London School of Economics and Political Science

ST456 - Deep Learning

High Frequency Research on Limit Order Book (LOB) with CNN and RNN

Charles Yuan Chang

Abstract

Classical trading strategy in low frequency data has limited arbitrage opportunities today. We need more advanced machine leaning and deep learning techniques to investigate high frequency data from the limit order book and form a profitable trading strategy. Therefore, we developed three models with deep neural network to analyse high frequency trading data and use Tensorflow to realize it. The architectures include Simple CNN, CNN with Inception and LSTM, Multi-input CNN with Inception and LSTM. We combined the high frequency 3-month limit order book data of TWTR, UBER, and TSLA as our training data, and use 1-month three different stocks to predict and evaluate the performance of our model. The result shows that the Simple CNN, and CNN with Inception and LSTM can achieve better result, while the Multi-input CNN with Inception and LSTM is less accurate and less steady. The contrast in the performance illustrates the universal property of the prediction models for financial market.

Keywords: LOB; CNN; LSTM; Universal; Tensorflow

1 Introduction

Financial markets offer discerning investors an opportunity to buy undervalued assets and short overvalued assets. People can gain investment return by inspecting the market and determining what is necessary and possible to do to generate the greatest profit with the lowest risk. The most popular mathematical models and quantitative finance methods to predict various aspects of financial markets are time series analysis, which allows us to make valuable predictions about various aspects of the market, such as volatility, trends or the actual value of an asset. However, these mathematical models rely on hand-crafted features and manually adjust their parameters through observation, which results bias in the predictions.

In addition, the development of modern trading system have greatly increased the volume of transactions that take place on a daily basis, thereby increasing the amount of data generated within exchanges. This allows financial institutions to analyse market situation and forecast the trend of individual stocks or certain portfolios, then form some profitable trading strategies with high frequency data. The high frequency trading data conveys precious information which is useful in market trend prediction and can be used to make the trading decisions simultaneously with proper algorithm. However, using machine learning to analyse such a large-scale data has not been easy. One obvious reason is that there are many noise trader in the market, which adds abnormal changes to the asset price movements. Another reason is that the stock prices are not only decided by its value and past trading volume, but also largely influenced by some unpredictable events such as political factors and pandemics, which cannot be captured by quantitative models.

Limit order in high frequency stock trading a is a type of order to purchase of sell a particular amount of shares at certain prices. For example, a sell limit order (ask) of \$100 with volume of 1000 means that the seller wants to sell the 1000 shares for no less than \$100 per share. Respectively, a buy limit order (bid) of \$100 means that the buyer hopes to buy a certain number of shares for no more than \$100 each. Obviously, there are two sides in the order book, the bid side, which includes buy orders with prices $P_b(t)$ and volumes $V_b(t)$, and the ask side, which includes sell orders with prices $P_a(t)$ and volumes $V_a(t)$. The orders are sorted on both sides based on the price. $p_b^{(1)}(t)$ is the is the highest price on the bid side and $p_a^{(1)}(t)$ is the lowest price on the ask side.

In this paper, we use 3 different deep learning architectures and compare the price prediction results of them. Model 1 contains the CNN, Inception Module, and LSTM layers. Model 2 only contains CNN layers. The first two model learn from the architectures designed by Zhang (2020) and Tsantekidis (2017) respectively. To highlight the property of universality of

the first two models, we also design Model 3 which has multi-input CNN with Inception and LSTM layers by ourselves. The result shows that Model 1 and Model 2 have overall better performance compared to Model 3. The main contribution of this work is that we compared different deep neural network models in predicting the trend of high frequency stock price and identified the effective architectures. This could be used to form possible trading strategies with reinforcement learning.

2 Related work

Many of the past research focus on stock price prediction (Ang, 2006). The stock market is predictable to some extent although there are many different views of the efficiency of the market (Bollerslev & Marrone, 2009). One view is that the financial data is predictable only when the market is in the weak form of efficiency (Ferreira & Santa-Clara, 2011). Another view is that the market is predictable in general and it is not related to the efficiency of the market (Mandelbrot & Hudson, 2007).

In terms of the analytical methods, there are mainly two streams in financial time series data, statistical parametric models, which are under the assumption that time series comes from a parametric process (Agrawal etc, 2013), and machine learning techniques, which can catch and distinguish characteristics on a longer time horizon (Cavalcante etc, 2016).

With the development of deep learning techniques and improvement of computational power, researchers tend to use more sophisticated architectures in analysis. It is well acknowledged that deep learning can improve the result accuracy compared to ordinary machine learning techniques in some specific areas (Tsantekidis, 2018). As famous deep learning models, convolutional neural networks and recurrent neural networks have made a significant contribution to those areas with much more accurate results by performing better feature identification and summarisation (Dixon etc, 2017).

Particularly, the filters in CNN can capture spatial patterns. It has been widely applied to many fields but less in finance (Chen, 2016). However, unlike traditional financial time series data, the limit order book has some spatial patterns because of the different price levels at a certain time point (Doering etc, 2017). It is possible to use CNN to extract the spatial patterns that existed in the high-frequency limit order book data (Tsantekidis, 2017). Meanwhile, Long Short-Term Memory (LSTM) was initially invented to solve the vanishing gradients in a long series of data and was widely applied in speech recognition and text generation (Bengio, 1994). Due to its adaptivity to series data, many researchers use it to model financial data together with other modifications, which improves the overall result to a large extent (Zhang etc, 2019).

3 Data process

We extracted the trading information of TWTR, UBER, and TSLA during 01/09/2021 – 30/11/2021 from the NYSE Trade and Quote (TAQ) database , which contains intraday transactions data (trades and quotes) for all securities list on NYSE, AMEX, as well as Nasdaq (NMS) and SmallCap issues, from the WRDS database. The data from the TAQ database can be viewed as a limit order book (LOB), which contains information of bid orders and ask orders. For the definition, a limit bid order refers to an order for investors to buy a certain volume of stocks at a certain price; a limit ask order refers to an order for investors to sell a certain volume of stocks at a certain price. In our problem context, we do not consider some potential influential factors, such as order cancellation, dark pool, hidden liquidity. We only downloaded the ask price, ask volume, bid price, bid volume of the stocks. The TWTR, UBER, and TSLA are three of the most liquid stocks in the market so that the price formation process will be less affected some unexpected outlier order price, which is beneficial for us to train a model to predict the price movement of them.

3.1 Input Data

We denote the price of bid side as $P_b(t)$, the volume as $V_b(t)$, the price of ask side as $P_a(t)$, the volume as $V_a(t)$. We choose the best ten bid prices and their corresponding cumulative volumes, which are the highest ten prices from bid side and therefore the most likely to be traded. For ask side, we choose the lowest ten prices and their corresponding cumulative volumes. In this way, we can organize our data as the form of $P_a^i, V_a^i, P_b^i, V_b^i$, which $i = 0, 1, ..., 9$.

