CS230 Project Report:
Crypto Exchange Price Prediction using Limit Order Book

Ben Gilboa
(SUID# - 06278930)

Tamal Biswas
(SUID# — 05107984)

Ashwin Selka Padmanabhan
(SUID# — 06246676)

Stanford University
Spring 2018
GitHub: https://github.com/gilboab/CS230_Project

Abstract

In this project we develop models for prediction of future Bitcoin price trends using limit order book data as inputs.

1. Introduction

High frequency trading or Algo trading is gaining significant momentum in stock exchanges. In today's market, a sizable portion of the daily traded volume is done by specialized companies using those techniques. In the elaborated stock market, it is almost impossible for individuals not using heavy machinery and very fast access to data to gain any advantage, as margins and arbitrages are closed in fraction of a second.
The rise of the crypto market and exchanges might reveal opportunities that are long gone in the stock market for small scale algorithmic trading.

In this project we explore and develop a deep machine learning model that predicts the future price of digital asset such as bitcoin. We developed RNN (Recurrent Neural Network) that predicts the future price trend of a tradable and volatile digital asset such as the Bitcoin. The input to the model will be a limit order book data along with other historical indicators for demand and supply to develop our predictor. Although we chose a digital asset for this project, the principals and methods we develop are transferable to any asset that is tradable in an exchange.

2. Prior work

Prior work in this area can be split into two categories namely Mathematical models and the Deep Learning models. Tian Guo and Nino Antulov-Fantulin [1] try to predict the short term bitcoin price fluctuations mathematically using their own custom model derived from the volatility of the order book which is more reliable than the related time series and moving average models like ARIMA, ARIMAX etc. Huisu Jang and Jaewook Lee [2] use information from Blockchain transaction data and try proving that a Bayesian neural network performs well in predicting the Bitcoin price time series associated with its high volatility. Muhammad J Amjad and Devarat Shah [3] improve on the current time series prediction algorithms. More specifically, they develop a framework for time series analysis and then present a scalable real time algorithm with an intent to predict the next state of Bitcoin with high accuracy. Justin A Srignano [5] developed a new Neural Network architecture in 2015 for modeling spatial distributions of the limit order books. While this work was mostly around regular stocks and not the highly volatile crypto currencies. The paper presents a good motivational factor for combining Neural Networks and Limit Order books for future price predictions and fluctuations.

3. Dataset Characteristics and Acquisition

The data that is primarily used in for our predictor is the data from limit order book.

3.1. Limit order book

The limit order book captures all pending orders of bids and asks. Graphical presentation (figure 1) shows an accumulative view of a single limit order book snapshot.

The limit order book snapshot represents the demand and supply in the market in a certain point in time. In the figure, it is clearly seen that the demand is "stronger". There are much more buyers who are willing to buy the asset for a price that is lower by 3% from last price than sellers who are willing to sell in a price that is higher by 3% than the last price. This might indicate that the price is about to increase. We look at the 500 highest bid orders and the 500 lowest ask orders in every snapshot of the order book.

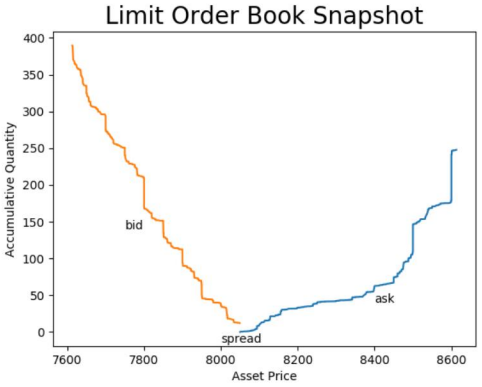


Figure 1: Limit Order Book Snapshot

3.2. Bitcoin historical price

For each limit order book sample data point we look at the corresponding bitcoin price. This is basically the "last" price of a transaction at the same time when the order book was sampled. This data is used to generate the classifier for price increase or decrease. Consider a point in time 't_0' that corresponds to sample in our dataset 's_0'. By considering certain number of examples (s_{-1}, s_{-2}, … , s_{-n}) we get historical feature to the training set. For predicting future trend at time $t_{-future}$ we compare the Bitcoin price at t_0 and $t_{-future}$ to label a price increase or decrease.

