IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 28, 2019, accepted June 17, 2019, date of publication June 21, 2019, date of current version July 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924353

Feature Engineering for Mid-Price Prediction With Deep Learning

ADAMANTIOS NTAKARIS[1], GIORGIO MIRONE[2], JUHO KANNIAINEN[1], MONCEF GABBOUJ[1], (Fellow, IEEE), AND ALEXANDROS IOSIFIDIS[3], (Senior Member, IEEE)

[1]Faculty of Information Technology and Communication Sciences, Tampere University of Technology, FI-33720 Tampere, Finland
[2]Danmarks Nationalbank, 1093 Copenhagen, Denmark
[3]Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus, Denmark

Corresponding authors: Adamantios Ntakaris (adamantios.ntakaris@tuni.fi) and Giorgio Mirone (gmi@nationalbanken.dk)

This work was supported by the H2020 Project BigDataFinance MSCA-ITN-ETN under Grant 675044.

ABSTRACT Mid-price movement prediction based on the limit order book data is a challenging task due to the complexity and dynamics of the limit order book. So far, there have been very limited attempts for extracting relevant features based on the limit order book data. In this paper, we address this problem by designing a new set of handcrafted features and performing an extensive experimental evaluation on both liquid and illiquid stocks. More specifically, we present an extensive set of econometric features that capture the statistical properties of the underlying securities for the task of mid-price prediction. The experimental evaluation consists of a head-to-head comparison with other handcrafted features from the literature and with features extracted from a long short-term memory autoencoder by means of a fully automated process. Moreover, we develop a new experimental protocol for online learning that treats the task above as a multi-objective optimization problem and predicts: 1) the direction of the next price movement and; 2) the number of order book events that occur until the change takes place. In order to predict the mid-price movement, features are fed into nine different deep learning models based on multi-layer perceptrons, convolutional neural networks, and long short-term memory neural networks. The performance of the proposed method is then evaluated on liquid and illiquid stocks (i.e., TotalView-ITCH US and Nordic stocks). For some stocks, results suggest that the correct choice of a feature set and a model can lead to the successful prediction of how long it takes to have a stock price movement.

INDEX TERMS Deep learning, econometrics, high-frequency trading, limit order book, mid-price, U.S. data.

I. INTRODUCTION

The automation of financial markets has increased the complexity of information analysis. This complexity can be effectively managed by the use of ordered trading universes like the limit order book (LOB). LOB is a formation that translates the daily unexecuted trading activity in price levels according to the type of orders (i.e., bid and ask side). The daily trading activity is a big data problem, since millions of trading events take place inside a trading session. Information extraction and digital signal (i.e., time series) analysis from every trading session provide the machine learning (ML) trader with useful instructions for orders, executions, and cancellations of trades.

The associate editor coordinating the review of this manuscript and approving it for publication was Bora Onat.

Traditional time series analysis methods have failed to capture the complexity of the contemporary trading markets adequately. For instance, the work in [1] and [2] suggest that classical machine learning and deep learning methods for financial metric predictions achieve better results compared to ARIMA and GARCH models. On the contrary, machine and deep learning methods have proved to be very effective mechanisms for time series analysis and prediction (e.g., [3], [4], [5]). The main advantage of these methods is their ability to capture non-linearities of the input data and filter them consecutively by creating new weighted features more relevant to the suggested problem.

Despite their efficacy to predict time series, machine and deep learning methods are developed mainly through empirical testing. The majority of the literature (e.g., [6], [7], [8]) that focuses on deep learning frameworks solely relies either

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

on raw data or a limited number of features. So far, very little attention has been paid to the information a neural network should analyze for the mid-price prediction task. In this paper, we shed light on the information that the ML trader should consider utilizing in mid-price movement prediction. To this end, we employ an extensive list of econometric features[1] for mid-price prediction and make a head-to-head comparison with indicators derived from: i) technical and quantitative analysis (i.e., [11]), ii) time-sensitive and time-insensitive features (i.e., [12] and [13]), and iii) features extracted through a fully automated process. This fully automated feature extraction process is conducted by a long short-term memory (LSTM) autoencoder (AE).

We choose econometrics as motivation for our handcrafted features since it is the field of financial engineering that captures the empirical evidence of microstructure noise and causality of the data. Our data comes with variations in prices, known in the financial literature as volatility – a measure that we incorporate into our handcrafted features. Despite the general perception in academic literature that volatility itself is not a factor that affects stock returns, ample evidence exists to support the opposite. For instance, in [14] the author finds that volatility together with other proxies that are not directly observable in the data, like liquidity premium, affect stock returns. In the same direction, Lettau and Ludvigson [15] provide evidence that consumption-to-wealth ratio offers information for excess stock market returns, with volatility explaining a significant portion of these returns. Another example is the work by Chung and Chuwonganant [16], where authors find strong evidence that market volatility affects individual stock returns. Under this light, we believe that these are reliable indicators in considering econometrics as features for the task of mid-price movement prediction.

We perform our analysis based on deep learning models which have recently been proposed for financial time series analysis. These models vary from multi-layer perceptrons (MLP) to convolutional neural networks (CNN) and recurrent neural networks (RNN) like LSTM. For our experiments, we use two TotalView-ITCH datasets from the US and the Nordic stock markets. We formulate these experiments based on two protocols: the first one (i.e., ''Protocol I'' in our experiments) is introduced here for the first time, and is based on online learning. The prediction of the mid-price movement takes place every next event and is treated as a multi-objective optimization problem, since it predicts when and in which direction the mid-price movement will happen. The second protocol (i.e., ''Protocol II'' in our experiments) is an existing protocol based on the work of Tsantekidis *et al.* [17], ,according to which the mid-price movement prediction is treated as a three-class classification problem (i.e., up, down or stationary mid-price states) for every next 10^{th} event.

[1]Econometrics features were used in the past for tasks such as identification of big changes in exchange rate volatility (i.e., [9]), or bankruptcy prediction in [10].

The main contribution of our work lies on three pillars. The first pillar refers to the utilization of an extensive econometric features list as input to deep learning models for mid-price movement prediction. The second pillar is related to an extensive evaluation of the newly introduced features with two other handcrafted feature sets and a feature set based on a fully automated process. We conduct a fair evaluation of these feature sets by using the same nine deep learning models for liquid and illiquid stocks, as well as unbalanced and balanced feature sets. Next, we test them not only on the newly introduced experimental protocol but also on a protocol suggested in the literature for the Nordic dataset (also utilized here). Our findings indicate that handcrafted features, which overperformed the fully automated feature extraction process (i.e., based on LSTM AE), transform the forecasting universe of high-frequency trading. More specifically, the present evaluation facilitates traders' task of selecting suitable features according to data, stock, and model availability. The third pillar, finally, refers to the development of a new experimental protocol that takes into consideration every trading event and is unaffected by time irregularities in high-frequency data. Our work suggests that feature extraction should be customized according to stock and model selection; similar findings can be in seen in [18]. The present research opens avenues for several other applications. For instance, the same sets of features can be tested for time series such as exchange rates or bitcoin price predictions. Furthermore, the newly introduced protocol can be the basis of every time series problem since it is event-driven and unaffected by time irregularities. Ultimately, there is no need for any type of data sampling, even for high-frequencies time resolution environments where datasets are massive.

The remainder of the paper is organized as follows. We provide a comprehensive literature review in Section II. The problem statement is provided in Section III. The list of handcrafted features follows in Section IV. In Section V, we describe the various deep learning models adopted in our analysis, while in Section VI we describe details of the datasets and the experimental protocol. In Section VII we provide the empirical results and Section VIII concludes the paper. A detailed description of the econometric features used in our experiments are provided in Appendix together with results for Protocol II.

II. LITERATURE REVIEW

High-frequency LOB data analysis has captured the interest of the machine learning community. The complex and chaotic behavior of the data inflow gave space to the use of non-linear methods like the ones that we see in the machine and deep learning. For instance, Zhang *et al.* [19] utilize neural networks for the prediction of Baltic Dry index and provide a head-to-head comparison with econometric models. The author in [20] develops a new type of deep neural network that captures the local behavior of a LOB for spatial distribution modeling. Dixon applies RNN [21] on S&P500 E-mini futures data for a metric prediction like price change

forecasting. Minh *et al.* [22] also propose RNN architecture for short-term stock predictions by utilizing successfully financial news and sentiment dictionary. In [23], authors apply a combined neural network model based on CNN and RNN for mid-price prediction.

Metrics prediction, like mid-price, can be facilitated by the use of handcrafted features. Handcrafted features reveal hidden information as they are capable of translating time-series signals to meaningful trading instructions for the ML trader. Several authors worked towards this direction, like [12], [24], [13], [25], [26], [27] and [20]. These works present a limited set of features which varies from raw LOB data to change of price densities and imbalance volume metrics. Another work that provides a wider range of features is presented by Ntakaris *et al.* [11]. The authors there extract handcrafted features based on the majority of the technical indicators and develop a new quantitative feature based on logistic regression, which outperformed the suggested feature list.

Handcrafted features represent only one part of the experimental protocol in the quest for mid-price movement prediction. Classification, via deep learning methods, is the continuation of a machine learning protocol. Many authors have used deep learning in financial literature for several problems. For example, Alberg and Lipton [28] use MLPs and RNNs for companies' future fundamentals forecasting. Qian [1] utilizes machine and deep learning methods, like support vector machines (SVM), MLPs, denoising auto-encoder (DAE), and an assembled DAE-SVM model in order to predict future trends of stock's index prices. These machine and deep learning models outperformed traditional time series models like ARIMA and generalized autoregressive conditional heteroskedasticity (GARCH). Sezer *et al.* [29] use MLPs and the three most commonly used technical indicators as inputs for stock price movement predictions.

Many authors utilize LOB data as input to their models. For instance, Nousi *et al.* [4] examine the performance of several machine learning methods, like autoencoders (AE), bag-of-features algorithm, single hidden layer feedforward neural networks (SLFN), and MLPs for mid-price prediction. Han *et al.* [30] apply decision trees on LOB data and outperform support vector machines (SVM) for the problem of mid-price prediction. In the same direction, authors in [31] apply similar methods on market order book data for market movement predictions. Doering *et al.* [32] utilize event flow and limit order datasets for price-trend and price-volatility predictions based on a deep learning architecture. Mäkinen *et al.* [33] predict price jumps with the use of LSTM, where the input data is based on LOB data. A similar work, in terms of the neural model, is conducted in [34] in order to forecast LOB's mid-price.

To the best of our knowledge, this is the first time that an extensive list of econometric features based on high-frequency LOB data is proposed as input to several neural networks for mid-price prediction. We conduct a head-to-head comparison with state-of-the-art handcrafted features is conducted together with features based on a fully automated process; Finally, we report results extracted from two high-frequency datasets with two US and five Nordic stocks for both balanced and unbalanced sets.

TABLE 1. Message list example.

Timestamp	Id	Price	Quantity	Event	Side
1275386347944	6505727	126200	400	Cancellation	Ask
1275386347981	6505741	126500	300	Submission	Ask
1275386347981	6505741	126500	300	Cancellation	Ask
1275386348070	6511439	126100	17	Execution	Bid
1275386348070	6511439	126100	17	Submission	Bid
1275386348101	6511469	126600	300	Cancellation	Ask

TABLE 2. Order book example.

			Level 1				...
			Ask		Bid		
Timestamp	Mid-price	Spread	Price	Quantity	Price	Quantity	
1275386347944	126200	200	126300	300	126100	17	...
1275386347981	126200	200	126300	300	126100	17	...
1275386347981	126200	200	126300	300	126100	17	...
1275386348070	126050	100	126100	291	126000	2800	...
1275386348070	126050	100	126100	291	126000	2800	...
1275386348101	126050	100	126100	291	126000	2800	...

III. PROBLEM STATEMENT

The problem under consideration is the mid-price movement prediction based on high-frequency LOB data. More specifically, we use message and limit order books as input for the suggested features. Message book (MB), as seen in Table 1, contains the flow of information which takes place at every event occurrence. The information displayed by every incoming event includes the timestamp of the order, execution or cancellation, the id of the trade, the price, the volume, the type of the event (i.e., order, execution or cancellation), and the side of the event (i.e., ask or bid).

LOB (Table 2) works under specific rules based on the operation of the trading system-exchange. The main advantage of an order book is that it accepts orders under limits (i.e., limit orders) and market orders. In the former case, the trader/broker is willing to sell or buy a financial instrument under a specific price. In the latter case, the action of buying or selling a stock at the current price takes place. LOBs accept orders by the liquidity providers who submit limit orders and the liquidity takers who submit market orders. These limit orders, which represent the unexecuted trading activity until a market order arrives or cancellation takes place, construct the LOB that is divided into levels. The best level consists of the highest bid and the lowest ask price orders, and their average price defines the so-called mid-price, whose movement we try to predict.

We treat the mid-price movement prediction as a multi-objective optimization problem with two outputs – one is related to classification and the other one to regression. The first part of our objective is to classify whether the mid-price will go up or down and the second part – the regression part is to predict in how many events in the future this movement will happen. To further explain this, let us consider the following example: in order to extract the intraday labels, we measure

starting from time t_k, in how many events the mid-price will change and in which direction (i.e., up or down). For instance, the mid-price will change in 10 events from now, and will go up. This means that our label at time k is going to be $\{1,10\}$, where 1 is the direction of mid-price and 10 is the number of events that need to pass in order to see that movement taking place.

We depart from this labeling system to answer the critical question of whether handcrafted features derived from econometrics can boost deep learning classification and regression performance. We conduct extensive experiments based on nine neural topologies (i.e., five MLPs, two CNNs, and two LSTMs) and two TotalView-ITCH datasets, and compare the performance of econometric features to three other feature sets. The first set is based on time-sensitive and time-insensitive features as presented in [12] and [13], the second feature set is based on technical and quantitative analysis, introduced in [11], and the third one is based on feature representations extracted automatically for the train of an LSTM AE with a description provided in Section V-D .

IV. HANDCRAFTED FEATURE POOL

In this section we provide the nominal list (see Table 3) of the extensive econometric feature list together with the two other state-of-the-art handcrafted feature sets from the literature that are based on technical and quantitative analysis and time-insensitive and time-sensitive indicators. Description of the econometric features is seen in Appendix while the description of technical and quantitative feature set and time-sensitive and time-insensitive set extracted from the LOB can be found in [11].

We extract our econometric features from both MB and LOB and divide them into four main categories: Statistical features, volatility measures, noise measures, and price discovery features. The first category encompasses basic statistical features that are widely used in the literature (e.g., [12], [20]). The logic behind the choice of the volatility measure features is the intimate relation between the volatility of the price process and the price movement itself. As such, we regard the volatility measures included in the present article to retain information useful to real-time price prediction. This is particularly true when the predicted objective is the next price movement. Additionally, the econometric literature widely evidences the significant detrimental impact of the so-called microstructure noise in the measurement of fundamental quantities when working at the highest frequencies. Furthermore, the noise process directly affects the underlying price process itself and as such contributes to the observed price movements. For these reasons we implement a number of estimates of the characteristics of the noise process, which we identify as the noise measures features set.[2] The last group of features includes all those features related to the price discovery process; i.e., those that take into account the interaction of the two sides of the LOB. Several articles in the literature (e.g., [11], [33]) have focused and demonstrated the importance of accounting for the differences between the ask and bid side in order to improve the mid-price forecasting accuracy.

Each of the features in Table 3 operates under a different time duration. Time duration of the features plays an important role in capturing information about underline behavior of time series. More specifically, the feature extraction process consists of low frequency (e.g., technical indicators based on interpolation) and high-frequency features (e.g., adaptive logistic regression), which complement each other. Low frequency features identify long-term trends and structural data components, while high-frequency features capture discontinuities and rapid metric changes. This combination of features facilitates improves neural network perfromance (e.g., [12] and [38]).

V. DEEP LEARNING

The goal of this paper is to forecast the movement of the mid-price. The predicted output has dual information: the direction of the mid-price movement and the prediction of the number of events taking the mid-price to move up or down. An efficient way to do that is by using deep learning architectures. We consider three different neural networks types (i.e., MLPs, CNNs, and LSTMs) and run them seperately. We, then, examine their validity with respect to our optimization problem.