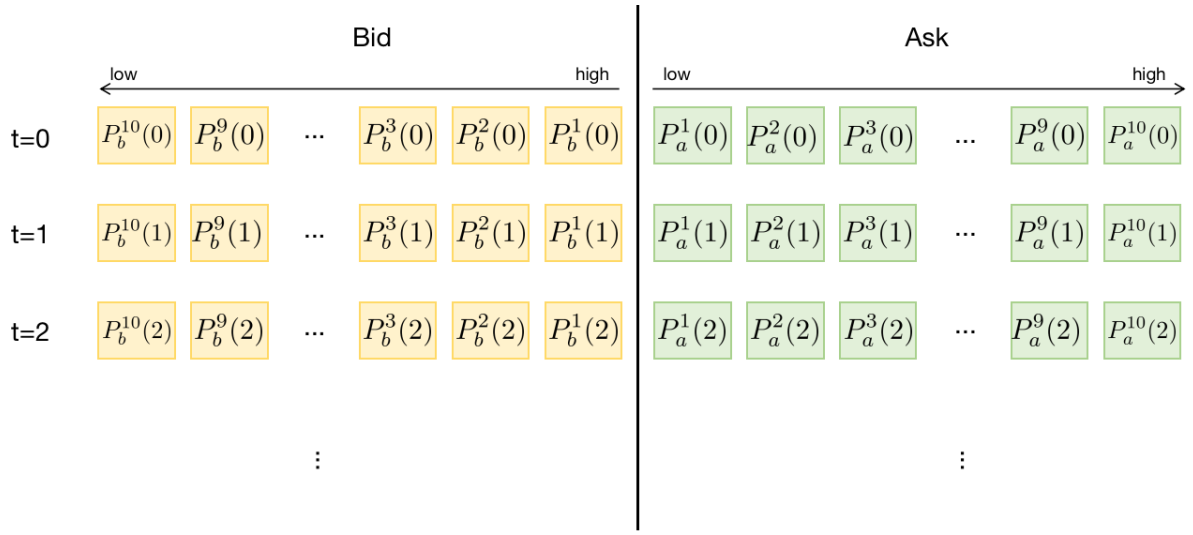


Figure 1: Structure of the Price Data

Step1: To make sure our input dataset contain the characteristic of high frequency and also be tractable, we decide to sample at every 10 seconds. In this way, we will get about 2520 timestamps every day (6 samples per minute, 360 sample per hour, 2520 sample per trading day). Another problem we met is that we cannot get enough orders to form the order pool with 10 best bid prices and 10 best ask prices if we only include orders sampled every 10 seconds. We solve this problem by adding orders happened on one and two seconds before and after the every precise 10 seconds and categorizing the orders on the four extra seconds to the precise 10-second timestamps.

Step2: After step 1, we get a dataframe with approximately less than 2520 timestamps a day. (for example, trading gets inactive during 16:00 - 16:30, therefore, the data may be vacant in this period for many days so that it is obviously less than 2520.) In the second step, we need to organize the data contained in each timestamp into the structure of $P_a^i, V_a^i, P_b^i, V_b^i$, which $i = 0, 1, ..., 9$. Thus, we just sort the bid (ask) price and choose the best ten bid (ask) prices. And then we sum up the order bid (ask) volume of each bid (ask) at the timestamp and arrange them in the sequence of bid_price_i, bid_volume_i, ask_price_i, ask_volume_i.

Step3: We are going to use the 100 most recent states of the LOB as an input to our model. Thus, we restructure our data to the form of many 100*40*1 matrix by the method of rolling window.

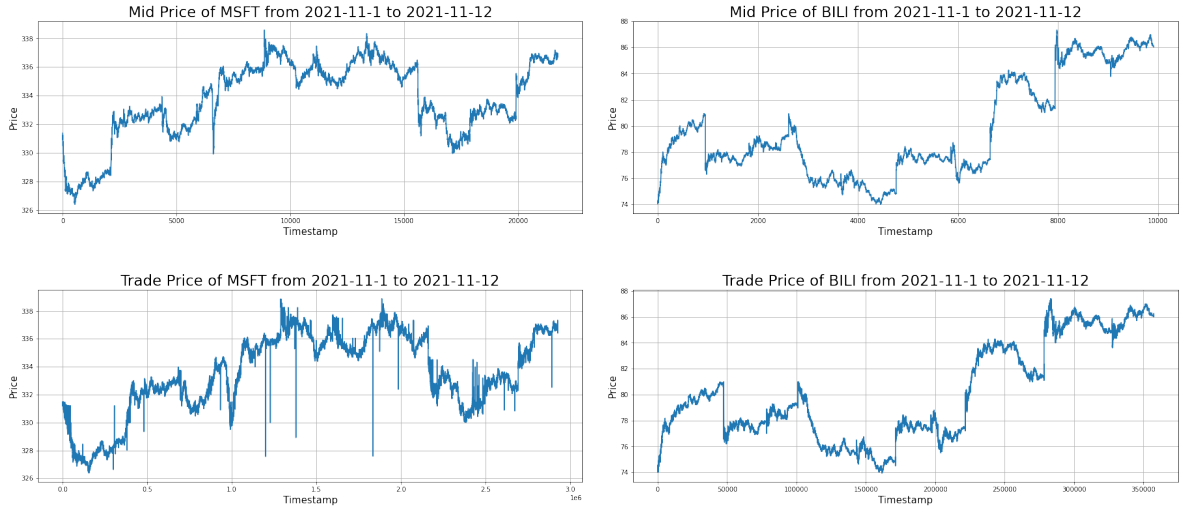
3.2 Data Labelling and Smoothing

First, we calculate the mid price of each timestamp, which is the average of the best bid price and the best ask price,

$$p_t = \frac{p_a^{(0)}(t) + p_a^{(0)}(t)}{2} \tag{3.2.1}$$

to create labels that represent the direction of price changes. Although no order can transact exactly at the mid-price, it expresses a general market value for an asset and it is frequently quoted when we want a single number to represent an asset price. Figure 2 represents the comparison between the mid price calculated from raw quotes and the price of trade that exactly happened. We can see from the plots that the price of real trades exhibits same trend with the mid price we calculated precisely but is more noisy and more dense than the mid price we calculated. And sometimes they do not act exactly synchronous because there are some missing timestamps of our data due to inactive trading during certain time period. However, it will not produce large negative influence on our model because our model is quite invariant

to time adjustment of the data.



