

Optimizing the Output of Long Short-Term Memory Cell for High-Frequency Forecasting in Financial Markets

Adamantios Ntakaris*, Moncef Gabbouj[†], *Fellow, IEEE*, and Juho Kannianen[†]

*Business School, University of Edinburgh, Edinburgh, EH8 9JS, UK

[†] Department of Computing Sciences, Tampere University, Tampere, 33720, Finland

Abstract—High-frequency trading requires fast data processing without information lags for precise stock price forecasting. This high-paced stock price forecasting is usually based on vectors that need to be treated as sequential and time-independent signals due to the time irregularities that are inherent in high-frequency trading. A well-documented and tested method that considers these time-irregularities is a type of recurrent neural network, named long short-term memory neural network. This type of neural network is formed based on cells that perform sequential and stale calculations via gates and states without knowing whether their order, within the cell, is optimal. In this paper, we propose a revised and real-time adjusted long short-term memory cell that selects the best gate or state as its final output. Our cell is running under a shallow topology, has a minimal look-back period, and is trained online. This revised cell achieves lower forecasting error compared to other recurrent neural networks for online high-frequency trading forecasting tasks such as the limit order book mid-price prediction as it has been tested on two high-liquid US and two less-liquid Nordic stocks.

Index Terms—Limit order book, high-frequency trading, long short-term memory cell, online learning, stock forecasting.

I. INTRODUCTION

HIGH-FREQUENCY trading (HFT), which represents more than 50% of the trading activity in the US stock market, is the process where signals and trades are analyzed and executed in a fraction of a second (double-digit nanoseconds). This speed race creates several opportunities for the participants who can capitalize on their infrastructure, like fiber cables and microwave signals, and also on their scientific and technical expertise. One way to organize this trading activity is the so-called limit order book (LOB), where market participants, including liquidity providers and liquidity takers, form its order flow dynamics. The liquidity providers typically play a more active role in the trading environment. This symbiotic relationship between these two parties creates competition based on the order flow information analysis and trading strategy execution. In this paper, we focus on the first part, and more specifically on real online forecasting models for

the prediction of the LOB’s mid-price movements, which is the main objective of our experimental protocol.

In the HFT LOB universe [1]–[4], every forecasting task poses challenges related to digesting large data sizes (several million order book events per trading day) and inspecting time irregularities. These two factors create a non-linear environment that can be handled properly by specific types of neural networks (NN). A NN type that has exhibited good predictive power is a variant that belongs to the recurrent neural network (RNN) family, named long short-term memory neural network (LSTM) [5], which filters current and look-back/past information of the input data. The effectiveness of LSTMs is quite extensive in several fields such as in acoustics [6]–[8], natural language processing [9]–[11], biology and medicine [12]–[14], and computer vision [15]–[17] among others. One particular area of research, where LSTMs have been applied extensively, is finance and algorithmic trading [2], [18]–[23].

An additional line of challenge in building real online machine learning experimental protocols is the development of dynamically-adjusted NNs that are suitable for short training based on fewer training epochs. To address the challenge of creating real online forecasting models following the RNN architecture we have to better understand the internal computational processes of their cells, such as the LSTM cell. The LSTM cell is a wrapper system that contains several gates and states that filter the input vector(s). These gates and states are combinations of activation functions that act as the source of information for the next LSTM cell. The order of these calculations is fixed, stale during training/learning, and their position within the cell is not justified by any specific metric. That means that the LSTM cell and the LSTM NN in general exhibit a static behavior against a dynamic and ultra-fast trading environment. A critical question is whether autonomous and dynamic cell intervention yields superior and more computationally efficient results.

In this paper, we tackle these challenges by presenting a truly dynamic and shallow NN topology, achieved by rearranging and selecting the optimal LSTM cell’s final output online. This is accomplished by online evaluation

of the importance of the internal gates and states. This internal within-the-LSTM-cell feature importance mechanism has been developed around two critical components: a simple optimization method that follows the gradient descent (GD) learning algorithm and a non-forecasting supervised regression problem that has similar but not the same targets/labels as the main objective of our experimental protocol. The GD method and the non-forecasting supervised regression problem are combined internally and the optimized outcome is passed to the next LSTM cell. Since we capitalize on the existing LSTM cell architecture without adding, removing, or changing the core LSTM cell calculations, we name our cell architecture as Optimizable Output LSTM (OPTM-LSTM) cell.¹

The rest of this paper is organized as follows. In Section II, we expand on related work about different RNN formations. In Section III, we provide the technical details of the OPTM-LSTM cell and in Section IV we provide the experimental details and the comparative performance results. Section V, summarizes our findings and discusses limitations and future lines of research.