A. MLP FOR CLASSIFICATION AND REGRESSION

MLP (i.e., [39]) is a type of neural network that shows a high degree of connectivity among its components/neurons (see Fig. 1). The strength of this connectivity is determined by the synaptic weights of the neural network. These synaptic weights are determined by a differentiable nonlinear activation function. These basic characteristics of the neural network complicate the analysis of MLPs' behavior. As a result, several MLP architectures have to be examined in order to see whether input data (i.e., handcrafted features) affect the outcome/prediction. The way that an MLP can be trained is based on a sequential data feeding process called batch learning. Batch learning is a process according to which the neural network adjusts the synaptic weights after the presentation of all the samples $J = \{x(i), d(i)\}_{i=1}^{N}$ in the training process, where $x(i)$ is the input multi-dimensional vector and $d(i)$ the response vector of the supervised problem at instance i, and the error function at instance i is:

$$e^{(i)} = d^{(i)} - y^{(i)} \tag{1}$$

where $d^{(i)}$ is the i^{th} element of the $d(i)$ and $y^{(i)}$ is the produced output term at instance i. The error function that we use for our experiments is bespoke to our supervised problem and its components are based on the binary cross entropy (for the classification task) and the mean squared error (for the

[2]Most of the presented measures have been developed and are consistent estimators under broad assumptions on the underlying price process and contaminating noise process; we will not discuss these assumptions into details here as outside the scope of the article. Interested readers are referred to [37] and references within for an exhaustive review of the literature

TABLE 3. Feature list for the three feature sets: Description for the newly introduced, based on Econometrics, handcrafted features can be found in Appendix, where description for the Tech & Quant and LOB feature sets can be found in [11].

Econometric features	Tech & Quant features	LOB features
Statistical Features	**Technical Indicators**	**Basic**
Mid-Price	Accumulation Distribution Line	n LOB Levels
Financial Duration	Awesome Oscillator	
Average Mid-Price Financial Duration	Accelerator Oscillator	**Time-Insensitive**
Log-Returns	Average Directional Index	Spread & Mid-Price
	Average Directional Movement Index Rating	Price Differences
Volatility Measures	Displaced Moving Average	Price& Volume Means
Realized Volatility	Absolute Price Oscillator	Accumulated Differences
Realized Kernel	Aroon Indicator	
Realized Pre-Averaged Variance	Aroon Oscillator	**Time-Sensitive**
Realized Semi-Variance	Average True Range	Price & Volume Derivation
Realized Bipower Variation	Bollinger Bands	Average Intensity per Type
Realized Bipower Variation (lag 2)	Ichimoku Clouds	Relative Intensity Comparison
Realized Bipower Semi-Variance	Chande Momentum Oscillator	Limit Activity Acceleration
Jump Variation	Chaikin Oscillator	
Spot Volatility	Chandelier Exit	
Average Spot Volatility	Center of Gravity Oscillator	
	Donchian Channels	
Noise and Uncertainty Measures	Double Exponential Moving Average	
Realized Quarticity	Detrended Price Oscillator	
Realized Quarticity Tripower	Heikin-Ashi	
Realized Quarticity Quadpower	Highest High and Lowest Low	
Noise Variance [35]	Hull MA	
Noise Variance [36]	Internal Bar Strength	
	Keltner Channels	
Price Discovery Features	Moving Average Convergence/Divergence Oscillator	
Weighted Mid-Price by Order Imbalance	Median Price	
Volume Imbalance	Momentum	
Bid-Ask Spread	Variable Moving Average	
Normalized Bid-Ask Spread	Normalized Average True Range	
	Percentage Price Oscillator	
	Rate of Change	
	Relative Strength Index	
	Parabolic Stop and Reverse	
	Standard Deviation	
	Stochastic Relative Strength Index	
	T3-Triple Exponential Moving Average	
	Triple Exponential Moving Average	
	Triangular Moving Average	
	True Strength Index	
	Ultimate Oscillator	
	Weighted Close	
	Williams %R	
	Zero-Lag Exponential Moving Average	
	Fractals	
	Linear Regression Line	
	Digital Filtering: Rational Transfer Function	
	Digital Filtering: Savitzky-Golay Filter	
	Digital Filtering: Zero-Phase Filter	
	Remove Offset and Detrend	
	Beta-like Calculation	
	Quantitative Indicators	
	Autocorrelation	
	Partial Correlation	
	Cointegration based on Engle-Granger test	
	Order Book Imbalance	
	Logistic Regression for Online Learning	

regression task), as follows:

$$\mathcal{L}_{all} = \underset{\mathcal{L}_1,\mathcal{L}_2}{arg\,min}\,\{\lambda\mathcal{L}_1 + (1-\lambda)\mathcal{L}_2\} \tag{2}$$

where $\mathcal{L}_1 = -t\ log\ \hat{y}^{(i)} - (1-t)\ log(1-\hat{y}^{(i)}), t \in \{0, 1\}$ and $\mathcal{L}_2 = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$ with a free parameter λ, where $y^{(i)}$ and $\hat{y}^{(i)}$ be the ground truth and the predicted values of the i^{th} training sample which belongs to $\mathbb{R}^N$, respectively. This customized function is part of the backpropagation algorithm that helps the neural network (e.g. MLP) to correct the synaptic weights in order to optimize Eq. 2. Backpropagation in our case follows the automatic differentiation (AD) reverse mode (i.e., [40]). Reverse AD facilitates the process of correcting the synaptic weights and it can be done as follows: Initially we define the input variables as $v_{i-n} = x_i, i = 1, ..., n$, all the intermediate variables of the neural network as $v_i, i = 1, ..., k$ and $y_{m-i} = v_{k-i}, i = m-1, ..., 0$ be the output variables. Derivatives calculation is a two-step process. During the first phase the intermediate variables v_i are populated and create the graph trace, whereas during the second phase derivatives are calculated based on the propagation of the adjoints $\bar{v}_i = \frac{\partial y_l}{\partial v_i}$. In general, the reverse AD performs the calculations from the output to the input starting from the output as seed:

$$\frac{\partial f}{\partial y_{m-i}} \leftarrow 1$$

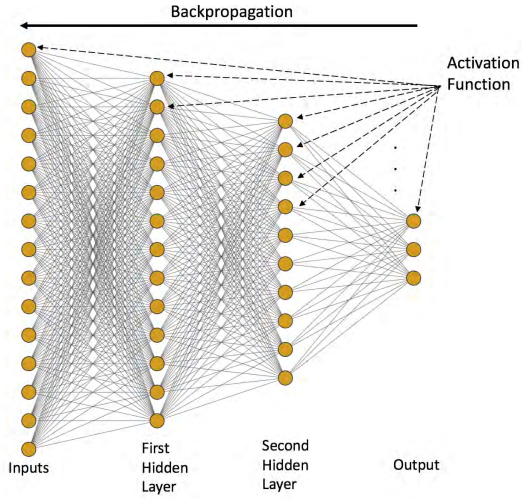


FIGURE 1. Example of an MLP neural network with two hidden layers and 4 units output.

and moves to the inputs via the intermediate states based on the calculation:

$$x_i \leftarrow \sum_{j:i\in Pa(j)} \frac{\partial f}{\partial x_j}\frac{\partial g_j}{\partial x_i}$$

where $Pa(j)$ denotes the parent formation of node j and g_j the intermediate functions of the graph. The next part of the MLP training is the learning process, which is defined as the method through which the loss function will reach the optimal solution via proper parameter updates. For this reason we choose the Nesterov accelerated gradient (NAG) method incorporated into the adaptive moment estimation (Adam) method named as Nadam by Dozat [41]. Nadam applies the momentum step only once and takes into consideration the current momentum – rather than the previous momentum – vectors. This gives us the Nadam update parameters rule:

$$\theta_{t+1} := \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}(\beta_1\hat{m}_t + \frac{(1-\beta_1)\nabla_{\theta_t}\mathcal{L}_{all}(\theta_t)}{1-\beta_1^t}) \quad (3)$$

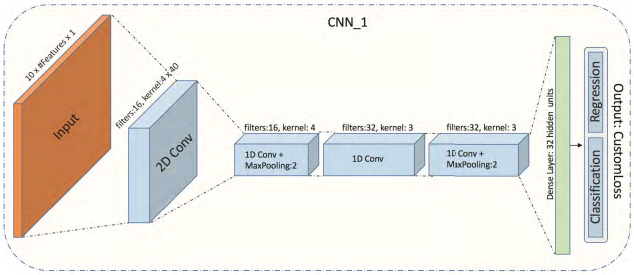
where the first (i.e., mean) and second (i.e., variance) moment for the current momentum vector are, respectively:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{u}_t = \frac{v_t}{1-\beta_2^t} \quad (4)$$

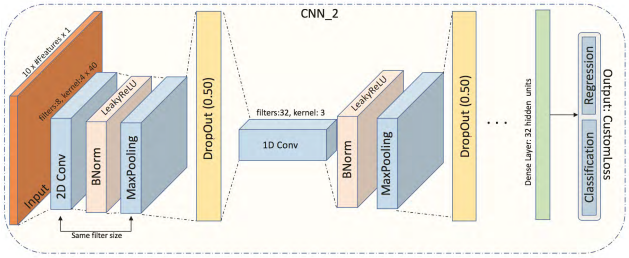
with $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_{\theta_t}\mathcal{L}_{all}(\theta_t)$, $v_t = \beta_2 v_{t-1} + (1-\beta_1)\nabla^2_{\theta_t}\mathcal{L}_{all}(\theta_t)$ and learning rate $\eta = 0.002$.

B. CNN FOR CLASSIFICATION AND REGRESSION

CNN, as described in [42], is a type of neural network that handles time series of multidimensional data for metric prediction. The main motivation for choosing this type of neural network is its capability for sparse connectivity between neural layers, for sharing the so-called tied weights and equivariant representation properties. More specifically, sparse connectivity can be achieved by using a kernel smaller than the sample input. This action reduces the amount of memory that is required for the training process. The second



(a) CNN based on the work presented in [43].



(b) CNN with deeper topology that will be used later in the experimental protocols.

FIGURE 2. Two CNN examples that demonstrate their operation mechanisms. These two CNNs (i.e., CNN_1 and CNN_2) will later on utilized in the experiments.

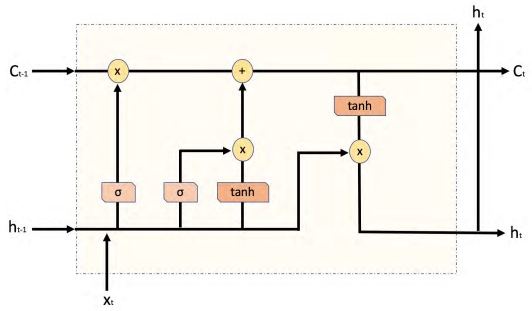


FIGURE 3. Visual representation of LSTM's internal cell calculations.

advantage of a CNN is the use of tied weights. Tied weights are shared among the inputs since the same amount of weights is applied to the inputs.

CNN has three main parts: the convolution layer, the pooling layer, and the fully connected layer. The convolution layer extracts features from the input multi-dimensional signal expressed usually as a tensor or matrix. This process creates linear activations that run via a non-linear activation function such as the rectified linear activation function (ReLU) and the Leaky ReLU. Then the pooling layer will convert the local output based on a summary statistic related to the local outputs (e.g. max-pooling). The last step of the process is the connection to the fully connected layers (see examples in Fig. 2) that will perform the classification and regression tasks. These tasks are based on discrete time series events that formulate the (forward) convolution layer calculation as follows:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^{H}\sum_{j=0}^{W}\sum_{d=0}^{D} f_{i,j,d} \times x^{l}_{i^{l+1}+i,j^{l+1}+j,d} \quad (5)$$

TABLE 4. List of the nine deep learning models that are used for the two experimental protocols. Output, in the neural networks above, means that for Protocol I the output is a dense layer with 1 unit and linear activation function for the regression task and a dense layer with two units and softmax activation function. For Protocol II, the output is a dense layer with three units and softmax activation function.

Model	Topology
MLP_1	• Dense layer with 4 units with Tanh activation • Output
MLP_2	• Dense layer with 512 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • Output
MLP_3	• Dense layer with 256 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • Output
MLP_4	• Dense layer with 256 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • Output
MLP_5	• Dense layer with 128 units and Tanh activation • 20% Dropout • Dense layer with 128 units and Tanh activation • 20% Dropout • Dense layer with 128 units and Tanh activation • Output
CNN_1	• 2D Convolution layer with 16 filters, 4 x 40 kernel size • 1D Convolution layer with 16 filters, 4 as kernel size • Maxpooling size of 2 • 1D Convolution layer with 32 filters, 3 as kernel size • 1D Convolution layer with 32 filters, 3 as kernel size • Maxpooling size of 2 • Dense layer with 32 neurons • Output
CNN_2	• 2D Convolution layer with 8 filters, 4 x 40 kernel size, 2 x 2 stride size and same output size • BatchNormalization • LeakyReLU with 0.1 slope • 2 x 2 MaxPooling with the same output size • 50% Dropout • 1D Convolution layer with 32 filters, 3 as kernel size, 5 as stride and same output size • BatchNormalization • LeakyReLU with 0.1 slope • Maxpooling size of 2 with the same output size • 50% Dropout • 1D Convolution layer with 32 filters, 3 as kernel size, 5 as stride and same output size • LeakyReLU with 0.1 slope • Maxpooling size of 2 with the same output size • 50% Dropout • Dense layer with 8 units • Output
LSTM_1	• LSTM layer with 32 units • Dropout • PReLU • Dense layer with 64 units • Output
LSTM_2	• LSTM layer with 40 units • PReLU • Attention layer • Dense layer with 40 units • Output

where H, D,and D are the row, columns and depth dimension of the input tensor $x \in \mathbb{R}^{H^l \times W^l \times D^l}$ respectively, $f \in \mathbb{R}^{H^l \times W^l \times D^l}$ is the filter bank, and the indexing $(i^{l+1}+i, j^{l+1}+j, d)$ refers to the iterative local convolution of the filter bank on the suggested input for the l-layer. Pooling is performed right after convolution; to conduct our experiments, we choose the formation of max pooling. The final step is the use of fully connected layers. The structure of these fully connected layers is the same as in Sec. V-A. The process that we follow in order to train our CNN parameters (i.e., filter banks and synaptic/tied weights) is based on batch learning combined with reverse AD (i.e., backpropagation) as we did for the MLP case.

