DeepLOB: Deep Convolutional Neural Networks for Limit Order Books

Zihao Zhang, Stefan Zohren, and Stephen Roberts

Abstract—We develop a large-scale deep learning model to predict price movements from limit order book (LOB) data of cash equities. The architecture utilises convolutional filters to capture the spatial structure of the limit order books as well as LSTM modules to capture longer time dependencies. The proposed network outperforms all existing state-of-the-art algorithms on the benchmark LOB dataset [1]. In a more realistic setting, we test our model by using one year market quotes from the London Stock Exchange and the model delivers a remarkably stable out-of-sample prediction accuracy for a variety of instruments. Importantly, our model translates well to instruments which were not part of the training set, indicating the model's ability to extract universal features. In order to better understand these features and to go beyond a "black box" model, we perform a sensitivity analysis to understand the rationale behind the model predictions and reveal the components of LOBs that are most relevant. The ability to extract robust features which translate well to other instruments is an important property of our model which has many other applications.

I. INTRODUCTION

IN today's competitive financial world more than half of the markets use electronic Limit Order Books (LOBs) [2] to record trades [3]. Unlike traditional quote-driven marketplaces, where traders can only buy or sell an asset at one of the prices made publicly by market makers, traders now can directly view all resting limit orders[1] in the limit order book of an exchange. Because limit orders are arranged into different levels based on their submitted prices, the evolution in time of a LOB represents a multi-dimensional problem with elements representing the numerous prices and order volumes/sizes at multiple levels of the LOB on both the buy and sell sides.

A LOB is a complex dynamic environment with high dimensionality, inducing modelling complications that make traditional methods difficult to cope with. Mathematical finance is often dominated by models of evolving price sequences. This leads to a range of Markov-like models with stochastic driving terms, such as the vector autoregressive model (VAR) [4] or the autoregressive integrated moving average model (ARIMA) [5]. These models, to avoid excessive parameter spaces, often rely on handcrafted features of the data. However, given the billions of electronic market quotes that are generated everyday, it is natural to employ more modern data-driven machine learning techniques to extract such features.

In addition, limit order data, like any other financial time-series data is notoriously non-stationary and dominated by stochastics. In particular, orders at deeper levels of the LOB are often placed and cancelled in anticipation of future price moves and are thus even more prone to noise. Other problems, such as auction and dark pools [6], also add additional difficulties, bringing ever more unobservability into the environment. The interested reader is referred to [7] in which a number of these issues are reviewed.

In this paper we design a novel deep neural network architecture that incorporates both convolutional layers as well as Long Short-Term Memory (LSTM) units to predict future stock price movements in large-scale high-frequency LOB data. One advantage of our model over previous research [8] is that it has the ability to adapt for many stocks by extracting representative features from highly noisy data.

In order to avoid the limitations of handcrafted features, we use a so-called Inception Module [9] to wrap convolutional and pooling layers together. The Inception Module helps to infer local interactions over different time horizons. The resulting feature maps are then passed into LSTM units which can capture dynamic temporal behaviour. We test our model on a publicly available LOB dataset, known as FI-2010 [1], and our method remarkably outperforms all existing state-of-the-art algorithms. However, the FI-2010 dataset is only made up of 10 consecutive days of down-sampled pre-normalised data from a less liquid market. While it is a valuable benchmark set, it is arguable not sufficient to fully verify the robustness of an algorithm. To ensure the generalisation ability of our model, we further test it by using one year order book data for 5 stocks from the London Stock Exchange (LSE). To minimise the problem of overfitting to backtest data, we carefully optimise any hyper-parameter on a separate validation set before moving to the out-of-sample test set. Our model delivers robust out-of-sample prediction accuracy across stocks over a test period of three months.

As well as presenting results on out-of-sample data (in a timing sense) from stocks used to form the training set, we also test our model on out-of-sample (in both timing and data stream sense) stocks that are not part of the training set. Interestingly, we still obtain good results over the whole testing period. We believe this observation shows not only that the proposed model is able to extract robust features from order books, but also indicates the existence of universal features in the order book that modulate stock demand and price. The ability to transfer the model to new instruments opens up a

The authors are with the Oxford-Man Institute of Quantitative Finance, Department of Engineering Science, University of Oxford (e-mail: zihao@robots.ox.ac.uk).

Github: https://github.com/zcakhaa

[1]Limit orders are orders that do not match immediately upon submission and are also called passive orders. This is opposed to orders that match immediately, so-called aggressive orders, such as a market order. A LOB is simply a record of all resting/outstanding limit orders at a given point in time.

number of possibilities that we consider for future work.

To show the practicability of our model we use it in a simple trading simulation. We focus on sufficiently liquid stocks so that slippage and market impact are small. Indeed, these stocks are generally harder to predict than less liquid ones. Since our trading simulation is mainly meant as a method of comparison between models we assume trading takes place at mid-price[2] and compare gross profits before fees. The former assumption is equivalent to assuming that one side of the trade may be entered into passively and the latter assumes that different models trade similar volumes and would thus be subject to similar fees. Our focus here is using a simulation as a measure of the relative value of the model predictions in a trading setting. Under these simplifications, our model delivers significantly positive returns with a relatively small risk.

Although our network achieves good performance, a complex "black box" system, such as a deep neural network, has limited use for financial applications without some understanding of the rationale behind the model predictions. Here we exploit the model-agnostic LIME method [10] to highlight highly relevant components in the order book to gain a better understanding between our predictions and model inputs. Reassuringly, these conform to sensible (though arguably unusual) patterns of activity in both price and volume within the order book.

Outline: The remainder of the paper is as follows. Section II introduces background and related work. Section III describes limit order data and the various stages of data preparation. We present our network architecture in Section IV and give justifications behind each component of the model. In Section V we compare our work with a large group of popular methods. Section VI summarises our findings and considers extensions and future work.

II. BACKGROUND AND RELATED WORK

Research on the predictability of stock markets has a long history in the financial literature e.g., [11, 12]. Although opinions differ regarding the efficiency of markets, many widely accepted studies show that financial markets are to some extent predictable [13, 14, 15, 16]. Two major classes of work which attempt to forecast financial time-series are, broadly speaking, statistical parametric models and data-driven machine learning approaches [17]. Traditional statistical methods generally assume that the time-series under study are generated from a parametric process [18]. There is, however, agreement that stock returns behave in more complex ways, typically highly nonlinearly [19, 20]. Machine learning techniques are able to capture such arbitrary nonlinear relationships with little, or no, prior knowledge regarding the input data [21].