(a) MSFT (b) BILI

Figure 2: Visualization of Cleaned LOB Data

Because the stochastic property of high frequent trading data, the price movement will be greatly unsteady if we denote the increase or decrease by the difference by p_t and p_{t+1} or p_{t+k}. Nevertheless, a noisy price movement means we need to adjust our position very frequently, which will cause great transaction cost in real trading scenarios. Therefore, we need to apply a smooth process to the mid price to label the price movement. First, let $m_-(t)$ denote the mean of the previous k mid-prices and $m_+(t)$ denote the mean of the next k mid-prices:

$$m_-(t) = \frac{1}{k}\sum_{i=0}^{k} p_{t-i} \tag{3.2.2}$$

$$m_+(t) = \frac{1}{k}\sum_{i=1}^{k} p_{t+i} \tag{3.2.3}$$

where p_t is the mid-price calculated from in Equation 3.2.1 and k is the prediction horizon. To define the price movement of each stock, we choose to use the percentage change to of the mid price to decide the movement direction because of different scales of each stock.

$$l_t = \frac{m_+(t) - m_-(t)}{m_-(t)} \tag{3.2.4}$$

Then we need to choose a threshold for l_t, if $l_t > \alpha$ or $l_t < \alpha$, we define the price movement as up (+1) or down (-1); If the absolute value of l_t is smaller than the α, we regard it as stationary (0). In our context, we choose α as 10^{-4} to label the price movement.

3.3 Data Normalization and Prediction Horizon

After labelling the price movement based on the mid price trend, we should normalize our data before fitting it into the machine learning models. Due to the property of high volatility and potential regime change of finance data, we should not to normalize the data in the traditional method of static normalization. Instead, we apply dynamic normalization to the data with normalization window equal to the prediction window using $z - score$ method, which is deducting the mean of k before data and dividing the result by the standard deviation of k before data.

As for the prediction window k, we choose $k = 10, 20, 50$, which sets us with different horizons. Figure 3 shows the label result of the first 2000 timestamps of TWTR. Green represents label (+1), red represents label (-1), white represents label (0). We can also see that longer the prediction horizon is, more steady the price trend is so that we do not need to change our position frequently. For the whole procedure, we firstly fit the data into equation (2) (3) to get the price movement label. After that, we normalize our data using the dynamic regime to get the final input of the model.

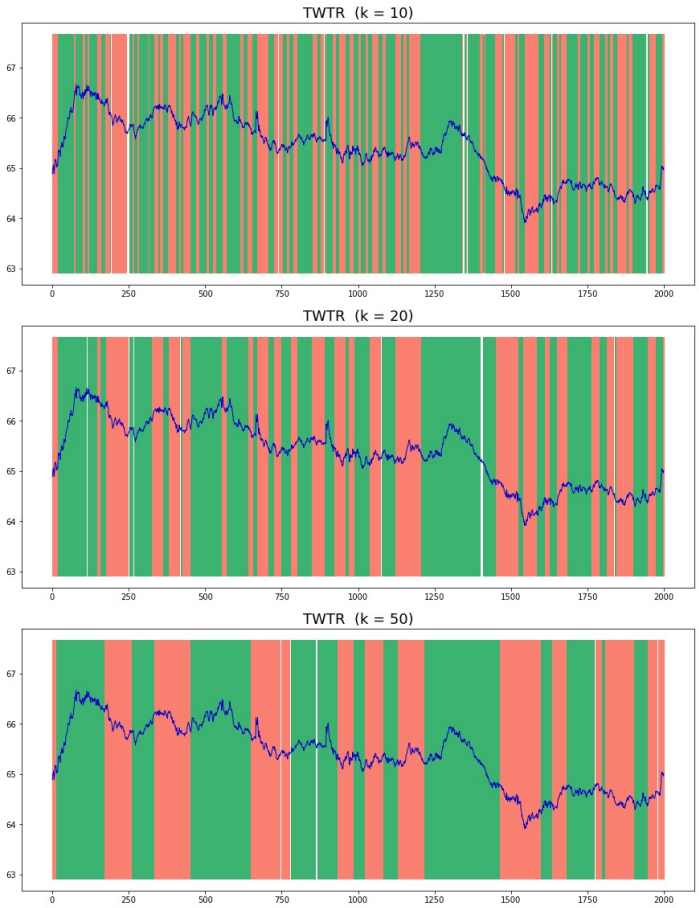


Figure 3: Label of the Stock Price Movement

4 Model Description

We build 3 different neural network architectures, which comprises standard convolutional layers, an Inception Module, an LSTM layer, and a fully connected layer, as seen in Figure 4. Standard CNN layers and Inception Module are used to capture the feature along the second dimension of the inputs (prices and volumes of 10 best bid prices and 10 lowest best ask price at each sample point), and the LSTM layer is used to capture the time feature of input data.

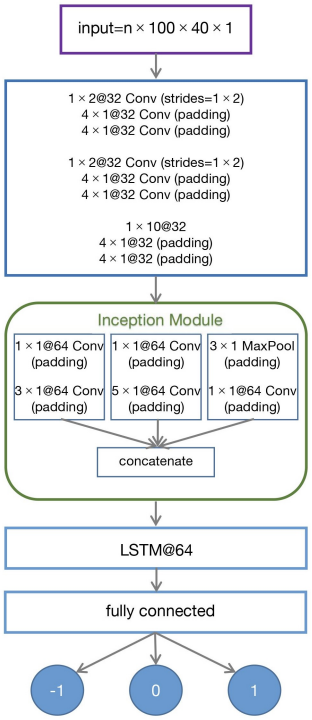
4.1 Introduction of Each Layer

4.1.1 Convolutional Layers

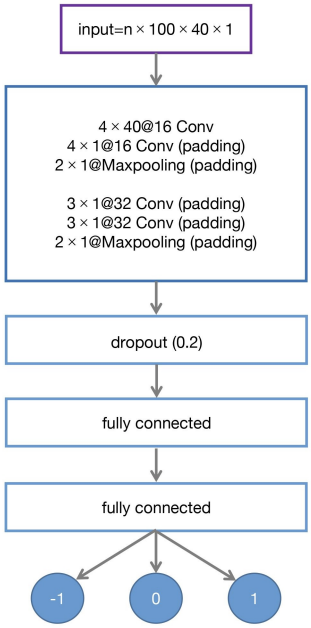
CNNs are most commonly applied to analyze visual images, which are based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses. Here we reasonably assume that our historical data of bid and ask prices and volumes somehow show spatial structure which can be captured by using CNN.

For each sample point, we take 100 consecutive updates from the order book, each of which contains 40 features and the 40 features can be denoted as $p_a^{(i)}(t), v_a^{(i)}(t), p_b^{(i)}(t), v_b^{(i)}(t)_{i=1}^{n=10}$, where i is the i-level of a limited order at a time point, p means prices, v means volumes, a means ask, and b means bid. Here CNN can be used to capture the features along this dimension and the information from ask and bid prices and volumes could be processed and combined together.

