International Journal of Forecasting 40 (2024) 1189–1205

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Short-term stock price trend prediction with imaging high frequency limit order book data

Wuyi Ye, Jinting Yang, Pengzhan Chen *

International Institute of Finance, School of Management, University of Science and Technology of China, Hefei, PR China

ARTICLE INFO

Keywords:
Limit order book
Mid-price prediction
Convolutional neural network
Image classification
Feature engineering

ABSTRACT

Predicting price movements over a short period is a challenging problem in high-frequency trading. Deep learning methods have recently been used to forecast short-term prices via limit order book (LOB) data. In this paper, we propose a framework to convert LOB data into a series of standard images in 2D matrices and predict the mid-price movements via an image-based convolutional neural network (CNN). The empirical study shows that the image-based CNN model outperforms other traditional machine learning and deep learning methods based on raw LOB data. Our findings suggest that the additional information implicit in LOB images contributes to short-term price forecasting.

1. Introduction

High-frequency trading (HFT), a kind of automatic trading, is occupying an increasing proportion of the electronic securities market, with improvements in communication technology and computing efficiency (Gregoriou, 2015). To predict asset price movements in a short time interval, traditional low-frequency data, such as daily OHLC (open, high, low, close price) data of stocks, cannot meet our analytical needs for predicting asset price movements in a short time interval. High-frequency data, which reflect the instantaneous market supply and demand, such as limit order book (LOB) data, are the main object of research in HFT. In other words, enhancing short-term forecasts of price trends based on high-frequency data is a hot topic in academia.

The relevant research in HFT can be categorized into two main strands. One stream focuses on model-based methods, in which theoretical models are constructed based on high-frequency price dynamics described as stochastic processes, and the properties derived from the models are applied to real trading. For example, An and Chan (2017) proposed a novel stochastic model that considers both market event occurrence frequency and order size to describe LOB dynamics, and they examined the performance of the model in real trading. Other related works include Avellaneda, Reed, and Stoikov (2011), Avellaneda and Stoikov (2008), Bayer, Horst, and Qiu (2017), Bayraktar and Ludkovski (2014), Cattivelli and Pirino (2019), Cont, Stoikov, and Talreja (2010), Horst and Paulsen (2017), Horst and Xu (2019), Lu and Abergel (2018), Morariu-Patrichi and Pakkanen (2022) and Rosu (2009). Although the mathematical tractability of the stochastic models provides fascinating insights into high-frequency price dynamics and trading strategies, they are limited by some assumptions that either fail to hold or are challenging to validate in real data. In addition, some model parameters may not be observed. These drawbacks potentially hinder the models' performance in real trading (see Kercheval & Zhang, 2015).

The second stream of research focuses on data-driven models, such as machine learning (ML) and deep learning (DL) methods. These methods play a crucial role in modeling large-scale high-frequency data, predicting price dynamics using the patterns learned from historical data, and constructing trading strategies. Regarding ML methods, Kercheval and Zhang (2015) applied multi-class support vector machine (SVM) models to recognize

* Corresponding author.
E-mail address: cpz@ustc.edu.cn (P. Chen).

https://doi.org/10.1016/j.ijforecast.2023.10.008
0169-2070/

the patterns of LOB dynamics with handcrafted features from LOB data and predicted the mid-price movements and direction of bid–ask spread crossings. Their handcrafted features were used in many subsequent studies; see, e.g., Ntakaris, Magris, Kanniainen, Gabbouj, and Iosifidis (2018). In terms of DL methods, Sirignano (2019) designed a new neural network architecture called the spatial neural network, which utilizes the special structure of LOB data to model spatial distributions. This architecture is less computationally expensive and better-performing than traditional neural networks. Other works of ML and DL modeling high-frequency data for forecasting include Dixon (2018), Nousi et al. (2019), Ntakaris, Mirone, Kanniainen, Gabbouj, and Iosifidis (2019), Tsantekidis et al. (2020), and Zhang, Zohren, and Roberts (2019). See Zaznov, Kunkel, Dufour, and Badii (2022) for a review.

In general, most of the above-mentioned ML and DL articles focus on directly modeling raw high-frequency data or handcrafted features extracted from them. By contrast, there has been a recent surge of interest in "imaging" financial data and employing image-based ML and DL methods to predict stock price trends. In particular, a nonlinear transformation process known as "imaging" is employed on the raw data, converting time-series numerical data into a series of images that capture the physical attributes of the data, e.g., spatial relationships. This approach enables specialized image analysis algorithms, such as convolutional neural networks (CNNs), to better utilize their strengths in prediction tasks.

Numerous recent studies have made significant contributions to image-based prediction. For high-frequency price trend predictions, Niño, Hernández, Arévalo, León, and Sandoval (2018) encoded LOB series into image-like representations stored in 3D tensors with RGBA (red, green, blue, alpha channels) as the inputs for a CNN to predict mid-price movements, and they achieved promising results. Nakayama et al. (2019) converted order flow information into images with multiple channels based on the characteristics of order flow data. By using ML for prediction, the results indicated that the execution information has higher predictive power than the order and cancellation information. In other prediction areas, Hao and Lenskiy (2023) established a novel connection between imaging market data with short-term realized volatility. They employed an approach for encoding order flow information into images and making predictions of short-term realized volatility with a CNN. However, the imaging process in these articles has a common drawback: images with multiple channels need to be stored in high-dimensional tensors. With the vast amount of high-frequency data, the storage space requirement increases exponentially. In addition, compared to low-dimensional data, the computation time for high-dimensional data is relatively higher. This is detrimental in high-frequency trading where decisions need to be made within milliseconds. These observations motivated us to optimize the imaging process for data dimension reduction to improve computational and storage efficiency while retaining sufficient information.

Recently, Jiang, Kelly, and Xiu (2020) proposed a novel imaging process that converts daily stock OHLC data into a series of 2D metric black-and-white images. These images serve as inputs for a CNN model to capture nonlinear associations within daily price curves. However, given the substantial structural disparities between high-frequency LOB data and daily OHLC data, their method may not be directly applicable to high-frequency scenarios. Here, we extend the method introduced by Jiang et al. (2020) and present a novel imaging approach. This approach is better tailored to the structural characteristics of LOB data, making a valuable contribution to the field of expert-based feature engineering. Specifically, compared with existing high-frequency stock price trend prediction methods, our proposed method offers several advantages. First, it standardizes raw LOB data with varying distributions across different stocks. Second, it reconstructs and portrays raw data features from a new perspective, such as transforming price spread information into position relationships within the image. This transformation is particularly well suited for image analysis methods. Third, the resulting image, generated through our imaging process, is presented in 2D grayscale. This not only conserves storage space but also ensures computational efficiency. Finally, the imaging process serves as a data preprocessing step that smooths the data and eliminates less influential information. This helps reduce data dimensionality and noise while amplifying significant signals.

In what follows, we first describe our imaging framework to convert LOB data with different price and volume distributions into a set of standard images stored in 2D tensors step by step. By means of this transformation, a mid-price trend prediction problem becomes an image classification problem. Then, we briefly introduce the CNN architecture as our main classifier and present the specific model parameter settings used in this research. Furthermore, we designed experiments with real data from the Chinese stock market to analyze the impact of different input image sizes on the prediction results. We compared our image-based CNN models to traditional ML and DL methods that directly model raw LOB data. Our model achieved the best performance in the competitive environments. To investigate the reasons behind our model's outperformance, we conducted an ablation experiment to examine the impact of the model's architecture on the classification. The results indicate that the performance of our method is not sensitive to variations in the model's structure. The outperformance primarily stems from the imaging process. Finally, to evaluate the effectiveness of our proposed method in a real trading environment, we designed a simple back-testing framework based on real data. The trading strategy derived from our proposed methodology outperformed other benchmark strategies.

The rest of this paper is organized as follows: Section 2 introduces concepts related to the limit order book and describes our transformation framework for imaging LOB data. In Section 3, we introduce the CNN architecture and specific model parameter settings. In Section 4, we discuss the experimental results of the proposed method and other ML methods on real data and test our method in a back-testing framework. Finally, Section 5 presents the conclusion.

Table 1
Limit order book.

	Ask1		Bid1		Ask2		Bid2		…
Time	Price	Volume	Price	Volume	Price	Volume	Price	Volume	…
09:33:21.000	25.21	500	25.20	26,200	25.22	500	25.18	11,800	…
09:33:24.000	25.20	400	25.18	10,800	25.25	300	25.17	500	…
…	…	…	…	…	…	…	…	…	…

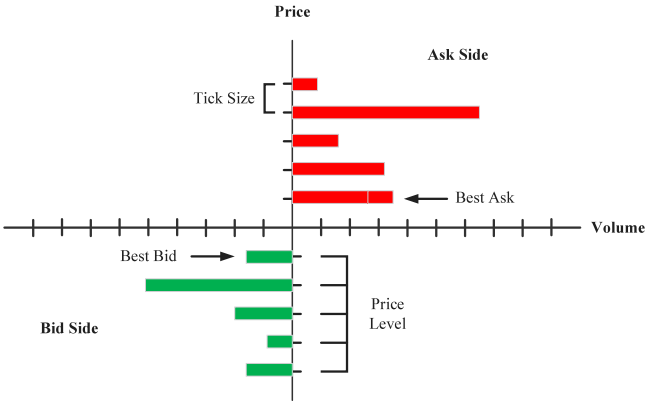

Fig. 1. A common representation of a limit order book.

2. Imaging limit order book data

In this section, we first introduce the concept of an LOB and then propose a framework for representing LOB data as an image. The framework presented in this section plays a crucial role in utilizing alternative information in images of the LOB to predict short-term stock price trends.

2.1. Limit order book

The electronic trading market is composed of various orders that can be classified as market orders or limit orders. Market orders are typically filled immediately at the best prices[1] after they are submitted. Limit orders are recorded in the LOB as shown in Table 1, which represent the intention of the trader to trade at a specific price or better.

A typical record of the LOB contains comprehensive information about both the prices and order volumes of accumulated limit orders submitted up to the current time. As in the existing literature (see, e.g., An & Chan, 2017), a common visual depiction of the LOB is illustrated in Fig. 1. In particular, ask-side levels are shown in red and bid-side levels are shown in green. At each price level, the accumulated limit order volumes are recorded by the length of the rectangle. The "Tick Size" in Fig. 1 is the minimum price fluctuation in discrete price levels.[2] The lowest ask price and the highest bid price are the best ask price and best bid price, respectively.