C. LSTM FOR CLASSIFICATION AND REGRESSION

The ML trader has to consider the temporal behavior of time series. The events that we have to deal with in the LOB universe are likewise formed in a sequential manner. Sequential systems, like RNNs, are based on computational graphs and are, thus, ideal for time series analysis. RNNs provide much flexibility in terms of architecture formation, which is described in Eq. 6:

$$h_t = f(h_{t-1}, x_t; \theta) \tag{6}$$

where h and x are the state and the input at time t and θ are the shared parameters for a transition function f at time t. Since we use RNN for empirical calculations we

choose to forecast mid-price by using gated RNNs (named LSTM) as presented in [44]. Motivation for choosing this type of gated RNN is its ability to create connections through time and account for the problem of vanishing (or exploding) gradients. Instead of applying just element-wise nonlinear input transformations, LSTM units (see LSTM's internal cell calculations in Fig. 3), contain processes which that take into consideration the sequential nature of time series. More specifically, an LSTM cell is equipped with gates that filter the information flow by applying weights internally. The first pass is the forget gate vector f_i^t:

$$f_i^{(t)} = \sigma\Big(\sum_j W_{i,j}^f h_j^{(t-1)} + \sum_j U_{i,j}^f x_j^{(t)} + b_i^f\Big) \tag{7}$$

where $x_i^{(t)}$ and $h_i^{(t)}$ are the current input and hidden state vectors of cell i at time t, respectively. The attached weight matrices to these vectors are W^f and U^f for the forget gate vector with b^f the bias term. The next pass is related to the information to be saved to the so-called "cell state". The cell state can be divided in two parts - the input vector and a *tanh* layer as follows:

$$C_i^{(t)} = f_i^{(t)} C_i^{(t-1)} + g_i^{(t)} \sigma\Big(\sum_j W_{i,j}^C h_j^{(t-1)} + \sum_j U_{i,j}^C x_j^{(t)} + b_i^C\Big) \tag{8}$$

where $g^{(t)}$ is the input gate:

$$g_i^{(t)} = \sigma\Big(\sum_j W_{i,j}^g h_j^{(t-1)} + \sum_j U_{i,j}^g x_j^{(t)} + b_i^g\Big) \tag{9}$$

The last remaining part is the filtered output. More specifically, the LSTM output/hidden state will be formulated by the output gate vector $o_i^{(t)}$ which is calculated as follows:

$$o_i^{(t)} = \sigma\Big(\sum_j W_{i,j}^o h_j^{(t-1)} + \sum_j U_{i,j}^o x_j^{(t)} + b_i^o\Big) \tag{10}$$

and the final output $h_i^{(t)}$ is equal to:

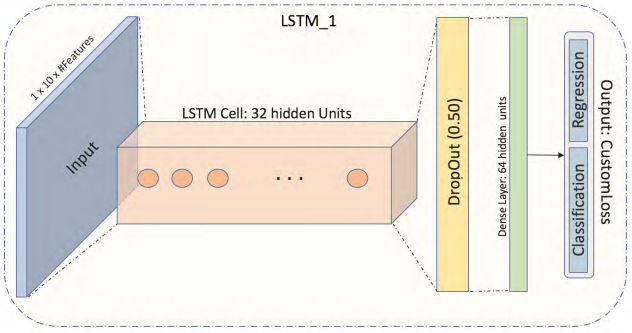
$$h_i^{(t)} = o_i^{(t)} * tanh(C_i^{(t)}). \tag{11}$$

The formation above refers to the case of a typical LSTM neural network, which we implement in Section VI. We also apply an attention mechanism to the LSTM architecture in order to weight/measure the significance of the input sequence. We follow the implementation in [45] and [33] where the sequential LSTM outputs (i.e., hidden states $H^{(t)}$, $t \in \{1, ..., T\}$) are filtered via the following steps for every K-dimensional vector w:

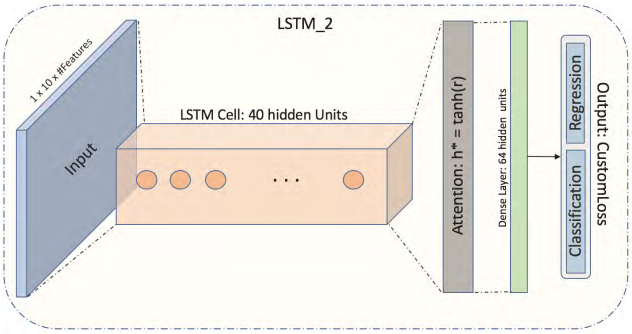
$$M = tanh(H^{(t)}) \tag{12}$$

$$\alpha = \frac{e^{w_i^T * M}}{\sum_{k=1}^{K} e^{w_k^T * M}} \tag{13}$$

$$r = H^{(t)} * \alpha \tag{14}$$



(a) LSTM based on the work in [17]



(b) LSTM with attention layer

FIGURE 4. Two LSTM examples with one main LSTM block (orange colored box) with several hidden cell units (orange cycles). These two LSTMs (i.e., LSTM_1 and LSTM_2) will later on utilized in the experiments.

and the final LSTM with attention output is:

$$h^* = tanh(r). \tag{15}$$

Here we use the same backpropagation mechanism as we did for MLPs. Examples of LSTM neural networks can be seen in Fig. 4

D. FULLY AUTOMATED FEATURE EXTRACTION BASED ON AUTOENCODERS

Autoencoders (AE) (i.e., [46], [47]) are neural networks which operate on a self-feedback loop fashion. They do not require any labeling system since they depend on this semi-supervised protocol. This type of neural network is divided in three main parts; the encoder, the latent representation, and the decoder (i.e. encoder and decoder). An example of AE can be seen in Fig. 5.

The basic structure of AE is defined as a mapping from encoder to decoder, the main objective being the following minimization problem:

$$f, g = \underset{f,g}{arg\,min}\ ||X - (f \circ g)X||^2 \tag{16}$$

where $f : X \to F$ and $g : F \to X$ with X be the input raw LOB data in the present work.

The fully automated feature extraction process is based on the latent representation. This latent representation in

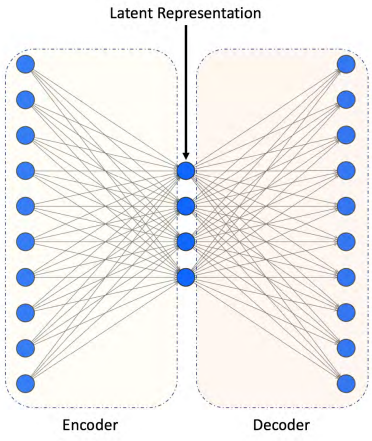


FIGURE 5. AE Example.

the present work plays the role of the vector representation, which will, later on act as input to each of the suggested nine deep neural networks. In order to use this latent space as feature set we train an LSTM AE.[3] LSTM AE has exactly the same structure as a simple AE with the difference that the filtering is based on LSTM layers for the encoding and decoding part. We choose LSTM AE since they take into consideration the temporal behavior of our time series.

VI. DATA DESCRIPTION AND EXPERIMENTAL PROTOCOLS

Our objective is to provide informative handcrafted features to ML traders and market makers for the task of mid-price movement prediction. Prediction of this movement requires in-depth analysis in terms of data selection (e.g., liquid or illiquid stocks) and experimental protocol development. For these reasons, our analysis consists of two TotalView-ITCH based on two US and five Nordic stocks and two experimental protocols. The first protocol, named Protocol I, is based on online prediction for every 10-block rolling events, and we introduce it here for the first time. The second protocol, named Protocol II, is derived from the literature (i.e., [17]) and is based on mid-price movement prediction with 10-event lag. Both protocols are event driven, which means that there are no-missing values. However, Protocol II is based on independent 10-block events, which creates a lag of 10 events. Some of the suggested features can partially overcome this problem by finding averages or other types of transformations inside these blocks, but, still some information will be parsed. A possible solution to this problem comes from Protocol I where every single trading event is taken into consideration and, as a result, there are no missing values. We should also mention that LOB data is exposed to bid-ask[4] bounce effect which may inject bias. We leave this topic for future research, where we plan to increase the rolling event block size in Protocol I since a wider block will, potentially, improve stability.

A. DATA

We utilize two TotalView-ITCH datasets based on two US and five Nordic stocks. The time resolution of the datasets is in milliseconds. For the US datasets, we use two stocks, Amazon and Google, whereas for the Nordic dataset we use Kesko Oyj, Outokumpu Oyj, Sampo Oyj, Rautaruukki, and Wartsila Oyj. We use ten business days for both datasets covering the periods: from 22.09.15 to 05.10.15 for the US dataset and from 01.06.10 to 14.06.10 for the Nordic dataset, respectively. The trading activity for these ten business days is 13,000,000 events for the US dataset and 4,000,000 events for the Nordic dataset. We use MBs in order to create relevant LOBs. We utilize super clustering computational power based on HP Apollo 6000 XL230a/SL230s supercluster to convert MBs to LOBs (i.e., LOBs are of depth 10 for both sides). We follow several pre-processing steps before we start training the deep learning models. A general description of the pre-processing process can be seen in Fig. 6

B. PROTOCOL I

Both TotalView-ITCH datasets convey asynchronous information varying from events taking place at the same millisecond to events several minutes apart from each other. In order to address this issue, we develop Protocol I, which utilizes all the given events in an online manner. More specifically, our protocol extracts feature representation every ten events with an overlap of nine events for every next feature representation. We decided to use a 10-window block for our experiments due to the frequency [5] of the stationarity present in both datasets. In order to identify whether our time series have unit roots, we perform an Engle-Granger cointegration test,[6] with focus on the augmented Dickey-Fuller test, on the pair $Ask - Bid$ prices from LOBs level I. The hypothesis test shows that there is a constant alternation between its states (i.e. 1 for non-stationarity and 0 for stationarity of the suggested time series), which occurs several times during the day. This is indicative for both datasets as seen in Figure 7. These stationarity breaks supports the initial idea, as this presented by many authors (e.g., [48], [49], [50]), that neural networks are capable of identifying underlying processes of a non-stationary time series. Neural networks are nonlinear and non-parametric adaptive-learning filters which operate with fewer assumptions compare to more traditional time series models like ARIMA and GARCH.

A visual description of our protocol can be seen in plot (a) in Fig. 8. The problem under consideration in Protocol I is to predict the movement of mid-price (i.e., classification: up or down) together with the number of events it takes for that movement to occur in the future (i.e., regression:

[3]Details of the training are provided in Section VII.

[4]Bid-ask bounce is the rapid stock's price bounce between bid and ask side.

[5]The average rate of change of the non-stationarity condition, for both TotalView-ITCH datasets, is changing in average every ten events.

[6]Test implementation can be found in [11].

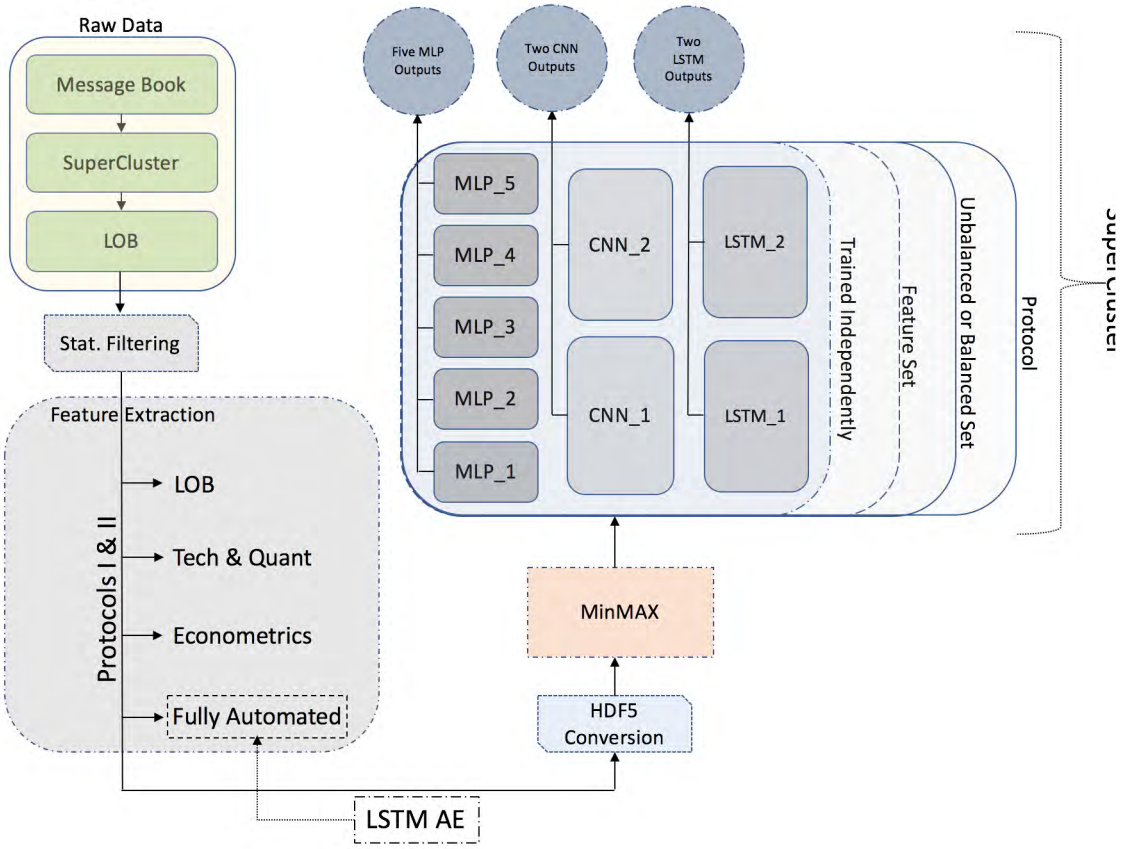


FIGURE 6. This is a higher-level explanation of the steps that we follow for the present analysis. From left top to right bottom: The first step is to obtain the datasets for the US and Nordic stocks and send raw data (i.e., message books) to CSC superclusters and obtain the LOBs. The next step is to apply statistical filtering (description can be found in Section VI-D). What follows is the process of feature extraction for the four different feature sets for Protocols I & II. An HDF5 conversion takes place right afterwards, and a MinMax normalization follows for every feature set case for both protocols. Next, each of the nine neural networks is trained independently for the four different feature lists based on unbalanced and balanced sets. The training process is based on python scripts, which are sent to CSC superclaster in order to obtain results for Protocol I & II.

number of events until next mid-price's movement change). More specifically, in order to testing performance evaluation, we utilize f1 score for the classification task and RMSE (i.e., Root Mean Square Error) for the regression task. F1 score is defined as:

$$f1 = \frac{2 \times Recall \times Precision}{Recall + Precision}, \tag{17}$$

with

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

and

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

where TP, FN, and FP are the True-Positives, False-Negatives, and False-Positives, respectively, and RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}, \tag{20}$$

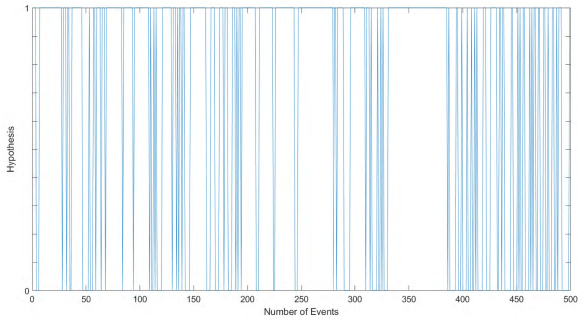
where P_i and O_i are the predicted and observed values of n samples, respectively.

We have a labeling system that requires classification and regression. The first part of the dual labeling format contains the binary information 1 and -1 for the up and down mid-price movement, respectively. The second part of the labeling format represents the discretization of the numeric data expressed as the steps until the next mid-price change. A pictorial example of the above labeling system is in Fig. 9. The label extraction is described as follows:

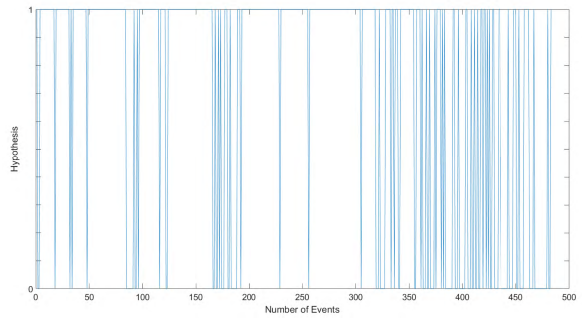
1) $d(i) = 1_{MP_{(i)} - MP_{(i-1)} > 0}$ OR $-1_{MP_{(i)} - MP_{(i-1)} < 0}$, where $i \in \mathbb{R}^{N-1}$, with N be the number of the mid-prices (MP) samples,
2) $L(p) \leq d(i) < L(p+1)$, $1 \leq p < Q$, where $L(p)$ is a vector that contains the bin limits in a monotonically increasing order and Q is the number of bins equal to the total number of the non-zero elements in the vector of mid-price differences.

C. PROTOCOL II

Protocol II is based on independent 10-event blocks for the creation of the feature representations as this can be seen in

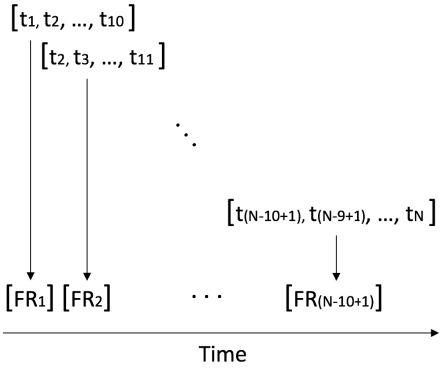


(a) Hypothesis test for stationarity check for the Nordic stock, Kesko Oyj. The plot represents a sample of 500 consecutive events.

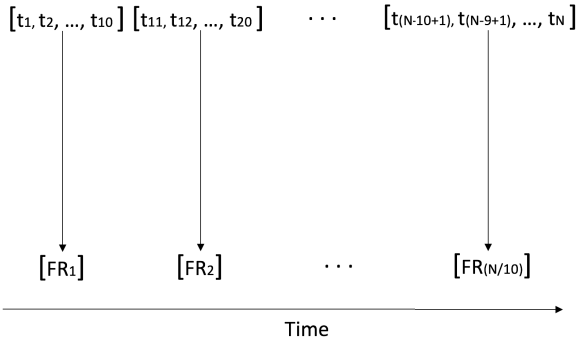


(b) Hypothesis test for stationarity check for the US stock, Amazon. The plot represents a sample of 500 consecutive events.

FIGURE 7. Hypothesis test for stationarity check, where constant transition from state 0 to state 1 is present.