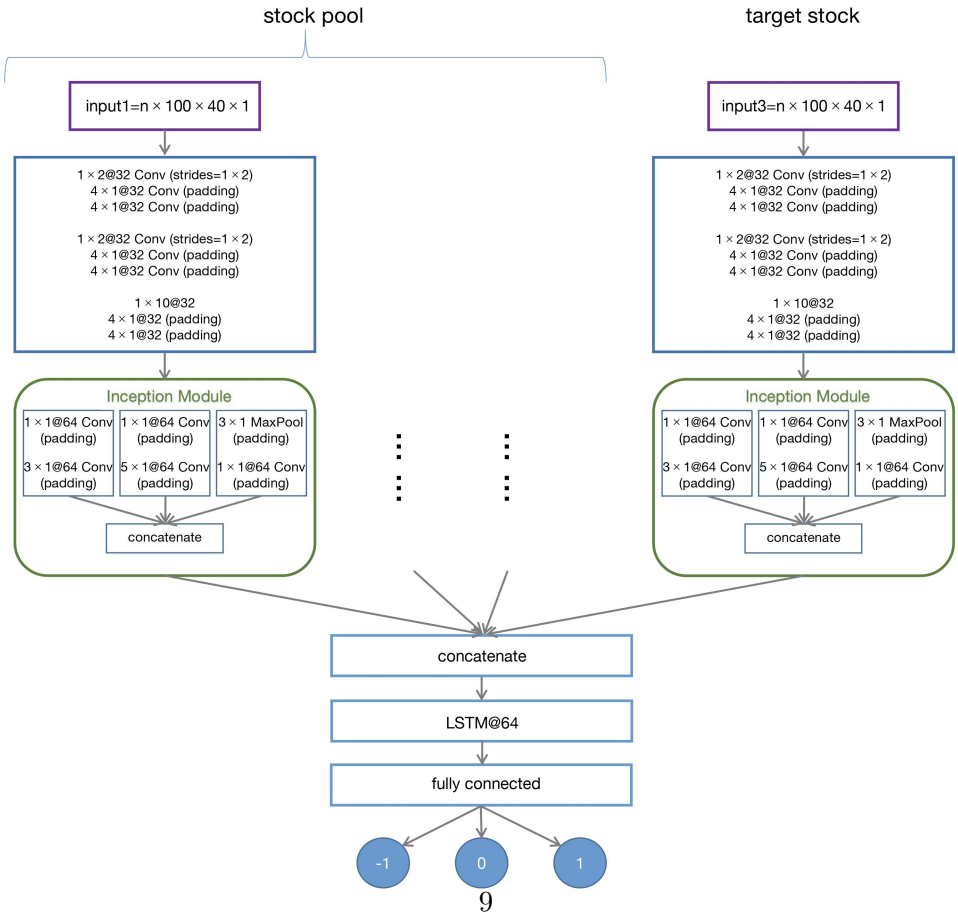
The first CNN layer applies $(1 * 2)$ filters with strides of $(1 * 2)$, which captures and combines the information between prices and volumes for bid and ask sides and for each level respectively, namely $p^{(i)}, v^{(i)}$. The fourth CNN layer also applies $(1 * 2)$ filters with strides of $(1 * 2)$, which captures and combines the information between processed ask side and bid side for each level. The seventh CNN layer applies $(1 * 10)$ filters, which combines the information among all 10 levels. Moreover, two more CNN layers with $(4 * 1)$ zero padding filters also follow each CNN layer mentioned above, which has a function of capturing the local feature over four time steps along the time dimension and not changing the length of time dimension at the same time. Furthermore, we use Leaky-ReLu as the activation functions in this architecture. All the CNN layers in this part have 32 filters. Therefore, the dimension of output is $(100 * 1 * 32)$.



(a)



(b)



(c)

Figure 4: Architectures of 3 models

4.1.2 Inception Module

Inception Module was first published by researchers in Google in 2014. It can extract features at varying scales, and can solve the problem of computational expense, as well as overfitting. Therefore, in our architecture, by using Inception Module after the standard CNN layers, we can capture dynamic behaviours over multiple timescales. We split the data, which has been processed by the above standard CNN layers, into a small set with lower dimension by using $(1 * 1)$ CNN layer. Then, we first use CNN layer with $(3 * 1)$ zero padding filter to obtain the dynamic temporal feature over 3 time steps and use CNN layer with $(5 * 1)$ zero padding filter to obtain feature over 5 time steps. A max-pooling layer is also used in our Inception Module. Finally, we combine those branches up. Compared with the standard CNN layers, Inception Module is able to capture non-linear properties of the data, which may significantly improve the accuracy of the architecture. All the CNN layers in Inception Module have 64 filters. Therefore, after concatenating the 3 branches, the dimension of the output $(100 * 192)$.

4.1.3 LSTM Layer

Long Short Term Memory (LSTM) Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. It can process not only single data points, but also entire sequences of data. The LSTM layer in our network has two essential functions. First, it significantly reduces the parameters to be estimated compared with a simple fully connected layer with units of the same number. Second, an LSTM layer is able to further capture the dynamic temporal structure of the data whose spatial structure has been captured by standard CNN layers and Inception Module. The number of units we set is 64. Therefore, the dimension output of this layer is 64.

In the last step, we apply an output dense layer with activation function as 'softmax', so we can obtain the probabilities of three kinds of movements at each time step and the kind of movement with the largest estimated probability can be seen as our prediction.

4.2 Introduction of the Three Models

4.2.1 Model 1: CNN with Inception and LSTM

Combining the layers introduced above, we construct Model 1 as our standard model.[13] The input samples of both models are consecutive 100 time steps of prices and volumes from ask and bid sides and the output is the probability of 3 kinds of movements of smoothed stock

price at the 101st time step. The models learn this pattern by training on historical data of multiple stocks over time, but not just one stock. Therefore we can say that Model 1 are able to capture the relationship between the features in the first 100 time steps and the movement at the 101st step over all stocks in the market, which can be seen as a kind of universality. The fact that both models can extract robust features from the order book also indicates the existence of universal features in the order book that modulate stock price.

4.2.2 Model 2: Simple CNN

In order to make a comparison with Model 1, we construct another Model 2 which also has the property of universality but has a quite different structure.[4] In model 2, the information along the feature dimension is roughly captured in the first CNN layer with filters of $(4 * 40)$ in one time. Then three more CNN layers with $(4 * 1)$, $(3 * 1)$, $(3 * 1)$ filters respectively are used to capture the information along the time dimension. Furthermore, LSTM layer in Model 1 is also replaced by a simple Dense layer.