Although the common representation of an LOB record shown in Fig. 1 clearly reflects the market information at a given moment, there are some drawbacks when we use it as an image input for a CNN classification model. On the one hand, the distribution of both price and order volume in the LOB vary with time and stocks, leading to inconsistent image standards. On the other hand, tensors of at least three dimensions must be used to represent the images with RGB (red, green, blue channels), leading to more complicated computational problems, as presented in Niño et al. (2018). To overcome these difficulties, we develop a transformation method, which is described in detail in the next subsection.

2.2. Framework for imaging limit order book data

The symbols and their respective meanings used in this subsection are summarized in Table 2. The detailed description of the imaging is divided into three steps as follows:

Step 1. Transformation of the price spread and order volume.

We consider the relative price s defined as the difference between a given price and the best bid price[3]:

$$s = p - p_b,$$

where p_b is the best bid price, and p is the limit order price.

Two stocks (600436.SH and 000002.SZ) in the Chinese stock market are taken as an example. As shown

[1] The best ask (bid) price is the lowest (highest) offer price available in the LOB to sell (buy) a security.

[2] E.g., the tick size of an order is RMB 0.01 for the A shares in China, which are the shares in mainland China-based companies.

[3] Notably, using the best ask price instead makes no essential difference. In addition, Tashiro, Matsushima, Izumi, and Sakaji (2019) use the mid-price to calculate the relative price, rather than best bid or ask prices. The reason we use the best bid price is that it is a real price in the LOB, whereas the mid-price is not.

Table 2
Summary of mathematical symbols and their meanings.

Symbols	Meanings
p, p_a, p_b, p_{mid}	Price, best ask price, best bid price, mid-price
$\mathbb{N}^+$	The set of positive integers
$s, \tilde{s}$	The relative price and the transformed relative price
$\mathbb{S}$	The set of relative prices
$\tilde{\mathbb{S}}$	The set of transformed relative prices
$\mathbb{S}^+, \tilde{\mathbb{S}}^-$	The sets of transformed positive and negative relative prices
δ	The tick size in the stock market
I	The number of intervals in the transformed relative price scale
M	The number of elements in each interval
K	The difference between transformed adjacent relative prices
h	The height spread in images
c_i	The pixel values
v_i, v_{max}	Order volume at price level i, the maxima of the order volume

Table 3
Descriptive statistics of two stocks from the Chinese stock market.

Code	Mid-price		Relative price		
	Mean	Std	Max	Min	Max−Min
600436.SH	83.99	3.81	0.92	−0.91	1.83
000002.SZ	24.94	0.89	0.17	−0.32	0.49

Notes. This table shows the simple descriptive statistics of 600436.SH and 000002.SZ, which are stock codes from the Chinese stock market. Their data are used in the subsequent experiments. "Mean" is the average mid-price of the stock, and "Std" is the standard deviation. "Max" and "Min" are the maximum and minimum value of the relative price, respectively. "Max−Min" is the maximum relative price spread in the LOBs.

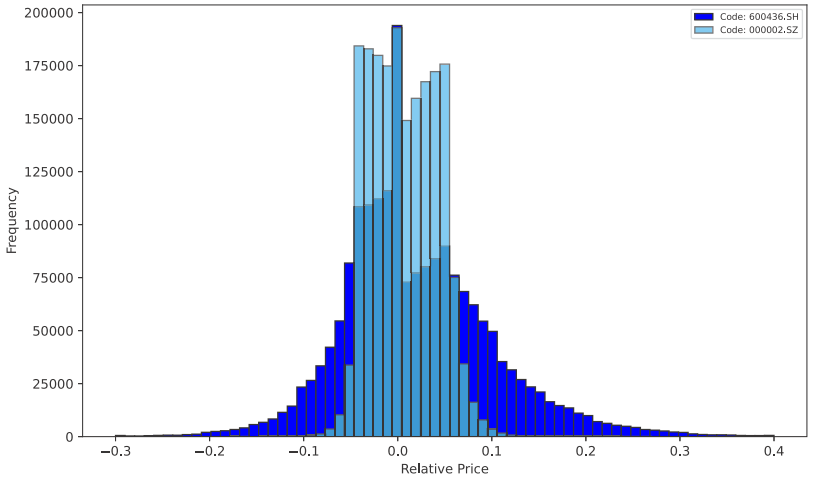

Fig. 2. Relative price distribution of stock codes 600436.SH and 000002.SZ.

in Fig. 2, the distribution of the relative price for different stocks can differ substantially. The distribution of 600436.SH falls mostly in $[-0.2, 0.3]$, which is approximately triple the length of $[-0.07, 0.1]$ of 000002.SZ, because the average stock price level of 600436.SH is higher (see Table 3). However, the information embedded in the tail has less influence on short-term price movements (Tashiro et al., 2019). Hence, if we use the same spread scale for all stocks, the redundant information of the big price spread is given the same weight as the small price spread, thereby weakening the effect of the important information of the small price spread in forecasting.

Next, we introduce the transformation. Suppose $\delta > 0$ is the tick size in the stock market; then, the set of relative prices is denoted as $\mathbb{S} = \bigcup_{i=1}^{I} \mathbb{S}^+ \cup \mathbb{S}^- \cup \{0\}$ with $\mathbb{S}^{\pm} = \{\pm n\delta\}_{n\in\mathbb{N}^+}$. For a given $K \in \mathbb{N}^+$, we propose to scale the relative price s to the following form:

$$\tilde{\mathbb{S}} = \bigcup_{i=1}^{I} \tilde{\mathbb{S}}_i^+ \cup \tilde{\mathbb{S}}_i^- \cup \{0\}, \tag{1}$$

where $\tilde{\mathbb{S}}_{i+1}^{\pm} = \{K \times \tilde{s} + M : \text{for } \tilde{s} \in \tilde{\mathbb{S}}_i^{\pm}\}$ and $\tilde{\mathbb{S}}_1^{\pm} = \{\pm k\}_{k=1,\ldots,M}$ with $M \in \mathbb{N}^+$, $K \in \mathbb{N}^+$, and $I \in \mathbb{N}^+$. There are $2I$ intervals in the transformed relative price scale, each of which has M elements. Within the same interval, the

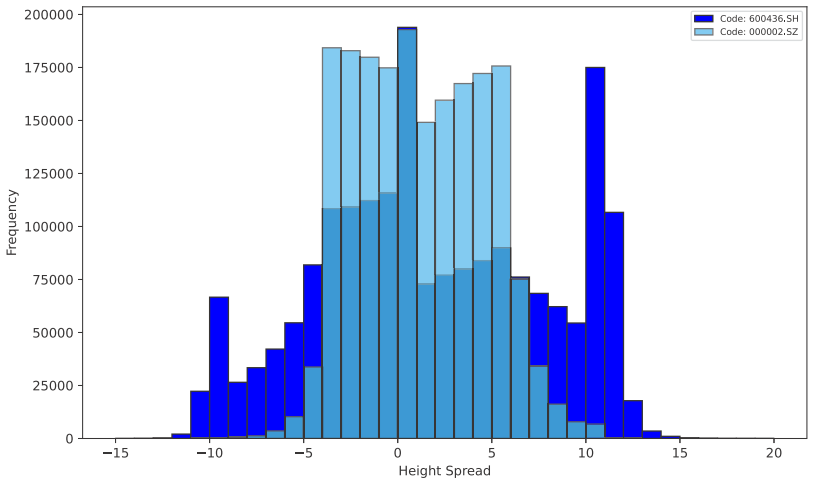

Fig. 3. The height spread distribution of 600436.SH and 000002.SZ after transformation by Eq. (1) (2).

difference between adjacent transformed relative prices is K. Then, we use the following function to map the elements in $\tilde{S}$ to the height spread in the image:

$$h^{\pm} = \begin{cases} 0 & \tilde{s} = 0 \\ \tilde{s} & \tilde{s} \in \tilde{\mathbb{S}}_1^{\pm} \\ \pm M + h^{\pm}((\tilde{s} \mp M)/K) & \tilde{s} \in \tilde{\mathbb{S}}_i^{\pm} \end{cases}. \tag{2}$$

We can control the image size by choosing different M, K, and I corresponding to the statistical characteristics of the LOB in different trading markets. In practice, we choose $M = 10$, $K = 10$, and $I = 2$. The transformation above is portrayed in Fig. 3, in which the height spread distribution of 600436.SH, ranging from -12 to 15, is similar to that of 000002.SZ, ranging from -7 to 11. Furthermore, the shape of the tail of 600436.SH changes significantly after the transformation. The result intuitively shows that the information from big spreads with less influence are categorized together; thus, the height of the image is controlled.

To control the width of the images, we use only one pixel, with the grayscale ranging from 0 to 255 to represent the order volume information at each price level, in which the darker the color, the lower the order volume. Due to the large difference in the distribution of order volumes for different stocks, we convert the order volume into pixel values in 0–255 via the following formula:

$$c_i = \lfloor \frac{v_i}{v_{max}} \times 255 \rfloor, \tag{3}$$

where v_i is the order volume at price level i in an LOB record, and v_{max} is the maxima of the order volume in the LOB records in a certain time interval. For example, if an image contains the past five LOB records, v_{max} is the maxima of the order volume at all price levels in the past five records.

Step 2. Generate an image with one record of the limit order book.

Using the transformation of the relative price and the order volume given in Step 1, a typical process of converting a single LOB record into an image is illustrated in Fig. 4. Firstly, we design a $(2MI + 1) \times 3$ matrix[4] to represent the image with a single LOB record. As shown in Fig. 4, the relative price axis is converted to the vertical white line in the middle column of the image. We use the best bid price as the origin of the axis in the image, which is located at the center of the image (hereafter, the "center price"). On the price axis, numbers of pixels above and below the center price represent the relative prices converted by Eqs. (1) and (2); thus, the total height of the image is $(2MI + 1)$. The length of the price axis is determined by the height spread after transformation between the level-5 ask price and the level-5 bid price in this record. Secondly, the pixels representing the order volume are placed on the side of the corresponding price levels, with bid orders placed on the left side of the price axis and ask orders placed on the right side of the price axis. The order volume is scaled into color depth by Eq. (3). Therefore, the width of the image is 3. Fig. 4 shows that the best ask price has the largest order volume, i.e., $Volume_{max}$ in this record. The pixel representing the best ask price order volume is the brightest point in the image. In addition, the level-4 ask price and the level-5 ask price have large price spreads, and their information is categorized together. Their volume pixel values are added together and displayed as one pixel in the image. If the sum of pixel values is greater than 255, we always treat it as 255 in data preprocessing.