(a) **Protocol I**: Feature extraction in an online manner with zero lag delay



(b) **Protocol II**: Feature extraction with 10 events lag

FIGURE 8. Feature extraction based on the two protocols for the task of mid-price movement prediction. For given time series $(t_1, t_2, ..., t_N)$ there $N - 10 + 1$ feature representations (FR) for Protocol I and $\frac{N}{10}$ FR for Protocol II.

the plot (b) in Fig. 8. More specifically, feature representations are based on the information that can be extracted from 10 events each time with these 10-event blocks independent from each other. Protocol II treats the problem of mid-price movement prediction as a three-class classification problem, with three states: up, down, and stationary condition for the mid-price movement. These changes in the mid-price are defined by means of the following calculations:

$$l_t = \begin{cases} 1, & \text{if } \dfrac{m_a(t)}{MP(t)} > 1 + \alpha \\ -1, & \text{if } \dfrac{m_a(t)}{MP(t)} < 1 - \alpha \\ 0, & \text{otherwise} \end{cases}$$

where $MP(t)$ is the mid-price at time t, $m_a(t) = \frac{1}{r}\sum_{i=1}^{r} MP(t+1)$ is the average of the future mid-price events with window size $r = 10$, and α determines the significance of the mid-price movement which is equal to 2×10^{-5}.

D. DATA NORMALIZATION AND FILTERING

The next step of the pre-processing step is data normalization. We perform a filtering and a normalization method during the feature extraction process and training. The first one is a statistical filtering method , while the second one is based on MinMax. More specifically, we perform the filtering methodology first and apply it directly on the raw MB data. The main idea of the methodology is to identify and eliminate any observation that does not reflect market activity. In the financial econometrics literature this is often referred to as data cleaning and its importance has been widely discussed

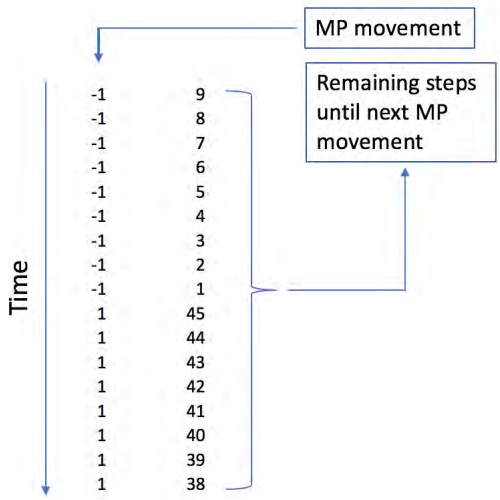


FIGURE 9. Labeling sample for the dual prediction problem of our classification and regression objective. The left part represents the direction (i.e., up or down) of the mid-price (MP) movement while the right part represents the remaining steps until the next change in MP will take place.

in the literature (e.g., [51], [52], and [53]).[7] In more detail, to filter the raw data for outliers we follow a two-step procedure. We initially remove all transactions recorded outside official trading time and clearly misrecorded transactions.[8] We then proceed by implementing a more elaborate filtering algorithm, which takes into account the statistical properties of the series to assess the validity of each observation according to its likelihood of being an outlier.[9] More specifically, for a k size window, we identify a set of (centered) neighboring observations for each data point. To avoid including prices too distant in time, the window size k should be chosen according to the trading intensity of the series. We then compute the trimmed mean of the neighboring set and mark as an outlier the considered observation if it falls more than $\alpha+\gamma$ standard deviations away from the neighbors' mean. Where γ is a granularity parameter, which should be chosen as a multiple of the tick size. The idea behind γ is to create a lower positive bound for the price variation. This is particularly important for the cleaning procedure as it is not uncommon to observe a sequence of equal mid prices in the LOB, which would lead to a zero variance and a consequent rejection of every price different from the mean value. Technically, be X_i the i^{th} element of a time series of observations X, we check:

$$(|X_i - \bar{X}_i(k)| < \alpha * s_i(k) + \gamma) \tag{21}$$

where $s_i(k)$ and $\bar{X}_i(k)$ are respectively the sample standard deviation and the trimmed mean computed over a neighborhood of k observations around X_i. Hence, we identify and remove observation X_i if (21) is true and keep it otherwise. The normalization procedure is based on MinMax for the handcrafted features, as follows:

$$MM = \frac{X_{(i)} - X_{min}}{X_{max} - X_{min}}, i \in \mathbb{R}^N, \tag{22}$$

where N is the total sample size for every feature vector X and $X_{(i)}$ is the i^{th} element of X.

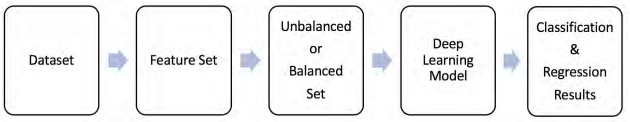
VII. RESULTS & DISCUSSION

In this section, we provide results of the experiments we conducted, based on two massive LOB datasets from the US (i.e., two stocks: Amazon and Google) and Nordic (i.e., five stocks: Kesko Oyj, Outokumpu Oyj, Sampo Oyj, Rautaruukki, Wartsila Oyj) stock markets. We also discuss the perfromance of the handcrafted feature extraction universe for mid-price movement prediction and test its efficacy against a fully automated process. What is more, we make a head-to-head comparison of the three handcrafted feature sets, namely: i) "Limit Order Book (LOB):L", based on the works of [12] and [13], ii) "Tech-Quant:T-Q", based on [11], and iii) "Econ:E", which uses econometric features. Finally,

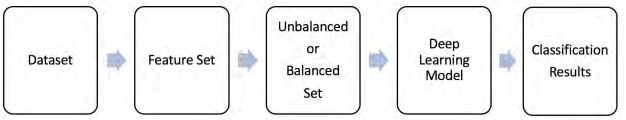
[7]While the advancement of technology has drastically reduced the number of outliers and misrecorded observations, their effect on the statistical analysis is still significant and the implementation of a cleaning procedure is, to this day, required to avoid biased results.

[8]These can be, for example, observations with a price equal to zero.

[9]The methodology follows closely [52]



(a) This is Protocol I, where we test the four sets of features (i.e., Econ, Tech-Quant, LOB, and fully automated), via nine deep learning models (i.e., five MLPs, two CNNs, and two LSTMs) for mid-price prediction. The mid-price prediction in this protocol is a combined prediction of when the next mid-price movement will happen and in which direction. This protocol is based on online learning architecture. We test this protocol for both US and Nordic stocks.



(b) This is Protocol II, where we test the four sets of features (i.e., Econ, Tech-Quant, LOB, and fully automated), via nine deep learning models (i.e., five MLPs, two CNNs, and two LSTMs) for mid-price prediction. The mid-price prediction in this protocol is a three-class problem with states for up, down, and stationary mid-price movement. Protocol I predicts every 10th event from the current mid-price state. We test this protocol for both US and Nordic stocks.

FIGURE 10. Plots (a) and (b) show the process for predicting the mid-price movement based on Protocol I and Protocol II, respectively. In both protocols, the first step is the choice of dataset. The ML trader has to choose the US or Nordic stock(s) (e.g., there is the option of choosing a stock or the 'Joint' case where all the stocks from the US or Nordic markets used for training). The second step is to choose the feature set. The ML trader has to choose one of the four suggested feature sets, which are: The newly introduced econometric set, the one that is based on technical and quantitative indicators, another one based on time-sensitive and time-insensitive LOB features, and the last one based on fully automated features. The third step is whether the prediction should be based on a balanced or unbalanced set. The fourth step is the choice of one of the suggested nine deep learning models. The final step is the one that differs in Protocol I and Protocol II. The difference lies in the fact that Protocol I is a combined classification and regression optimization problem with zero event lag and Protocol II is a three-class classification problem based on a 10-event lag.

we compare these three sets of handcrafted features with features extracted based on an LSTM autoencoder.

Latent representations are extracted after training an LSTM AE. This training employs an extensive grid search, in which the best perfromance is reported. The grid search is based on symmetrical, assymetrical, shallow, deep, overcomplete, and undercomplete LSTM AE. The provided options vary from: i) the encoder with maximum depth up to four hidden LSTM layers with different numbers of filters varying according to the list {128, 64, 18, 9}, ii) the decoder with maximum depth up to four hidden LSTM layers with different numbers of filters varying according to the list {128, 64, 18, 9}, and iii) the latent representation with different options varying according to the list {5, 10, 20, 50, and 130}. The best performance reported is based on a symmetrical and undercomplete LSTM AE of four hidden LSTM layers with 128, 64, 18, and 9 filters respectively, and 10 for the latent representation vector size. The list of the suggested grid search is limited; however, we believe it provides a wide range of combinations in order to make a fair comparison of a fully automated feature extraction process against advanced

TABLE 5. Protocol I: f1 scores for the US stocks. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Amazon	0.32	0.39	0.52	0.33	0.44	0.31
	Google	0.38	0.32	0.48	0.34	0.52	0.32
	Joint	0.32	0.32	0.44	0.33	0.54	0.31
MLP_2	Amazon	0.32	0.49	0.38	0.50	0.36	0.30
	Google	0.36	0.43	0.47	**0.57**	0.34	0.31
	Joint	0.33	0.52	0.38	0.33	0.35	0.32
MLP_3	Amazon	0.33	0.50	0.54	0.48	0.49	0.29
	Google	0.40	0.51	0.52	0.51	0.38	0.30
	Joint	0.32	0.52	0.40	**0.56**	0.51	0.31
MLP_4	Amazon	0.32	0.53	0.38	0.33	0.36	0.28
	Google	0.51	0.32	0.44	0.44	0.32	0.30
	Joint	0.52	0.52	0.37	0.34	0.35	0.28
MLP_5	Amazon	0.30	0.42	0.33	0.33	0.31	0.31
	Google	0.43	0.45	0.34	0.44	0.48	0.31
	Joint	0.52	0.33	0.37	0.34	0.35	0.30
CNN_1	Amazon	0.50	0.49	0.53	0.40	0.43	0.51
	Google	0.49	0.49	0.57	0.48	0.48	0.51
	Joint	0.49	0.48	0.54	0.41	0.53	0.45
CNN_2	Amazon	0.51	0.45	0.55	0.52	0.56	0.51
	Google	0.45	0.51	0.55	0.49	0.50	0.47
	Joint	0.53	0.50	0.57	0.43	0.56	0.46
LSTM_1	Amazon	0.52	0.46	0.51	0.50	0.44	0.44
	Google	0.46	0.49	**0.58**	0.49	0.42	0.51
	Joint	0.52	0.48	0.56	0.49	0.50	0.49
LSTM_2	Amazon	0.45	0.49	0.56	0.54	0.52	0.47
	Google	0.52	0.51	0.54	0.53	0.45	0.51
	Joint	0.40	0.51	**0.59**	0.55	0.55	0.51

TABLE 6. Protocol I: RMSE scores for the US stocks. *Note: Highlighted text shows the best RMSE performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Amazon	28.99	**33.02**	80.64	79.63	86.22	79.61
	Google	131.18	37.53	96.75	98.25	94.59	96.11
	Joint	32.32	38.43	88.88	87.70	87.49	87.72
MLP_2	Amazon	**28.39**	224.91	80.67	381.22	90.65	603.53
	Google	45.57	282.40	101.91	404.47	96.51	550.23
	Joint	33.34	266.32	92.79	628.33	87.86	608.31
MLP_3	Amazon	30.44	227.90	81.37	393.43	793.08	615.51
	Google	61.28	312.42	101.74	498.35	95.90	563.60
	Joint	33.31	255.48	88.30	474.82	87.78	671.49
MLP_4	Amazon	28.61	253.89	79.60	634.33	79.60	634.31
	Google	95.73	296.81	101.54	688.81	96.43	167.81
	Joint	**32.29**	269.86	87.71	671.49	**87.70**	671.67
MLP_5	Amazon	28.40	264.38	79.78	628.99	82.89	629.01
	Google	42.07	271.69	100.01	664.08	96.47	654.19
	Joint	33.22	496.80	87.73	663.93	87.70	663.89
CNN_1	Amazon	31.21	337.89	90.35	592.95	104.90	686.93
	Google	190.19	201.45	203.65	319.34	99.39	421.14
	Joint	34.85	367.70	289.76	278.90	260.46	836.35
CNN_2	Amazon	30.47	342.41	380.90	671.19	144.22	767.75
	Google	189.87	178.98	167.89	302.33	97.63	753.54
	Joint	37.46	362.68	200.67	165.23	352.20	737.46
LSTM_1	Amazon	29.08	277.94	110.86	454.85	84.64	774.62
	Google	137.65	207.48	123.36	599.94	99.13	418.62
	Joint	36.05	240.14	92.58	604.54	96.58	421.04
LSTM_2	Amazon	32.03	235.14	86.81	440.10	82.03	500.47
	Google	38.37	262.28	98.38	398.17	97.45	449.66
	Joint	36.65	309.80	89.69	487.73	86.95	527.83

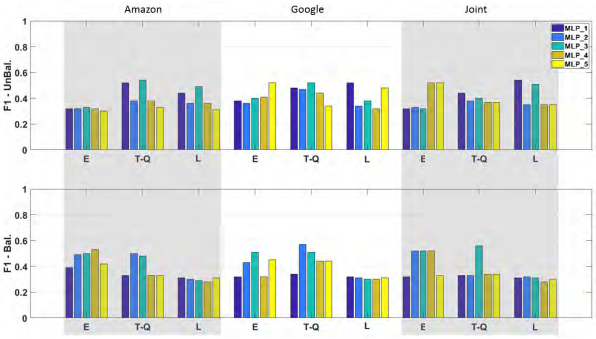
handcrafted features. We should also mention that, despite the extensive grid search on the LSTM AE, we limited our search to up to four hidden units for the encoding and decoding parts with four different filter options. Further analysis on the topic is required.

In order to scrutinize the efficacy of the handcrafted and fully automated features, we use two experimental protocols and nine deep learning models, and present results based on unbalanced and balanced inputs. In particular, we test the four feature sets according to two protocols: the newly introduced experimental protocol (i.e., Protocol I) for online learning, as we explain in Section VI, and Protocol II, that follows [17]. Protocol I is suitable for online learning, whose main objective is to predict when a change in the mid-price will happen (i.e., regression problem) and in which direction, for instance, up or down (i.e., two-class classification problem). Protocol II predicts the mid-price movement direction for every next 10^{th} event, where feature representations are based on independent 10-event blocks. Authors in [17] used a joint training set of the five Nordic stocks for seven trading days and the next three days as testing for mid-price movement prediction (i.e., up, down, and stationary movement). We incorporate the same idea here, under the name "Joint", and we also use the same 7-3 training and testing proportion for each stock individually for both US and Nordic datasets. A general idea for both protocols can be seen in Fig. 10.

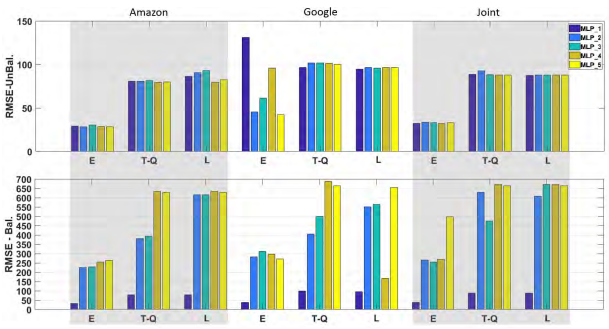
Protocol I and Protocol II use three types of deep neural networks as classifiers and regressors. In particular, we utilize five different MLPs, two CNNs, and two LSTMs. Motivation for choosing MLPs is the fact that such a simple neural network can perform extremely well when descriptive handcrafted features are used as input. The next type of neural network that we use is CNN. The first CNN, named "CNN_1" is based on [43], whereas the second one, named "CNN_2" is based on the grid search that we describe below. The last type of neural network that we utilize is LSTM. We use two different architectures: the first one, named "LSTM_1", is based on [17], and the second one, named "LSTM_2" is based on LSTM with attention mechanism. In total, we train independently nine deep neural networks for each of the two experimental protocols separately. Details of these nine topologies can be found in Table 4.