4.2.3 Model 3: Multi-input CNN with Inception and LSTM

In order to make a comparison with models with the property of universality, we also construct an architecture without the property of universality which is Model 3. First we assume that there exists a connection between a stock's behaviour during the first 100 time steps and the movement of itself at the 101st time step which is also the idea of Model 1 and Model 2. However, in Model 3 we don't expect the universality of this connection among all the stocks in the market. We assume that different stocks have different patterns for predicting the movement at the next time step according to the previous. Furthermore, we believe that only the information of the stock itself is not enough to predict, so we further add market information into the model. We set up a stock pool containing 3 of the most liquid stocks in NYSE: TWTR, UBER, and TSLA which can be seen as representatives of market information. Therefore, now we can assume that there exists a specific relationship between the behaviours of the stock itself and the stock pool in the first 100 time steps, and the movement at the 101st time step. So we can predict the movement of a stock given the previous behaviour of them. Moreover, we assign different weights on different stocks in the stock pool and the stock itself, thus we set different inputs in the model using functional API. Because of the exclusiveness, we need to train multiple times to obtain multiple weights for multiple target stocks.

5 Model Implementation

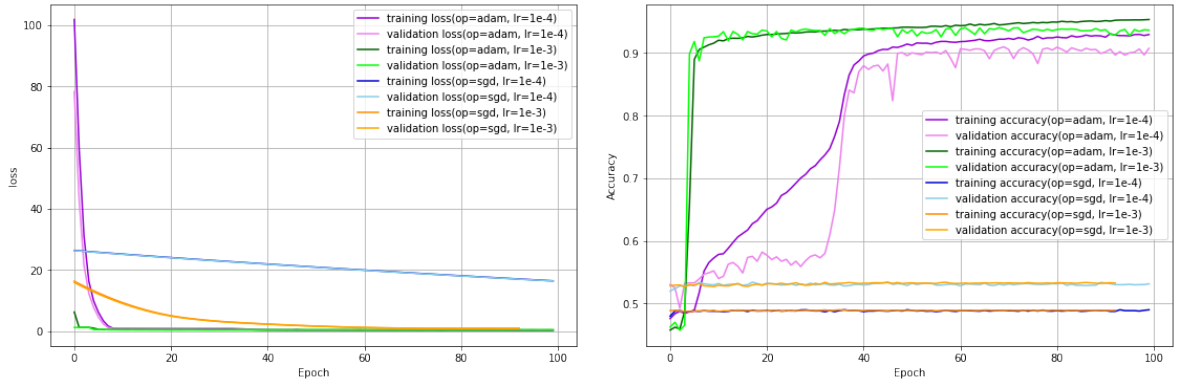
- For Model 1 and Model 2, We implement training on processed data of TWTR, UBER, and TSLA from 01/09/2021 to 30/11/2021. Then we apply transfer learning on three new stocks, BILI, AAPL, and MSFT from 01/11/2021 to 12/11/2021 to test the test accuracy and universality of the first two models.
- For Model 3, using TWTR, UBER, and TSLA, three of the most liquid stocks in NYSE, as our stock pool, we predict the movement of BILI and IBM using processed data of themselves and the stock pool from 01/09/2021 to 30/11/2021. Then we test Model 3 using data from 01/12/2021 to 14/12/2021.

5.1 Experiment Setting and Selecting

We learn the parameters of three models by minimising the categorical cross-entropy loss and set batch size as 128, number of epochs as 100, and the rate of validation split as 0.25. Furthermore, using checkpoint function in keras, models save the best weights when the validation losses are minimised. Using earlystopping function, models stop automatically when losses do not improve for 10 more epochs. Furthermore, when facing the problem of overfitting, we add a kernel regularizer with L1 regularisation factor of 0.05 to apply penalties on layer parameters or layer activity during optimization.

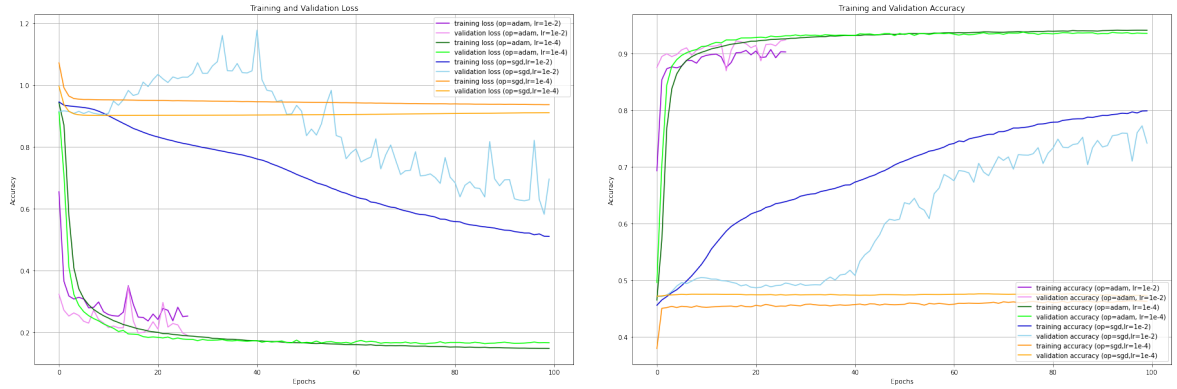
Since the setting of the optimizer and learning rate is essential to the neural network model, we adjust these two tuning parameters to optimise the model implementation according to training loss and accuracy, validation loss and accuracy when prediction horizon $k = 10$. We assume that for each model, best tuning parameters also apply to situations when $k = 20$ and $k = 50$. We choose the optimizer from Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam), and choose learning rate from 10^{-4} and 10^{-2} or 10^{-3}. Here we plot the loss and accuracy for different optimizer and parameter settings.

From Figure 5, 6, 7, we want to choose the best parameter under validation set for fitting the models. And We further assume that these settings should also be applied to other situations when $k = 20$ and $k = 50$ and we will conduct our models using the best parameters in the next analysis.



