Predicting Prices from Order Book Data using a Deep Learning Model

Chenghao Zhang, Nicholas Harper, Kevin Bai, Chan Hyuk Yoon

Introduction

Electronic trading of stocks and assets happens using limit order books which allow trading orders to be queued and matched in a centralized exchange. Limit order books allow for orderly buying and selling based on (1) limit orders that add liquidity to the book and remain in the order book until they are matched with and (2) market orders that take liquidity off the order book.

Price formation happens via the order book as well. The price of a stock is often reported as a single number which is the last price that it was transacted at, but another way to describe the price of an asset is via the bid-ask spread i.e. the distance between the best bid in the market and the best offer. Order books contain a tremendous amount of information beyond the latest price; each snapshot shows the number of buyers and sellers ready to transact which can convey buying or selling pressure, large cancellation of orders, and momentum or mean reversion in prices.

Based on the information contained inside the order book, Deep Learning models can be utilized to capture the microstructure and detect implicit patterns contained in the limit order book. This project focuses on predicting and classifying the future movement of the price of bitcoin using a Deep Learning Model. After utilizing convolutional neural network layers and an LSTM layer, we were able to achieve an accuracy of around 62% on our test set after a short model run time (3 minutes).

Literature

Recent literature has utilized deep learning models to capture the hidden patterns inside the limit order book. One key feature of modern deep learning is the use of feature extraction and representation of the model. While CNN's have been successful in domains such as speech recognition and object detection, it has not been extensively used in the realm of financial microstructure, and the implementation using a CNN with LSTM layers is even more rare in this domain.

One notable implementation which utilizes the LSTM model is Jha et al. "Deep Learning for Digital Asset Order Books". This implementation trains the models on 100ms snapshots of order books with focus on two feature labels: price and volume. The authors use a depth of 100 price levels at each snapshot, where each price level is divided into 50 levels on the bid and 50 levels on the ask, and for each price level two features of price and volume are used.

Another interesting paper is "DeepLOB: Deep Convolutional Neural Networks for Limit Order Books". This paper utilizes the deep learning scheme to classify the price movement (ie. "up", "down", "stable"). The implementation uses stacked convolutional neural networks and an LSTM layer followed by a softmax layer to output probabilities for price movements. The price movement class which has been assigned the highest probability would be selected as the final output.

Data Collection

We pulled 159,105 time stamps from FTX over a depth of 10. After data collection, panel data is indexed by timestamp and contains 40 columns for each timestamp (10 bid prices, 10 ask prices, 10 bid volumes, and 10 ask volumes). Times between timestamps were not consistent, although average time between was typically very small (around 3 milliseconds). An example of a several rows of data is pasted below:

```
                               bid1     bid2     bid3  ...  asize8  asize9  asize10
time                                                   ...
2021-08-02 19:06:22.029600  41041.0  41035.0  41032.0  ...  0.0080  1.1345   0.1001
2021-08-02 19:06:22.071000  41041.0  41035.0  41032.0  ...  1.1345  0.1001   0.0300
2021-08-02 19:06:22.099900  41041.0  41035.0  41032.0  ...  0.0080  1.1345   0.1001
2021-08-02 19:06:22.117100  41041.0  41035.0  41032.0  ...  8.5262  0.0005   0.0080
2021-08-02 19:06:22.152500  41041.0  41035.0  41032.0  ...  8.5262  0.0005   0.0080
...                             ...      ...      ...  ...     ...     ...      ...
2021-08-02 21:54:59.524300  41078.0  41076.0  41074.0  ...  0.0800  0.0080   0.9744
2021-08-02 21:54:59.548000  41078.0  41076.0  41074.0  ...  0.1902  0.0800   0.0324
2021-08-02 21:54:59.576400  41078.0  41076.0  41074.0  ...  0.1902  0.0800   0.0324
2021-08-02 21:54:59.598600  41078.0  41076.0  41074.0  ...  0.1902  0.0800   0.0324
2021-08-02 21:54:59.923200  41078.0  41076.0  41074.0  ...  0.1902  0.0800   0.0324
```

Data Preprocessing & Scaling

The collected data contained a trivial amount of missing (NaN) values, which have been removed for preprocessing purposes. In several of the papers, scaling and normalization of the dataset was conducted. The intuition behind this is to enable the model to capture the pattern and shape of the order book regardless of the mid-price at a specific time. For our research, scaling is introduced on the bid/ask prices by taking the mid-price, defined as 1/2*(best bid price + best ask price), and subtracting each price in the column from this mid-price. Again, scaling is conducted so that relative price differences are now the only aspects the model can learn, whereas previously it cannot distinguish between changes and overall prices.