Recently, there has been a surge of interest to predict limit order book data by using machine learning algorithms [1, 22, 23, 24, 25, 26, 27, 20, 28, 29]. Among many machine learning techniques, pre-processing or feature extraction is often performed as financial time-series data is highly stochastic. Generic feature extraction approches have been implemented, such as the Principal Component Analysis (PCA) and the

Linear Discriminant Analysis (LDA) in the work of [24]. However these extraction methods are static pre-processing steps, which are not optimised to maximise the overall objective of the model that observes them. In the work of [25, 24], the Bag-of-Features model (BoF) is expressed as a neural layer and the model is trained end-to-end using the backpropagation algorithm, leading to notably better results on the FI-2010 dataset [1]. These works suggest the importance of a data driven approach to extract representative features from a large amout of data. In our work, we advocate the end-to-end training and show that the deep neural network by itself not only leads to even better results but also transfers well to new instruments (not part of the training set) - indicating the ability of networks to extract "universal" features from the raw data.

Arguably, one of the key contributions of modern deep learning is the addition of feature extraction and representation as part of the learned model. The Convolutional Neural Network (CNN) [30] is a prime example, in which information extraction, in the form of filter banks, is automatically tuned to the utility function that the entire network aims to optimise. CNNs have been successfully applied to various application domains, for example, object tracking [31], object-detection [32] and segmentation [33]. However, there have been but a few published works that adopt CNNs to analyse financial microstructure data [34, 35, 26] and the existing CNN architectures are rather unsophisticated and lack of thorough investigation. Just like when moving from "AlexNet" [36] to "VGGNet" [37], we show that a careful design of network archiecture can lead to better results compared with all existing methods.

The Long Short-Term Memory (LSTM) [38] was originally proposed to solve the vanishing gradients problem [39] of recurrent neural networks, and has been largely used in applications such as language modelling [40] and sequence to sequence learning [41]. Unlike CNNs which are less widely applied in financial markets, the LSTM has been popular in recent years, [42, 28, 43, 44, 45, 46, 47, 20] all utilising LSTMs to analyse financial data. In particular, [20] uses limit order data from 1000 stocks to test a four layer LSTM model. Their results show a stable out-of-sample prediction accuracy across time, indicating the potential benefits of deep learning methods. To the best of our knowledge, there is no work that combines CNNs with LSTMs to predict stock price movements and this is the first extensive study to apply a nested CNN-LSTM model to raw market data. In particular, the usage of the Inception Model in this context is novel and is essential in inferring the optimal "decay rates" of the extracted features.

III. DATA, NORMALISATION AND LABELLING

A. Limit Order Books

We first introduce some basic definitions of limit order books (LOBs). For classical references on market microstructure the reader is referred to [48, 49] and for a short review on LOBs in particular we refer to [7]. Here we follow the conventions of [7]. A LOB has two types of orders: bid orders and ask orders. A bid (ask) order is an order to buy (sell) an

---

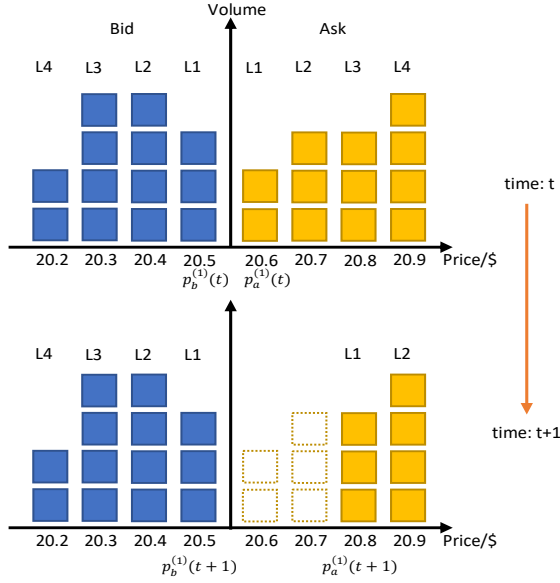[2]The average of the best buy and best sell prices in the market at the time.

Figure 1. A slice of LOB at time t and $t + 1$. L1 represents the respective first level, L2 the second, etc. $p_a^{(1)}(t)$ is the lowest ask price (best ask) and $p_b^{(1)}(t)$ is the highest bid price (best bid) at time t.

asset at or below (above) a specified price. The bid orders have prices $\mathbf{P}_b(t)$ and sizes/volumes $\mathbf{V}_b(t)$, and the ask orders have prices $\mathbf{P}_a(t)$ and sizes/volumes $\mathbf{V}_a(t)$. Both $\mathbf{P}(t)$ and $\mathbf{V}(t)$ are vectors representing values at different price levels of an asset.

Figure 1 illustrates the above concepts. The upper plot shows a slice of a LOB at time t. Each square in the plot represents an order of nominal size 1. This is done for simplicity, in reality different orders can be of different sizes. The blue bars represent bid orders and the yellow bars represent ask orders. Orders are sorted into different levels based on their submitted prices, where L1 represents the first level and so on. Each level contains two values: price and volume. On the bid side, $\mathbf{P}_b(t)$ and $\mathbf{V}_b(t)$ are 4-vectors in this example. We use $p_b^{(1)}(t)$ to denote the highest available price for a buying order (first bid level). Similarly, $p_a^{(1)}(t)$ is the lowest available selling order (first ask level). The bottom plot shows the action of an incoming market order to buy 5 shares at time $t + 1$. As a result, the entire first and second ask-levels are executed against that order and $p_a^{(1)}(t + 1)$ moved to 20.8 from 20.6 at time t.

B. Input Data

We test our model on two datasets: the FI-2010 dataset [1] and one year length of limit order book data from the London Stock Exchange (LSE). The FI-2010 dataset [1] is the first publicly available benchmark dataset of high-frequency limit order data and extracted time series data for five stocks from the Nasdaq Nordic stock market for a time period of 10 consecutive days. Many earlier algorithms are tested on this dataset and we use it to establish a fair comparison to other algorithms. However, 10 days is an insufficient amount of data to fully test the robustness and generalisation ability of an algorithm as the problem of overfitting to backtest data

is severe and we often expect a signal to be consistent over a few months.

To address the above concerns, we train and test our model on limit order book data of one year length for Lloyds Bank, Barclays, Tesco, BT and Vodafone. These five instruments are among the most liquid stocks listed on the London Stock Exchange. It is generally more difficult to train models on more liquid stocks, but at the same time, those instruments are easier to trade without price impact so making the simple trading simulation used to assess performance more realistic. The data includes all LOB updates for the above names. It spans all trading days from 3rd January 2017 to 24th December 2017 and we restrict it to the interval between 08:30:00 and 16:00:00, so that only normal trading activities occur and no auction takes place. Each state of the LOB contains 10 levels on each side and each level contains information on both price and volume. Therefore, we have a total of 40 features at each timestamp. Note that the FI-2010 dataset is actually downsampled limit order book data because the authors followed [50] to create additional features by using every non-overlapping block of 10 events. We did not perform any processing on our data and only feed raw order book information to our algorithm.