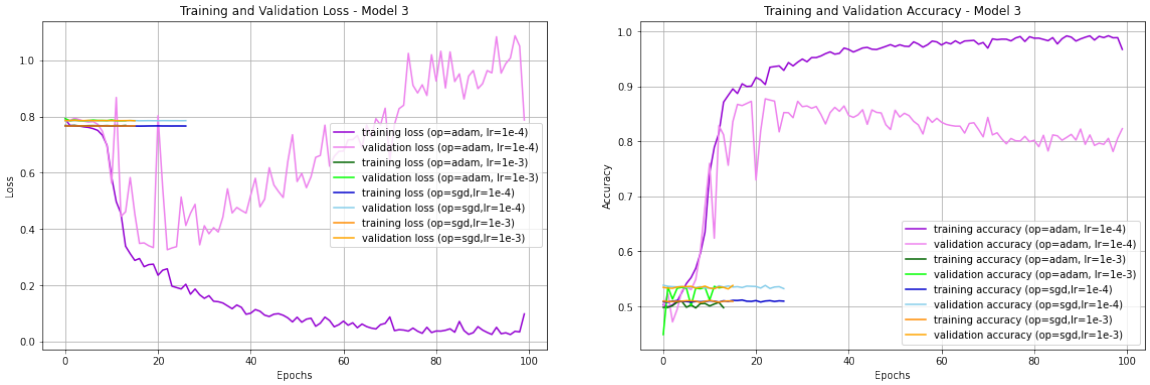
(a) Loss - Model 1 (b) Accuracy - Model 1

Figure 5: Model 1 comparison over Optimizer and Learning Rate



(a) Loss - Model 2 (b) Accuracy - Model 2

Figure 6: Model 2 comparison over Optimizer and Learning Rate



(a) Loss - Model 3 (b) Accuracy - Model 3

Figure 7: Model 3 comparison over Optimizer and Learning Rate

- For Model 1, model with optimizer of Adam and learning rate of 10^{-3} performs the best, which is slightly better than the model with learning rate 10^{-4}. What is special here is that we would still choose learning rate of 10^{-4} for the other prediction horizons of $k = 20$ and $k = 50$ because $lr = 10^{-3}$ only performs slightly better than model 1 with

$lr = 10^{-4}$ and Model 1 with $lr = 10^{-3}$ does not perform well under other prediction horizons and triggers early stop within the first few steps.

- For Both Model 2 and Model 3, model with optimizer of Adam and learning rate of 10^{-4} is the best. Especially, we can see that, only when optimizer is Adam and learning rate is 10^{-4} can Model 3 learn till the last epoch. Otherwise, Model 3 cannot learn well and stops early when the number of epochs goes to about 20. However, there exits a slight problem of overfitting, which may be caused by the exhaustiveness of the architecture. By using checkpoint, we save the weights when validation loss reaches it minimum which is a good way to alleviate this problem..

5.2 Apply Models to Different Prediction Horizons k

Here the prediction horizon k means the k in Equation (3.2.2), which is a parameter of the length of previous and next time steps for telling whether the stock goes up or down when labelling y. We find out that k has an obvious impact on the performances of three models. Here we plot the validation accuracies of three models versus $k = 10$, $k = 20$, and $k = 50$ using the best tuning parameters selected above.

From Figure 8, We can see that all the three models have a good performance of predicting the stock price movement. But there are some interesting and inspiring discoveries among them:

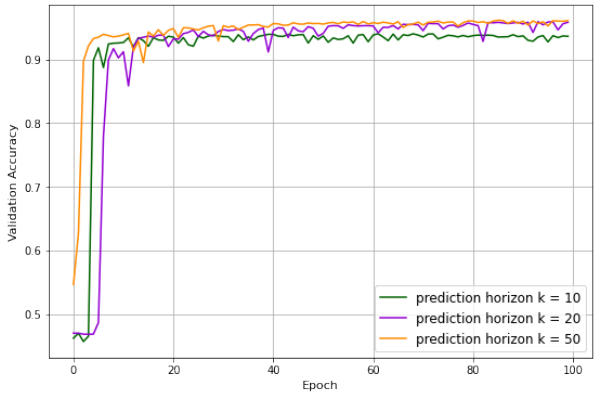
1. Model 1 and Model 2 achieve higher accuracy for all prediction horizons than Model 3. And the performance of Model 1 and Model 2 is more steady among the 100 epochs while Model 3 exhibits overfitting problem after around 50 epochs.

   The reason is that the first two model follow the principle of universal property of the prediction model, the main idea of which is to capture the general features that can affect price movement in the whole market, while Model 3 tries to explore whether we can make use of some idiosyncratic features of some stocks to predict price movement of another specific stock.

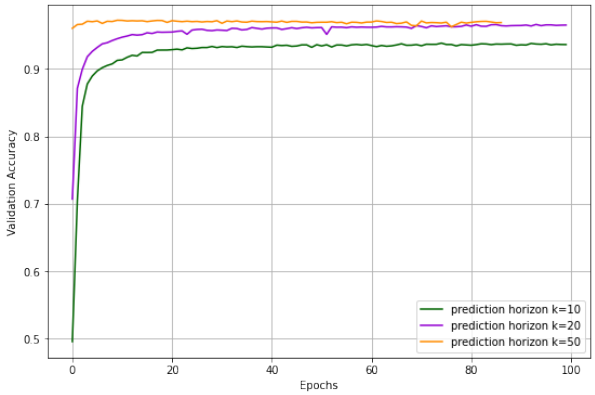
2. The accuracy of validation set for Model 1 and Model 2 has sharp increase in the first few epochs and keeps the high level. The accuracy, however, is more steady for Model 2 than Model 1 for all three prediction horizons.

   The reason is that Model 2 only makes use of convolution layers and has fewer parameters than Model 1. And from the comparison, we can say that the convolution layers can capture most of the features that can be influential to the movement of stock price.

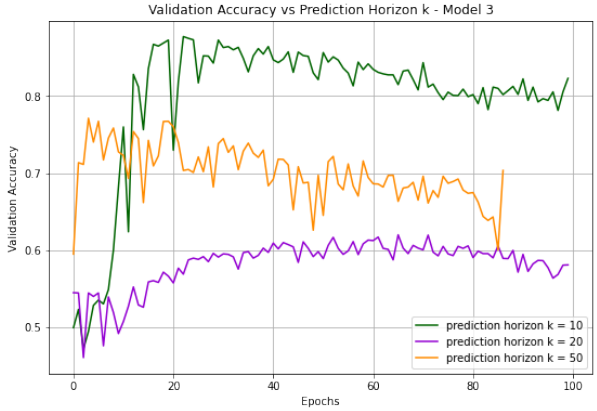
3. For Model 1 and Model 2, the prediction performance for prediction horizons ranks as $k = 50 \geq k = 20 \geq k = 10$. And the validation accuracy converges after around 90 epochs for $k = 20$ and $k = 50$. Nevertheless, for Model 3, we can see that all three lines in the plot fluctuate much and in the practical training processes we do find out that the performances of Model 3 show randomness. Therefore, we cannot compare the effects of different k of Model 3 just in one training. Multiple training for different k will be done in the next session.