Step 3. Generate the image with multiple records.

To generate an image with multiple LOB records, we display the past N records side by side in an image with a $(2MI + 1) \times 3N$ matrix[5] after transforming them via the above steps, where the center price is represented by the best bid price in the most recent record.

After the transformation, our proposed method successfully converts prices and order volumes between different stocks to a comparable scale and represents them

[4] In this paper, we set $M = 10, K = 10, and\ I = 3$, The absolute value of the maximum price spread higher and lower than the best bid price can be represented by $10 \times 3 = 30$, with the center price. The total height is 61.

[5] The standard image size is $61 \times 3N$ in this article.

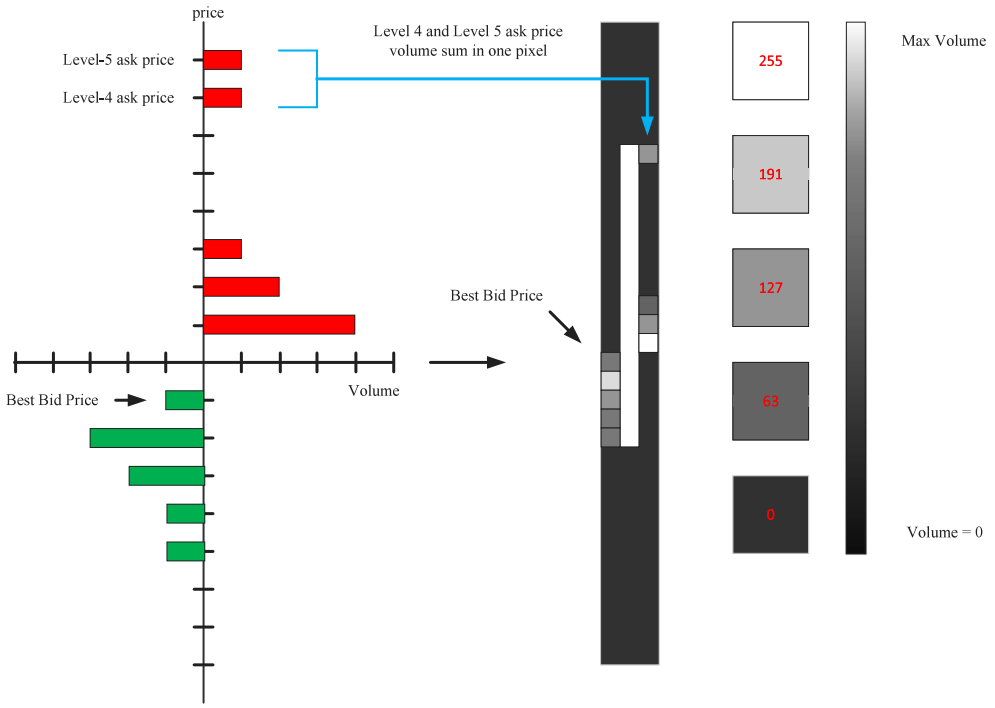

Fig. 4. A typical example of limit order book transformation.

with standard size images. In this study, we generate five different sizes of images containing the past 1, 2, 3, 5, or 10 records as the input for the CNN predictor shown in Fig. 5.

3. Model

In this section, we first discuss the main task of forecasting. Then we provide a brief introduction to CNN and the specific settings used in this study.

3.1. Problem

We are interested in predicting the mid-price movement in the next 30 seconds using the CNN with the transformed LOB image. The mid-price p_{mid} is defined as

$$p_{mid} = \frac{1}{2}(p_a + p_b),$$

where p_a and p_b are the best ask and bid prices, respectively. The mid-price can be seen as a "fair" price of the instantaneous market supply and demand Stoikov (2018); hence, its movement can partially represent market trends. In general, the future movement of the mid-price can be classified as up, down, or neutral. If we can accurately predict the future trend of the mid-price, we have the opportunity to obtain trading profits. The prediction process is illustrated in Fig. 6.

3.2. Convolutional neural networks

We chose a CNN as the primary forecasting model for the following reasons. First, CNNs can automatically extract features from images without the need for manual feature extraction, which means that we do not need to spend a considerable amount of effort searching for indicators in high-frequency data. Instead, it allows the model to capture them directly from images. Second, the convolutional kernels in the CNN can capture the relationships between adjacent LOB records. We focus on the variations between adjacent records by designing the size and stride of the filters, which provide important information for the future trends. Third, the CNN model is relatively simple. Our images are uniform and standardized, with a simple structure, making them well-suited for classical image recognition models rather than more complex models like transformers (Vaswani et al., 2017).

A CNN consists of three main modules: convolution, activation, and max-pooling. Through these modules, the CNN can automatically extract features from images and transform them into a set of feature vectors that serves as input to a fully connected network (FCN), finally outputting the probability of mid-price movement by means of a softmax function. Furthermore, these main modules can be flexibly combined with other DL modules to adapt the model to varying application scenarios (He, Zhang, Ren, & Sun, 2016; Kim, 2014; Krizhevsky, Sutskever, & Hinton, 2017).

In convolution modules, a set of low-dimension 2D-matrix filters scan the LOB images horizontally and vertically to generate a new set of 2D-matrix images. The filters—in other words, the feature extraction machine—apply convolution operations on original images, and each pixel value of the newly generated image represents the value of the convolution sum in the corresponding region.

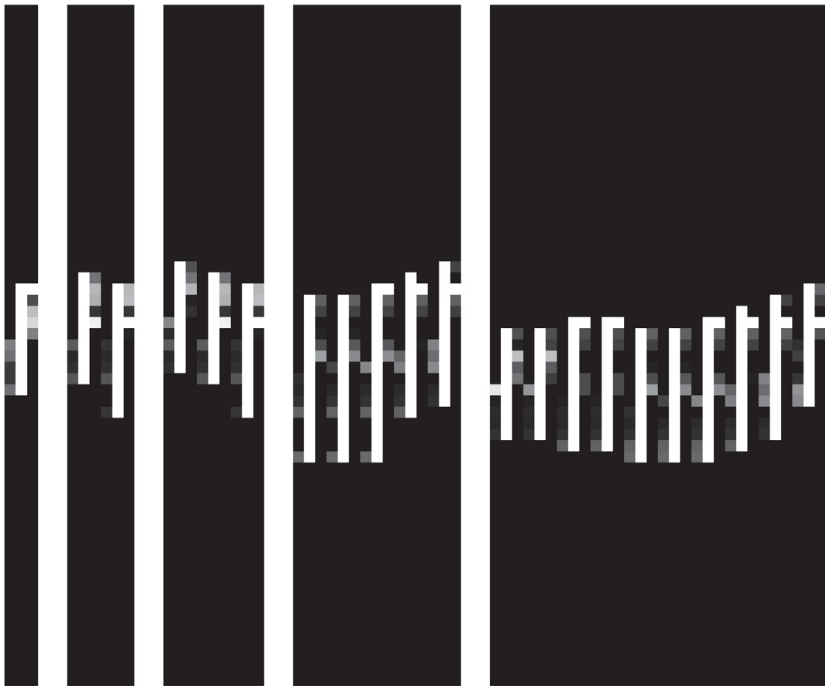

Fig. 5. LOB images with the past 1, 2, 3, 5, and 10 records.

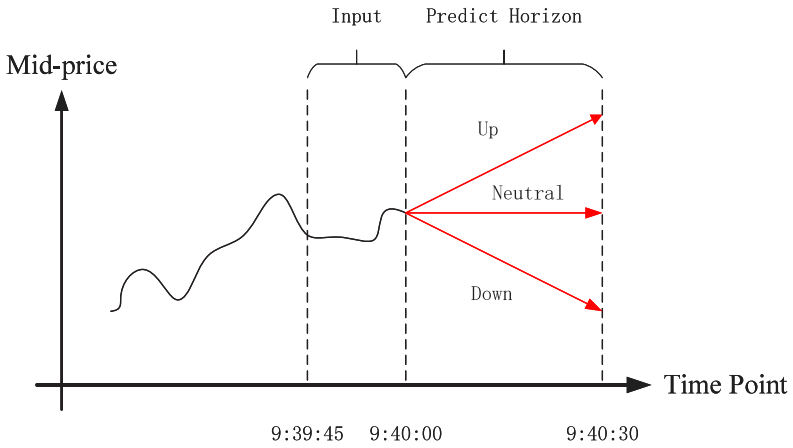

Fig. 6. Main prediction process of the mid-price.

The second module is the activation module, which applies the nonlinear activation function called leaky ReLU (Maas et al., 2013) to the images generated by the convolution modules. This module generally performs a nonlinear transformation over the input images so that the model can identify the nonlinear connections between features.

The final module is max-pooling. In this module, a lower-dimension filter scans the input matrix after activation and extracts the maximum value at corresponding regions in the image, which can greatly reduce the dimension of the input matrix and data noise (Jiang et al., 2020).

3.3. Model training

We independently trained CNN models based on images with five different input sizes. During the parameter selection process, the choice of initial parameters was inspired by Jiang et al. (2020) and adjusted to be better suited to the characteristics of our images. Once the initial parameters were determined, we applied a tuning process on a small subset of training samples to determine our final parameters. The following is a detailed explanation of the logic behind selecting these parameters.

Our main CNN architecture consists of three blocks and a fully connected layer. Each block has a convolution,

activation, and max-pooling module. To accurately capture the price–volume relationship between two adjacent records, we apply a series of 6×6 filters. We use a horizontal stride of one and a vertical stride of two in the first convolution module, because of the sparsity in the vertical dimension in the first layer. Then, 2×2 filters with a horizontal stride of one and a vertical stride of one extract deeper information in other convolution modules. According to Zeiler and Fergus (2014), the features in deeper layers become more complex, and we increase the number of filters by a factor of 2. In other words, we set 256, 512, and 1024 filters in the three convolution modules so that the model can identify the complex relationships in deeper layers to predict future trends. Moreover, we set max-pooling filters of 2×2 in the first pooling modules and 2×1 in the other layers for the model to retain more accurate horizontal information.