Classification Scheme and Data Labelling

The deep learning model requires a labeled dataset to train and improve model accuracy. The purpose of classification schemes is to transform time-series bitcoin price data into a labeled dataset. We will use the classification scheme used by major papers which labels each time stamp as in one of [“up”, “down”, “stable”] which denotes the future actual price movement given the current orderbook.

The following describes the procedure in more detail :

1. Define the midpoint price as $p_t = (p_a(t) + p_b(t))/2$ where $p_a(t)$ is the price of the best ask and $p_b(t)$ is the price of the best bid at time t.
2. Define m_+ and m_- which represents the average of k mid prices before and including time t(-) and immediately after t(+). Details in Equation1.

3. Compare m_+ and m_- to classify and label bitcoin price movement in Equation2. In equation α is a hyperparameter that separates “stable” from other classification with default value of 1e-5.

$$m_-(t) = \sum_{i=0}^{k-1} p_{t-i}$$

$$m_+(t) = \sum_{i=1}^{k} p_{t+i}$$

Equation 1

$$l_t = \begin{cases} [label = 0]down & m_-(t) > m_+(t)(1+\alpha) \\ [label = 2]up & m_-(t) \leq m_+(t)(1+\alpha) \\ [label = 1]stable & otherwise \end{cases}$$

Equation2

Class Imbalance Modification

Because prices do not change much during such small fluctuations in time, prices mainly were classified as 1 or "stable". To fix this, the alpha was adjusted using L2 norm so that the classes were balanced as close as possible. We obtained an α = 4.64e-6 and were able to re-balance the classes as seen below:

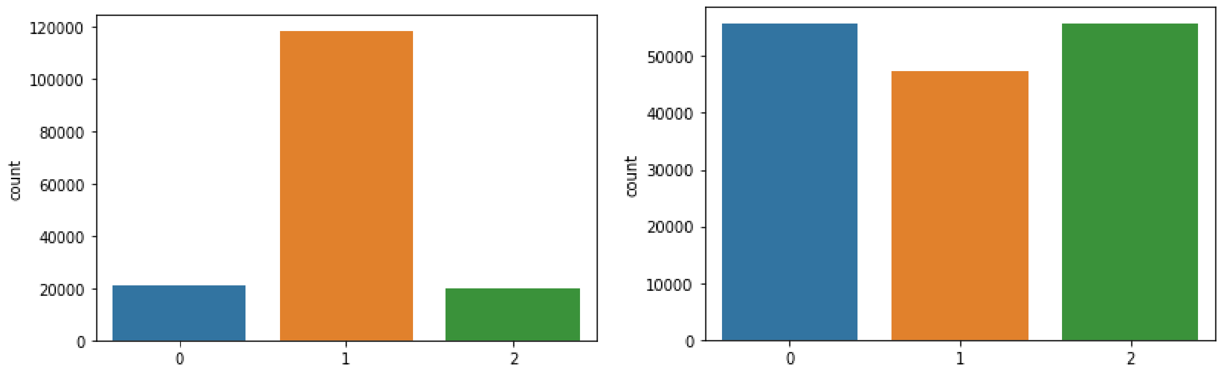

Figure 1: Classes before (left) and after (right) rebalancing

Model Architecture and Data Input/Output

Deep learning neural networks utilize multiple layers to capture a complex non-linear pattern which is not easily detected by simple machine learning techniques. Specifically, one distinctive feature of the convolutional layer is the use of a kernel to scan through the order book price levels. Just as human eyes detect objects inside the picture, convolutional neural networks scan through the price level to detect patterns that are relevant to the output classification. After three convolutional layers follows a bidirectional LSTM layer. The LSTM layer captures the temporal structural dependencies of time-series order book data. This enables us to take into account the impact of the shape change of the order book to the output classification. The overall model is trained to minimize cross entropy loss, which is a typical classification minimization scheme.