(a) Validation Accuracy vs k - Model 1



(b) Validation Accuracy vs k - Model 2



(c) Validation Accuracy vs k - Model 3

Figure 8: Validation Accuracy vs k

6 Empirical Evaluation

6.1 Prediction Score Metrics

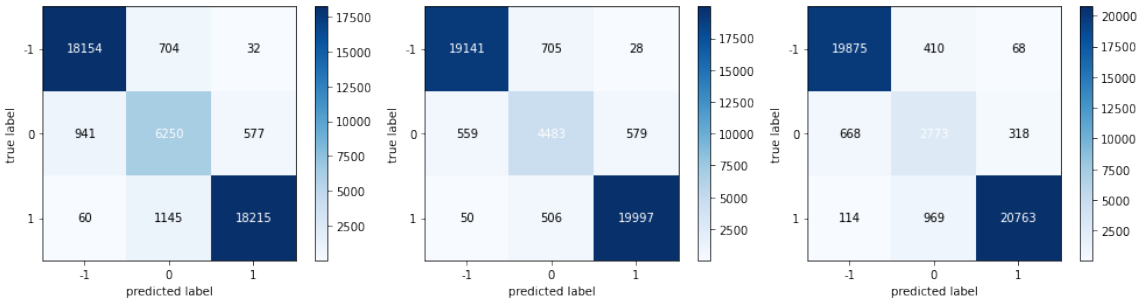
To test the stability of our models, we apply transfer learning to new stocks: BILI, MSFT, AAPL from 01/11/2021 to 12/11/2021 for Model 1 and Model 2, as well as BILI, IBM from 01/12/2021 to 14/12/2021 for Model 3. To illustrate the prediction results more concisely, we put the prediction results, including accuracy scores, precision, recall, and $F1$, of the new stocks together into the following confusion matrix for $k = 10$, $k = 20$, and $k = 50$. More specifically, test data from BILI, MSFT, and AAPL for Model 1 and Model 2 are simply concatenated together for prediction, while for Model 3, because of the idiosyncratic property, we need to train the model on BILI, IBM separately. The accuracy score, precision, recall, and F1 of Model 3 are the arithmetic means of BILI and IBM.

Table 1: Experiment Result of the Three Models

Prediction Horizon	Accuracy %	Precision %	Recall %	F1 %
Model 1 on New Stocks				
$k = 10$	92.31	92.43	92.31	92.35
$k = 20$	94.70	94.74	94.70	94.72
$k = 50$	94.19	94.44	94.19	94.29
Model 2 on New Stocks				
$k = 10$	92.55	92.38	92.55	92.44
$k = 20$	94.73	94.71	94.73	94.72
$k = 50$	95.65	95.62	95.65	95.63
Model 3 on New Stocks				
$k = 10$	54.92	52.80	54.93	52.95
$k = 20$	68.16	67.14	68.17	66.89
$k = 50$	84.50	83.11	84.50	83.77

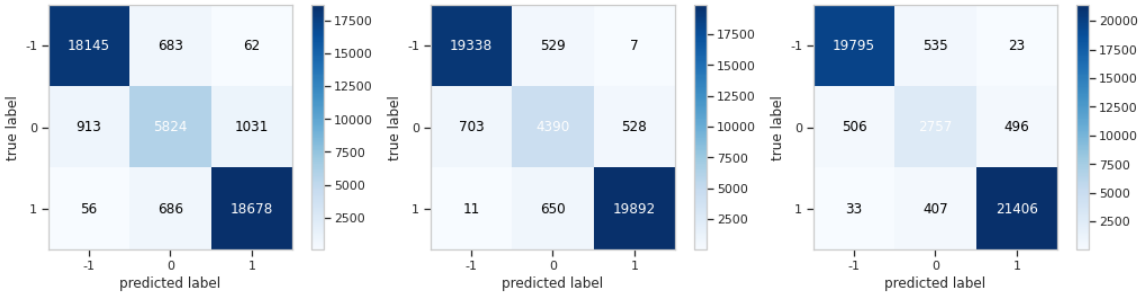
6.2 Confusion Matrix and Boxplots

We first plot the confusion matrix for three models. For each model with specific k, we simply add up the matrix training on multiple new stocks for simplicity and intuitiveness.



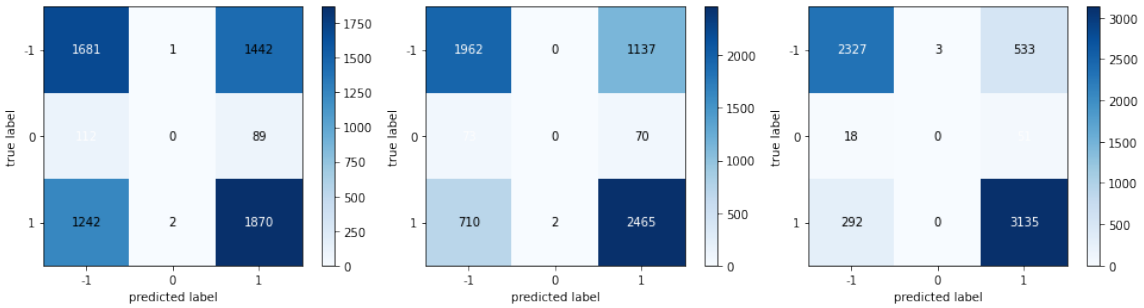
(a) k = 10 (b) k = 20 (c) k = 50

Figure 9: Confusion Matrix for Model 1 on New Stocks



(a) k = 10 (b) k = 20 (c) k = 50

Figure 10: Confusion Matrix for Model 2 on New Stocks



(a) k = 10 (b) k = 20 (c) k = 50

Figure 11: Confusion Matrix for Model 3 on New Stocks

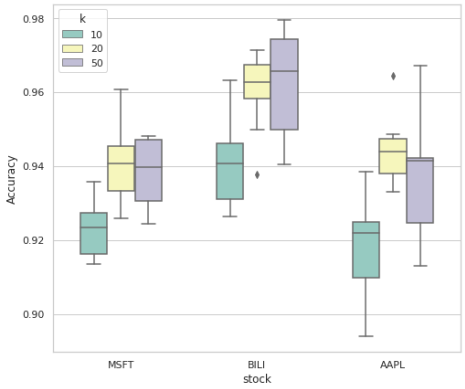
From Figure 9, we can see that for Model 1 and Model 2, most points distribute along the diagonal of the confusion matrix, especially for the elements on two ends. Though the elements on the middle of the diagonals representing "label 0" (price will stay the same level) are not very large, it is because the fact that the true samples with "label 0" only take a very small part of the whole data with our labelling algorithm. For Model 3, it is quite obvious that model does not perform very well when k is small, while confusion matrix of Model 3 with $k = 50$ performs nearly as well as first two models. Therefore we can say that Model 3, which aims to capture idiosyncratic features between stocks, has a better ability to predict price over a

relatively longer period.