Overall, our LSE dataset is made up of 12 months, and has more than 134 million samples. On average, there are 150,000 events per day per stock. The events are irregularly spaced in time. The time interval, $\Delta_{k,k+1}$, between two events can vary considerably from a fraction of a second to seconds, and $\Delta_{k,k+1}$ is on average 0.192 seconds in the dataset. We take the first 6 months as training data, the next 3 months as validation data and the last 3 months as test data. In the context of high-frequency data, 3 months test data corresponds to millions of observations and therefore provides sufficient scope for testing model performance and estimating model accuracy.

C. Data Normalisation and Labelling

The FI-2010 dataset [1] provides 3 different normalised dataset: z-score, min-max and decimal precision normalisation. We used data normalised by z-score without any emendation and found subtle difference when using the other two normalisation schemes. For the LSE dataset, we again use standardisation (z-score) to normalise our data, but use the mean and standard deviation of the previous 5 days' data to normalise the current day's data (with a separate normalisation for each instrument). We want to emphasize the importance of normalisation because the performance of machine learning algorithms often depends it. As financial time-series usually experiences regime shifts, using a static normalisation scheme is not appropriate for a dataset of one year length. The above method is dynamic and the normalised data often falls into a reasonable range. We use the 100 most recent states of the LOB as an input to our model for both datasets. Specifically, a single input is defined as $X = [x_1, x_2, \cdots, x_t, \cdots, x_{100}]^T \in \mathbb{R}^{100 \times 40}$, where $x_t = [p_a^{(i)}(t), v_a^{(i)}(t), p_b^{(i)}(t), v_b^{(i)}(t)]_{i=1}^{n=10}$. $p^{(i)}$ and $v^{(i)}$ denote the price and volume size at i-th level of a limit order book.

After normalising the limit order data, we use the mid-price

$$p_t = \frac{p_a^{(1)}(t) + p_b^{(1)}(t)}{2}, \tag{1}$$

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.

Because financial data is highly stochastic, if we simply compare p_t and p_{t+k} to decide the price movement, the resulting label set will be noisy. In the works of [1] and [26], two smoothing labelling methods are introduced. We briefly recall the two methods here. First, let m_- denote the mean of the previous k mid-prices and m_+ denote the mean of the next k mid-prices:

$$m_-(t) = \frac{1}{k} \sum_{i=0}^{k} p_{t-i}$$
$$m_+(t) = \frac{1}{k} \sum_{i=1}^{k} p_{t+i} \tag{2}$$

where p_t is the mid-price defined in Equation (1) and k is the prediction horizon. Both methods use the percentage change (l_t) of the mid-price to decide directions. We can now define

$$l_t = \frac{m_+(t) - p_t}{p_t} \tag{3}$$

$$l_t = \frac{m_+(t) - m_-(t)}{m_-(t)} \tag{4}$$

Both are methods to define the direction of price movement at time t, where the former, Equation 3, was used in [1] and the latter, Equation 4, in [26].

The labels are then decided based on a threshold (α) for the percentage change (l_t). If $l_t > \alpha$ or $l_t < -\alpha$, we define it as up ($+1$) or down (-1). For anything else, we consider it as stationary (0). Figure 2 provides a graphical illustration of two labelling methods on the same threshold (α) and the same prediction horizon (k). All the labels classified as down (-1) are shown as red areas and up ($+1$) as green areas. The uncoloured (white) regions correspond to stationary (0) labels.

The FI-2010 dataset [1] adopts the method in Equation 3 and we directly used their labels for fair comparison to other methods. However, the produced labels are less consistent as shown on the top of Figure 2 because this method fits closer to real prices as smoothing is only applied to future prices. This is essentially detrimental for designing trading algorithms as signals are not consistent here leading to many redundant trading actions thus incurring larger transaction costs.

Further, the FI-2010 dataset was collected in 2010 and the instruments were less liquid compared to now. We experimented with this approach in [1] on our data from the London Stock Exchange and found the resulting labels are rather stochastic, therefore we adopt the method in Equation 4 for our LSE dataset to produce more consistent signals.
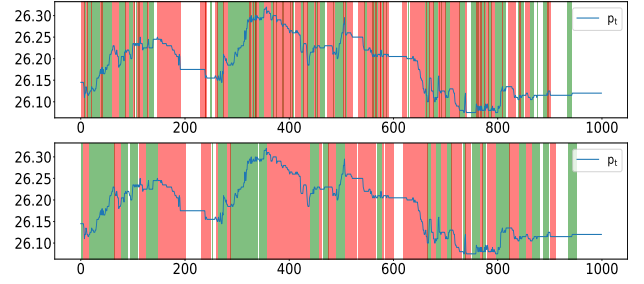

Figure 2. An example of two smoothed labelling methods based on a same threshold (α) and same prediction horizon (k). Green shading represents a +1 signal and red a -1. **Top:** [1]'s method and **Bottom:** [26]'s method.

IV. MODEL ARCHITECTURE

A. Overview

We here detail our network architecture, which comprises three main building blocks: standard convolutional layers, an Inception Module and a LSTM layer, as shown in Figure 3. The main idea of using CNNs and Inception Modules is to automate the process of feature extraction as it is often difficult in financial applications since financial data is notoriously noisy with a low signal-to-noise ratio. Technical indicators such as MACD and the Relative Strength Index are included as inputs and preprocessing mechanisms such as principal component analysis (PCA) [51] are often used to transform raw inputs. However, none of these processes is trivial, they make tacit assumptions and further, it is questionable if financial data can be well-described with parametric models with fixed parameters. In our work, we only require the history of LOB prices and sizes as inputs to our algorithm. Weights are learned during inference and features, learned from a large training set, are data-adaptive, removing the above constraints. A LSTM layer is then used to capture additional time dependencies among the resulting features. We note that very short time-dependencies are already captured in the convolutional layer which takes "space-time images" of the LOB as inputs.

B. Details of Each Component

a) Convolutional Layer: Recent development of electronic trading algorithms often submit and cancel vast numbers of limit orders over short periods of time as part of their trading strategies [52]. These actions often take place deep in a LOB and it is seen [7] that more than 90% of orders end in cancellation rather than matching, therefore practitioners consider levels further away from best bid and ask levels to be less useful in any LOB. In addition, the work of [53] suggests that the best ask and best bid (L1-Ask and L1-Bid) contribute most to the price discovery and the contribution of all other levels is considerably less, estimated at as little as 20%. As a result, it would be otiose to feed all level information to a neural network as levels deep in a LOB are less useful and can potentially even be misleading. Naturally, we can smooth these signals by summarising the information contained in deeper levels. We note that convolution filters used in any CNN architecture are discrete convolutions, or finite impulse response (FIR) filters, from the viewpoint of
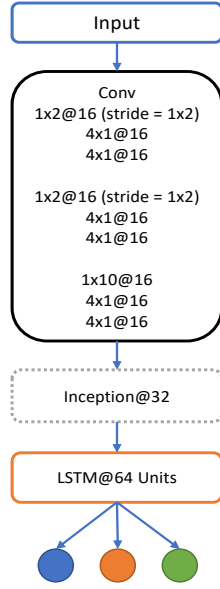
Figure 3. Model architecture schematic. Here 1x2@16 represents a convolutional layer with 16 filters of size (1×2). '1' convolves through time indices and '2' convolves different limit order book levels.

signal processing [54]. FIR filters are popular smoothing techniques for denoising target signals and they are simple to implement and work with. We can write any FIR filter in the following form:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \qquad (5)$$

where the output signal $y(n)$ at any time is a weighted sum of a finite number of past values of the input signal $x(n)$. The filter order is denoted as M and b_k is the filter coefficient. In a convolutional neural network, the coefficients of the filter kernel are not obtained via a statistical objective from traditional signal filtration theory, but are left as degrees of freedom which the network infers so as to extremise its value function at output.