We report results for nine different neural networks, two of which are based on existing works as shown above. For the remaining seven neural networks we conduct the following grid search:

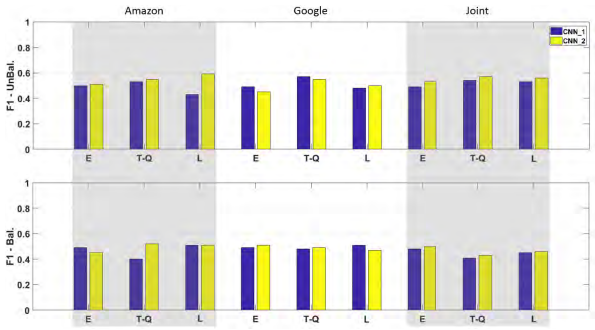
- For MLPs we set a limit up to three hidden layers, where for the number of nodes we set the options { 4, 9, 18, 64, 128, 256, and 512} nodes per layer and for dropout 20% and 50%. We report results based on five MLPs since these neural networks achieved good results for several cases (see Section VII-A for results discussion).
- For CNN we conduct an extensive grid search limited to up to three convolutional layers (with the option of 1-dimensional and 2-dimensional convolutional layer



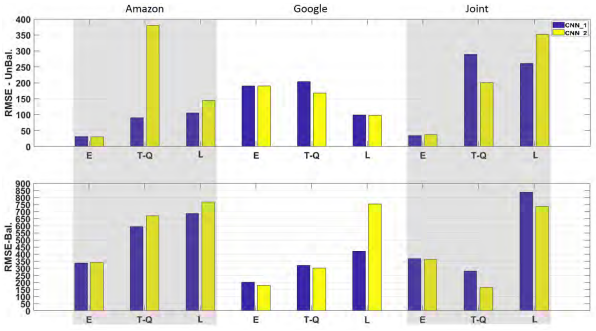
(a) F1 scores based on the unbalanced (top) and balanced (bottom) sets



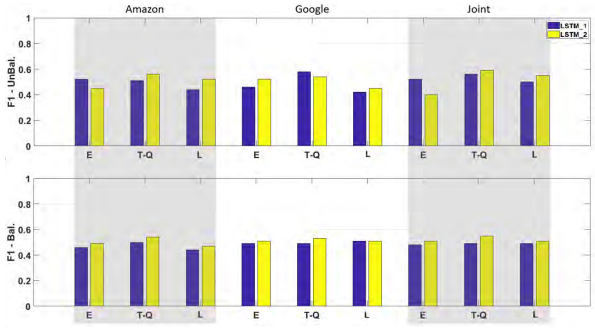
(b) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



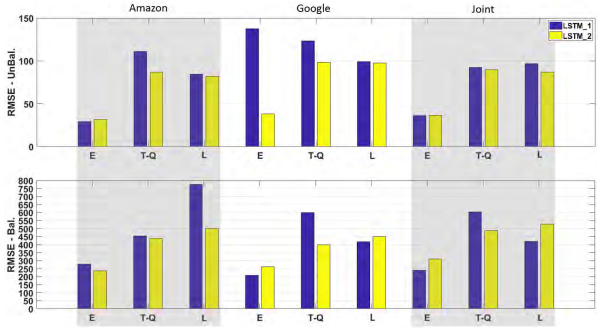
(c) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(d) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



(e) F1 scores based on the unbalanced (top) and balanced (bottom) sets



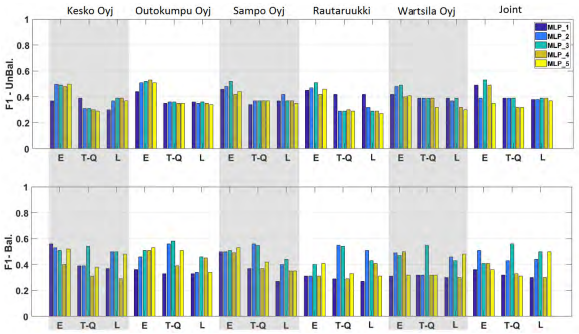
(f) RMSE scores based on the unbalanced (top) and balanced (bottom) sets

FIGURE 11. F1 (left column plots) and RMSE (right column plots) scores for the nine deep learning models based on the US data.

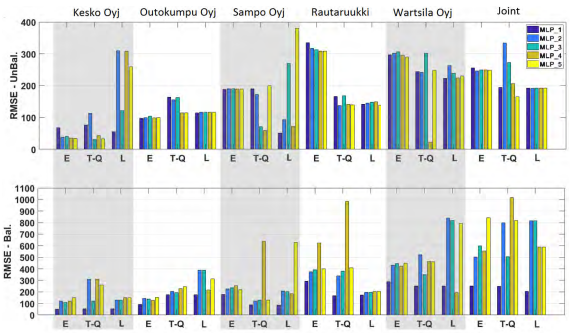
types) with 8, 16, and 32 filters and kernels size options {4×10, 4×20, 4×30, and 4×40 for the 2-dimensional case and 3 and 4 for the 1-dimensional case}. Dropout options are restricted to 20% and 50%. We report only one CNN since we noticed that shallower CNN architectures had very poor performance and no significant difference for the deeper ones.

- For LSTM we follow the same approach with up to three hidden layers and five options for hidden LSTM units {9, 18, 32, 64, 128} and the option of attention layer. We report only one LSTM performance since all other topologies performed worse for our task.

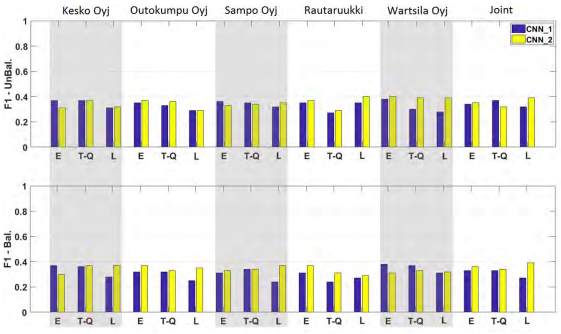
The training of these nine neural networks takes place at CSC super-cluster where we use Pascal P100 and K80 GPUs. We use multi-GPUs, under Keras (i.e., [54]) framework, in order to reduce the training time. The models, apart from CNN_1 and LSTM_1, use the Nesterov-Adam optimizer with a learning rate of 0.002, with mean squared error and binary cross-entropy for the dual output of Protocol I where this dual output is weighted by 0.01 and 0.99, respectively, and categorical cross-entropy as loss function for Protocol II. Additionally, we use 250 epochs to train our models with data shuffling and validation ratio of 0.2. Finally, in order to control overfitting we utilize Dropout to the majority of the suggested neural networks. By dropping out some nodes (i.e., a dropped out node have a zero output) from neural network topologies we control node dependencies and we achieve more robust results.



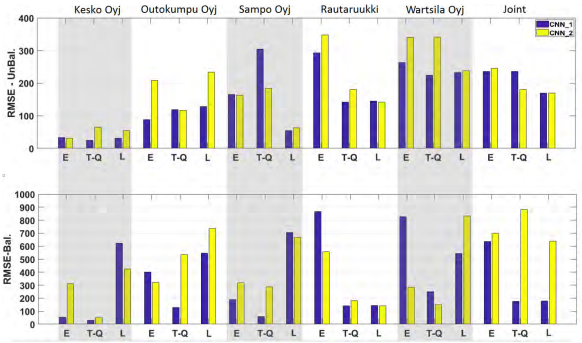
(a) F1 scores based on the unbalanced (top) and balanced (bottom) sets



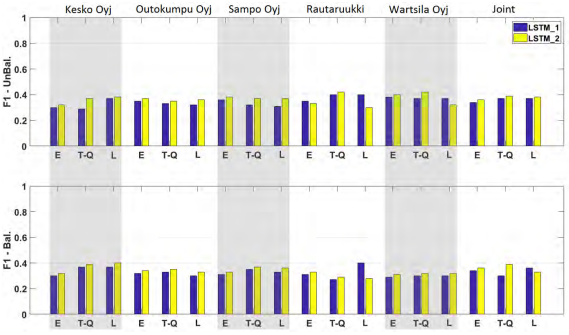
(b) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



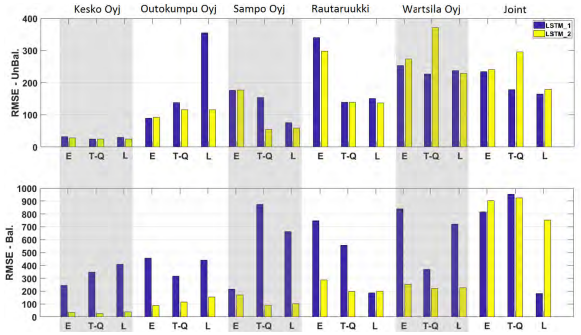
(c) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(d) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



(e) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(f) RMSE scores based on the unbalanced (top) and balanced (bottom) sets

FIGURE 12. F1 (left column plots) and RMSE (right column plots) scores for the nine deep learning models based on the Nordic data.

A. RESULTS

We present our results in separate tables for Protocol I (see Table 5 - Table 10) and Protocol II (see Appendix VIII-B). For each protocol, we split the results (i.e., f1 score and RMSE for Protocol I and f1 scores for Protocol II) for both US and Nordic datasets. We would like to mention that results derived from the LSTM AE, for both f1 and RMSE scores, are presented in separate tables (see Tables 9 & 10 for Protocol I and Tables A.2 & A.4 for Protocol II). Since handcrafted feature results overperformed the fully automated feature set we emphasize more on their perfromance by providing tables together with bar plots (see Fig. 11 & 12). Each of the tables contains the full head-to-head comparison for the three handcrafted features sets for each of the nine different deep learning models separately. For instance, Table 7 contains f1 scores for the Nordic stocks based on Protocol I. The table has five main columns (i.e., Model, Stock, Econ, Tech-Quant, and LOB) and six subcolumns divided into three pairs (i.e., UnBal. and Bal.). The first main column contains the nine deep neural networks; the second main column contains the five independent and different Nordic stocks, in which the sixth row for every model is the joint training set based on these five stocks; and the third, fourth and fifth main columns represent the three handcrafted feature sets. Moreover, for every feature set, we present results for unbalanced and balanced cases, whereas for the balanced cases we use random undersampling for the majority class. Even though balanced datasets do not project a realistic

TABLE 7. Protocol I: f1 scores for the Nordic stocks. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	0.37	0.56	0.39	0.39	0.39	0.37
	Outokumpu Oyj	0.44	0.36	0.35	0.33	0.36	0.33
	Sampo Oyj	0.46	0.50	0.34	0.37	0.37	0.35
	Rautaruukki	0.45	0.31	0.42	0.29	0.42	0.27
	Wartsila Oyj	0.42	0.31	0.39	0.32	0.39	0.30
	Joint	0.49	0.36	0.39	0.32	0.38	0.30
MLP_2	Kesko Oyj	0.50	0.53	0.31	0.39	0.37	0.50
	Outokumpu Oyj	0.51	0.46	0.36	0.56	0.35	0.34
	Sampo Oyj	0.48	0.50	0.37	0.56	0.42	0.40
	Rautaruukki	0.47	0.31	0.29	0.55	0.32	0.51
	Wartsila Oyj	0.48	0.49	0.39	0.32	0.37	0.46
	Joint	0.39	0.51	0.39	0.43	0.38	0.44
MLP_3	Kesko Oyj	0.49	0.51	0.31	0.54	0.39	0.50
	Outokumpu Oyj	0.52	0.51	0.36	**0.58**	0.36	0.46
	Sampo Oyj	0.52	0.51	0.37	0.55	0.37	0.44
	Rautaruukki	0.51	0.40	0.29	0.54	0.29	0.43
	Wartsila Oyj	0.49	0.47	0.39	0.55	0.39	0.43
	Joint	**0.53**	0.41	0.39	**0.56**	0.39	0.50
MLP_4	Kesko Oyj	0.48	0.40	0.30	0.31	0.39	0.29
	Outokumpu Oyj	**0.53**	0.51	0.35	0.39	0.35	0.45
	Sampo Oyj	0.42	0.49	0.37	0.37	0.37	0.35
	Rautaruukki	0.42	0.31	0.30	0.29	0.29	0.41
	Wartsila Oyj	0.40	0.50	0.39	0.32	0.32	0.30
	Joint	0.49	0.41	0.32	0.33	0.39	0.30
MLP_5	Kesko Oyj	0.50	0.52	0.29	0.38	0.37	0.48
	Outokumpu Oyj	0.51	0.53	0.35	0.51	0.34	0.34
	Sampo Oyj	0.44	0.53	0.37	0.42	0.35	0.35
	Rautaruukki	0.46	0.41	0.29	0.33	0.27	0.31
	Wartsila Oyj	0.41	0.32	0.32	0.32	0.30	0.48
	Joint	0.35	0.36	0.32	0.31	0.37	0.50
CNN_1	Kesko Oyj	0.37	0.37	0.37	0.36	0.31	0.28
	Outokumpu Oyj	0.35	0.32	0.33	0.32	0.29	0.25
	Sampo Oyj	0.36	0.31	0.35	0.34	0.32	0.24
	Rautaruukki	0.35	0.31	0.27	0.24	0.35	0.27
	Wartsila Oyj	0.38	0.38	0.30	0.37	0.28	0.31
	Joint	0.34	0.33	0.37	0.33	0.32	0.27
CNN_2	Kesko Oyj	0.31	0.32	0.37	0.31	0.32	0.39
	Outokumpu Oyj	0.37	0.37	0.36	0.33	0.29	0.35
	Sampo Oyj	0.33	0.33	0.34	0.34	0.35	0.37
	Rautaruukki	0.37	0.37	0.29	0.31	0.40	0.29
	Wartsila Oyj	0.40	0.31	0.39	0.33	0.39	0.32
	Joint	0.35	0.36	0.32	0.34	0.39	0.39
LSTM_1	Kesko Oyj	0.30	0.30	0.29	0.37	0.37	0.37
	Outokumpu Oyj	0.35	0.32	0.33	0.33	0.32	0.30
	Sampo Oyj	0.36	0.31	0.32	0.35	0.31	0.33
	Rautaruukki	0.35	0.31	0.40	0.27	0.40	0.40
	Wartsila Oyj	0.38	0.29	0.37	0.30	0.37	0.30
	Joint	0.34	0.34	0.37	0.30	0.37	0.36
LSTM_2	Kesko Oyj	0.32	0.32	0.37	0.39	0.38	0.40
	Outokumpu Oyj	0.37	0.34	0.35	0.35	0.36	0.33
	Sampo Oyj	0.38	0.33	0.37	0.37	0.37	0.36
	Rautaruukki	0.33	0.33	0.42	0.29	0.30	0.28
	Wartsila Oyj	0.40	0.31	0.42	0.32	0.32	0.32
	Joint	0.36	0.36	0.39	0.39	0.38	0.33