Since LOBs are converted into images, our final task becomes a multi-class image classification problem to minimize the objective function, i.e., the cross-entropy loss function:

$$\text{Loss} = -\sum_{i=1}^{n} y_i \log y_i',$$

where $y_i = 1$ when the sample true class is i; otherwise, $y_i = 0$. y_i' is the probability generated by the softmax function. We use the PyTorch default Kaiming initializer method (He, Zhang, Ren, & Sun, 2015) to initialize the CNN parameters and the Adam algorithm (Kingma & Ba, 2014) with learning rate of 1×10^{-5} and a batch size of 128 to optimize the loss function. In order to improve the training speed and convergence of the model, we incorporate batch normalization (Ioffe & Szegedy, 2015) layers between the convolutions and nonlinear activations. We additionally apply 50% dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) to the fully connected layer to prevent the CNN from overfitting. Appendix records the parameter settings of all input size models in detail. All experiments were performed in Windows, with a 3.8 GHz Intel Core i7 CPU, 32 GB of 1600 MHz memory, and an 8 GB NVIDIA GeForce RTX 2070 SUPER GPU. Our main programming language was Python.

4. Experiments

In this section, we describe the preliminary work of our experiments and summarize their results in detail.

4.1. Preliminaries

Our datasets contain LOB data of 20 real stocks from 20 industries on the CSI 300 index from the Shanghai Stock Exchange and Shenzhen Stock Exchange of China, including 42 trading days from November 1, 2018, to December 31, 2018. The datasets are obtained from the Wind database. The data from November 2018 is used as the training set. For each stock, the training set has approximately 100,000 samples. The data from December 2018 is randomly divided into validation and test sets at a 1:1 ratio. To prevent the accumulation of overnight information that could significantly disrupt the predictions, we removed the samples from the first and the last 5 min of each trading day.

Table 4 shows descriptive statistics of our datasets in both the training and testing periods. During the same period, Table 4 conveys that each stock has significant differences in volatility and liquidity. We independently re-train the CNN model for each stock with the same parameter settings five times, for the model to capture the unique characteristics of each stock more effectively. In each training, we choose the model with the best performance on the validation set as the final model for testing.

The Proportion column in Table 4 records the percentage of down, up, and neutral categories in the full dataset of each stock. Stocks with higher price levels exhibit a greater imbalance among the three categories (see, e.g., 603986.SH). This imbalance can be attributed to the fixed constant smallest trading unit (i.e., the tick size of an order) shared by all stocks, resulting in a decreased likelihood for stocks with higher price levels to fall into the neutral category. To ensure efficient feature extraction and model training, we use the weighted random sampling (WRS) technique in our experiments. WRS is a methodology that assigns distinctive weights to training samples with the aim of controlling the probability of selecting each individual sample during the process of sampling. Essentially, samples with a smaller number of classes are more likely to be selected. Some relevant algorithms can be found in Efraimidis and Spirakis (2006). In this article, we employ the *torch.utils.data.WeightedRandomSampler* in PyTorch to tackle the challenge of imbalanced class distributions.

To evaluate the performance of our models, we use the average of the F_1 scores of three categories as metrics. The F_1 score is defined as the harmonic mean of precision and recall and can be used to comprehensively evaluate the performance of a model on an imbalanced dataset. The F_1 score is calculated by the following formula:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall},$$

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

where TP, FP, and FN respectively represent the number of true-positive samples, false-positive samples, and false-negative samples in one category.

In addition, we employed two statistical tests—namely the Friedman test and Nemenyi test, as introduced in J. (2006)—to compare the performance of multiple classifiers across various datasets.

The Friedman test (Friedman, 1940) is employed to compare the average ranks of classifiers, in which the average rank is denoted as $R_j = \frac{1}{N}\sum_i r_i^j$, where r_i^j represents the rank of the jth classifier of Z classifiers on the ith dataset out of N datasets. The null hypothesis of the Friedman test assumes that all the classifiers are

Table 4
Summary statistics of datasets.

Code	Mid-price		Bid–ask spread		Relative price		Proportion
	Mean	Std	Mean	Std	Max	Min	
000002.SZ	24.94	0.89	0.014	0.009	17	−32	40.4% Down, 38.3% Up, 21.3% Neutral
000063.SZ	19.93	0.88	0.011	0.004	13	−15	37.5% Down, 37.5% Up, 25.0% Neutral
000651.SZ	37.56	1.25	0.012	0.007	16	−23	40.0% Down, 35.0% Up, 25.0% Neutral
000858.SZ	51.58	1.46	0.015	0.012	36	−47	43.5% Down, 40.3% Up, 16.2% Neutral
002460.SZ	24.59	1.00	0.013	0.007	19	−29	42.6% Down, 38.9% Up, 18.5% Neutral
002594.SZ	53.59	3.33	0.026	0.025	74	−75	46.9% Down, 44.2% Up, 8.9% Neutral
002841.SZ	60.16	2.89	0.064	0.062	138	−189	46.0% Down, 44.0% Up, 10.0% Neutral
300144.SZ	21.38	1.12	0.019	0.014	22	−30	39.1% Down, 39.1% Up, 21.8% Neutral
300498.SZ	25.25	0.80	0.023	0.019	32	−55	40.0% Down, 40.0% Up, 20.0% Neutral
600111.SH	9.81	0.51	0.010	0.002	8	−11	18.8% Down, 18.8% Up, 62.4% Neutral
600309.SH	30.12	1.22	0.014	0.009	35	−39	42.6% Down, 38.9% Up, 18.5% Neutral
600398.SH	8.31	0.31	0.011	0.004	11	−15	22.2% Down, 22.2% Up, 55.6% Neutral
600436.SH	83.99	3.81	0.037	0.040	92	−91	45.9% Down, 45.0% Up, 9.1% Neutral
600585.SH	31.87	1.60	0.014	0.009	23	−27	37.8% Down, 35.1% Up, 27.1% Neutral
600893.SH	23.31	0.81	0.020	0.015	25	−24	38.1% Down, 38.1% Up, 23.8% Neutral
600900.SH	14.84	0.43	0.012	0.005	12	−14	25.0% Down, 25.0% Up, 50.0% Neutral
601088.SH	19.31	0.64	0.014	0.008	18	−19	30.8% Down, 30.8% Up, 38.4% Neutral
601100.SH	19.98	0.61	0.024	0.020	50	−54	40.0% Down, 37.8% Up, 22.2% Neutral
601360.SH	22.48	1.01	0.016	0.011	23	−29	40.4% Down, 38.3% Up, 21.3% Neutral
603986.SH	72.04	4.76	0.039	0.039	17	−15	48.7% Down, 44.7% Up, 6.6% Neutral

Notes. This table shows descriptive statistics of the stocks in our datasets. The first column records the mean and standard deviation of mid-price. The second column shows the mean and standard deviation of bid–ask spread, which is a measure of liquidity. The third column presents the maximum (Max) and minimum (Min) value of the relative price in their LOBs. The fourth column records the proportion of down, up, and neutral categories.

equivalent, and their ranks on multiple datasets are randomly assigned, implying that their ranks R_j are equal. The standard Friedman statistic is given by

$$\chi_F^2 = \frac{12N}{Z(Z+1)}\left[\sum_j R_j^2 - \frac{Z(Z+1)^2}{4}\right],$$

which is distributed according to χ_F^2 with $Z-1$ degrees of freedom. In this paper, we use the modified Friedman statistic proposed by Iman (1980):

$$F_F = \frac{(N-1)\chi_F^2}{N(Z-1)-\chi_F^2},$$

which is distributed according to the F-distribution with $Z-1$ and $(Z-1)(N-1)$ degrees of freedom.

The Nemenyi test (Nemenyi, 1963) is a post hoc test used to compare all classifiers with each other after the null hypothesis of the Friedman test is rejected. The performance of two classifiers is considered significantly different if the difference in their average ranks is equal to or greater than the critical difference (CD), which is calculated as follows:

$$CD = q_\alpha\sqrt{\frac{Z(Z+1)}{6N}},$$

where the critical value q_α is determined based on the significance level α chosen for the test. The values of q_α with $\alpha = 0.05, \alpha = 0.10$ can be found in Table 5 (J., 2006).

In the testing procedure, we first use the Friedman test. If the null hypothesis is significantly rejected, we then proceed to use the Nemenyi post hoc test to compare the performance of each model across multiple datasets.

Table 5
Critical values for the two-tailed Nemenyi test.

#Classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

4.2. Prediction with different image sizes

The first set of analyses examines the impact of different input image sizes on the model prediction F_1 score. We consider five input image sizes, including the past 1, 2, 3, 5, and 10 records of the LOB information (model training with these image sizes is respectively denoted as CNN-I1, CNN-I2, CNN-I3, CNN-I5, and CNN-I10), and our task is to predict the mid-price trends in the next 30 s. We introduce two naive benchmark models, labeled Naive 1 and Naive 2 in Fig. 7 for comparison.[6] Naive 1 is a pure guessing model. We assume that the distribution of the classes during the training time remains consistent during testing, and we randomly generate a prediction according to the assumed distribution. For Naive 2, we utilize the class from the previous interval as the prediction for the subsequent interval.

The F_1 score achieved for the two naive models and five different input sizes of 20 stocks is shown in Fig. 7. For each stock, the prediction F_1 score across all imaged-based models surpasses that of the Naive 1 model. This finding implies that our image-based CNN models are capable of identifying distinct patterns within images, thereby offering relatively reliable predictions of mid-price future trends, as opposed to relying solely on random guessing.

[6] Because the results of the Naive 2 model did not vary after repeated experiments, it is depicted with short lines in a box plot.