The details of the first convolutional layer inevitably need some consideration. As convolutional layers operate a small kernel to "scan" through input data, the layout of limit order book information is vital. Recall that we take the most 100 recent updates of an order book to form a single input and there are 40 features per time stamp, so the size of a single input is (100×40). We organise the 40 features as following:

$$\{p_a^{(i)}(t), v_a^{(i)}(t), p_b^{(i)}(t), v_b^{(i)}(t)\}_{i=1}^{n=10} \qquad (6)$$

where i denotes the i-th level of a limit order book. The size of our first convolutional filter is (1×2) with stride of (1×2). The first layer essentially summarises information between price and volume $\{p^{(i)}, v^{(i)}\}$ at each order book level. The usage of stride is necessary here as an important property of convolutional layers is parameter sharing. This property is attractive as less parameters are estimated, largely avoiding overfitting problems. However, without strides, we would apply same parameters to $\{p^{(i)}, v^{(i)}\}$ and $\{v^{(i)}, p^{(i+1)}\}$. In other words, $p^{(i)}$ and $v^{(i)}$ would share same parameters

because the kernel filter moves by one step, which is obviously wrong as price and volume form different dynamic behaviors.

Because the first layer only captures information at each order book level, we would expect representative features to be extracted when integrating information across multiple order book levels. We can do this by utilising another convolutional layer with filter size (1×2) and stride (1×2). The resulting feature maps actually form the micro-price defined by [55]:

$$p^{\text{micro price}} = I p_a^{(1)} + (1 - I) p_b^{(1)}$$
$$I = \frac{v_b^{(1)}}{v_a^{(1)} + v_b^{(1)}} \qquad (7)$$

The weight I is called the imbalance. The micro-price is an important indicator as it considers volumes on bid and ask side, and the imbalance between bid and ask size is a very strong indicator of the next price move. This feature of imbalances has been reported by a variety of researchers [56, 57, 58, 59, 60]. Unlike the micro-price where only the first order book level is considered, we utilise convolutions to form micro-prices for all levels of a LOB so the resulting features maps are of size $(100, 10)$ after two layers with strides. Finally, we integrate all information by using a large filter of size (1×10) and the dimension of our feature maps before the Inception Module is $(100, 1)$.

We apply zero padding to every convolutional layer so the time dimension of our inputs does not change and Leaky Rectifying Linear Units (Leaky-ReLU) [61] are used as activation functions. The hyper-parameter (the small gradient when the unit is not active) of the Leaky-ReLU is set to 0.01, evaluated by grid search on the validation set.

Another important property of convolution is that of equivariance to translation [62]. Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. For example, suppose that there exists a main classification feature m located at (x_m, y_m) of an image $I(x, y)$. If we shift every pixel of I one unit to the right, we get a new image I' where $I'(x, y) = I(x - 1, y)$. We can still obtain the main classification feature m' in I' and $m = m'$, while the location of m' will be at $(x_{m'}, y_{m'}) = (x_m - 1, y_m)$. This is important to time-series data, because convolution can find universal features that are decisive to final outputs. In our case, suppose a feature that studies imbalance is obtained at time t. If the same event happens later at time t' in the input, the exact feature can be extracted later at t'.

We do not use any pooling layer except in the Inception Modules. Although pooling layers help us find representations invariant to translations of the input, the smoothing nature of pooling can cause under-fitting. Common pooling layers are designed for image processing tasks, and they are most powerful when we only care if certain features exist in the inputs instead of where they exist [62]. Time-series data has different characteristics from images and the location of representative features is important. Our experiences show that pooling layers in the convolutional layer, at least, cause under-fitting problems to the LOB data. However, we think pooling is important and new pooling methods should be designed to process time-series data as it is a promising
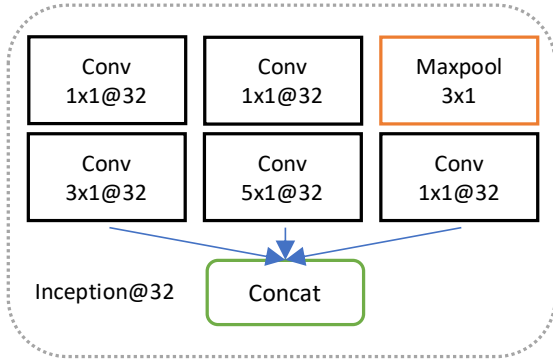
Figure 4. The Inception Module used in the model. For example, 3×1@32 represents a convolutional layer with 32 filters of size (3×1).

solution to extract invariant features.

b) Inception Module: We note that all filters of a standard convolutional layer have fixed size. If, for example, we employ filters of size (4×1), we capture local interactions amongst data over four time steps. However, we can capture dynamic behaviours over multiple timescales by using Inception Modules to wrap several convolutions together. We find that this offers a performance improvement to the resultant model.

The idea of the Inception Module can be also considered as using different moving averages in technical analysis. Practitioners often use moving averages with different decay weights to observe time-series momentum [63]. If a large decay weight is adopted, we get a smoother time-series that well represents the long-term trend, but we could miss small variations that are important in high-frequency data. In practice, it is a daunting task to set the right decay weights. Instead, we can use Inception Modules and the weights are then learned during back-propagation.

In our case, we split the input into a small set of lower-dimensional representations by using 1×1 convolutions, transform the representations by a set of filters, here 3×1 and 5×1, and then merge the outputs. A max-pooling layer is used inside the Inception Module, with stride 1 and zero padding. "Inception@32" represents one module and indicates all convolutional layers have 32 filters in this module, and the approach is depicted schematically in Figure 4. The 1×1 convolutions form the Network-in-Network approach proposed in [64]. Instead of applying a simple convolution to our data, the Network-in-Network method uses a small neural network to capture non-linear properties of our data. We find this method to be effective and it gives us an improvement on prediction accuracy.

c) LSTM Module and Output: In general, a fully connected layer is used to classify the input data. However, all inputs to the fully connected layer are assumed independent of each other unless multiple fully connected layers are used. Due to the usage of Inception Module in our work, we have a large number of features at end. Just using one fully connected layer with 64 units would result in more than 630,000 parameters

to be estimated, not to mention multiple layers. In order to capture temporal relationship that exist in the extracted features, we replace the fully connected layers with LSTM units. The activation of a LSTM unit is fed back to itself and the memory of past activations is kept with a separate set of weights, so the temporal dynamics of our features can be modelled. We use 64 LSTM units in our work, resulting in about 60,000 parameters, leading to 10 times fewer parameters to be estimated. The last output layer uses a softmax activation function and hence the final output elements represent the probability of each price movement class at each time step.