TABLE 8. Protocol I: RMSE scores based on Nordic stocks for the handcrafted features. *Note: Highlighted text shows the best RMSE performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	68.62	50.97	76.99	55.23	173.92	54.94
	Outokumpu Oyj	97.76	91.29	164.00	176.92	114.58	176.93
	Sampo Oyj	187.98	178.04	190.84	87.32	51.71	86.69
	Rautaruukki	334.61	292.36	166.61	167.00	142.40	173.41
	Wartsila Oyj	297.42	289.06	244.18	250.69	223.17	250.74
	Joint	255.92	252.13	195.15	250.02	192.19	205.41
MLP_2	Kesko Oyj	38.65	119.89	113.20	309.49	539.66	128.88
	Outokumpu Oyj	99.89	143.86	155.78	203.97	116.83	388.18
	Sampo Oyj	190.41	226.04	172.69	119.33	93.07	206.68
	Rautaruukki	317.66	374.41	137.48	340.20	144.91	196.54
	Wartsila Oyj	302.42	433.15	241.99	522.49	262.87	839.49
	Joint	246.80	502.29	334.22	799.74	191.54	815.36
MLP_3	Kesko Oyj	40.20	108.74	31.13	122.02	447.78	128.45
	Outokumpu Oyj	103.84	136.95	163.30	194.48	117.14	388.12
	Sampo Oyj	190.81	236.69	71.01	128.70	269.54	201.89
	Rautaruukki	312.41	390.14	168.69	378.60	149.69	196.54
	Wartsila Oyj	306.61	446.21	302.83	349.45	224.24	820.50
	Joint	249.72	599.45	272.35	504.21	192.46	815.36
MLP_4	Kesko Oyj	102.52	121.94	43.46	308.61	105.17	148.69
	Outokumpu Oyj	98.44	126.24	114.98	229.33	116.83	216.99
	Sampo Oyj	189.59	254.34	59.40	639.52	72.42	185.65
	Rautaruukki	309.03	625.19	142.40	985.50	149.69	204.93
	Wartsila Oyj	295.85	423.50	22.84	464.79	224.24	716.44
	Joint	250.11	555.76	206.55	1015.87	192.46	589.94
MLP_5	Kesko Oyj	99.79	148.71	33.19	259.19	650.56	148.43
	Outokumpu Oyj	99.79	151.41	114.83	246.73	117.14	312.73
	Sampo Oyj	189.36	220.94	199.50	129.55	379.95	629.82
	Rautaruukki	308.72	399.87	140.08	409.47	138.74	205.36
	Wartsila Oyj	290.03	450.53	248.25	460.97	230.90	793.99
	Joint	248.34	841.92	**165.29**	818.41	193.10	589.15
CNN_1	Kesko Oyj	32.73	55.43	25.32	30.37	30.98	623.58
	Outokumpu Oyj	88.04	401.17	118.89	128.44	128.44	545.72
	Sampo Oyj	164.81	189.73	304.42	59.40	54.07	704.06
	Rautaruukki	292.60	865.16	141.77	150.66	145.31	261.21
	Wartsila Oyj	262.92	827.61	224.24	248.79	232.48	544.67
	Joint	235.99	636.80	236.17	**176.25**	169.12	178.67
CNN_2	Kesko Oyj	30.75	310.08	64.88	54.35	54.85	424.82
	Outokumpu Oyj	208.71	321.86	116.56	535.32	233.86	737.63
	Sampo Oyj	163.52	317.88	184.45	288.18	62.67	669.31
	Rautaruukki	347.81	556.37	181.09	168.62	142.08	304.91
	Wartsila Oyj	340.68	284.84	340.92	149.73	238.14	832.23
	Joint	245.35	700.03	180.72	880.98	169.03	639.19
LSTM_1	Kesko Oyj	32.50	245.49	25.24	346.99	30.37	409.85
	Outokumpu Oyj	89.83	455.90	138.15	315.56	354.52	440.01
	Sampo Oyj	176.26	216.65	154.00	875.29	75.96	662.33
	Rautaruukki	339.78	746.22	138.94	556.63	150.66	187.29
	Wartsila Oyj	253.11	839.99	226.97	369.42	237.22	719.71
	Joint	234.65	816.23	178.73	952.69	166.10	180.74
LSTM_2	Kesko Oyj	28.22	34.63	**24.81**	**24.66**	24.88	39.29
	Outokumpu Oyj	92.34	89.71	116.09	114.20	115.78	153.77
	Sampo Oyj	177.53	171.03	54.07	90.62	59.23	100.59
	Rautaruukki	297.96	285.85	139.15	198.07	136.83	200.85
	Wartsila Oyj	273.83	253.96	370.96	220.45	229.40	227.30
	Joint	240.97	903.34	295.32	925.90	179.12	752.62

trading scenario (i.e., trading fees are not applicable), it is important to give an equal opportunity to the minority class, which can be an ML trader's trading position. More specifically, for Protocol I and the classification task, the Nordic dataset has 45% for the downward movement and 55% for the upward, while for the US dataset is 47% for the downward movement and 53% for the upward. The undersampling offers an 85% data reduction for the Nordic set and 90% for the US set. For better interpretation of Protocol I we provide bar plots which show the reaction of every deep learning model and dataset for the unbalanced and balanced cases (see Fig. 11 and Fig. 12). Protocol II and the Nordic dataset exhibits a 75% for the stationary condition, with the remaining 25% being equally divided to the upward and downward mid-price movement before undersampling. For the US dataset 73% belongs to the stationary condition, 20% to the upward movement and the remaining 7% to the downward movement. The undersampling offers a 30% data reduction for the Nordic dataset and 10% data reduction for the US dataset.

B. DISCUSSION

The conducted experiments reveal some interesting results for both experimental protocols and datasets selection. Both protocols forecast the mid-price movement, with Protocol I forecasting the mid-price movement every next event and Protocol I with a lag of 10 events. Protocol I provides

TABLE 9. Protocol I: f1 and RMSE scores based on US stocks for the fully-automated features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE - f1		LSTM AE - RMSE	
Model	Stock	UnBal.	Bal.	UnBal.	Bal
MLP_1	Amazon	0.37	0.36	79.81	90.79
	Google	0.34	0.34	109.80	110.98
	Joint	0.31	0.35	121.49	100.78
MLP_2	Amazon	0.31	0.36	88.37	116.29
	Google	0.34	0.34	96.51	136.51
	Joint	0.35	0.35	101.44	125.44
MLP_3	Amazon	0.31	0.36	83.73	116.89
	Google	0.33	0.34	96.05	134.96
	Joint	0.31	0.35	121.54	126.52
MLP_4	Amazon	0.36	0.36	**79.50**	117.54
	Google	0.32	0.34	103.76	134.17
	Joint	0.31	0.35	133.24	126.10
MLP_5	Amazon	0.31	**0.37**	81.86	105.38
	Google	0.32	0.34	98.29	124.58
	Joint	0.35	0.35	88.80	114.09
CNN_1	Amazon	0.36	0.36	187.06	90.92
	Google	0.34	0.34	95.92	109.60
	Joint	0.35	0.35	363.45	110.31
CNN_2	Amazon	0.36	0.31	82.66	90.18
	Google	0.34	0.32	113.39	109.24
	Joint	0.36	0.35	216.62	101.36
LSTM_1	Amazon	0.31	0.31	79.89	**87.70**
	Google	0.34	0.32	98.02	109.24
	Joint	0.31	0.35	87.70	100.52
LSTM_2	Amazon	0.46	0.31	80.03	90.17
	Google	**0.53**	0.32	97.13	109.06
	Joint	0.31	0.35	90.17	87.75

more information regarding the high-frequency activity since it takes into consideration every trading event. We cannot directly compare the two protocols since both tackle the problem of mid-price forecasting from a different angle. We also observe that in general, larger data samples increase deep learning models' performance. However, by focusing on each protocol seperately, we can see that: for Protocol I, the best classification score comes from US dataset and best regression score from Nordic dataset, while, for Protocol II, the best classification score comes again from the US dataset.

Each one of the nine neural networks has to perform a dual task, regression and classification simultaneously. To begin with, the Joint (i.e., the full range of stocks is used for training) reports for the Nordic dataset the best f1 performance that comes from MLP_3, for both unbalanced and balanced datasets under the Econ feature set with 53% and 56% for the Tech-Quant set. This MLP did not perform well for the regression task where the RMSE was above 165.29. For the stock specific case: we achieve the best classification performance of 53% f1 score for Outokumpu Oyj under MLP_4 and the Econ feature set with RMSE of 98.44. Thisvstock-specific performance of the MLP_4 is the best trade-off between classification and regression for the Nordic dataset. If we want to focus on the regression task only, we can choose the more advanced model, LSTM_2, with RMSE of approximately 24 for both unbalanced and balanced Tech-Quant feature sets for Kesko Oyj.

TABLE 10. Protocol I: f1 and RMSE scores based on Nordic stocks for the fully-automated features. *Note: Highlighted text shows the best RMSE performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE - f1		LSTM AE - RMSE	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	0.29	0.37	39.11	48.90
	Outokumpu Oyj	0.33	0.33	186.68	197.53
	Sampo Oyj	0.35	0.35	63.78	79.56
	Rautaruukki	0.27	0.27	182.43	201.86
	Wartsila Oyj	0.37	0.34	282.11	309.47
	Joint	0.37	0.37	207.90	190.24
MLP_2	Kesko Oyj	0.37	0.29	51.70	47.35
	Outokumpu Oyj	0.34	0.34	202.91	199.90
	Sampo Oyj	0.35	0.32	87.96	146.54
	Rautaruukki	0.27	0.40	210.69	212.94
	Wartsila Oyj	0.43	0.36	299.77	395.76
	Joint	0.30	0.37	236.10	207.82
MLP_3	Kesko Oyj	0.29	0.37	47.12	52.91
	Outokumpu Oyj	0.34	0.32	204.05	183.82
	Sampo Oyj	0.32	0.32	92.06	67.94
	Rautaruukki	0.40	0.40	207.93	174.24
	Wartsila Oyj	0.37	0.36	303.98	396.01
	Joint	0.30	0.37	232.29	209.93
MLP_4	Kesko Oyj	0.29	0.37	55.23	52.55
	Outokumpu Oyj	0.34	0.32	202.99	200.38
	Sampo Oyj	0.35	0.33	90.66	58.94
	Rautaruukki	0.40	0.40	209.65	202.04
	Wartsila Oyj	0.37	0.36	299.49	327.36
	Joint	0.30	0.37	233.67	204.09
MLP_5	Kesko Oyj	0.29	0.37	45.40	52.50
	Outokumpu Oyj	0.33	0.32	195.46	202.28
	Sampo Oyj	0.35	0.33	79.83	111.58
	Rautaruukki	0.40	0.40	196.63	245.04
	Wartsila Oyj	0.37	0.37	292.99	344.89
	Joint	0.30	0.37	222.40	202.40
CNN_1	Kesko Oyj	0.33	0.38	44.30	288.28
	Outokumpu Oyj	0.33	0.37	186.07	299.16
	Sampo Oyj	0.35	0.32	61.76	655.34
	Rautaruukki	0.40	0.27	173.71	510.65
	Wartsila Oyj	0.37	0.30	279.80	300.04
	Joint	0.40	0.37	206.89	305.67
CNN_2	Kesko Oyj	0.37	0.37	40.20	70.80
	Outokumpu Oyj	0.33	0.37	185.48	181.14
	Sampo Oyj	0.35	0.35	62.34	429.33
	Rautaruukki	0.27	0.27	176.98	350.67
	Wartsila Oyj	0.43	0.30	280.23	432.86
	Joint	**0.49**	0.37	**204.61**	350.43
LSTM_1	Kesko Oyj	0.37	0.38	39.10	42.10
	Outokumpu Oyj	0.35	0.33	185.28	179.89
	Sampo Oyj	0.35	0.35	61.65	67.34
	Rautaruukki	0.35	0.43	181.27	223.70
	Wartsila Oyj	0.35	0.30	279.46	256.78
	Joint	0.37	0.37	234.78	110.54
LSTM_2	Kesko Oyj	0.42	0.37	**39.03**	40.19
	Outokumpu Oyj	**0.47**	0.45	185.24	178.67
	Sampo Oyj	0.36	0.35	61.69	69.42
	Rautaruukki	0.35	0.30	180.49	167.89
	Wartsila Oyj	0.42	0.30	279.13	243.32
	Joint	0.38	0.37	212.89	89.90

For the US dataset, the new protocol presents more interesting results. For the Joint case, where both Amazon and Google used for training, the LSTM_2 achieves 59% f1 score and RMSE of 89.69, whereas, for the stock specific case, LSTM_1 under the Tech-Quant feature set achieves 58% f1 score and high RMSE of 123.36 for Google and the unbalanced case. If we focus only on the regression part, we can choose the entire MLP universe and the Econ feature set for Amazon and the Joint case. The newly introduced Econ feature set performed very well for the regression task also

for LSTM_2 across the entire protocol for the unbalanced dataset. One more interesting observation is that the Econ feature set together with the shallower MLP_1 and the balanced set reports very low RMSE for Amazon, Google, and the Joint cases, respectively. That means that the Econ feature set, for the Amazon and Joint case, were able to predict that the mid-price will change its direction in a millisecond duration. Here, it is vital to report that the daily trading activity, for the US and Nordic stocks, contains several trades with the same timestamp/millisecond. Approximately 30% of the trades, in the US dataset, occur in a millisecond, whereas this percentage for the Nordic dataset is 36%.

For Protocol II and the Joint case we achieve the best forecasting performance of 51% f1 for the Nordic dataset based on MLP_4 (which is one of our deeper MLP architectures) under the Tech-Quant feature set and the unbalanced case. For the Joint case in the US dataset, we achieve the best f1 performance of 65% based on MLP_4 under the Tech-Quant feature set and the balanced case. In terms of individual stock performance for the Nordic case we achieve 63% f1 score for Kesko Oyj, and our shallower MLP (i.e., MLP_1) under the Tech-Quant set, while for the US dataset we achieve an f1 performance of 65% for Google based on MLP_4 for the balanced case. We can see that MLPs for Protocol II were able to retain the information that the Tech-Quant feature set carries. The majority of the Tech-Quant features was derived from technical analysis, a type of analysis which is based on geometrical pattern identification of agglutinated times series like ours. What is more, the data size affected the performance of models and feature sets. For instance, Kesko Oyj, which scored the highest f1 score, is the stock with the least daily trading activity compared to the rest of the Nordic stocks and of course compared to the massive US dataset. Finally, we would like to point out that we limited the experiments to two US and five Nordic stocks; we leave the extension of the present evaluation on wider LOB datasets for future research that will help us to identify similarities among stock categories and time periods.

VIII. CONCLUSION

In this paper, we extracted handcrafted features based on the econometric literature for mid-price prediction using deep learning techniques. Our work is the first of its kind since we do not only utilize an extensive feature set list, based on econometrics for the mid-price prediction task, but we also provide a fair comparison with two other existing state-of-the-art handcrafted and fully automated feature sets . Our extensive experimental setup, based on liquid and illiquid stocks (i.e., two US and five Nordic stocks) showed superiority of the suggested handcrafted feature sets against the fully automated process derived from an LSTM AE. What is more, our research sheds light on the area of deep learning and feature engineering by providing information based on online mid-price predictions. Our findings suggest that extensive analysis of the input signal leads to high forecasting performance even with simpler neural network architects like shallow MLPs, particularly when advanced features capture the relevant information edge. More specifically, econometric features and deep learning predicted that the mid-price would change direction in a millisecond duration for Amazon and the Joint (i.e., training on both Amazon and Google) cases. Although these results are promising, our study here also suggests that selection of features and models should be differentiated for liquid and illiquid stocks

APPENDIX

A. FEATURE POOL

See Figure 11–12 and Tables 7–10.

1) STATISTICAL FEATURES

- Mid price is defined as:

$$MP = \frac{Ask_{best} + Bid_{best}}{2} \tag{A.1}$$

- Financial duration is defined as:

$$FD = T_t - T_{t-1}, \tag{A.2}$$

where T denotes the time instance at time t.

- Average mid-price financial duration is defined as:

$$AMPD_l = \frac{\left\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\right\}_{i=1}^{N}}{\left\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\right\}_{i=1}^{N}}, \tag{A.3}$$

where $\left\{\mathcal{T}_k\right\}_{k=1}^{N}$, and $\left\{\mathcal{P}_k\right\}_{k=1}^{N}$ are the partial cumulative sums of time and price differences for every LOB level for N samples.

- Mid price deeper levels are equal to:

$$DMP = \frac{Ask_l + Bid_l}{2},\ l = 2:10 \tag{A.4}$$

where l denotes the depth of the LOB.

- Log returns are defined as:

$$r(X)_i = X_i - X_{i-1}; \tag{A.5}$$

where X_i is the logarithmic price

2) VOLATILITY MEASURES

The features in this category aim to estimate, either the integrated variance (IV), that is the process

$$IV_t = \int_0^t \sigma_s^2 \mathrm{d}s \tag{A.6}$$

or, more generally, the quadratic variation (QV)

$$[X, X]_t = \int_0^t \sigma_s^2 \mathrm{d}s + \sum_{0<s\leq t} (\zeta_s \mathrm{d}N_s)^2. \tag{A.7}$$

Here X is the logarithmic price of some given asset. We assume that X_t follows an Itô semimartingale; that is,

$$X_t = X_0 + \int_0^t b_s \mathrm{d}s + \int_0^t \sigma_s \mathrm{d}W_s + \int_o^t \zeta_s \mathrm{d}N_s, \tag{A.8}$$

where b is locally bounded, σ is cádlág and predictable, and W is a standard Weiner process, ζ is a thin (i.e., finite) process mapping the jump size, and N is the counting process associated to the jump times of X. We define Δ_n the time elapsed between two adjacent observations; specifically, if we assume the observations are equidistant in time we have $\Delta_n = \lfloor \frac{t}{n} \rfloor$. As we do not work in calendar time we will have $\Delta_n = \frac{1}{n}$.