II. RELATED WORK

So far, RNNs, including LSTM NNs with various topologies, have been widely used in LOB research for a range of forecasting tasks. Specifically, [26]–[29] have employed RNNs, alongside other ML and DL techniques, for predicting mid-price movements. The capability of RNNs to accurately forecast future ask and bid movements has been emphasized in [30]. The predictive performance of LSTMs was further improved by integrating an attention mechanism, as detailed in [31]. Additionally, earlier [32] merged LSTMs with an attention mechanism for forecasting stock price jumps. More recently, LSTMs have been adopted for devising market-making strategies, as shown in [33].

The LSTM cell architecture is based on a pre-defined sequence of operations that were set based on ad-hoc assumptions about the ordering of the internal calculations, the number of gates/states, and the cell’s output information. This is a common approach for other RNN variants such as standard recurrent cells, gated recurrent units (GRU), and several other LSTM variants. Outside the HFT LOB universe, only a limited number of studies have adjusted the internal gates in accordance with the information flow, albeit with some shortcomings. In one study, over ten thousand RNN architectures were evaluated, revealing that a combination of certain calculations, similar to GRU development, yielded slightly better performance compared to the standard LSTM NN [34]. However, their results are not suitable for HFT LOB forecasting tasks. The extensive topological grid search is time-consuming until a suitable sequence and type

of linear algebra calculations have to be used. Another problem is that the re-arrangement or the elimination of the RNN’s gate(s) is not directly connected to the final objective(s). That means that the RNN’s weights will be updated first on the full batch (or mini-batch) sequence and then a different architecture will be tested. On a different note, variants of RNN cells that exhibited good predicting performance have been suggested, but their architecture remained unchanged when the training process started. For instance, [35] suggested three simplified LSTM cells with similar or better performance to the original LSTM cell structure. In the same fashion, [36] and [37] proposed modifications to the most important LSTM’s cell gates. Lighter RNN cells are also suggested in [38]–[42].

More complicated structures were also proposed. For instance, in [43], an additional term was attached to several internal LSTM gates, known as peephole LSTM NN. Other variants of such complicated cells can be found in [44]–[49]. Few additional RNN variants exist, such as bidirectional LSTM, which were utilized effectively in [50] for predicting the open, high, low, and close stock prices. Similarly, bidirectional LSTMs architectures were employed for daily stock price forecasting in [51] and [52]. An additional line of research of LSTMs is their use as hybrid models. There are several examples, such as [53], [54], [55], and [56] that combined with convolutional neural networks (CNN) for the task of stock price forecasting. These models, despite their forecasting efficacy, are not agile in terms of high-paced information flow analysis since the cells remain stale during the training process and are disconnected from their predicting task.

One common approach for the training process is the use of gradient-based optimizers such as the Adam optimizer that has $O(N)$ computational complexity per iteration, where N is the number of parameters. This makes it apt for large-scale, high-dimensional models, facilitating rapid convergence critical for processing high-frequency trading data [57]. Authors in [58] suggested an alternative to gradient-based optimization methods based on Particle Swarm Optimization (PSO). Despite the effectiveness of this approach, the PSO algorithm exhibits a computational complexity of $O(M \times F)$, where M denotes the number of particles and F the complexity of the objective function evaluation. This presents a significant challenge in terms of trade execution within the context of ultra-high frequency data. This challenge is corroborated by findings in broader optimization literature, where PSO’s performance is sensitive to factors like swarm size and function complexity, see [59], [60].

III. PROPOSED METHOD

The dynamics of HFT LOB is ultra-fast, and therefore, the LSTM NN needs to be ready to identify, in a fraction of a second, these dynamics and provide the most optimized

¹The coding part is based on TensorFlow [24] and Keras [25] libraries and our code can be found at <https://github.com/DeQmE/OPTMLSTM/tree/main>.

suggestion/information. This we aim to tackle by improving the original mechanics of the LSTM cell. The cell, as part of the LSTM NN, so far has been over-engineered, either by adding or removing complexity in terms of the number and order of its internal states and gates. For this reason, we propose a real data-driven and adaptive LSTM cell architecture, named OPTM-LSTM cell, that is different from the existing RNN cell mechanism in two key areas:

- LSTM’s cell gates and states are treated as features and we measure their ability to handle the online information flow via an internal feature importance mechanism. More specifically, the LSTM cell is equipped with this internal within-the-cell feature importance mechanism which acts as a non-forecasting supervised regression problem. That means that this non-forecasting supervised regression relies on labels that represent the current observable LOB’s mid-price and act as an indicator for the importance of the internal LSTM’s cell gates and states. The indicator will attach trained weights, via the GD learning algorithm, to every LSTM’s cell internal gates and states.
- The indicator mechanism, which is an internal block within the LSTM cell, operates based on labels that represent the current observable LOB’s mid-price. By this way, the OPTM-LSTM cell is equipped with a robust internal instructing method that is connected to the overall forecasting objective. In other words, this instructing method is using a lagged version of the labels that are utilized in the actual prediction problem. Therefore, we have two supervised regression problems: the first one is part of the indicator mechanism within the OPTM-LSTM cell to optimally select the best gate and the other one a part of the actual forecasting problem, which, in our case is the prediction of the next LOB mid-price.

This way, we do not add or remove any arbitrary calculations within the LSTM cell but instead, we capitalize on its existing formation. Thus, we can train fast without any extensive grid search calculations. The cell will be updated constantly by every incoming trading event with the present mid-price. That means that the internal gates and states will be evaluated according to the GD method with respect to the mean squared error (MSE) and this mechanism will measure the importance of gates and states. Before diving into the method’s specifics, let’s first define the LSTM cell and provide background information regarding its learning and training processes.

In Fig. 1 we can see that every LSTM cell considers only one time-step input. An example of this input can be just the present full LOB level data (i.e., the continual operation of a minimal time-step input length) or the current and previous full LOB levels data (i.e., a sliding

window based on larger historical input LOB data). These time-step inputs are sequential order book events of a specific length and we can call it a look-back period. The look-back period will determine whether the number of sequential LSTM cells can retain any useful information in their memory. The LSTM’s cell memory is based on nine components, four internal gates, four states, and one input feature vector, which are:

- $f_t \in \mathbb{R}^{1 \times U}$, forget gate
- $i_t \in \mathbb{R}^{1 \times U}$, input gate
- $\tilde{c}_t \in \mathbb{R}^{1 \times U}$, cell input gate
- $o_t \in \mathbb{R}^{1 \times U}$, output gate
- $c_t \in \mathbb{R}^{1 \times U}$, cell state
- $h_t \in \mathbb{R}^{1 \times U}$, hidden state
- $h_{t-1} \in \mathbb{R}^{1 \times U}$, previous hidden state
- $c_{t-1} \in \mathbb{R}^{1 \times U}$, previous cell state
- $x_t \in \mathbb{R}^{1 \times D}$, input vector,

where the first dimension of the tensors above represents the current or the previous time-step of the look-back period, U are the number of LSTM units (e.g., under the TensorFlow framework), and D the number of input features. We also include the bias terms b_f , b_i , $b_{\tilde{c}}$, and $b_o \in \mathbb{R}^{1 \times U}$ (which are activated by default in TensorFlow). Each of these gates and states represents the following transformations:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

$$\tilde{c}_t = \tanh(W_{\tilde{c}} x_t + U_{\tilde{c}} h_{t-1} + b_{\tilde{c}}) \quad (3)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$

$$h_t = o_t \odot \tanh(c_t), \quad (6)$$

where σ is the sigmoid function, \tanh is the hyperbolic tangent function, the recurrent weights W_f , W_i , $W_{\tilde{c}}$, W_o , U_f , U_i , $U_{\tilde{c}}$, and U_o are multiplied with their corresponding input x_t and previous hidden state h_{t-1} vectors based on the dot product algebraic operation, and finally, the \odot algebraic operation is the Hadamard product. The first dimension of the tensors represents the input time-step, which in our case will be one, otherwise it will be equal to the selected batch size. We need to mention here the importance of the relationship between the batch size and the look-back period in an online learning experimental protocol. The batch or mini-batch setting updates the NN weights with a lag equal to the length of the look-back period and this creates several contradictory facts about the online training scenarios. These scenarios are the sliding window or the continual learning approaches. More specifically, if the length of the look-back period