V. EXPERIMENTAL RESULTS

A. Experiments Settings

We apply the same architecture to all our experiments in this section and the proposed model is denoted as DeepLOB. We learn the parameters by minimising the categorical cross-entropy loss. The Adaptive Moment Estimation algorithm, ADAM [65], is utilised and we set the parameter "epsilon" to 1 and the learning rate to 0.01. The learning is stopped when validation accuracy does not improve for 20 more epochs. This is about 100 epochs for the FI-2010 dataset and 40 epochs for the LSE dataset.

We train with mini-batches of size 32. We choose a small mini-batch size due to the findings in [66] in which they suggest that large-batch methods tend to converge to narrow deep minima of the training functions, but small-batch methods consistently converge to shallow broad minima. All models are built using Keras [67] based on the TensorFlow backend [68], and we train them using a single NVIDIA Tesla P100 GPU.

B. Experiments on the FI-2010 Dataset

There are two experimental setups using the FI-2010 dataset. Following the convention of [24], we denote them as Setup 1 and Setup 2. Setup 1 splits the dataset into 9 folds based on a day basis (a standard anchored forward split). In the i-th fold, we train our model on the first i days and test it on the $(i+1)$-th day where $i = 1, \cdots, 9$. The second setting, Setup 2, originates from the works [26, 28, 27, 25] in which deep network architectures were evaluated. As deep learning techniques often require a large amount of data to calibrate weights, the first 7 days are used as the train data and the last 3 days are used as the test data in this setup. We evaluate our model in both setups here.

Table I shows the results of our model compared to other methods in Setup 1. Performance is measured by calculating the mean accuracy, recall, precision, and F1 score over all folds. As the FI-2010 dataset is not well balanced, [1] suggests to focus on F1 score performance as fair comparisons. We have compared our model to all existing experimental results including Ridge Regression (RR) [1], Single-Layer-Feedforward Network (SLFN) [1], Linear Discriminant Analysis (LDA) [22], Multilinear Discriminant Analysis (MDA) [22], Multilinear Time-series Regression (MTR) [22], Weighted Multilinear Time-series Regression (WMTR) [22], Multilinear

Table I
SETUP 1: EXPERIMENT RESULTS FOR THE FI-2010 DATASET

Model	Accuracy %	Precision %	Recall %	F1 %
Prediction Horizon k = 10				
RR [1]	48.00	41.80	43.50	41.00
SLFN [1]	64.30	51.20	36.60	32.70
LDA [22]	63.83	37.93	45.80	36.28
MDA [22]	71.92	44.21	60.07	46.06
MCSDA [23]	83.66	46.11	48.00	46.72
MTR [22]	86.08	51.68	40.81	40.14
WMTR [22]	81.89	46.25	51.29	47.87
BoF [24]	57.59	39.26	51.44	36.28
N-BoF [24]	62.70	42.28	61.41	41.63
B(TABL) [25]	73.62	66.16	68.81	67.12
C(TABL) [25]	78.01	72.03	74.04	72.84
DeepLOB	78.91	78.47	78.91	**77.66**
Prediction Horizon k = 50				
RR [1]	43.90	43.60	43.30	42.70
SLFN [1]	47.30	46.80	46.40	45.90
BoF [24]	50.21	42.56	49.57	39.56
N-BoF [24]	56.52	47.20	58.17	46.15
B(TABL) [25]	69.54	69.12	68.84	68.84
C(TABL) [25]	74.81	74.58	74.27	74.32
DeepLOB	75.01	75.10	75.01	**74.96**
Prediction Horizon k = 100				
RR [1]	42.90	42.90	42.90	41.60
SLFN [1]	47.70	45.30	43.20	41.00
BoF [24]	50.97	42.48	47.84	40.84
N-BoF [24]	56.43	47.27	54.99	46.86
B(TABL) [25]	69.31	68.95	69.41	68.86
C(TABL) [25]	74.07	73.51	73.80	73.52
DeepLOB	76.66	76.77	76.66	**76.58**

Table II
SETUP 2: EXPERIMENT RESULTS FOR THE FI-2010 DATASET

Model	Accuracy %	Precision %	Recall %	F1 %
Prediction Horizon k = 10				
SVM [28]	-	39.62	44.92	35.88
MLP [28]	-	47.81	60.78	48.27
CNN-I [26]	-	50.98	65.54	55.21
LSTM [28]	-	60.77	75.92	66.33
CNN-II [27]	-	56.00	45.00	44.00
B(TABL) [25]	78.91	68.04	71.21	69.20
C(TABL) [25]	84.70	76.95	78.44	77.63
DeepLOB	84.47	84.00	84.47	**83.40**
Prediction Horizon k = 20				
SVM [28]	-	45.08	47.77	43.20
MLP [28]	-	51.33	65.20	51.12
CNN-I [26]	-	54.79	67.38	59.17
LSTM [28]	-	59.60	70.52	62.37
CNN-II [27]	-	-	-	-
B(TABL) [25]	70.80	63.14	62.25	62.22
C(TABL) [25]	73.74	67.18	66.94	66.93
DeepLOB	74.85	74.06	74.85	**72.82**
Prediction Horizon k = 50				
SVM [28]	-	46.05	60.30	49.42
MLP [28]	-	55.21	67.14	55.95
CNN-I [26]	-	55.58	67.12	59.44
LSTM [28]	-	60.03	68.58	61.43
CNN-II [27]	-	56.00	47.00	47.00
B(TABL) [25]	75.58	74.58	73.09	73.64
C(TABL) [25]	79.87	79.05	77.04	78.44
DeepLOB	80.51	80.38	80.51	**80.35**

Table III
AVERAGE COMPUTATION TIME OF STATE-OF-THE-ART MODELS

Models	Forward (ms)	Number of parameters
BoF [24]	0.972	86k
N-BoF [24]	0.524	12k
CNN-I [26]	0.025	768k
LSTM [28]	0.061	-
C(TABL) [25]	0.229	-
DeepLOB	0.253	60k

Class-specific Discriminant Analysis (MCSDA) [23], Bag-of-Feature (BoF) [24], Neural Bag-of-Feature (N-BoF) [24], and Attention-augmented-Bilinear-Network with one hidden layer (B(TABL)) and two hidden layers (C(TABL)) [25]. More methods such as PCA and Autoencoder (AE) are actually tested in their works but, for simplicity, we only report their best results and our model achieves better performance.

