**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**

*This midterm report reviews the problem that the project is intended to investigate and solve. It explains the dataset format and acquisition process along with the initial model that we started to build and evaluate. The last section describes the remaining activities, methods and tasks that we plan to do for completing the project*

# **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 intend to build a machine learning RNN (Recurrent Neural Network) that predicts the future price 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.

# **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.

# **Dataset Characteristics and Acquisition**

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

## **Limit order book**

Figure 1 in the next page is a ledger maintained by the exchange of all limit orders that are pending. The order book has a sorted list of all bid and ask orders with the quantity and associated price for each order. It is usually presented graphically as accumulative plot of all bid and ask orders.

The limit order book snapshot represent the demand and supply in the market in a certain point in time. In the above 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.

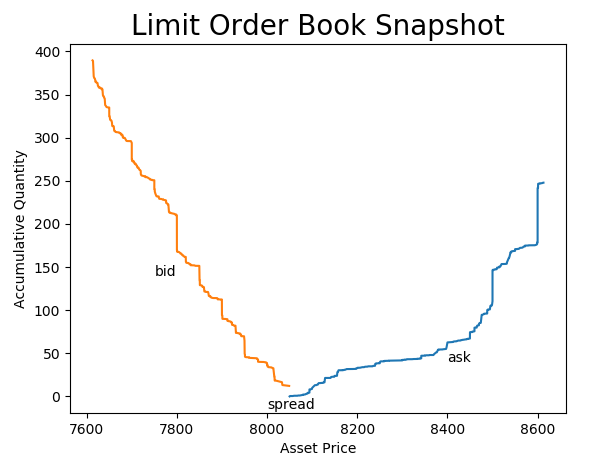
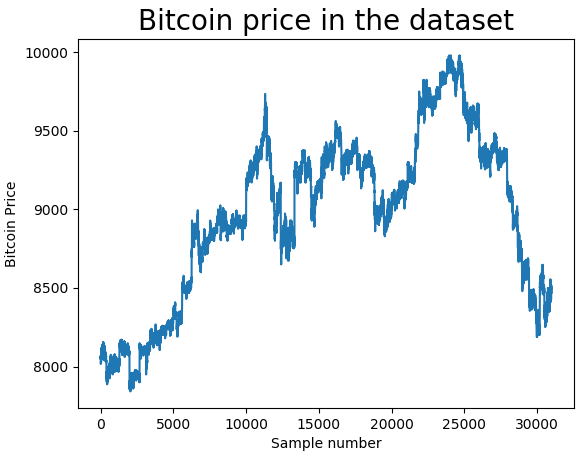


Figure 1: Limit Order Book Snapshot

## **Bitcoin historical price**

Apart from the limit order book we also 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 will serve both as features in the training examples as well as in generating the classifier for price increase or decrease. Consider a point in time ‘t0’ that corresponds to sample in our dataset ‘s0’. By considering certain number of examples (s-1, s-2, …, s-n) we get historical feature to the training set. By considering the samples (s1, s2, …, sn) we build our label for the classifier.



## **Order history**

In addition to the bitcoin price history and the limit order book history, we have data that represent the last 100 orders that were placed in the exchange. We plan to check if this data can contribute to the prediction. The data contains the number of bids and asks and the accumulative quantities of each. For example, one training example contains 60 bids at total of 5 bitcoins and 40 asks at total of 3 bitcoin. The delta time of these last 100 orders is also known to us and might add value.

## **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 far over 20,000 samples that represent 2 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

Sample raw Sell Order Book data from API - <https://bittrex.com/api/v1.1/public/getorderbook?market=USDT-BTC&type=sell>

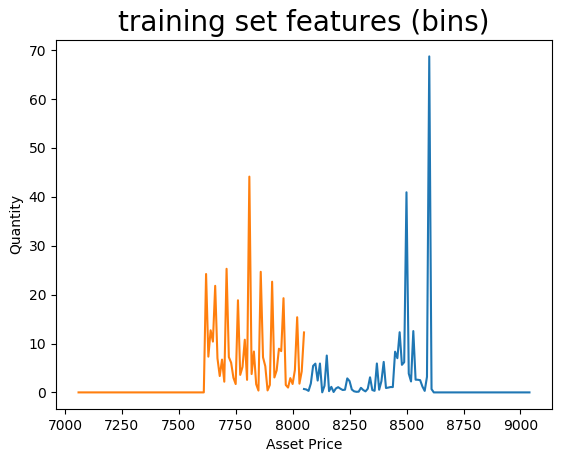
Sample raw Buy Order Book data from the API - https://bittrex.com/api/v1.1/public/getorderbook?market=USDT-BTC&type=buy

# **Initial model**

As a starting point we use only the limit order book to predict future price increase or decrease and we use only one snapshot of the order book meaning that we predict a future change based on the current status without looking at the history.