- Realized variance
  The realized variance [55] is the most natural estimator of the quadratic variation process and is equal to:

$$RV_t = \sum_{i=1}^{n} (r(X)_i)^2. \tag{A.9}$$

- Realized kernel
  Realized kernels [56] are used to obtain a noise robust estimate of QV as follows:

$$RK_t = \gamma_0(X_{\Delta_n}) + \sum_{h=1}^{H} k\left(\frac{h}{H}\right)\{\gamma_h(X_{\Delta_n}) + \gamma_{-h}(X_{\Delta_n})\}, \tag{A.10}$$

  with H the kernel bandwidth, $\gamma_h(X_{\Delta_n})$ the autocovariation process, k is the kernel function of choice. In particular we use a non-flat-top Parzen and our implementation follows closely [53].
- Realized pre-averaged variance
  The pre-averaged realized variance [57] is akin to the realized kernel estimator (in fact they are asymptotically equivalent). As for the realized kernel, the pre-averaged realized variance is used to retrieve a noise-free measurement of the quadratic variation of our price process and it is calculated as follows:

$$PA - RV_t = \frac{\sqrt{\Delta_n}}{\theta\psi_2} \sum_{i=0}^{n-H+1} (\bar{X}_i^n)^2 - \frac{\psi_1 \Delta_n}{2\theta^2\psi_2} \sum_{i=0}^{N} (r(X))^2. \tag{A.11}$$

  As before we have H the kernel bandwidth and θ the pre-averaging horizon. Further, given a nonzero real-valued function $g : [0, 1] \to \mathbb{R}$ with $g(0) = g(1) = 0$ and which is further continuous and piecewise continuously differentiable such that its derivative g' is piecewise Lipschitz. Then, we define:

$$\psi_1 = \int_0^1 (g'(s))^2 \mathrm{d}s, \qquad \psi_1 = \int_0^1 (g(s))^2 \mathrm{d}s.$$

  In our application we follow [58] and set $H = \theta\sqrt{n}$ and $\theta = 1$, $g(x) = x \wedge (1 - x)$. Hence we will have $\psi_1 = 1$ and, $\psi_2 = \frac{1}{12}$.
- Realized semi-variance $(+, -)$
  Positive $(+)$ and negative $(-)$ realized semi-variances [59] measure upside and downside risk respectively, as follows:

$$RSV^+(X)_t = \sum_{i=1}^{n} r(X)_i^2 1_{(r(X)_i > 0)}$$

$$RSV^-(X)_t = \sum_{i=1}^{n} r(X)_i^2 1_{(r(X)_i < 0)} \tag{A.12}$$

  where 1 is a simple indicator function.
- Realized bipower variation
  The realized bipower variation [60] measures the diffusive component of the price process, isolating it from the variation caused by the jump components and it is equal to:

$$BV(X)_t := \frac{\pi}{2} \sum_{i=2}^{n} |r(X)_i||r(X)_{i-1}| \tag{A.13}$$

- Realized bipower variation (lag 2)

$$BV(X)_t := \frac{\pi}{2} \sum_{i=3}^{n} |r(X)_i||r(X)_{i-2}| \tag{A.14}$$

- Realized bipower semivariance $(+, -)$
  Realized bipower semivariances [59] are used to measure the upside and downside risk of the diffusive component:

$$BV^+(X)_t := \frac{\pi}{2} \sum_{i=2}^{n} |r(X)_i||r(X)_{i-1}| 1_{(r(X)_i > 0)}$$

$$BV^-(X)_t := \frac{\pi}{2} \sum_{i=2}^{n} |r(X)_i||r(X)_{i-1}| 1_{(r(X)_i < 0)}. \tag{A.15}$$

- Jump variation
  We use a modified version of the jump variation estimator [58] which is both non-negative and consistent. As hinted by the name, the jump variation estimator provides a measures of the discontinuous variability component:

$$JV(X)_t := \max(RV(X)_t - BV(X)_t, 0). \tag{A.16}$$

- Spot volatility
  We only compute the spot volatility (i.e., [61] and [37]) estimates on the block. The spot volatility measures the instantaneous volatility. The definition is consistent with the terminology commonly used in the literature on parametric stochastic volatility models in continuous-time:

$$SV(X)_t := \lim_{h \to 0} \{\mathbb{E}[([X, X]_{t+h} - [X, X]_t)/h] | \mathcal{F}_t\}. \tag{A.17}$$

  with $h \to 0$ being the time interval upon which the measure is computed.
- Average spot volatility
  The average spot volatility provides an historical average of the estimated spot volatilities:

$$\overline{SV}(X)_t := \frac{1}{t} \sum_{i=0}^{t} SV(X)_i. \tag{A.18}$$

TABLE 11. Protocol II: f1 scores based on US stocks for the handcrafted features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Amazon	0.28	0.26	0.20	0.62	0.31	0.45
	Google	0.28	0.28	0.18	0.63	0.30	0.55
	Joint	0.36	0.25	0.23	0.56	0.27	0.51
MLP_2	Amazon	0.35	0.26	0.20	0.58	0.25	0.31
	Google	0.31	0.35	0.19	0.35	0.32	0.50
	Joint	0.21	0.25	0.24	0.49	0.29	0.50
MLP_3	Amazon	0.25	0.26	0.26	0.46	0.26	0.44
	Google	0.27	0.33	0.19	0.47	0.26	0.48
	Joint	0.28	0.21	0.33	0.63	**0.51**	0.56
MLP_4	Amazon	0.21	0.23	0.19	0.53	0.15	0.41
	Google	0.27	0.27	0.20	**0.65**	0.27	0.56
	Joint	0.21	0.26	0.27	**0.65**	0.21	0.59
MLP_5	Amazon	0.31	0.26	0.21	0.56	0.20	0.39
	Google	0.33	0.31	0.20	0.62	0.21	0.56
	Joint	0.36	0.30	0.24	0.52	0.35	0.57
CNN_1	Amazon	0.34	0.25	0.19	0.16	0.24	0.18
	Google	0.33	0.34	0.21	0.19	0.27	0.22
	Joint	0.35	0.27	0.19	0.22	0.29	0.20
CNN_2	Amazon	0.31	0.26	0.22	0.21	0.26	0.21
	Google	0.35	0.22	0.25	0.20	0.31	0.22
	Joint	0.37	0.29	0.23	0.24	0.30	0.23
LSTM_1	Amazon	0.33	0.22	0.23	0.22	0.28	0.15
	Google	0.31	0.35	0.22	0.23	0.33	0.14
	Joint	0.35	0.23	0.19	0.25	0.21	0.19
LSTM_2	Amazon	0.34	0.26	0.21	0.25	0.26	0.21
	Google	0.32	0.37	0.24	0.26	**0.41**	0.19
	Joint	0.37	0.25	0.20	0.27	0.42	0.22

TABLE 12. Protocol II: f1 scores based on US stocks for the fully automated features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE	
Model	Stock	UnBal.	Bal.
MLP_1	Amazon	0.27	0.28
	Google	0.28	**0.31**
	Joint	0.27	0.25
MLP_2	Amazon	0.11	0.11
	Google	0.15	0.18
	Joint	0.19	0.25
MLP_3	Amazon	0.22	0.22
	Google	0.27	0.24
	Joint	0.23	0.21
MLP_4	Amazon	0.25	0.25
	Google	0.24	0.23
	Joint	0.22	0.26
MLP_5	Amazon	0.21	0.21
	Google	0.23	0.28
	Joint	0.24	**0.30**
CNN_1	Amazon	0.27	0.21
	Google	0.24	0.22
	Joint	0.25	0.19
CNN_2	Amazon	0.27	0.25
	Google	0.29	0.26
	Joint	0.30	0.25
LSTM_1	Amazon	0.19	0.21
	Google	0.33	0.27
	Joint	**0.33**	0.22
LSTM_2	Amazon	0.21	0.23
	Google	**0.34**	0.21
	Joint	0.28	0.24

3) NOISE AND UNCERTAINTY MEASURES

In this category, we incorporate two kinds of measures which are intimately linked to each other. We provide three different estimates for the integrated quarticity and two different estimates for the variance of the contaminating noise process. The integrated quarticity measures the degree of estimation error in the realized variance and can be consistently estimated through the realized quarticity estimators presented below for a fixed window size of 2000 events. The noise variance estimates provide a measure of the intensity of the noise process affecting the underlying price, as follows:

$$IQ_t = \int_0^t \sigma_s^4 \mathrm{d}s \tag{A.19}$$

with the noise variance estimates providing a measure of the contaminating:

- Realized quarticity [62]:

$$RQ_t = \frac{n}{3}\sum_{i=1}^{n}(X_i - X_{i-1})^4 \tag{A.20}$$

- Realized quarticity Tripower
  The tri-power quarticity [62] is a generalization of the realized bipower variation and is a consistent estimator for the integrated quarticity in the presence of jumps:

$$RQ_t = n\mu_{4/3}^{-3}\sum_{i=3}^{n}|r(X)_i|^{4/3}|r(X)_{i-1}|^{4/3}|r(X)_{i-2}|^{4/3} \tag{A.21}$$

with $\mu_p = \mathbb{E}(|Z|^p)$,where Z denotes a standard normally distributed random variable.

- Realized quarticity Quadpower
  A generalization of multipower variation measures led to the realized quadpower quarticity estimator proposed by [62] and it is equal to:

$$RQ_t = n\mu_1^{-4}\sum_{i=4}^{n}|r(X)_i||r(X)_{i-1}||r(X)_{i-2}||r(X)_{i-3}| \tag{A.22}$$

- Noise variance [35]:

$$NV_t = -\frac{1}{n-1}\sum_{i=2}^{n}(r(X)_i r(X)_{i-1}). \tag{A.23}$$

- Noise variance [36]:

$$NV_t = \frac{1}{2n}\sum_{i=1}^{n}(X_i - X_{i-1})^2. \tag{A.24}$$

4) PRICE DISCOVERY FEATURES

- Mid price weighted by order imbalance:

$$MidPrice_t = \frac{Ask * V_{Ask} + Bid * V_{Bid}}{V_{Ask} + V_{Bid}}. \tag{A.25}$$

- Volume imbalance:

$$VolImbalance = \frac{V_{Bid}}{V_{Ask} + V_{Bid}} \tag{A.26}$$

TABLE 13. Protocol II: f1 scores based on Nordic stocks for the handcrafted features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	0.42	0.29	**0.63**	**0.56**	0.59	0.25
	Outokumpu Oyj	0.30	0.25	0.51	0.45	0.52	0.40
	Sampo Oyj	0.32	0.30	0.56	0.48	0.57	0.36
	Rautaruukki	0.19	0.37	0.42	0.48	0.42	0.41
	Wartsila Oyj	0.30	0.35	0.43	0.37	0.36	0.41
	Joint	0.37	0.34	0.48	0.43	0.45	0.42
MLP_2	Kesko Oyj	0.45	0.38	0.55	0.53	0.51	0.44
	Outokumpu Oyj	0.30	0.31	0.46	0.45	0.51	0.45
	Sampo Oyj	0.30	0.30	0.49	0.48	0.55	0.45
	Rautaruukki	0.31	0.35	0.38	0.37	0.40	0.41
	Wartsila Oyj	0.34	0.36	0.35	0.38	0.36	0.39
	Joint	0.32	0.34	0.34	0.42	0.37	0.44
MLP_3	Kesko Oyj	0.44	0.39	0.56	0.48	0.53	0.43
	Outokumpu Oyj	0.30	0.31	0.47	0.46	0.50	0.44
	Sampo Oyj	0.30	0.29	0.50	0.48	0.54	0.52
	Rautaruukki	0.34	0.37	0.40	0.42	0.41	0.41
	Wartsila Oyj	0.30	0.37	0.36	0.41	0.34	0.33
	Joint	0.33	0.33	0.49	0.43	0.48	0.43
MLP_4	Kesko Oyj	0.44	0.36	0.54	0.52	0.52	0.41
	Outokumpu Oyj	0.30	0.29	0.45	0.45	0.52	0.46
	Sampo Oyj	0.31	0.31	0.51	0.49	0.54	0.46
	Rautaruukki	0.33	0.35	0.41	0.40	0.42	0.42
	Wartsila Oyj	0.33	0.35	0.40	0.40	0.38	0.40
	Joint	0.30	0.33	**0.51**	0.41	0.46	0.41
MLP_5	Kesko Oyj	0.45	0.37	0.49	0.49	0.53	0.39
	Outokumpu Oyj	0.30	0.30	0.49	0.48	0.51	0.43
	Sampo Oyj	0.32	0.30	0.51	0.49	0.53	0.50
	Rautaruukki	0.32	0.35	0.43	0.40	0.42	0.41
	Wartsila Oyj	0.29	0.35	0.41	0.38	0.37	0.41
	Joint	0.32	0.33	0.50	0.44	0.44	**0.45**
CNN_1	Kesko Oyj	0.42	0.26	0.51	0.46	0.48	0.09
	Outokumpu Oyj	0.31	0.26	0.46	0.29	0.48	0.18
	Sampo Oyj	0.34	0.27	0.45	0.38	0.50	0.34
	Rautaruukki	0.32	0.38	0.40	0.38	0.40	0.29
	Wartsila Oyj	0.31	0.23	0.36	0.24	0.33	0.20
	Joint	0.32	0.25	0.44	0.26	0.45	0.29
CNN_2	Kesko Oyj	0.45	0.13	0.52	0.37	0.54	0.20
	Outokumpu Oyj	0.28	0.17	0.51	0.30	0.51	0.22
	Sampo Oyj	0.33	0.19	0.52	0.26	0.55	0.20
	Rautaruukki	0.40	0.16	0.41	0.29	0.40	0.28
	Wartsila Oyj	0.33	0.30	0.36	0.26	0.38	0.28
	Joint	0.31	0.30	0.49	0.26	0.47	0.27
LSTM_1	Kesko Oyj	0.43	0.28	0.52	0.50	0.54	0.14
	Outokumpu Oyj	0.31	0.24	0.45	0.34	0.49	0.24
	Sampo Oyj	0.32	0.31	0.50	0.39	0.51	0.30
	Rautaruukki	0.28	0.24	0.38	0.31	0.40	0.30
	Wartsila Oyj	0.33	0.32	0.35	0.29	0.39	0.27
	Joint	0.32	0.27	0.46	0.27	0.45	0.29
LSTM_2	Kesko Oyj	0.44	0.21	0.54	0.47	0.48	0.13
	Outokumpu Oyj	0.32	0.23	0.45	0.19	0.49	0.19
	Sampo Oyj	0.29	0.25	0.50	0.40	0.52	0.31
	Rautaruukki	0.32	0.30	0.38	0.30	0.39	0.30
	Wartsila Oyj	0.31	0.31	0.35	0.30	0.38	0.27
	Joint	0.34	0.26	0.46	0.26	0.48	0.27

TABLE 14. Protocol II: f1 scores based on Nordic stocks for the fully automated features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE	
Model	Stock	UnBal.	Bal.
MLP_1	Kesko Oyj	0.35	0.30
	Outokumpu Oyj	0.20	0.19
	Sampo Oyj	0.28	0.26
	Rautaruukki	0.19	0.21
	Wartsila Oyj	0.28	0.22
	Joint	**0.37**	0.26
MLP_2	Kesko Oyj	0.34	0.29
	Outokumpu Oyj	0.20	0.18
	Sampo Oyj	0.27	0.21
	Rautaruukki	0.31	0.21
	Wartsila Oyj	0.27	0.22
	Joint	0.32	0.20
MLP_3	Kesko Oyj	0.32	**0.33**
	Outokumpu Oyj	0.20	0.17
	Sampo Oyj	0.26	0.29
	Rautaruukki	0.26	0.23
	Wartsila Oyj	0.26	0.26
	Joint	0.33	**0.29**
MLP_4	Kesko Oyj	0.30	0.31
	Outokumpu Oyj	0.20	0.19
	Sampo Oyj	0.28	0.32
	Rautaruukki	0.26	0.28
	Wartsila Oyj	0.33	0.27
	Joint	0.30	0.25
MLP_5	Kesko Oyj	0.30	0.28
	Outokumpu Oyj	0.20	0.19
	Sampo Oyj	0.18	0.17
	Rautaruukki	0.19	0.18
	Wartsila Oyj	0.18	0.18
	Joint	0.32	0.23
CNN_1	Kesko Oyj	0.28	0.26
	Outokumpu Oyj	0.29	0.27
	Sampo Oyj	0.26	0.27
	Rautaruukki	0.31	0.21
	Wartsila Oyj	0.30	0.22
	Joint	0.32	0.19
CNN_2	Kesko Oyj	0.29	0.13
	Outokumpu Oyj	0.27	0.17
	Sampo Oyj	0.29	0.19
	Rautaruukki	**0.36**	0.16
	Wartsila Oyj	0.31	0.24
	Joint	0.31	0.21
LSTM_1	Kesko Oyj	0.28	0.28
	Outokumpu Oyj	0.32	0.24
	Sampo Oyj	0.31	0.25
	Rautaruukki	0.27	0.25
	Wartsila Oyj	0.31	0.24
	Joint	0.33	0.27
LSTM_2	Kesko Oyj	0.31	0.21
	Outokumpu Oyj	0.33	0.23
	Sampo Oyj	0.33	0.23
	Rautaruukki	0.34	0.22
	Wartsila Oyj	0.32	0.22
	Joint	0.31	0.25

- Bid-ask spread:

$$BA_{\text{spread}} = Ask - Bid. \tag{A.27}$$

- Normalized bid-ask spread
  The normalized bid-ask spread expresses the spread as the number of ticks between the bid and the ask price:

$$BA_{\text{spread}} = \frac{Ask - Bid}{TickSize}. \tag{A.28}$$

B. PROTOCOL II RESULTS

See Tables 11–14.