However, the Setup 1 is not ideal for training deep learning models as we mentioned that deep network often requires a large amount of data to calibrate weights. This anchored forward setup leads to only one or two days' training data for the first few folds and we observe worse performance in the first few days. As training data grows, we observe remarkably better results as shown in Table II which shows the results of our network compared to other methods in Setup 2. In particular, the important difference between our model and CNN-I [26] and CNN-II [27] is due to network architecture and we can see huge improvements on performance here. In Table III, we compare the parameter sizes of DeepLOB with CNN-I [26]. Although our model has many more layers, there are far fewer parameters in our network due to the usage of LSTM layers instead of fully connected layers.

We also report the computation time (forward pass) in milliseconds (ms) for available algorithms in Table III. Due to the development of GPUs, training deep networks is now feasible and it is swift to make predictions, making it possible for high frequency trading. We will discuss this more in the next section.

C. Experiments on the London Stock Exchange (LSE)

As we suggested, the FI-2010 dataset is not sufficient to verify a prediction model - it is far too short, downsampled and taken from a less liquid market. To perform a meaningful evaluation that can hold up to modern applications, we further test our method on stocks from the LSE of one year length with a testing period of three months. As mentioned in Section III, we train our model on five stocks: Lloyds Bank (LLOY), Barclays (BARC), Tesco (TSCO), BT and Vodafone (VOD). Recent work of [20] suggests that deep learning techniques can extract universal features for limit order data. To test this universality, we directly apply our model to five more stocks that were not part of the training data set **(transfer learning)**. We select HSBC, Glencore (GLEN), Centrica (CNA), BP and ITV for transfer learning because they are also among the most liquid stocks in the LSE. The testing period is the same three months as before, and the classes are roughly balanced.

Table IV presents the results of our model for all stocks on different prediction horizons. To better investigate the results,

TABLE IV
EXPERIMENT RESULTS FOR THE LSE DATASET

Prediction Horizon	Accuracy %	Precision %	Recall %	F1 %
Results on LLOY, BARC, TSCO, BT and VOD				
k=20	70.17	70.17	70.17	70.15
k=50	63.93	63.43	63.93	63.49
k=100	61.52	60.73	61.52	60.65
Results on Transfer Learning (GLEN, HSBC, CNA, BP, ITV)				
k=20	68.62	68.64	68.63	68.48
k=50	63.44	62.81	63.45	62.84
k=100	61.46	60.68	61.46	60.77

Figure 5. Confusion matrices. **Top:** results on LLOY, BARC, TSCO, BT and VOD. From the left to right, prediction horizon (k) equals 20, 50 and 100; **Bottom:** results on transfer learning (GLEN, HSBC, CNA, BP, ITV).
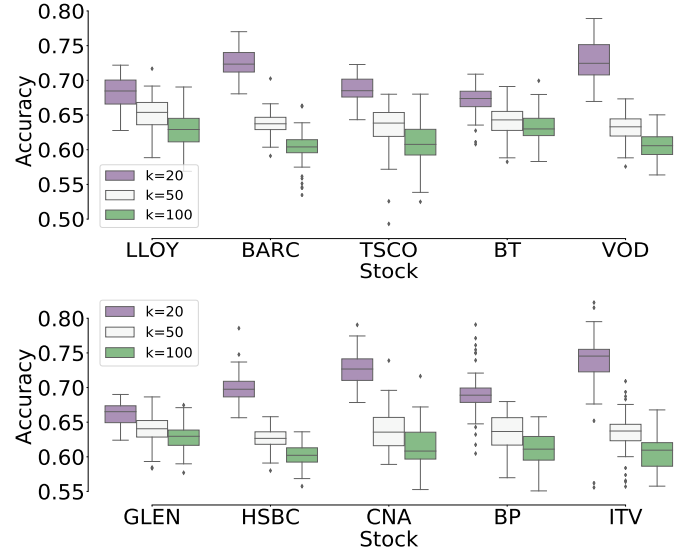

Figure 6. Boxplots of daily accuracy for the different prediction horizons. **Top:** results on LLOY, BARC, TSCO, BT and VOD; **Bottom:** results on transfer learning (GLEN, HSBC, CNA, BP, ITV).

we display the confusion matrices in Figure 5 and calculate the accuracy for every day and for every stock across the testing period. We use the boxplots in Figure 6 to present this information and we can observe consistent and robust performance, with narrow interquartile range (IQR) and few outliers, for all stocks across the testing period. The ability of our model that generalises well to data not in the training set indicates that the CNN block in the algorithms, acting to extract features from the LOB, can capture universal patterns that relate to the price formation mechanism. We find this observation most interesting.

D. Performance of the Model in a Simple Trading Simulation

A simple trading simulation is designed to test the practicability of our results. We set the number of shares per trade, μ, to one both for simplicity and to minimise the market impact, ensuring orders to be executed at the best price. Although μ can be optimised to maximise the returns, for example, prediction probabilities are used to size the orders in [69], we would like to show that our algorithm can work even under this simple set-up.

To reduce the number of trades, we use following rules to take actions. At each time-step, our model generates a signal from the network outputs $(-1, 0, +1)$ to indicate the price movements in k steps. Signals $(-1, 0, +1)$ correspond to actions (sell, wait and buy). Suppose our model produces a prediction of $+1$ at time t, we then buy μ shares at time $t+5$

(taking slippage into account), and hold until -1 appears to sell all μ shares (we do nothing if 0 appears). We apply the same rule to short selling and repeat the process during a day. All positions are closed by the end of the day, so we hold no stocks overnight. We make sure no trades take place at the time of auction, so no abnormal profits are generated.

As the focus of our work is on predictions and the above simple simulation is a way of showing that this prediction is in principle monetisable. In particular, our aim is not to present a fully developed, stand-alone trading strategy. Realistic high-frequency strategies often require a combination of various trading signals in particular to time the exact entry and exit points of the trade. For the purpose of the above simulation we use mid-prices without transaction costs. While in particular the second assumption is not a reasonable assumption for a standalone strategy, we argue that (i) it is enough for a relative comparison of the above models and (ii) it is a good indicator of the relative value of the above predictor to a more complex high-frequency trading model. Regarding the first assumption, a mid-mid simulation, we note that in high-frequency trading, many participants are involved in market making, as it is difficult to design profitable fully aggressive strategies with such short holding periods. If we assume that we are able to enter the trade passively, while we exit it aggressively, crossing the spread, then this is effectively equivalent to a mid-mid trade. Such a situation arises naturally for example in investment banks which are involved in client market making. Regarding the second assumption, careful timing of the entry points as well as more elaborate trading rules, such as including position upsizing, should be able to account for additional profits to cover the transaction costs. In any case, as merely a metric of testing predictability of our model, the above simple simulation suffices.