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$. 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 2: sample of one training example after structured in bins 

## **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

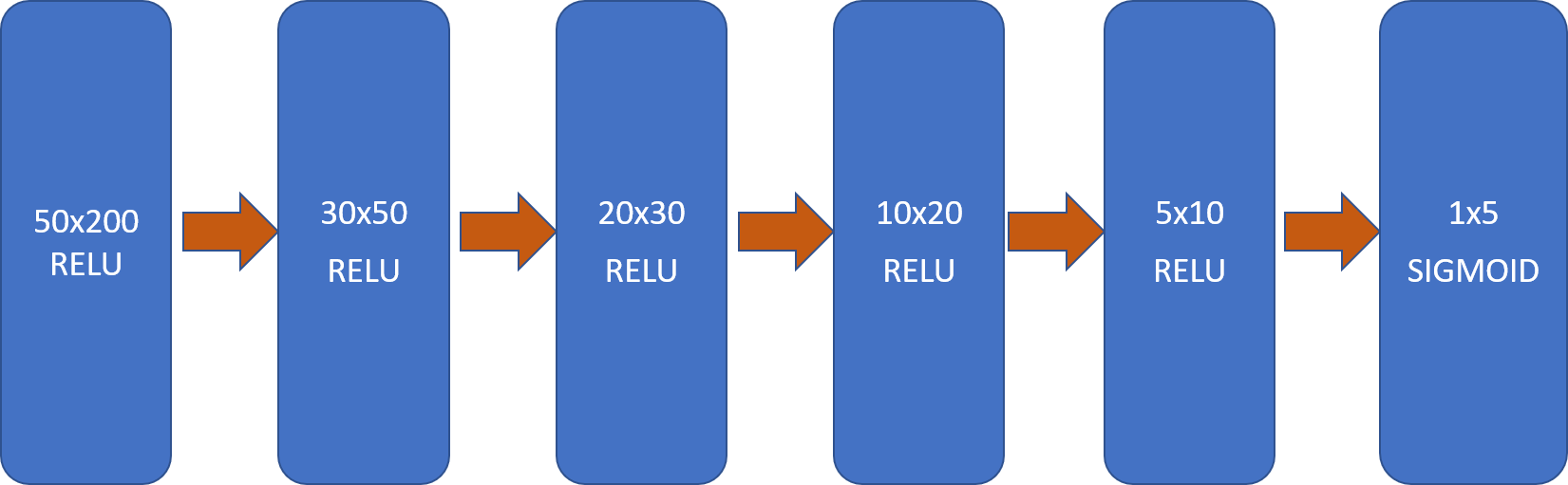


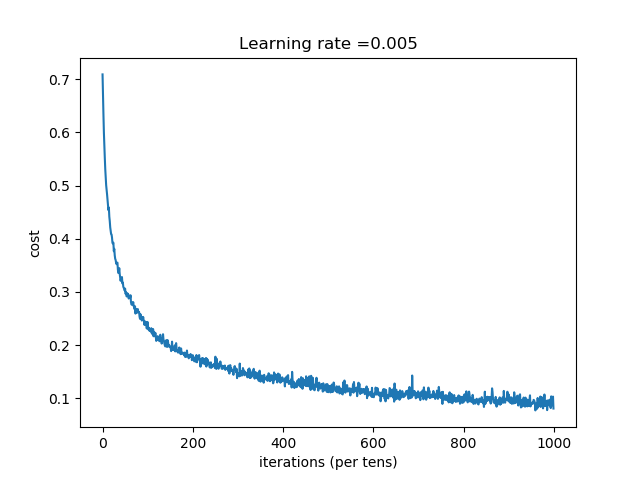
Figure 3: NN Architecture

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

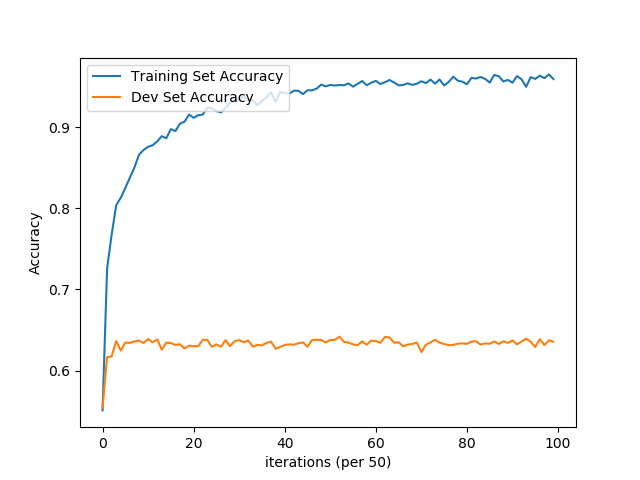
## **Initial Results**

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.

Below are some of the results associated with the final numbers after tuning the hyper parameters



**Figure 4: Cost Plot**



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

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.

# **RNN**

To develop the best predictor for future Bitcoin price we tried different approaches and RNN architectures.