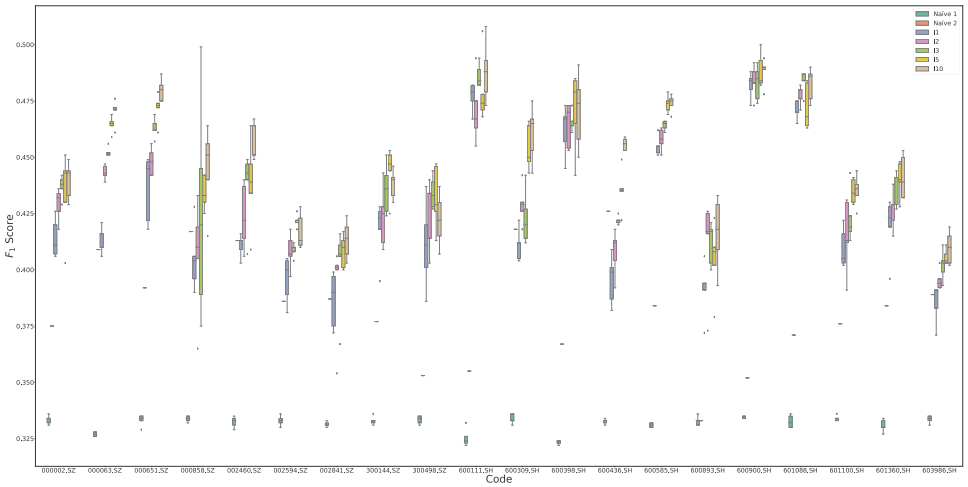

Fig. 7. The F_1 score of mid-price prediction with different input sizes.

Table 6
Numerical results of prediction with different input sizes.

Classifier	Naive 1	Naive 2	CNN-I1	CNN-I2	CNN-I3	CNN-I5	CNN-I10
F_1 score	0.332	0.383	0.422	0.432	0.440	0.445	0.450
Rank	7.00	5.85	5.05	3.85	2.75	2.10	1.45

Compared to the Naive 1 model, the performance of the Naive 2 model is superior. In short-term forecasting, a certain time interval exhibits a persistent trend. Incorporating the outcomes from the previous time period for forecasting the subsequent time period can adequately reflect this particular portion of information and leads to a higher F_1 score. Furthermore, our image-based models consistently outperform the Naive 1 and Naive 2 models on almost all of the datasets. Our models effectively capture an amount of specific information that is beneficial for predicting future trends.

Alternatively, Fig. 7 indicates that as the input size increases, the F_1 score of the model improves. In most datasets, as the size of input images increases, the box plot shows an upward trend. This conclusion is further verified in Table 6. Table 6 records the average F_1 score and ranks of the classifiers over multiple datasets. On the one hand, from CNN-I1 to CNN-I10, the F_1 score increases by an average of 2.8% per stock, and CNN-I10 achieves the highest F_1 score of 45.0%. For CNN-I1, the model can only analyze the relationship between different price levels and volumes at a fixed time point. Clearly, the patterns extracted from them can imply future market trends (an average F_1 of 42.2%). However, these patterns are static and cannot fully represent the historical dynamic trends, which are likely to have effective information about future trends such as price momentum and reversal in low-frequency trading. In comparison, CNN-I2 contains information from the past two timestamps. The model can analyze the connections between not only the price and volume at the fixed time point but also the dynamic variations in price and order volume crossing two past records. Thus, CNN-I2 has a 1% better F_1 score than that of CNN-I1. However, compared with that of CNN-I1, the F_1 score of CNN-I5 increases by an average of 2.3% per stock. The inputs of CNN-I10 double the image size compared to that of CNN-I5 but only improves the F_1 score by 0.5%.

On the other hand, the average ranks of the classifiers also show a clear upward trend, with CNN-I10 having the highest average rank and CNN-I1 having the lowest average rank. We employed the aforementioned Friedman test to examine whether measured average ranks significantly deviate from the expected average rank $R_j = 4$ under the null hypothesis: The $F_F \approx 204.75$ is distributed according to $F(6, 114)$, and the critical value of $F(6, 114)$ for $\alpha = 0.01$ is 2.96, so we strongly reject the null hypothesis. Then we proceeded with the Nemenyi post hoc test to check whether the performance of two classifiers is significantly different. Under $\alpha = 0.10, Z = 7$, the critical value $q_\alpha = 2.693$, and $CD \approx 1.84$. The performance of Naive 1, Naive 2, and CNN-I1 is significantly worse than that of CNN-I3, CNN-I5, and CNN-I10. This finding aligns with the previous analysis in Fig. 7 and Table 6. However, the differences between CNN-I3, CNN-I5, and CNN-I10 are not significant ($2.75 - 1.45 \approx 1.3 < 1.84$). It is evident that for specific datasets, the F_1 score of either CNN-I3 or CNN-I5 reaches its peak in Fig. 7, and CNN-I10 does not present significant improvements.

Compared to CNN-I10, CNN-I5 has a reliable F_1 score without requiring as much input information. This makes our computations highly efficient. In order to strike a balance between the two factors, we selected the CNN-I5 model as the primary subject of analysis in subsequent experiments.

4.3. Prediction with different models

The experiments in this section are concerned with the performance of our image-based CNN model (CNN-I5) and other ML and DL methods based on raw input data. Following Tashiro et al. (2019), we chose logistic regression (LR), a support vector machine (SVM), XGBoost, a multilayer perceptron (MLP), and a CNN (modeling raw LOB data) as comparison models. Logistic regression and the

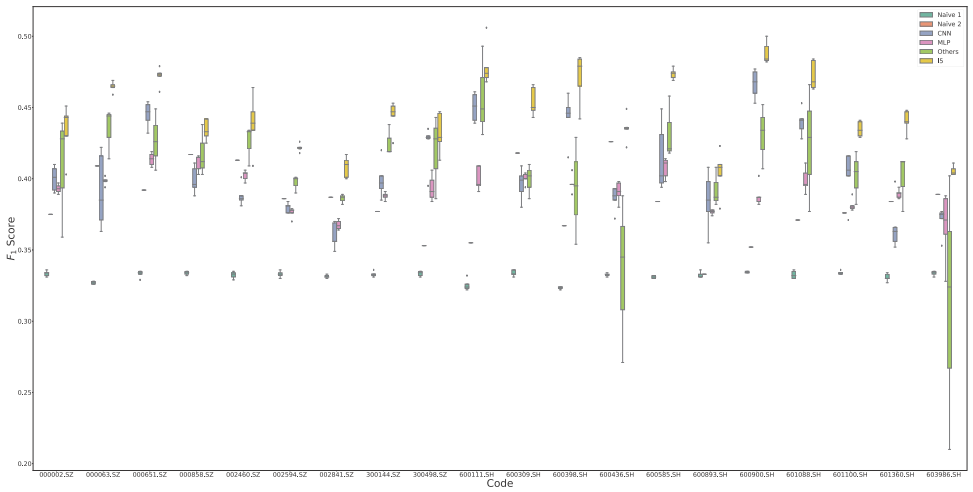

Fig. 8. The F_1 scores of mid-price prediction with different models.

SVM are classical linear classification models. They can efficiently capture the linear relationships between features and have good interpretability. They are commonly used as benchmark models in classification problems. Unlike LR and the SVM, XGBoost is a representative algorithm of gradient boosted regression trees in ensemble learning, which are fully nonparametric and can identify nonlinear relationships between features. The MLP and CNN are classical models in DL. They are nonlinear and highly parameterized, with broad applicability in classification tasks. The model structure and meta-parameters of the comparison models are recorded in Appendix.

For the comparison models, our input consists of the bid–ask price and order volume information from the past five records. Each model is re-trained five times, similar to CNN-I5. Fig. 8 shows a box plot of the F_1 score of the models. For some models (LR, SVM, and XGBoost), their results did not change after repeated experiments. So we combine their results into one box labeled "others". Naive 1 and Naive 2 still serve as our benchmark models.

In Fig. 8, CNN-I5 outperforms the comparison models on almost all stocks. After encoding the raw data, CNN-I5 identifies additional information that is difficult to directly extract from the raw data. This is the source of the high F_1 score for CNN-I5. Furthermore, other comparison models do not show distinct advantages. While they achieved higher F_1 scores than the Naive 1 model, they did not consistently defeat the Naive 2 model. The trend information captured by the Naive 2 model allows it to perform better on some datasets and exhibit stable performance over multiple datasets.

A similar conclusion can be inferred from Table 7. Table 7 records the average F_1 score and ranks of these machine learning models over multiple datasets. The CNN-I5 model achieves the highest F_1 score of 44.5%, which is approximately 2.2% higher than the second-highest, XGBoost. In comparison, the difference among Naive 2, LR, SVM, CNN, and MLP are not significant. The highest (CNN) F_1 score of them is only 2.2% better than the lowest (Naive 2). We further support our conclusion through the Friedman test and Nemenyi post hoc test. Firstly, in the

Table 7
Numerical results of prediction with different models.

Classifier	Naive 1	Naive 2	LR	SVM	XGB	CNN	MLP	CNN-I5
F_1 score	0.332	0.383	0.399	0.395	0.423	0.405	0.392	0.445
Rank	7.85	5.375	4.35	4.125	2.95	4.75	5.35	1.25

Friedman test: $F_F \approx 30.57$ is distributed according to the $F(7, 133)$, and the critical value of $F(7, 133)$ for $\alpha = 0.01$ is 2.78, so we strongly reject the null hypothesis. Thus, the measured average ranks are significantly different from the mean rank $R_j = 4.5$. Secondly, in Nemenyi post hoc test: under $\alpha = 0.10, Z = 8$, the critical value $q_\alpha = 2.780$, and $CD \approx 2.15$. The CNN-I5 model significantly outperforms all other comparison models except XGBoost. While the difference between the CNN-I5 and XGBoost is below the critical difference ($2.95 - 1.25 = 1.7 < 2.15$), they are close. In this case, although the difference does not reach statistical significance, we can still consider that CNN-I5 outperforms XGBoost to some extent. The differences between Naive 2, LR, SVM, CNN, and MLP are not significant. The difference between their highest and lowest ranks is just 1.25, which is consistent with our previous conclusion.

To gain a deeper understanding of reasons behind the superior performance of the CNN-I5 model, we conducted ablation studies on the image-based CNN. Here, we present our results on several stocks: 002594.SZ, 600398.SH, and 600585.SH. Each represents one of the three categories based on the number of instances: one with a higher number of up and down movements, one with a higher number of neutral instances, and one with a relatively balanced distribution of instances among the three movements. Table 8 records the results of ablation studies on three stocks. The F_1 score in each term is the average of forecasts from re-trained CNN models over five iterations.

The results in Table 8 indicate that while modifying the network structure can enhance the prediction F_1 score for a specific stock, it does not lead to a significant improvement in overall prediction performance. For

Table 8
Ablation studies on the image-based CNN (CNN-I5).