Figure 7 presents the boxplots for normalised daily profits (profits divided by number of trades in that day) for different
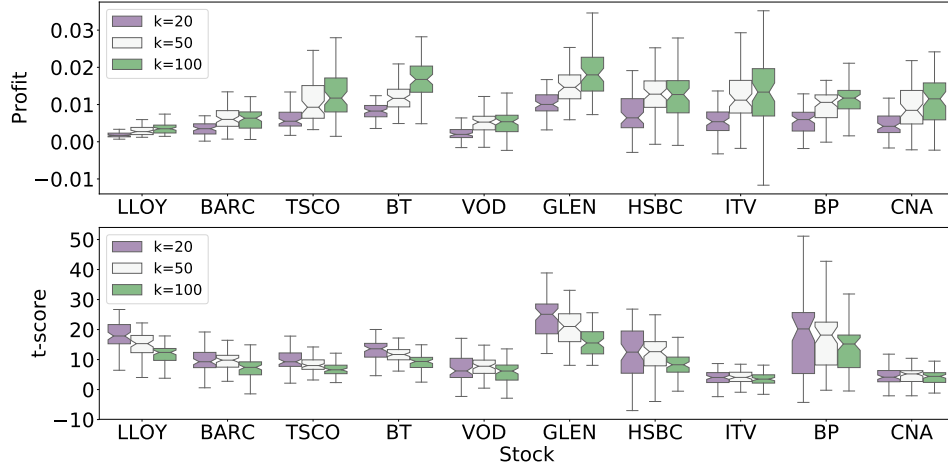
Figure 7. Boxplots for normalised daily profits and t-statistics for different stocks and prediction horizons (k). Profits are in GBX (= GBP/100).
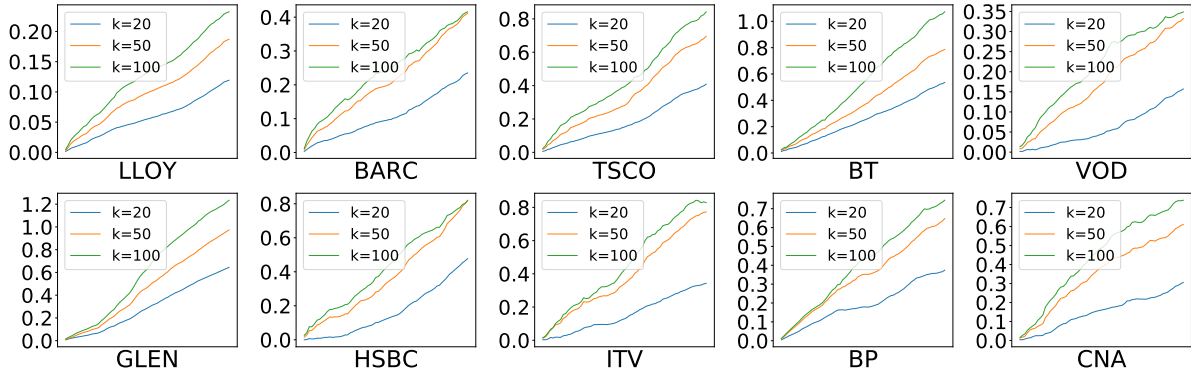

Figure 8. Normalised cumulative profits for test periods for different stocks and prediction horizons (k). Profits are in GBX(= GBP/100).

stocks and prediction horizons. We use a t-test to check if the profits are statistically greater than 0. The t-statistics is essentially the same as Sharpe ratios but a more consistent evaluation metric for high frequency trading. Figure 8 shows the cumulative profits across the testing period. We can observe consistent profits and significant t-values over the testing period for all stocks. Although we obtain worse accuracy for longer prediction horizons, the cumulative profits are actually higher as a more robust signal is generated.

E. Sensitivity Analysis

Trust and risk are fundamental in any financial application. If we take actions based on predictions, it is always important to understand the reasons behind those predictions. Neural networks are often considered as "black boxes" which lack interpretability. However, if we understand the relationship between the inputs' components (e.g. words in text, patches in an image) and the model's prediction, we can compare those relationships with our domain knowledge to decide if we can accept or reject a prediction.

The work of [10] proposes a method, which they call LIME, to obtain such explanations. In our case, we use LIME to reveal components of LOBs that are most important for predictions and to understand why the proposed model DeepLOB works

better than other network architectures such as CNN-I [26]. LIME uses an interpretable model to approximate the prediction of a complex model on a given input. It locally perturbs the input and observes variations in the model's predictions, thus providing some measure of information regarding input importance and sensitivity.

Figure 9 presents an example that shows how DeepLOB and CNN-I [26] react to a given input. In the figure we show the top 10 areas of pros (in green) and cons (in red) for the predicted class (yellow being the boundary). Not coloured areas represent the components of inputs that are less influential on the predicted results or "unimportant". We note that most components of the input are inactive for CNN-I [26]. We believe that this is due to two max-pooling layers used in that architecture. Because [26] used large-size filters in the first convolutional layer, any representation deep in the network actually represents information gleaned from a large portion of inputs. Our experiments applying LIME to many examples indicate this observation is a common feature.

VI. CONCLUSION

In this paper, we introduce the first hybrid deep neural network to predict stock price movements using high frequency limit order data. Unlike traditional hand-crafted models, where
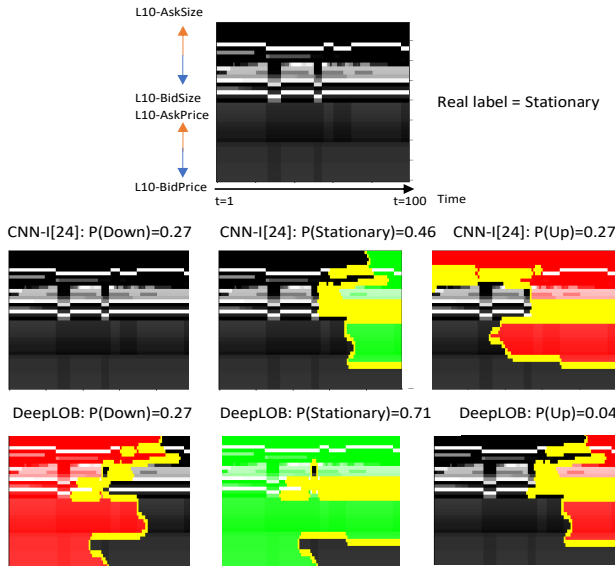
Figure 9. LIME plots. x-axis represents time stamps and y-axis represents levels of the LOB, as labelled in the top image. **Top:** Original image. **Middle:** Importance regions for CNN-I [26]. **Bottom:** Importance regions for DeepLOB model. Regions supportive for prediction are shown in green, and regions against in red. The boundary is shown in yellow.

features are carefully designed, we utilise a CNN and an Inception Module to automate feature extraction and use LSTM units to capture time dependencies.