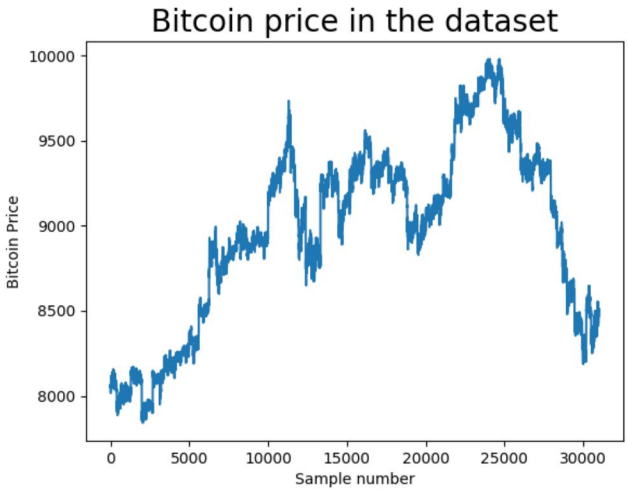


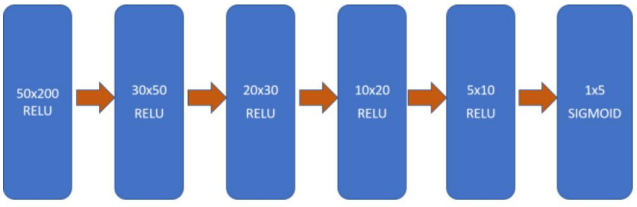
Figure 2: Bitcoin Price throughput our samples

3.3. Data acquisition

We obtain the above data by sampling the Bittrex exchange every 1 minute using the API it provides and storing the data. We obtained so over 30,000 samples that represent 3 weeks worth of trading data. The data is not 100% consecutive as sometimes the software crashes for several reasons due to networking or related issues on the Bittrex side

4. Initial model

As a starting point we use only one limit order meaning that we predict a future change based on the current status without looking at the history or consecutive trends. In fact,



we shuffle the samples and eliminate any timing notation.

Since every order in the book has 2 parameters (quantity and price) we can't use it as is. We apply a small modification to the data to extract a training example. We define "bins" of 10$ and we sum the quantities that relate to each bin. From 500 bid orders we create 100 bins that represent the last price down to last price minus 1000$. In later phases we modified the 10$ bins to 0.1% bins. Figure 2 present a result of the binning process and a visual representation of one training example that we feed to the initial NN. It is easy to observe that this training example corresponds to the one used in figure 1. After binning the data, we end up with 200 features for every training example.

For the labels we have the last Bitcoin price that corresponds to every training example. We make it a classification problem by comparing the next value of the bitcoin (1 min into the future) to the current price. If the price increased the label is '1' and if decreased or same it is '0'. This classification is very naïve and will not result in a successful trading strategy but it is good simple classification for initial design.

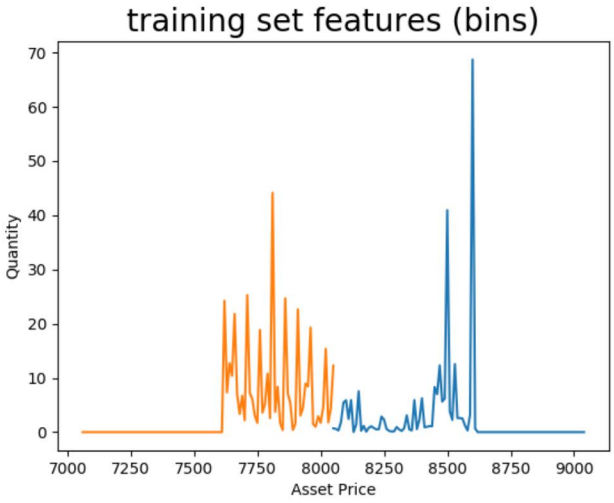


Figure 3: sample of one training example after structured in bins

4.1. Fully Connected Network Architecture

The objective of this initial phase is to find the correlation and validate the data from the order book as valid predictor. The architecture shown in figure 3 describes our current initial network

We had originally attempted using lesser number of layers and neurons and came up with the architecture in Figure 3 after some fine tuning and hyperparameter experimentation. More details below

Figure 4: NN Architecture

Initial Results

Our current architecture has 6 layers. We used about 21,000 training examples and shuffled them. Then we defined the training / dev sets as 80%/20% split. For the labels we compared bitcoin price 1min, 2min, 3min, 5min and 10min into the future to the current price. After adjusting the learning rate combined with Adam optimization and Early stopping, we achieved approximately 95% accuracy on the training set and approximately 64% on the dev set. The 95% accuracy is very encouraging result for us but the high variance is clearly a concern. We tried to add L2 Regularization and Dropout but it did not help to reduce variance. It only increased the bias.

