could separate the work. In general, they contributed equally to this work.

IX. Appendix

i. Accuracy for K = 1

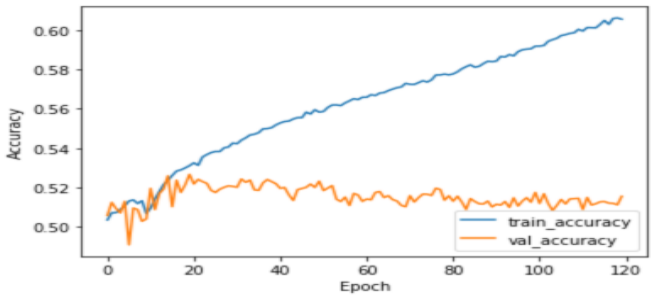

Figure 9: *K = 1 Valid Accuracy*

ii. NAV for K = 1

Figure 10: *K = 1 Valid NAV*

Figure 11: *K = 1 Test NAV*

iii. Accuracy and NAV for K = 3

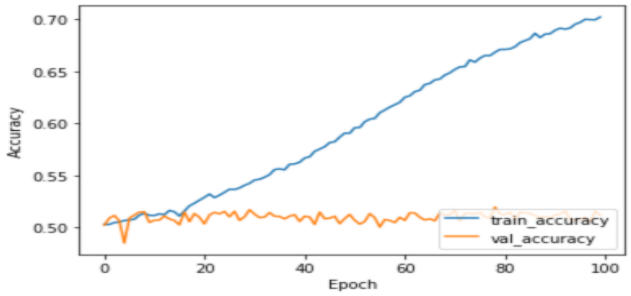

Figure 12: *K = 3 Accuracy*

Figure 13: *K = 3 NAV*

References

Alonso-Monsalve, S., Suarez-Cetrulo, A. L., Cervantes, A., Quintana, D. (2020). Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems with Applications*, *149*, 113250.

Bergmeir, C., Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192-213. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0020025511006773` (Data Mining for Software Trustworthiness) doi: https://doi.org/10.1016/j.ins.2011.12.028

Ji, S., Kim, J., Im, H. (2019). A comparative study of bitcoin price prediction using deep learning. *Mathematics*, *7*(10), 898.