The proposed method is evaluated against several baseline methods on the FI-2010 benchmark dataset and the results show that our model performs better than other techniques in predicting short term price movements. We further test the robustness of our model by using one year of limit order data from the LSE with a testing period of three months. An interesting observation from our work is that the proposed model generalises well to instruments that did not form part of the training data. This suggests the existence of universal features that are informative for price formation and our model appears to capture these features, learning from a large data set including several instruments. A simple trading simulation is used to further test our model and we obtain good profits that are statistically significant.

To go beyond the often-criticised "black box" nature of deep learning models, we use LIME, a method for sensitivity analysis, to indicate the components of inputs that contribute to predictions. A good understanding of the relationship between the input's components and the model's prediction can help us decide if we can accept a prediction. In particular, we see how the information of prices and sizes on different levels and horizons contribute to the prediction which is in accordance with our econometric understanding.

In a recent extension of this work we have modified the DeepLOB model to use Bayesian neural networks [69]. This allows to provide uncertainty measures on the network's outputs which for example can be used to upsize positions as demonstrated in [69].

In subsequent continuations of this work we would like to investigate more detailed trading strategies, using Rein-

forcement Learning, which are based on the feature extraction performed by DeepLOB.

REFERENCES

[1] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods," *Journal of Forecasting*, vol. 37, no. 8, pp. 852–866, 2018.

[2] C. A. Parlour and D. J. Seppi, "Limit order markets: A survey," *Handbook of financial intermediation and banking*, vol. 5, pp. 63–95, 2008.

[3] I. Rosu *et al.*, "Liquidity and information in order driven markets," Tech. Rep., 2010.

[4] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," *Modeling Financial Time Series with S-PLUS®*, pp. 385–429, 2006.

[5] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the ARIMA model," in *Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on*. IEEE, 2014, pp. 106–112.

[6] C. Carrie, "The new electronic trading regime of dark books, mashups and algorithmic trading," *Trading*, vol. 2006, no. 1, pp. 14–20, 2006.

[7] M. D. Gould, M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, and S. D. Howison, "Limit order books," *Quantitative Finance*, vol. 13, no. 11, pp. 1709–1742, 2013.

[8] W.-C. Chiang, D. Enke, T. Wu, and R. Wang, "An adaptive stock index trading decision support system," *Expert Systems with Applications*, vol. 59, pp. 195–207, 2016.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

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

[12] 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.

[13] 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.

[14] 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.

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

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

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

[18] 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.

[19] 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.

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

[21] 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.

[22] D. T. Tran, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. IEEE, 2017, pp. 1–7.

[23] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear class-specific discriminant analysis," *Pattern Recognition Letters*, vol. 100, pp. 131–136, 2017.

[24] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.

[25] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, 2018.

[26] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Business Informatics (CBI), 2017 IEEE 19th Conference on*, vol. 1. IEEE, 2017, pp. 7–12.

[27] ——, "Using Deep Learning for price prediction by exploiting stationary limit order book features," *arXiv preprint arXiv:1810.09965*, 2018.

[28] ——, "Using deep learning to detect price change indications in financial markets," in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 2511–2515.

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

[30] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[31] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in neural information processing systems*, 2013, pp. 809–817.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[33] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

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

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

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[37] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[40] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[41] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[42] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS one*, vol. 12, no. 7, p. e0180944, 2017.

[43] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on*. IEEE, 2017, pp. 1643–1647.

[44] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[45] L. Di Persio and O. Honchar, "Artificial neural networks architectures for stock price prediction: Comparisons and applications," *International Journal of Circuits, Systems and Signal Processing*, vol. 10, pp. 403–413, 2016.

[46] M. Dixon, "Sequence classification of the limit order book using recurrent neural networks," *Journal of computational science*, vol. 24, pp. 277–286, 2018.

[47] D. M. Nelson, A. C. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1419–1426.

[48] L. Harris, *Trading and exchanges: Market microstructure for practitioners*. Oxford University Press, USA, 2003.

[49] M. O'Hara, *Market microstructure theory*. Blackwell Publishers Cambridge, MA, 1995, vol. 108.

[50] A. N. Kercheval and Y. Zhang, "Modelling high-frequency limit order book dynamics with Support Vector Machines," *Quantitative Finance*, vol. 15, no. 8, pp. 1315–1329, 2015.

[51] A. Abraham, B. Nath, and P. K. Mahanti, "Hybrid intelligent systems for stock market analysis," in *International Conference on Computational Science*. Springer, 2001, pp. 337–345.

[52] T. Hendershott, C. M. Jones, and A. J. Menkveld, "Does algorithmic trading improve liquidity?" *The Journal of Finance*, vol. 66, no. 1, pp. 1–33, 2011.

[53] C. Cao, O. Hansch, and X. Wang, "The information content of an open limit-order book," *Journal of futures markets*, vol. 29, no. 1, pp. 16–41, 2009.

[54] S. J. Orfanidis, *Introduction to signal processing*. Prentice-Hall, Inc., 1995.

[55] J. Gatheral and R. C. Oomen, "Zero-intelligence realized variance estimation," *Finance and Stochastics*, vol. 14, no. 2, pp. 249–283, 2010.

[56] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 673–680.

[57] M. Avellaneda, J. Reed, and S. Stoikov, "Forecasting prices from Level-I quotes in the presence of hidden liquidity," *Algorithmic Finance*, vol. 1, no. 1, pp. 35–43, 2011.

[58] Y. Burlakov, M. Kamal, and M. Salvadore, "Optimal limit order execution in a simple model for market microstructure dynamics," 2012.

[59] L. Harris, "Maker-taker pricing effects on market quotations," *USC Marshall School of Business Working Paper. Avalable at http://bschool. huji. ac. il/. upload/hujibusiness/Maker-taker. pdf*, 2013.

[60] A. Lipton, U. Pesavento, and M. G. Sotiropoulos, "Trade arrival dynamics and quote imbalance in a limit order book," *arXiv preprint arXiv:1312.0514*, 2013.

[61] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[62] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[63] T. J. Moskowitz, Y. H. Ooi, and L. H. Pedersen, "Time series momentum," *Journal of financial economics*, vol. 104, no. 2, pp. 228–250, 2012.

[64] M. Lin, Q. Chen, and S. Yan, "Network in network," in *International Conference on Learning Representations*, 2014.

[65] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the International Conference on Learning Representations 2015*, 2015.

[66] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *International Conference on Learning Representations*, 2017.

[67] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[68] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[69] Z. Zhang, S. Zohren, and S. Roberts, "BDLOB: Bayesian Deep Convolutional Neural Networks for Limit Order Books," in *Third workshop on Bayesian Deep Learning (NeurIPS 2018)*, 2018.