The conclusion we got from the FCN exercise is that the architecture can't predict better than 65% on the dev set when learning for single order book sample and the dataset that we have.

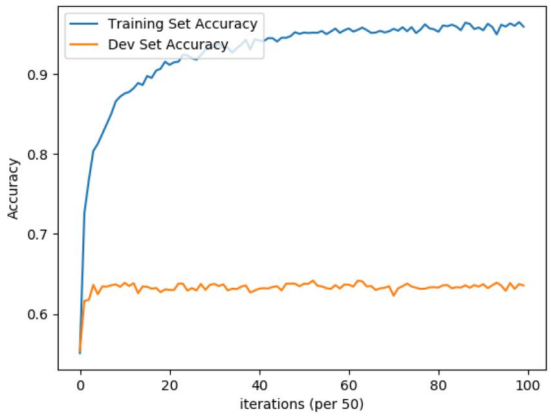


Figure 5: Training vs Dev Accuracy with max dev accuracy at around 3100 epochs

Figure 5 shows the accuracy associated with the final numbers after tuning the hyper parameters. We see that dev set accuracy does not reach more that 65%. These are results validate the correlation between the order book and the labels but they also tell us that the model is not good enough. To get a better model we wanted to bring back the sense of time to the samples and use RNNs for that.

5. RNN

To develop the best predictor for future Bitcoin price, we tried different approaches and RNN architectures. We used Tensorflow to develop our initial RNN models given its high flexibility, portability, performance and other advantages as explained in [6] .

The common theme among all the experiments below was to play with the hyperparameters and the related objects associated with the Network namely:

1. Learning Rate
2. Number of Time Steps
3. Number of hidden units
4. Number if iterations
5. Batch Size etc.
6. Static vs Dynamic RNN
7. Type of Cell (LSTM/GRU/Basic RNN etc.) etc.

5.1. Input and output similar to the Fully connected Network

In this experiment, we fed the limit order book input directly into an LSTM network followed by a sigmoid output prediction of an increase or a decrease The rationale here was to see how the RNN performed compared to a pure Fully Connected Network and its effect on the accuracy. This network trained slower compared to the raw non RNN network. From accuracy standpoint, we were able to achieve similar or better results on the training set but worse than expected performance on the dev set.

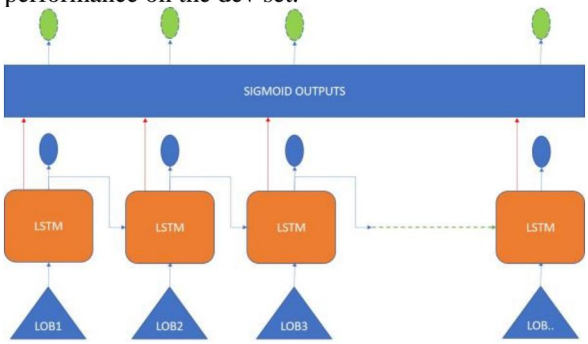


Figure 6: RNN Architecture where Inputs and Outputs are same as the Fully Connected Network

5.2. Input as Order book encoded with FC Network

In this experiment, we encoded the limit order book data using a Fully Connected Network and fed the activations from the last but one layer (layer before the sigmoid activation in the original FC network) into the RNN. From accuracy standpoint, we were not able to achieve a remarkable increase, and this performed very similar to the earlier RNN (5.1) where the order book was directly fed as input

As part of these variations (5.1 and 5.2), in addition to binning the quantity of bitcoins based on the distance from the current price, we doubled the number of input features by including the distance themselves. We believed that by doing so, we will help the network predict better, primarily because, while the bitcoin price can change by several 100 or 1000 dollars over time, the difference between the current price and the bid/ask will follow a pattern. For example, if the price today is $8000, many or most of the bid/ask orders would be closer to the $7000/$8000/$9000 range rather than the $20000 range. However, the neural network's dev set accuracy did not see any reasonable improvement

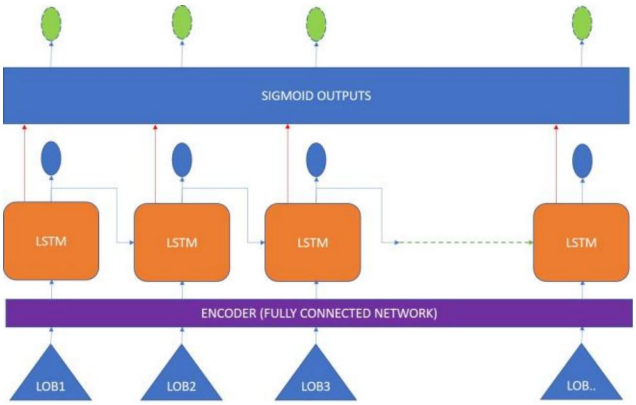


Figure 7: RNN Architecture where Fully connected network acting as encoder becomes RNN input