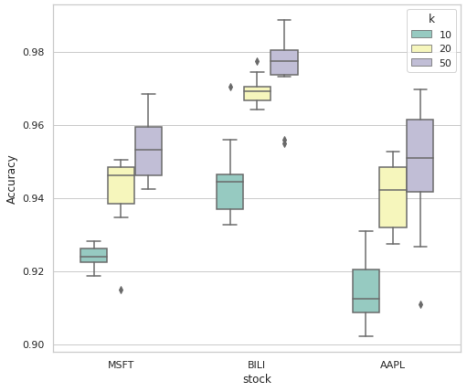
Moreover, we use the boxplots in Figure 12 to present prediction accuracy on new stocks of three models with different k. To get the boxplots, we divide our testing data into 10 parts evenly and each part represents approximately one day because our testing period is exact 10 days. Then we calculate the accuracy on each period and present the accuracy scores in the form of boxplot which shows the distribution of accuracies on ten periods for each model with specific k.

We can observe consistent and robust performance of Model 1 and Model 2, with narrow interquartile range (IQR) and few outliers, for all three new stocks across the testing period.

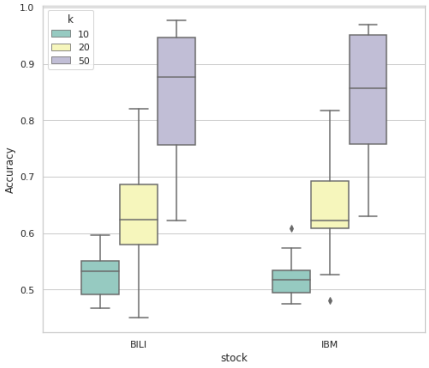
Furthermore, for Model 3, boxplots for BILI and IBM show quite similar pattern with each k. Therefore, we can say that it is reasonable to choose TWTR, UBER, and TSLA as our stock pool for providing market information, so there exist good connections between the stock pool and the target stock and we can get results with similar accuracies to predict different target stocks using Model 3 with this stock pool. However, Model 3 show large accuracy differences among different k, the larger k is, the higher accuracy is. It can be verified that Model 3 has a better ability to predict stock price over a relatively longer period.



(a) Model 1



(b) Model 2



(c) Model 3

Figure 12: Boxplot for the Three Models

7 Conclusion and Perspective

In this paper, we use 3 different deep learning architectures to learn the spatial and temporal features of prices and volumes at each time step to predict the movement of the target stock at the next time point. Model 1 applies CNN layers with an Inception Module to capture spatial feature of the order book and applies an LSTM layer to capture temporal feature among 100 time steps. Model 2 simply applies CNN layers. Model 3 applies same CNN and LSTM as Model 1 but has multiple inputs from stock pool and the target stock, which does not have the property of universality. We can summarize our conclusions as below.

1. Model 1 and Model 2 have overall better performance compared to Model 3. More specifically, the loss and accuracy of Model 3 over the epochs fluctuate and show the problem of overfitting and the testing accuracy is lower than first two models. Therefore, we can say that the property of universality not only capture the common features of all the stocks in the market but also show higher accuracy for prediction.

2. Model 1 and Model 2 show narrow interquaitile range and few outliers when applying transfer learning on new stocks which implies the stability of these two models.

3. Model 3 shows relatively high accuracy when prediction horizon k is as large as 50, which indicates that this model is better to be used for prediction over longer period (about 8 minutes). Furthermore, the similarity of the results of two stocks indicate that it is proper to choose TWTR, UBER, and TSLA as our stock pool.

However, there are also some limitations in our work. Even though we have achieved high accuracy in our models with certain proper parameters, we still cannot make sure that we can make profits if trading with an algorithm based on our price prediction model.

1. First, the accuracy of data labelling needs to be investigated. We can see from Figure 3 that there are some periods of mislabelling of the price movement, which will cause loss in real trading scenarios. One of the potential reasons that lead to this problem is that we smooth the data using a rolling window in the data processing. Thus, more research is needed to balance the accuracy of labelling and the data smooth process.

2. Second, in this work, we did not design a trading strategy to test the profitability of the results. The basic idea is to take a positive position in the periods with label 'up', to take a negative position in the periods with label 'down', and to clear all the position in the periods with label 'steady' while setting a certain transaction cost rate. However, in real trading scenarios, the condition we face will be much more complex. Therefore, it is

a good idea to optimize the trading algorithm to realize a higher profit. To achieve this, we can introduce reinforcement learning to get the optimized 'playing rule'.

3. Third, the dataset used for training can be expanded in time horizon and in stock choices to determine whether and how the amount of data could affect the model accuracy. Besides, attention mechanism can be tried to make the models be able to remember the more important information only and ignore the noise.

8 Reference

[1] A. Ang and G. Bekaert, "Stock return predictability: Is it there?" The Review of Financial Studies, vol. 20, no. 3, pp. 651–707, 2006.

[2] P. Bacchetta, E. Mertens, and E. Van Wincoop, "Predictability in financial markets: What do survey expectations tell us?" Journal of International Money and Finance, vol. 28, no. 3, pp. 406–426, 2009.

[3] T. Bollerslev, J. Marrone, L. Xu, and H. Zhou, "Stock return predictability and variance risk premia: Statistical inference and international evidence," Journal of Financial and Quantitative Analysis, vol. 49, no. 3, pp. 633– 661, 2014.

[4] M. A. Ferreira and P. Santa-Clara, "Forecasting stock market returns: The sum of the parts is more than the whole," Journal of Financial Economics, vol. 100, no. 3, pp. 514–537, 2011.

[5] B. Mandelbrot and R. L. Hudson, The Misbehavior of Markets: A fractal view of financial turbulence. Basic books, 2007.

[6] B. B. Mandelbrot, "How Fractals Can Explain What's Wrong with Wall Street," Scientific American, vol. 15, no. 9, p. 2008, 2008.

[7] J. Agrawal, V. Chourasia, and A. Mittra, "State-of-theart in stock prediction techniques," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, no. 4, pp. 1360– 1366, 2013.

[8] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira, "Computational intelligence and financial markets: A survey and future directions," Expert Systems with Applications, vol. 55, pp. 194–211, 2016.

[9] Q. Cao, K. B. Leggio, and M. J. Schniederjans, "A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market," Computers Operations Research, vol. 32, no. 10, pp. 2499–2512, 2005.

[10] J. Sirignano and R. Cont, "Universal features of price formation in financial markets: perspectives from deep learning," arXiv preprint arXiv:1803.06917, 2018.

[11] G. S. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques–Part II: Soft computing methods," Expert Systems with Applications, vol. 36, no. 3, pp. 5932–5941, 2009.

[12] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, July.

Forecasting stock prices from the limit order book using convolutional neural networks. In 2017 IEEE 19th conference on business informatics (CBI) (Vol. 1, pp. 7-12). IEEE. 2017.

[13] Z. Zhang, S. Zohren, and S. Roberts, Deeplob: Deep convolutional neural networks for limit order books. IEEE Transactions on Signal Processing, 67(11), pp.3001-3012, 2019.

[14] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," in Signal Processing Conference (EUSIPCO), 2017 25th European. IEEE, pp. 2511–2515, 2017.

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