		002594.SZ - Unbalanced up/down F_1 score	600398.SH - Unbalanced neutral F_1 score	600585.SH - Balanced F_1 score
Baseline		*0.422*	*0.471*	*0.474*
Filters (256)	64	0.425	0.462	0.475
	128	0.42	0.47	0.475
	512	0.412	0.448	0.469
Dropout (0.5)	0	0.421	0.46	0.471
	0.25	0.42	**0.479**	0.471
	0.75	0.42	0.473	0.471
Layers (3)	2	**0.426**	0.46	0.476
	4	0.415	0.463	0.46
BatchNorm (Yes)	No	0.42	0.464	**0.482**
Filter Size (6×6)	(3×3)	0.418	0.466	0.473
Max-pooling (2×1)	(2×2)	**0.426**	0.465	0.47
Activation (LReLU)	ReLU	0.417	0.474	0.471
Stride $(1, 2)$	$(2, 2)$	0.414	0.475	0.474

Notes. This table shows ablation studies on the image-based CNN, including the number of filters in the first layer (equal to 256 in the baseline model), dropout probability (baseline 0.5), number of convolutional layers (baseline 3), use of batch normalization (BatchNorm, baseline Yes), filter size in the first layer (baseline 6×6 pixels), size of max-pooling layers in the second and third layers (baseline 2×1), activation function (baseline leaky ReLU), and stride in the first layer (baseline $(1, 2)$).

example, when we exclude batch normalization during model training, it results in an enhanced F_1 score of 48.2% for 600585.SH. However, this modification in the network structure negatively affects the performance of 002594.SZ and 600398.SH. In other words, our main results are robust to the choice of different model structures. From another perspective, this also indicates that the outperformance of our image-based CNN is primarily due to the process of imaging, rather than the network structure itself. The imaging data provide the CNN with additional information that enables it to identify patterns relevant to future trends, which cannot be captured from the raw data alone. By utilizing this imaging process, we provide a more beneficial platform for the CNN to demonstrate its capabilities, without necessitating a specialized model structure. This conclusion is similar to the finding in Jiang et al. (2020).

4.4. Back-testing

In the final part of the experiments, we designed a back-testing framework to assess our proposed model in real trading. Our back-testing program runs from December 2018 (the data for the validation set and the test set) for 20 trading days based on real stock data. In each trading day, no trades are made within 5 min after the market opens and 5 min before the market closes to prevent overnight losses. If the position is still open 5 min before close, we immediately close the position at the market price.

In the order execution phase, we assume that the orders placed on the best ask or best bid price are executed randomly according to the probability of the exponential arrival rate proposed by Avellaneda and Stoikov (2008), as illustrated in the following formula:

$$P = Ae^{-B\Delta}dt,$$

where Δ is the spread between the mid-price and best bid or ask price, A and B are constants greater than 0, and dt is the quotation time interval.

Algorithm 1 Framework of the back-testing procedure.

Input: Limit order book data, Prediction model, Initialize the account state;
Output: Daily return record;
for $i = 1$ to 100 **do**
  for $j = 1$ to N do **do**
    Get a prediction y' based on the current state and prediction model;
    if y' is Neutral **then**
      if No position in account **then** Take no action;
      else if There is a position in the account **then** Close the position;
    else if y' is the Up or Down **then**
      if No position in account **then** Admit an order at the best price according to y';
      else if There is a position in the account **then**
        if y' is as same as the position direction **then** keep the position;
        else Close the position and admit an order at the best price according to y';
    Randomly decide whether the order is filled or not according to probability P;
    Update the account information according to the order status;
return The daily return records generated by the account information.

Our back-testing process is as follows: At each time moment t, the strategy generates a prediction of the mid-price movement in the next 30 s based on the models. If the prediction result is up, we submit a buy order of 1 unit at the best bid price. By contrast, if the prediction result is down, we submit a sell order of 1 unit at the best ask price. If the prediction result is neutral, we take no action. After the execution phase, the order transaction

Table 9
Strategy description.

Strategy	Model	Description
Strategy 1	Naive 1	Generate a prediction using the distribution from the training period.
Strategy 2	Naive 2	Generate a prediction at current t using the real values from $t-1$.
Strategy 3	XGBoost	Generate a prediction with the XGBoost model using the past five records.
Strategy 4	CNN-I5	Convert the past five records into images and generate a prediction with CNN-I5.

Notes. Since we re-train CNN-I5 multiple times, the predictions of CNN-I5 here are the average of the predictions from multiple models.

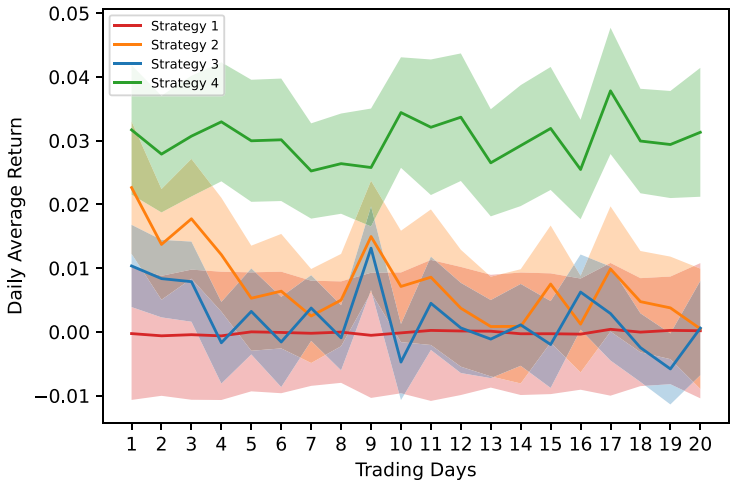

Fig. 9. Daily return of four strategies on each trading day.

state is obtained according to the transaction probability mentioned above. It the order is not executed, we cancel it and re-submit based on the prediction in the next 30 s. To avoid so-called inventory effects (Avellaneda & Stoikov, 2008), we implement a specific approach. After 30 s, we close the position at the prevailing market price, unless the predictions at current time match the previous step. If the predictions are opposite from the previous step, we not only close the position at the market price but also place a limit order for 1 unit at the best price available in the opposite direction. The back-testing procedure is described in Algorithm 1. We chose four different models (Naive 1, Naive 2, XGBoost, and CNN-I5) from the previous experiments and created four corresponding strategies. The application for these strategies is the same, and involves generating a prediction at time t. Descriptions of the strategies are presented in Table 9.

In the actual experiment, we applied the back-testing program to each individual stock independently and assessed the performance of each strategy on each stock. Moreover, we did not introduce any transaction cost, because this is a complex problem in the dealer model and real-world trading. The rationality of the transaction cost settings directly influences the back-testing results. Additionally, to account for the randomness in our framework, we repeated the test 100 times for each strategy. We used the average daily return and daily return standard deviation for each stock as our comparison results. Furthermore, we introduced a daily Sharpe ratio based on the assumption that the daily expected return is zero to measure the relationship between daily volatility and return:

$$SR = \frac{Return}{Std},$$

where *Return* is the average daily return, and *Std* is the daily return standard deviation. Table 10 shows the detailed results in back-testing for each stock.

As shown in Table 10, Strategy 4, derived from the image-based CNN, outperforms the other comparison strategies in terms of the daily return metrics on each stock. Strategy 4 has a 3.01% average daily return and 2.253 average Sharpe ratio, which are much higher than those of Strategies 1, 2, and 3. This indicates that our image-based CNN generates highly efficient signals in back-testing, and that these signals can enable us to achieve stable profits. Further evidence can be found in Fig. 9, which illustrates the average return on each trading day. The average return of Strategy 4 is greater than that of Strategies 1, 2, and 3 on each trading day.

Moreover, from Table 10 and Fig. 9, Strategy 1 obtains a daily return of close to 0%. This is natural because the signal in Strategy 1 is fully random and the frequency of occurrence of up trends and down trends in 30 s is almost equal. Surprisingly, the XGBoost model has a higher F_1 score than the Naive 2 model, but the corresponding Strategy 3's daily return and Sharpe ratio are lower than those of Strategy 2. A possible reason for this phenomenon is that, although the XGBoost model provides more accurate predictions, it may struggle to provide accurate forecasts in stable upward or downward trends.

Table 10
Strategy performance.

Code	Strategy 1			Strategy 2			Strategy 3			Strategy 4		
	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR
000002.SZ	0.0001	0.0109	0.013	0.0066	0.0144	0.456	0.0104	0.0127	0.824	0.0312	0.0121	2.587
000063.SZ	0.0000	0.0105	−0.003	0.0162	0.0147	1.105	0.0143	0.0132	1.078	0.0364	0.0158	2.301
000651.SZ	−0.0001	0.0060	−0.016	0.0073	0.0114	0.637	0.0010	0.0067	0.154	0.0178	0.0080	2.210
000858.SZ	0.0000	0.0084	−0.002	0.0190	0.0118	1.612	0.0120	0.0115	1.038	0.0276	0.0112	2.461
002460.SZ	−0.0005	0.0123	−0.038	0.0237	0.0208	1.141	0.0148	0.0136	1.089	0.0251	0.0149	1.681
002594.SZ	−0.0002	0.0126	−0.017	0.0189	0.0169	1.118	0.0077	0.0150	0.515	0.0349	0.0158	2.211
002841.SZ	−0.0010	0.0112	−0.093	0.0073	0.0110	0.659	−0.0008	0.0122	−0.066	0.0218	0.0127	1.711
300144.SZ	−0.0003	0.0109	−0.024	0.0017	0.0191	0.089	0.0010	0.0117	0.088	0.0371	0.0161	2.308
300498.SZ	0.0003	0.0121	0.024	−0.0097	0.0202	−0.478	0.0072	0.0133	0.540	0.0395	0.0178	2.217
600111.SH	−0.0001	0.0048	−0.017	−0.0041	0.0051	−0.792	−0.0043	0.0080	−0.538	0.0337	0.0111	3.049
600309.SH	0.0000	0.0119	0.002	0.0313	0.0272	1.151	−0.0020	0.0245	−0.081	0.0393	0.0192	2.048
600398.SH	−0.0002	0.0076	−0.031	−0.0040	0.0212	−0.189	0.0003	0.0157	0.018	0.0364	0.0159	2.294
600436.SH	−0.0005	0.0126	−0.043	0.0364	0.0180	2.028	0.0040	0.0200	0.202	0.0360	0.0154	2.341
600585.SH	0.0002	0.0065	0.025	0.0046	0.0100	0.455	−0.0017	0.0106	−0.160	0.0184	0.0079	2.337
600893.SH	−0.0002	0.0074	−0.022	−0.0088	0.0108	−0.817	−0.0036	0.0094	−0.379	0.0165	0.0102	1.614
600900.SH	0.0001	0.0056	0.012	−0.0117	0.0086	−1.366	0.0032	0.0082	0.394	0.0317	0.0113	2.819
601088.SH	0.0001	0.0068	0.011	−0.0071	0.0110	−0.652	−0.0022	0.0102	−0.211	0.0218	0.0114	1.910
601100.SH	−0.0003	0.0149	−0.019	−0.0075	0.0304	−0.248	−0.0073	0.0230	−0.320	0.0356	0.0304	1.174
601360.SH	−0.0001	0.0088	−0.016	0.0079	0.0134	0.594	−0.0040	0.0081	−0.490	0.0200	0.0107	1.875
603986.SH	0.0001	0.0124	0.009	0.0211	0.0195	1.081	−0.0079	0.0208	−0.382	0.0417	0.0107	3.908
Average	−0.0001	0.0097	−0.012	0.0074	0.0158	0.379	0.0021	0.0134	0.166	0.0301	0.0139	2.253