5.3. Single order book input split into time steps

In this experiment, we split the input into equal number of parts and fed each part to a time step in the LSTM network. For example, we spit the 200 feature inputs into 10 parts of 20 each and fed it to the LSTM network with 20 time steps. The rationale here was to follow a similar pattern associated with examples from another domain where an input image was split into rows and each row was fed into an LSTM cell as a time step. Unfortunately, we didn't find any remarkable change in accuracy with this approach

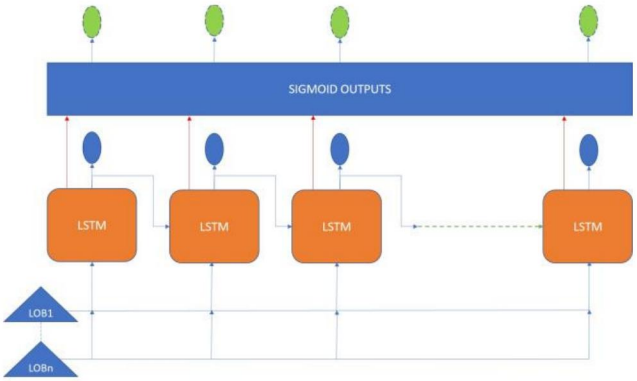


Figure 8: RNN Architecture with Single Limit order book spliced into multiple time steps

With the change to RNN architecture and experimentation, we were able to achieve 95% plus accuracy on the training set and the dev set accuracy improved to about 66% (a 3% increase compared to the FC network)

5.4. Categorical Model

Prediction of binary label is the simplest way to establish the correlation between the input dataset and the outcome but it is not a useful indication for successful trading algorithm.
We enhanced the output labels to 3 categories:

1. Increase by more than threshold percent
2. Decrease by more than threshold percent
3. Did not change by more than threshold percent

We created a signal with 4 dimensions so we can tune and find the best option. One dimension is the threshold (0.1%, 0.2%, 0.3%, 0.4%, 0.5%), second dimension is the future look ahead prediction (1min, 2min, 3min, 5min, 10min), other 2 dimensions are for the RNN time step (window size) and the batch.
We used time step of 4 and divided the data set to groups of 4 consecutive samples with overlap. Each input sample to the time distributed network is 4 samples of 200 dimensions representing 4 consecutive order book snapshots at 1 min intervals. For example, the first training example represent time (t_{-3}, t_{-2}, t_{-1}, t_0) and the second example represent time (t_{-2}, t_{-1}, t_0, t_1) etc'. This way the RNN gets the sense of time without the sense of artificial grouping. For the output we used one value that represent the trend of the bitcoin price for every sample. We only issue one label for the entire unrolled RNN of 4 time steps. This way, the model receive a sequence in time and the single result of this sequence. To select the label, we tried different options of look ahead times. Eventually, the best results are achieved for 2 min look ahead prediction and 0.2% threshold change.

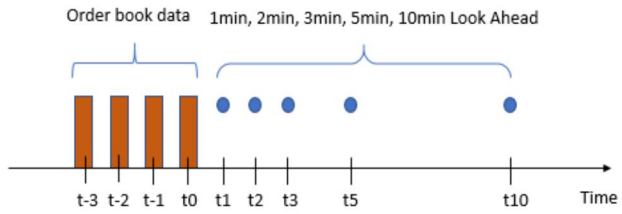


Figure 9: Categorical Model input / output preparation

5.4.1 Categorical model architecture

The architecture that has produced the best results so far has 3 convolutional layers (1 dimensional) connected to 2 layers of LSTM RNN with 256 hidden nodes each and one softmax layer with 3 states for the output.

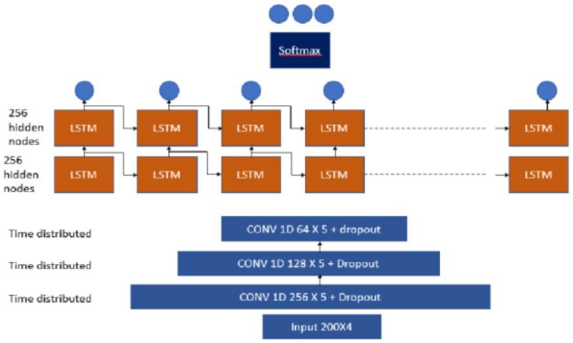


Figure 10: Categorical Model Architecture

The purpose of the convolutional layers is to smooth the extract features from the spiky order book sample.
When we started to train this network and we checked the labels for training and dev set we realized that the majority of the outputs are within the threshold.