ACKNOWLEDGMENT

The authors would like to thank CSC-IT Center for Science, Finland, for generous computational resources. The views and conclusions expressed in this paper are solely those of the authors and do not necessarily reflect the views of Danmarks Nationalbank.

REFERENCES

[1] X.-Y. Qian, "Financial series prediction: Comparison between precision of time series models and machine learning methods," 2017, *arXiv:1706.00948*. [Online]. Available: https://arxiv.org/abs/1706.00948

[2] S. Siami-Namini and A. S. Namin, "Forecasting economics and financial time series: ARIMA vs. LSTM," 2018, *arXiv:1803.06386*. [Online]. Available: https://arxiv.org/abs/1803.06386

[3] L. Chen, Z. Qiao, M. Wang, C. Wang, R. Du, and H. E. Stanley, "Which artificial intelligence algorithm better predicts the chinese stock market?" *IEEE Access*, vol. 6, pp. 48625–48633, 2018.
[4] P. Nousi, A. Tsantekidis, N. Passalis, A. Ntakaris, J. Kanniainen, A. Tefas, M. Gabbouj, and A. Iosifidis, "Machine learning for forecasting mid price movement using limit order book data," 2018, *arXiv:1809.07861*. [Online]. Available: https://arxiv.org/abs/1809.07861
[5] J. Sirignano and R. Cont, "Universal features of price formation in financial markets: Perspectives from deep learning," 2018, *arXiv:1803.06917*. [Online]. Available: https://arxiv.org/abs/1803.06917
[6] M. Velay and F. Daniel, "Stock chart pattern recognition with deep learning," 2018, *arXiv:1808.00418*. [Online]. Available: https://arxiv.org/abs/1808.00418
[7] R. Dash and P. K. Dash, "A hybrid stock trading framework integrating technical analysis with machine learning techniques," *J. Finance Data Sci.*, vol. 2, no. 1, pp. 42–57, 2016.
[8] M. U. Gudelek, S. A. Boluk, and A. M. Ozbayoglu, "A deep learning based stock trading model with 2-D CNN trend detection," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov./Dec. 2017, pp. 1–8.
[9] P. Wang, "Pricing currency options with support vector regression and stochastic volatility model with jumps," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 1–7, 2011.
[10] M. Zięba, S. K. Tomczak, and J. M. Tomczak, "Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction," *Expert Syst. Appl.*, vol. 58, pp. 93–101, Oct. 2016.
[11] A. Ntakaris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, *Mid-Price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators*. 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3213389
[12] A. N. Kercheval and Y. Zhang, "Modelling high-frequency limit order book dynamics with support vector machines," *Quant. Finance*, vol. 15, no. 8, pp. 1315–1329, Jun. 2015. doi: 10.1080/14697688.2015.1032546.
[13] 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," *J. Forecasting*, vol. 37, no. 8, pp. 852–866, 2018.
[14] H. Guo, "Limited stock market participation and asset prices in a dynamic economy," *J. Financial Quant. Anal.*, vol. 39, no. 3, pp. 495–516, 2004.
[15] M. Lettau and S. Ludvigson, "Consumption, aggregate wealth, and expected stock returns," *J. Finance*, vol. 56, no. 3, pp. 815–849, 2001.
[16] K. H. Chung and C. Chuwonganant, "Market volatility and stock returns: The role of liquidity providers," *J. Financial Markets*, vol. 37, pp. 17–34, Jan. 2018.
[17] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning for price prediction by exploiting stationary limit order book features," 2018, *arXiv:1810.09965*. [Online]. Available: https://arxiv.org/abs/1810.09965
[18] M. Göçken, M. Özçalici, A. Boru, and A. T. Dosdogru, "Integrating metaheuristics and Artificial Neural Networks for improved stock price prediction," *Expert Syst. Appl.*, vol. 44, pp. 320–331, Feb. 2016.
[19] X. Zhang, T. Xue, and H. E. Stanley, "Comparison of econometric models and artificial neural networks algorithms for the prediction of baltic dry index," *IEEE Access*, vol. 7, pp. 1647–1657, 2019.
[20] J. Sirignano, "Deep learning for limit order books," 2016, *arXiv:1601.01987*. [Online]. Available: https://arxiv.org/abs/1601.01987
[21] M. Dixon, "Sequence classification of the limit order book using recurrent neural networks," *J. Comput. Sci.*, vol. 24, pp. 277–286, Jan. 2018.
[22] D. L. Minh, A. Sadeghi-Niaraki, H. D. Huy, K. Min, and H. Moon, "Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network," *IEEE Access*, vol. 6, pp. 55392–55404, 2018.
[23] Z. Zhang, S. Zohren, and S. Roberts, "DeepLOB: Deep convolutional neural networks for limit order books," 2018, *arXiv:1808.03668*. [Online]. Available: https://arxiv.org/abs/1808.03668
[24] N. Passalis, A. Tsantekidis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Time-series classification using neural bag-of-features," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug./Sep. 2017, pp. 301–305.
[25] D. T. Tran, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov./Dec. 2017, pp. 1–7.
[26] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1407–1418, May 2019.
[27] B. Zheng, E. Moulines, and F. Abergel, "Price jump prediction in limit order book," 2012, *arXiv:1204.1381*. [Online]. Available: https://arxiv.org/abs/1204.1381
[28] J. Alberg and Z. C. Lipton, "Improving factor-based quantitative investing by forecasting company fundamentals," 2017, *arXiv:1711.04837*. [Online]. Available: https://arxiv.org/abs/1711.04837
[29] O. B. Sezer, A. M. Ozbayoglu, and E. Dogdu, "An artificial neural network-based stock trading system using technical analysis and big data framework," in *Proc. SouthEast Conf.*, 2017, pp. 223–226.
[30] J. Han, J. Hong, N. Sutardja, and S. F. Wong, "Machine learning techniques for price change forecast using the limit order book data," Tech. Rep., 2015. [Online]. Available: http://jcyhong.github.io/assets/machine-learning-price-movements.pdf
[31] K. Kanagal, Y. Wu, and K. Chen. (2017). *Market Making With Machine Learning Methods*. [Online]. Available: https://web.stanford.edu/class/msande448/2017/Final/Reports/gr4.pdf
[32] J. Doering, M. Fairbank, and S. Markose, "Convolutional neural networks applied to high-frequency market microstructure forecasting," in *Proc. 9th Comput. Sci. Electron. Eng. (CEEC)*, Sep. 2017, pp. 31–36.
[33] M. Mäkinen, A. Iosifidis, M. Gabbouj, and J. Kanniainen, *Predicting Jump Arrivals in Stock Prices Using Neural Networks With Limit Order Book Data*. 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3165408
[34] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug./Sep. 2017, pp. 2511–2515.
[35] R. C. A. Oomen, "Properties of realized variance under alternative sampling schemes," *J. Bus. Econ. Statist.*, vol. 24, no. 2, pp. 219–237, 2006. [Online]. Available: http://www.jstor.org/stable/27638871
[36] L. Zhang, P. A. Mykland, and Y. Aït-Sahalia, "A tale of two time scales: Determining integrated volatility with noisy high-frequency data," *J. Amer. Stat. Assoc.*, vol. 100, no. 472, pp. 1394–1411, 2005.
[37] T. G. Andersen, T. Bollerslev, and F. X. Diebold, "Parametric and nonparametric volatility measurement," in *Handbook of Financial Econometrics: Tools and Techniques*, vol. 1. Amsterdam, The Netherlands: Elsevier, 2010, ch. 2, pp. 67–137.
[38] S. Lahmiri, "Wavelet low- and high-frequency components as features for predicting stock prices with backpropagation neural networks," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 26, no. 2, pp. 218–227, 2014.
[39] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1995.
[40] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," 2015, *arXiv:1502.05767*. [Online]. Available: https://arxiv.org/abs/1502.05767
[41] T. Dozat, "Incorporating nesterov momentum into adam," in *Proc. ICLR*, 2016. [Online]. Available: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ
[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[43] 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 *Proc. IEEE 19th Conf. Bus. Inform. (CBI)*, vol. 1, Jul. 2017, pp. 7–12.
[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[45] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2016, pp. 207–212.
[46] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, Sep. 2014.
[47] C.-Y. Liou, J.-C. Huang, and W.-C. Yang, "Modeling word perception using the Elman network," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3150–3157, 2008.
[48] M. Qi and G. P. Zhang, "Trend time–series modeling and forecasting with neural networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 808–816, May 2008.
[49] M. Butler and D. Kazakov, "The effects of variable stationarity in a financial time-series on artificial neural networks," in *Proc. IEEE Symp. Comput. Intell. Financial Eng. Econ. (CIFEr)*, Apr. 2011, pp. 1–8.
[50] T. Y. Kim, K. J. Oh, C. Kim, and J. D. Do, "Artificial neural networks for non-stationary time series," *Neurocomputing*, vol. 61, pp. 439–447, Oct. 2004.

[51] M. M. Dacorogna, R. GenÃğgay, U. A. MÃijller, R. B. Olsen, and O. V. Pictet, Eds., "4—Adaptive data cleaning," in *An Introduction to High-Frequency Finance*. San Diego, CA, USA: Academic, 2001, pp. 82–120. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780122796715500071
[52] C. T. Brownlees and G. M. Gallo, "Financial econometric analysis at ultra-high frequency: Data handling concerns," *Comput. Statist. Data Anal.*, vol. 51, no. 4, pp. 2232–2245, 2006.
[53] O. E. Barndorff-Nielsen, P. R. Hansen, A. Lunde, and N. Shephard, "Realized kernels in practice: Trades and quotes," *Econ. J.*, vol. 12, no. 3, pp. C1–C32, 2009.
[54] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras
[55] T. G. Andersen and T. Bollerslev, "Answering the skeptics: Yes, standard volatility models do provide accurate forecasts," *Int. Econ. Rev.*, vol. 39, no. 4, pp. 885–905, 1998.
[56] O. E. Barndorff-Nielsen, P. R. Hansen, A. Lunde, and N. Shephard, "Designing realized kernels to measure the ex post variation of equity prices in the presence of noise," *Econometrica*, vol. 76, no. 6, pp. 1481–1536, 2008.
[57] J. Jacod, Y. Li, P. A. Mykland, M. Podolskij, and M. Vetter, "Microstructure noise in the continuous case: The pre-averaging approach," *Stochastic Processes Appl.*, vol. 119, no. 7, pp. 2249–2276, 2009.
[58] K. Christensen, R. C. A. Oomen, and M. Podolskij, "Fact or friction: Jumps at ultra high frequency," *J. Financial Econ.*, vol. 114, no. 3, pp. 576–599, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304405X14001548
[59] O. Barndorff-Nielsen, S. Kinnebrock, and N. Shephard, "Measuring downside risk—Realised semivariance," in *Volatility and Time Series Econometrics: Essays in Honor of Robert Engle*. Oxford Scholarship Online, 2010.
[60] O. E. Barndorff-Nielsen and N. Shephard, "Power and bipower variation with stochastic volatility and jumps," *J. Financial Econ.*, vol. 2, no. 1, pp. 1–37, 2004.
[61] O. E. Barndorff-Nielsen and N. Shephard, "Econometric analysis of realized volatility and its use in estimating stochastic volatility models," *J. Roy. Stat. Soc. B*, vol. 64, no. 2, pp. 253–280, 2002.
[62] O. E. Barndorff-Nielsen and N. Shephard, "Econometrics of testing for jumps in financial economics using bipower variation," *J. Financial Econ.*, vol. 4, no. 1, pp. 1–30, 2006. doi: 10.1093/jjfinec/nbi022.

ADAMANTIOS NTAKARIS received the B.Sc. degree in mathematics from the Aristotle University of Thessaloniki, in 2009, and the M.Sc. degree in financial modeling and optimization from The University of Edinburgh, in 2014. In 2014, he completed an industrial placement at Standard Life Investments, Edinburgh. Before commencing the Ph.D. degree, from 2014 to 2016, he was an Effective Interest Rate Analyst with CitiGroup Investment Bank, Edinburgh, and from 2010 to 2013, as a Math Olympiad Coach in Thessaloniki. He is currently an Early Stage Researcher within the Marie Curie BigDataFinance training network at the Department of Signal Processing, Tampere University of Technology.

GIORGIO MIRONE received the B.Sc. degree in economics and finance from the Universitat de Barcelona and The University of Melbourne, the M.Sc. degree in finance from the Universitá degli Studi di Siena, and the Ph.D. degree from in economics and business economics the Center for Research in Econometric Analysis of Time Series (CREATES), Aarhus University, in 2018. He held a visiting position at Oxford University with an Ass. Prof. A. B. Kock. He is currently a Quantitative Analyst with Danmarks Nationalbank and a former Early Stage Researcher within the Marie Curie BigDataFinance international training network. His research interests include econometrics, financial econometrics, volatility estimation and forecasting, statistical learning, and signal processing.

JUHO KANNIAINEN received the Ph.D. degree in quantitative finance from the Tampere University of Technology, where he is currently a Professor of financial engineering with the Faculty of Information Technology and Communication Sciences. His research agenda focused on statistical computing and data science in finance and risk management. In his research, he is not only using traditional quantitative methods but also modern data science approaches, namely complex networks techniques and machine learning methods with rich data sets. His focus in finance is on derivative pricing, financial econometrics, order book dynamics and liquidity, and financial networks. He has published finance and information technology in prestigious journals including *Review of Finance* and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. He has been coordinating two international EU projects, BigDataFinance and HPCFinance.

MONCEF GABBOUJ (F'11) received the B.S. degree in electrical engineering from Oklahoma State University, Stillwater, OK, USA, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1986 and 1989, respectively. He was an Academy of Finland Professor, from 2011 to 2015. He is currently a Professor of signal processing with the Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland. He guided 40 Ph.D. students and has published 650 papers. His current research interests include multimedia content-based analysis, indexing and retrieval, machine learning, nonlinear signal and image processing and analysis, voice conversion, and video processing and coding. He is a member of the Academia Europaea and the Finnish Academy of Science and Letters. He organized several tutorials and special sessions for major IEEE conferences and EUSIPCO. He is the past Chairman of the IEEE CAS TC on DSP and a Committee Member of the IEEE Fourier Award for Signal Processing. He served as an Associate Editor and a Guest Editor of many IEEE international journals and a Distinguished Lecturer for the IEEE CASS.

ALEXANDROS IOSIFIDIS (SM'16) held Postdoctoral Researcher positions with the Tampere University of Technology, Finland, and the Aristotle University of Thessaloniki, Greece. He is currently an Associate Professor in electrical and computer engineering with Aarhus University, Denmark. He has contributed more than ten R&D projects financed by EU, Greek, Finnish, and Danish funding agencies and companies. He has coauthored 55 articles in international journals and 78 papers in international conferences proposing novel machine learning techniques and their application in a variety of problems.

Dr. Iosifidis served as an Officer of the Finnish IEEE Signal Processing-Circuits and Systems Chapter, from 2016 to 2018. He received prestigious awards, including the Academy of Finland Postdoc Fellowship and the H. C. Oersted Forskerspirer Prize for research excellence at a young age. He served as an Area Chair for the IEEE ICIP, in 2018 and 2019, EUSIPCO 2019, and the Technical Program Chair for the IEEE ICASSP 2019. He has coorganized special issues/sessions in international journals/conferences focusing on topics of machine learning and big data analysis. He is currently serving as an Associate Editor for the IEEE ACCESS and *Neurocomputing* journals, and an Area Editor for the *Signal Processing: Image Communication* journal.