Even if the predictions are accurate, the profits are eroded by the friction of the closing positions at the market price. By contrast, the Naive 2 model excels at capitalizing on these stable trends, allowing it to accumulate substantial profits when they occur. Fig. 10 provides strong evidence for this argument, which illustrates the relationship between prediction behavior and profit changes. During stable trends, such as around 4000 s, the Naive 2 model consistently makes correct predictions, leading to significant profits accumulated by Strategy 2. Meanwhile, the XGBoost model produces a mix of correct and incorrect predictions, resulting in some losses by Strategy 3. However, Strategy 4 derived by the image-based CNN not only makes accurate predictions during stable trends but also achieves stable and accurate predictions during short-term oscillations, leading to continuous profit accumulation.

5. Conclusion

We proposed an imaging process to convert LOB data to a series of 2D grayscale images. During the imaging process, LOB data with different price and volume distributions are transformed into the same scale and presented as standard images. As more information is included, past price trends are clearly presented in the images. This framework can retain sufficient information after the transformation while reducing the image dimensionality and increasing the computational efficiency. After the imaging process, the mid-price prediction problem becomes a classification problem with up, down, and neutral labels for images. The CNN model, which can automatically extract features from images rather than modeling handcrafted features, was selected as our main prediction model. To analyze the capabilities and performance of our model, we designed three experiments. Firstly, we explored the impact of the size of the input image on the performance of the image-based models. The experimental results showed high F_1 scores for each type of image, and the CNN captured the features corresponding to different categories from imaging LOB data. The numerical results further illustrated that as the input image size increases, the F_1 score increases. Secondly, we compared CNN-I5 with traditional machine learning models, logistic regression, a support vector machine, a multilayer perceptron, XGBoost, and a CNN (modeling raw LOB data). The results of this experiment indicated that our proposed method performed best. The further ablation analysis showed that the outperformance of our CNN-I5 was primarily due to the process of imaging rather than the network structure itself. Finally, we tested the trading strategy derived from our proposed method in a simple back-testing environment. The proposed strategy clearly outperformed the comparison strategies. The results demonstrated that our proposed method produces useful forecasts of price trends for real high-frequency trading.

In future research, we will try to apply our imaging approach to cryptocurrency markets. Other information, such as the encoded order book in Tashiro et al. (2019), will be embedded to analyze cryptocurrency market price movements. In addition, we will introduce reinforcement learning to search for an optimal trading strategy based on our high-frequency imaging data and test our strategy in a virtual trading environment (Schnaubelt, 2022).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are indebted to the Editor Esther Ruiz, an Associate Editor, and two anonymous referees for detailed

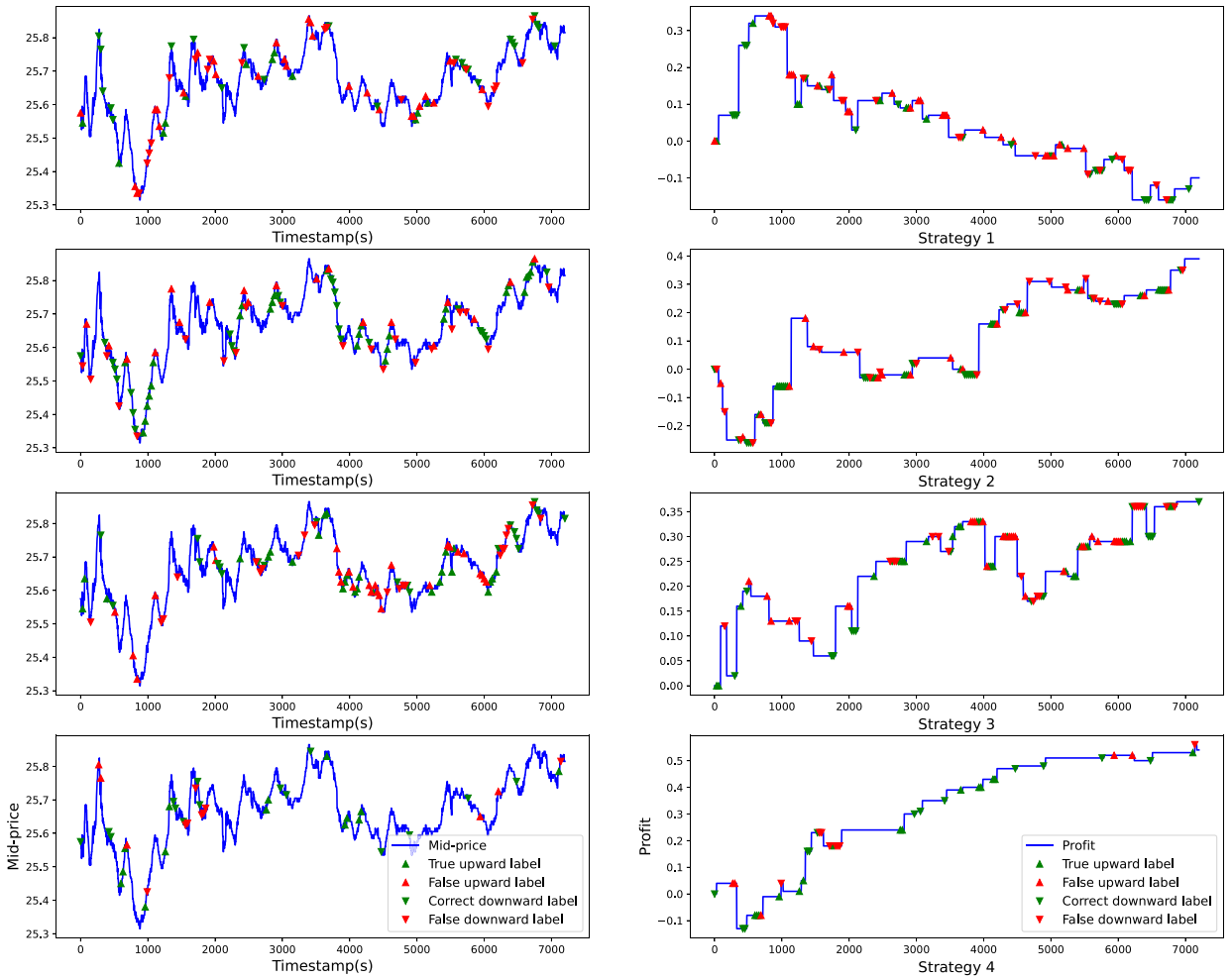

Fig. 10. Prediction behavior and profit curves of Strategies 1, 2, 3, and 4 over 7200 s. The *y*-axis above represents the stock price in CNY and the *y*-axis below represents the profit in CNY. The stock code is "002460".

comments. We are solely responsible for any remaining errors. Wuyi Ye's research is supported by the National Natural Science Foundation of China (No. 71973133) and Anhui Provincial Natural Science Foundation, China (No. 2208085J41). Pengzhan Chen's research is supported by a China Postdoctoral Science Foundation-funded project (No. 2022M 723018) and the Fundamental Research Funds for the Central Universities, China (WK2040000070).

Appendix. Parameter settings

A.1. Image-based CNN parameter settings

In this section, we provide a detailed description of parameter settings in this article. Table A.1 records the image-based CNN parameter settings with five different input sizes: CNN-I1, CNN-I2, CNN-I3, CNN-I5, and CNN-I10. Among the competition models, LR, SVM, XGBoost, and MLP take the limit prices and accumulated order volumes from the last five records as the inputs. LR and SVM use the default parameters of *scikit-learn*. XGBoost uses *max_depth=5, learning_rate=0.01, n_estimators=300* in *xgboost.XGBClassifier*. *max_depth=5* specifies a maximum depth of five for each tree. *learning_rate=0.01* controls the step size of 0.01 for weight updates in each iteration. *n_estimators=300* specifies the number of weak trees to be built. The model structure of the MLP is given in Table A.2. Moreover, the input of the CNN is a series of matrices stacked from the past five records, and its model structure is recorded in Table A.2.

References

An, Y., & Chan, N. H. (2017). Short-term stock price prediction based on limit order book dynamics. *Journal of Forecasting, 36*, 541–556. http://dx.doi.org/10.1002/for.2452.

Table A.1
Experimental parameters of CNN-I1, CNN-I2, CNN-I3,CNN-I5, and CNN-I10.