Ashwin to describe his TensorFlow network

## **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 (t0, t1, t2, t3) and the second example represent time (t1, t2, t3, t4) 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.

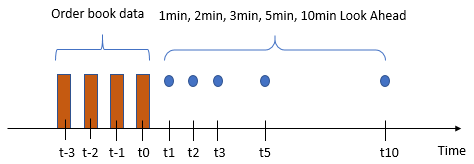


Figure 3: Categorical Model input / output preparation

### 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.

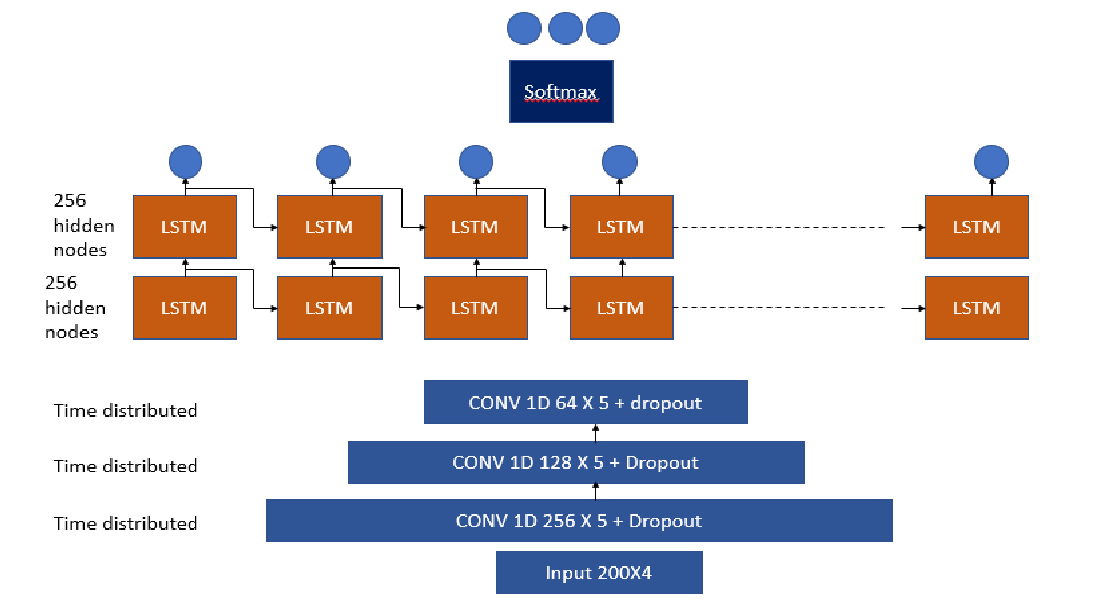


Figure 4: Categorical Model Architecture

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.

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.

### Categorical model results

So far, using the network above, 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.

**6. 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