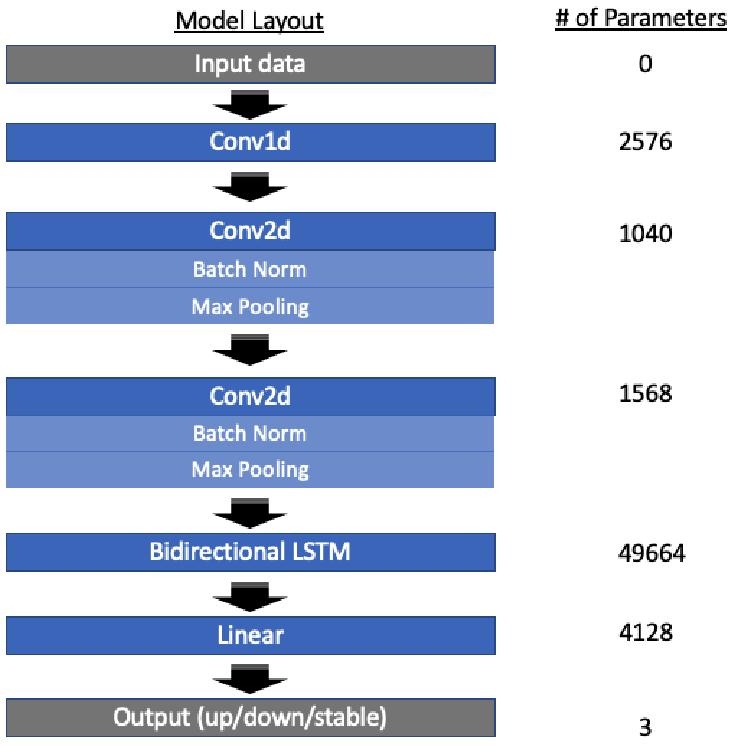

Figure 2: Deep Learning Model Layout

The input of the model follows a similar schema to the Jha et. al. methodology and the majority of papers researched. Namely, inputs of size 100 sliding timestamps with 40 features (10 bid price, 10 ask price, 10 bid level, and 10 ask level) made up each training instance. 30 timestamps were used because price fluctuations were generally not seen in singular training instances. In total, 159,065 instances were made possible using k = 20 and of these, the first 100,000 were used for training. The last 59,065 instances were used for validation/test.

The output from the model will be a classification of 0 (down) ,1 (stable), or 2 (up) for each input and was compared with the classification schema developed and mentioned earlier.

Model Comparison to Literature

Our model (when compared to results obtained in the literature) was not able to achieve as high of an accuracy (62% as opposed to 70-79%). We believe the reasons for the reasons for the suboptimal performance are as follows:

- The original data set used in the paper had a dataset containing over 1.6 Million data points while ours contains less than 150,000. Therefore, the model may not perform as well.
- Scaling was only applied to price and not to volume. The model may not have been able to learn relative discrepancies in volume.

- Parameters were reduced overall due to the number of data points. By nature of deep learning architectures, the number of parameters to be trained is significantly large (60,835 in the paper) which could capture more complex patterns inside the orderbook.
- The learning rate scheduler was not optimized correctly and was stuck in a local minimum.

Results

Our model architecture achieved an accuracy of 62~68% on the validation set. Achieving an accuracy of ~70% out-of-sample with such a small volume of trading data provides strong evidence that our model was able to distinguish patterns in the order book and generalized well across time periods.

The Training & Testing loss in figure 3 shows that testing loss is minimized at the early training stage. As the number of epochs increases, we can see that training loss decreases while the testing loss increases, which is a sign of overfitting. To cope with overfitting, we use the model at an earlier training stage to enhance testing predictability.

The confusion matrix in Figure3 shows that classification accuracy is fairly even across different classes with the value of 0.69 for predicting up movement, 0.62 for down movement, and 0.57 for stable movement. While uneven prediction accuracy usually represents deficiency of the model, balance in our prediction accuracy across classes may represent robustness of the model with predictive power regardless of where the market sits.