	Increase	No change	Decrease
Training Set	3126	16310	3060
Dev Set	278	1928	290

Table 1: 25,000 samples label distribution

With this type of distribution it is easy for the model to achieve high accuracy percentage simply by predicting no change. To overcome this problem we modified the traditional loss function.

$$Loss = -\sum_{i}^{batch\ size} output * \log(prediction)$$

For outputs that are labeled as increase or decrease we multiply by factor of 2. We tried different factors for that and 2 came out to perform reasonable.

5.4.2 Categorical model results

So far, using the network described, we achieved 70% accuracy for the training set and 67% accuracy for the dev set.

	Increase	No change	Decrease
Ground truth	278	1928	290
Predicted increase	87	247	17
Predicted no change	181	1505	196
Predicted decrease	10	176	77

Table 2: Dev set results for categorical model

Total samples in the dev sets is 278+1928+290=2496 and the green cells are the one predicted correctly 87+1505+77=1660. 1660/2496=66.5%.

While these results do not seem as good as we initially hoped, there are few things to note. If we consider the no change label as kind of "don't care" than the prediction becomes much better. Meaning that for prediction of increase, there are 25% of actual increase, 70% of no change and only 5% of decrease.

5.4.3 Trading Simulation

To further evaluate the model we developed a trading simulation on the dev set. The trading strategy is very simple. We either hold Bitcoin or Dollars. Meaning that every trade is using the entire amount that we simulate. If prediction is '1' (increase) than but Bitcoin or hold Bitcoin. If prediction is '-1' (decrease) sell bitcoin or hold dollars. If prediction is '0' (within thresholds) hold position.

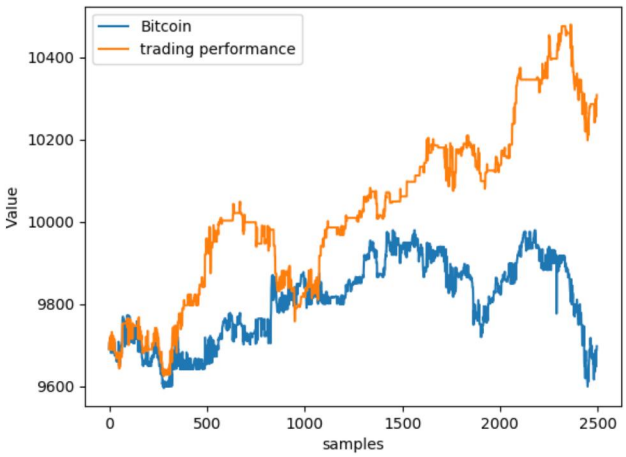


Figure 11: Trading Simulation

This simple simulation is zero trading cost model. It means that it ignores commissions as well as bid-ask spread. Under this model it proved itself profitable using our machine learning predictor running on the dev set gaining over 6% while Bitcoin itself gained 0%. We have tested the same on different test set data and it consistently outperforms the Bitcoin itself.

6. Conclusion

This project clearly demonstrated the correlation between limit order book and the future price. Starting from fully connected model of binary classification, continue to RNNs and concluding with categorical multiclass classification, we built different models and tested their performance on dev set. Although achieving over 70% accuracy on dev set is hard and we could only get close to that, the model that we have built on the categorical classification is consistently outperforming the Bitcoin itself under the zero trading cost model. Further work can be to consider the trading costs and find a profitable model.

7. References

[1] Tian Guo, NinoAntulov-Fantulin. Predicting short-term Bitcoin price fluctuations from buy and sell orders

[2] Huisu Jang, Jaewook Lee. An Empirical Study on Modeling and Prediction of Bitcoin prices with Bayesian Neural Networks Based on Blockchain information

[3] Muhammad J Amjad, Devarat Shah. Trading Bitcoinand Online Time Series Protection

[4] N.I. Indera, I.M.Yassinm A.Zabidi, Z.I.Rizman. Non-Linear AutoRegressive with Exegeneous Input (NARX) Bitcoin price prediction model using PSO-Optimized parameters and moving average technical indicators

[5] Justin A Sirignano. Deep Learning for Limit Order Books

[6] Tensorflow – Google's artificial intelligence system for large scale machine learning https://www.softwebsolutions.com/resources/tensorflow-googles-artificial-intelligence-system.html