CNN-I1		CNN-I2		CNN-I3		CNN-I5		CNN-I10	
Convolution layer 1		Convolution layer 1		Convolution layer 1		Convolution layer 1		Convolution layer 1	
Kernel size	(256, 6, 3)	Kernel size	(256, 6, 6)	Kernel size	(256, 6, 6)	Kernel size	(256, 6, 6)	Kernel size	(256, 6, 6)
Padding	(0, 5)	Padding	(0, 5)	Padding	(0, 5)	Padding	(0, 5)	Padding	(0, 5)
Stride	(1, 2)	Stride	(1, 2)	Stride	(1, 2)	Stride	(1, 2)	Stride	(1, 2)
Batch normalization		Batch normalization		Batch normalization		Batch normalization		Batch normalization	
Leaky ReLU		Leaky ReLU		Leaky ReLU		Leaky ReLU		Leaky ReLU	
Pooling kernel	2 × 2	Pooling kernel	2 × 2	Pooling kernel	2 × 2	Pooling kernel	2 × 2	Pooling kernel	2 × 2
Convolution layer 2		Convolution layer 2		Convolution layer 2		Convolution layer 2		Convolution layer 2	
Kernel size	(512, 2, 2)	Kernel size	(512, 2, 2)	Kernel size	(512, 2, 2)	Kernel size	(512, 2, 2)	Kernel size	(512, 2, 2)
Padding	(1, 1)	Padding	(1, 1)	Padding	(1, 1)	Padding	(1, 1)	Padding	(1, 1)
Stride	(1, 1)	Stride	(1, 1)	Stride	(1, 1)	Stride	(1, 1)	Stride	(1, 1)
Batch normalization		Batch normalization		Batch normalization		Batch normalization		Batch normalization	
Leaky ReLU		Leaky ReLU		Leaky ReLU		Leaky ReLU		Leaky ReLU	
Pooling kernel size	2 × 1	Pooling kernel size	2 × 1	Pooling kernel size	2 × 1	Pooling kernel size	2 × 1	Pooling kernel size	2 × 1
Convolution layer 3		Convolution layer 3		Convolution layer 3		Convolution layer 3		Convolution layer 3	
Kernel size	(1024, 2, 2)	Kernel size	(1024, 2, 2)	Kernel size	(1024, 2, 2)	Kernel size	(1024, 2, 2)	Kernel size	(1024, 2, 2)
Padding	(1, 1)	Padding	(1, 1)	padding	(1, 1)	Padding	(1, 1)	Padding	(1, 1)
Stride	(1, 1)	Stride	(1, 1)	Stride	(1, 1)	Stride	(1, 1)	Stride	(1, 1)
Batch normalization		Batch normalization		Batch normalization		Batch normalization		Batch normalization	
Leaky ReLU		Leaky ReLU		Leaky ReLU		Leaky ReLU		Leaky ReLU	
Pooling kernel size	2 × 1	Pooling kernel size	2 × 1	Pooling kernel size	2 × 1	Pooling kernel size	2 × 1	Pooling kernel size	2 × 1
Dropout	0.5	Dropout	0.5	Dropout	0.5	Dropout	0.5	Dropout	0.5
Fully connected layer	20 480	Fully connected layer	17 920	Fully connected layer	17 920	Fully connected layer	25 088	Fully connected layer	39 424
Softmax	3	Softmax	3	Softmax	3	Softmax	3	Softmax	3

Table A.2
Model structure of comparison MLP and CNN.

MLP		CNN	
Layer 1		Convolution layer 1	
Fully connected layer	200	Kernel size	256, 3, 3
Batch normalization		Padding	(2, 2)
Leaky ReLU		Stride	(2, 1)
		Batch normalization	
		Leaky ReLU	
		Pooling kernel size	(2, 2)
Layer 2		Convolution layer 2	
Fully connected layer	400	Kernel size	512, 2, 2
Batch normalization		Padding	1, 1
Leaky ReLU		Stride	1, 1
		Batch normalization	
		Leaky ReLU	
		Pooling kernel size	2, 1
Layer 3		Convolution layer 3	
Fully connected layer	200	Kernel size	1024, 2, 2
Batch normalization		Padding	1, 1
Dropout	0.5	Stride	1, 1
		Batch normalization	
		Leaky ReLU	
		Pooling kernel size	2, 1
		Dropout	0.5
		Fully connected layer	40 960
Softmax	3	Softmax	3

Avellaneda, M., Reed, J., & Stoikov, S. (2011). Forecasting prices from level-I quotes in the presence of hidden liquidity. *Algorithmic Finance*, *1*, 35–43. http://dx.doi.org/10.3233/AF-2011-004.

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*, 217–224. http://dx.doi.org/10.1080/14697680701381228.

Bayer, C., Horst, U., & Qiu, J. (2017). A functional limit theorem for limit order books with state dependent price dynamics. *Annals of Applied Probability*, *27*, 2753–2806. http://dx.doi.org/10.1214/16-AAP1265.

Bayraktar, E., & Ludkovski, M. (2014). Liquidation in limit order books with controlled intensity. *Mathematical Finance*, *24*, 627–650. http://dx.doi.org/10.1111/j.1467-9965.2012.00529.x.

Cattivelli, L., & Pirino, D. (2019). A SHARP model of bid–ask spread forecasts. *International Journal of Forecasting*, *35*, 1211–1225. http://dx.doi.org/10.1016/j.ijforecast.2019.02.008.

Cont, R., Stoikov, S., & Talreja, R. (2010). A stochastic model for order book dynamics. *Operations Research*, *58*, 549–563. http://dx.doi.org/10.1287/opre.1090.0780.

Dixon, M. (2018). Sequence classification of the limit order book using recurrent neural networks. *Journal of Computer Science*, *24*, 277–286. http://dx.doi.org/10.1016/j.jocs.2017.08.018.

Efraimidis, P. S., & Spirakis, P. G. (2006). Weighted random sampling with a reservoir. *Information Processing Letters*, *97*, 181–185. http://dx.doi.org/10.1016/j.ipl.2005.11.003.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, *11*, 86–92. http://dx.doi.org/10.1214/aoms/1177731944.

Gregoriou, G. N. (2015). *Handbook of high frequency trading*. Academic Press, http://dx.doi.org/10.1016/C2014-0-01732-7.

Hao, M., & Lenskiy, A. (2023). Short-term volatility prediction using deep CNNs trained on order flow. arXiv preprint arXiv:2304.02472.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034). http://dx.doi.org/10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 770–778). http://dx.doi.org/10.1109/CVPR.2016.90.

Horst, U., & Paulsen, M. (2017). A law of large numbers for limit order books. *Mathematics of Operations Research*, *42*, 1280–1312. http://dx.doi.org/10.1287/moor.2017.0848.

Horst, U., & Xu, W. (2019). A scaling limit for limit order books driven by Hawkes processes. *SIAM Journal on Financial Mathematics*, *10*, 350–393. http://dx.doi.org/10.1137/17M1148682.

Iman, R. L. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics. Theory and Methods*, *9*, 571–595. http://dx.doi.org/10.1080/03610928008827904.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).

J., Demšar (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Jiang, J., Kelly, B. T., & Xiu, D. (2020). (Re-)imag(in)ing price trends. In *Chicago booth research paper*. http://dx.doi.org/10.2139/ssrn.3756587.

Kercheval, A. N., & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, *15*, 1315–1329. http://dx.doi.org/10.1080/14697688.2015.1032546.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *EMNLP 2014-2014 conference on empirical methods in natural language processing, proceedings of the conference* (pp. 1746–1751). http://dx.doi.org/10.3115/v1/d14-1181.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*, 84–90. http://dx.doi.org/10.1145/3065386.

Lu, X., & Abergel, F. (2018). High-dimensional Hawkes processes for limit order books: Modelling, empirical analysis and numerical calibration. *Quantitative Finance*, *18*, 249–264. http://dx.doi.org/10.1080/14697688.2017.1403142.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th international conference on machine learning*. http://dx.doi.org/10.1145/3065386.

Morariu-Patrichi, M., & Pakkanen, M. (2022). State-dependent Hawkes processes and their application to limit order book modelling. *Quantitative Finance*, *22*, 563–583. http://dx.doi.org/10.1080/14697688.2021.1983199.

Nakayama, A., Izumi, K., Sakaji, H., Matsushima, H., Shimada, T., & Yamada, K. (2019). Short-term stock price prediction by analysis of order pattern images. In *2019 IEEE conference on computational intelligence for financial engineering & economics* (pp. 1–5). IEEE, http://dx.doi.org/10.1109/CIFEr.2019.8759057.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons*. Princeton University.

Niño, J., Hernández, G., Arévalo, A., León, D., & Sandoval, J. (2018). CNN with limit order book data for stock price prediction. In *Proceedings of the future technologies conference* (pp. 444–457). Springer, http://dx.doi.org/10.1007/978-3-030-02686-8_34.

Nousi, P., Tsantekidis, A., Passalis, N., Ntakaris, A., Kanniainen, J., Tefas, A., et al. (2019). Machine learning for forecasting mid-price movements using limit order book data. *IEEE Access*, *7*, 64722–64736. http://dx.doi.org/10.1109/ACCESS.2019.2916793.

Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2018). Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*, *37*, 852–866. http://dx.doi.org/10.1002/for.2543.

Ntakaris, A., Mirone, G., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). Feature engineering for mid-price prediction with deep learning. *IEEE Access*, *7*, 82390–82412. http://dx.doi.org/10.1109/ACCESS.2019.2924353.

Rosu, I. (2009). A dynamic model of the limit order book. *The Review of Financial Studies*, *22*, 4601–4641. http://dx.doi.org/10.1093/rfs/hhp011.

Schnaubelt, M. (2022). Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. *European Journal of Operational Research*, *296*, 993–1006. http://dx.doi.org/10.1016/j.ejor.2021.04.050.

Sirignano, J. (2019). Deep learning for limit order books. *Quantitative Finance*, *19*, 549–570. http://dx.doi.org/10.1080/14697688.2018.1546053.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Stoikov, S. (2018). The micro-price: A high-frequency estimator of future prices. *Quantitative Finance*, *18*, 1959–1966. http://dx.doi.org/10.1080/14697688.2018.1489139.

Tashiro, D., Matsushima, H., Izumi, K., & Sakaji, H. (2019). Encoding of high-frequency order information and prediction of short-term stock price by deep learning. *Quantitative Finance*, *19*, 1499–1506. http://dx.doi.org/10.1080/14697688.2019.1622314.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2020). Using deep learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing*, *93*, Article 106401. http://dx.doi.org/10.1016/j.asoc.2020.106401.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5999–6009).

Zaznov, I., Kunkel, J., Dufour, A., & Badii, A. (2022). Predicting stock price changes based on the limit order book: A survey. *Mathematics*, *10*(1234), http://dx.doi.org/10.3390/math10081234.

Zeiler, M. D., & Fergus, R. (2014). *Lecture notes in computer science (Including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*: *vol. 8689 LNCS*, *Visualizing and understanding convolutional networks* (pp. 818–833). http://dx.doi.org/10.1007/978-3-319-10590-1_53.

Zhang, Z., Zohren, S., & Roberts, S. (2019). Deep LOB: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, *67*, 3001–3012. http://dx.doi.org/10.1109/TSP.2019.2907260.