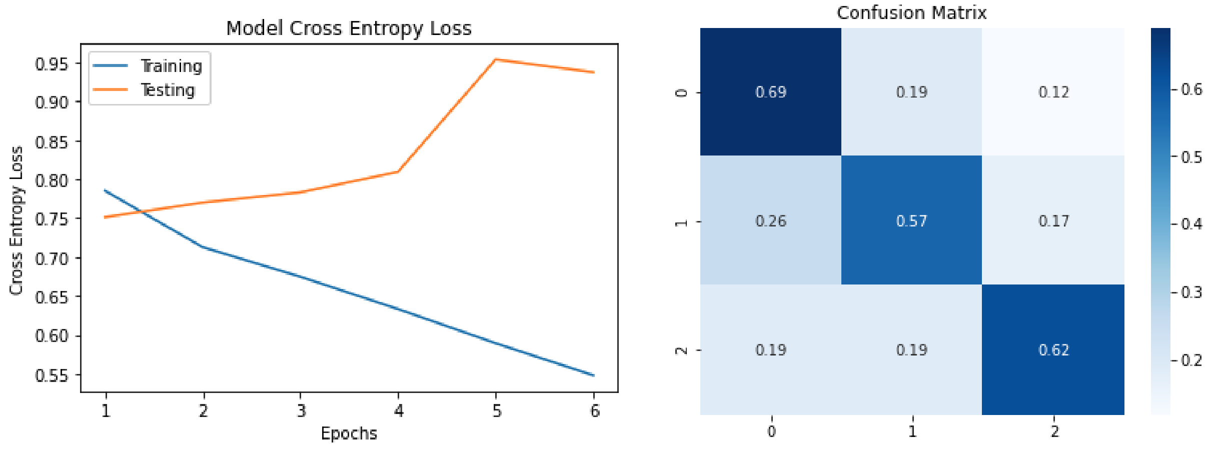

Figure 3: Training & Testing Loss (left) Model Accuracy Results using confusion matrix (right)

Market Impact Analysis

Financial markets can behave extremely differently depending on the time and day, so developing a model robust to these factors is crucial. For our specific time period, the class proportions of up / down / stable stayed fairly even. However, if the volatility of the market increases, the classification model which has been trained on relatively stable periods would have less predictive power which could affect the model's robustness.

Considering the impact of usage and execution of this model, 68% accuracy of forecasting upward and downward movement of the bitcoin prices will make the market more efficient, decreasing the profitability of informed traders.

References:

[1] Fan Fang, Waichung Chung, Carmine Ventre, Michail Basios, Leslie Kanthan, Lingbo Li, and Fan Wu. Ascertaining price formation in cryptocurrency markets with Deep Learning. arXiv preprint arXiv:2003.00803v1, 2020.

[2] Huisu Jang and Jaewook Lee. “An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information”. In: IEEE Access 6 (Dec. 2017), pp. 5427–5437. doi: 10.1109/access.2017.2779181.

[3] J. Doering, M. Fairbank, and S. Markose, “Convolutional neural networks applied to high-frequency market microstructure forecasting,” in Computer Science and Electronic Engineering (CEEC), 2017. IEEE, 2017, pp.31–36.

[4] J.-F. Chen, W.-L. Chen, C.-P. Huang, S.-H. Huang, and A.-P. Chen, “Financial time-series data analysis using deep convolutional neural networks,” in Cloud Computing and Big Data (CCBD), 2016 7th International Conference on. IEEE, 2016, pp. 87–92.

[5] M. Dixon, D. Klabjan, and J. H. Bang, “Classification-based financial markets prediction using deep neural networks,” Algorithmic Finance, vol. 6, no. 3-4, pp. 67– 77, 2017

[6] Rakshit Jha, Mattijs De Paepe, Samuel Holt, James West, and Shaun Ng. Deep Learning for Digital Asset Limit Order Books. arXiv preprint. arXiv:2010.01241v1, 2020.

[7] Zihao Zhang, Stefan Zohren, and Stephen Roberts. “DeepLOB: Deep Convolutional Neural Networks for Limit Order Books”. In: IEEE Transactions on Signal Processing 67.11 (Jan. 2019), pp. 3001–3012. doi: 10.1109/tsp.2019.2907260.