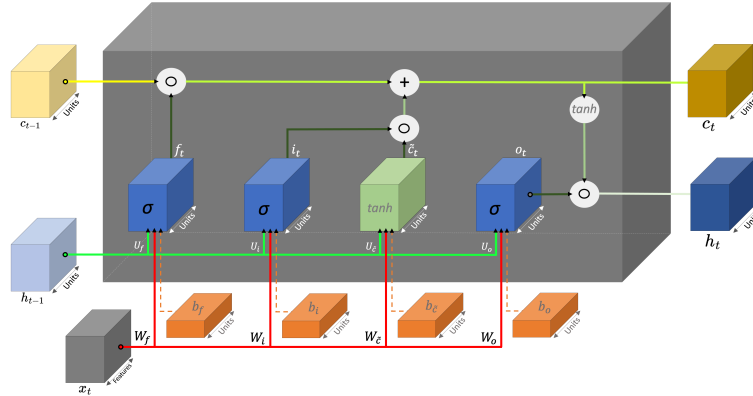


Fig. 1: LSTM Cell: This is a three-dimensional interpretation of the core LSTM cell operations. There are four internal gates: f_t , i_t , \tilde{c}_t , and o_t . Moreover, there are previous states, c_{t-1} and h_{t-1} , and next states, c_t and h_t . There are three input tensors and these are: the feature vector $x_t \in \mathbb{R}^{1 \times d}$, which represents just one time-step of the look-back period and the full range of the feature space, the previous LSTM cell output $h_{t-1} \in \mathbb{R}^{1 \times U}$, where its dimensions stem from the previous time-step and the number of LSTM units (the units are based on Tensorflow's notation where the ML user usually selects 8, 16, 32, 64, 128, 256 or 512 units), and the third input is the previous cell state $c_{t-1} \in \mathbb{R}^{1 \times U}$ with a similar dimension profile as the previous hidden state h_{t-1} . Note: The LSTM cell is designed in a three-dimensional space. More specifically, every gate and state are presented as three-dimensional tensors but this is only for convenience to demonstrate the different building blocks. In reality, these are two-dimensional tensors with a batch size of 1.

is selected based on some experimental criteria and not randomly, then the batch or mini-batch size will be less important and only then the attention will be paid to the topology of the LSTM NN. This is the main motivation that enables us to scrutinize the LSTM cell topology under an online continual learning protocol.

The main idea of our OPTM-LSTM cell is to enable the ML trader to focus only on the selection of the look-back period and discard the selection of the batch or mini-batch size per training iteration. This can happen by creating a dynamically adjusted LSTM cell that can capitalize on its existing internal gates and states and process the information flow without any lags. In Fig. 2 we see an overview of the OPTM-LSTM cell. This cell contains the same number of gates and states as the original LSTM cell with the only difference that just before the generation of the two output tensors, which are hidden states and cell states at time t , we employ a feature importance mechanism based on an internal non-forecasting supervised regression which acts as a feature importance indicator. The indicator considers only the current and already known mid-price and it will highlight the most important feature among f_t , i_t , \tilde{c}_t , o_t , c_t , and h_t . This mid-price will be the label/target for the internal non-forecasting supervised problem and it will define the importance/order of the LSTM's cell states and gates. This importance mechanism is utilizing the GD as the learning weights optimizer.

The internal supervised problem is not a forecasting problem but a calibration problem. That means that the provided labels represent the current LOB mid-price. This

approach offers a robust calibration of the feature repository (or else Feature Repo in Fig. 2) selection mechanism based on GD. The GD learning algorithm will converge after a few iterations (we tested that seven to ten iterations suffice) with the input data vector being the current full LOB data state. The feature repository then extracts the best gate or state and this will be the final/optimal output named hidden state h_t . The cell state c_t stays intact. We observed that the replacement of the cell state c_t by the second most important feature according to the MSE score did not exhibit significant improvements to the overall regression objective outside the OPTM-LSTM cell, which, in our case, is the prediction of the next mid-price.

A closer look inside the feature repository (Feature Repo) block can be seen in Fig. 3. The Feature Repo block contains several critical components which are:

- the collection of the six internal gates and states,
- the gradient weights which are updated based on the GD learning algorithm,
- the labels that reflect only the current mid-price for non-forecasting regression problem,
- and the input time-step tensor x_t .

The combination of these components will highlight which state or gate is the most important feature by averaging the highest price in terms of the learned weights per internal gate or state. We need to mention that the GD weights are not part of the backpropagation through time training method (BPTT) - the mechanism that optimizes the LSTM's trainable parts W_f , W_i , $W_{\tilde{c}}$, W_o , U_f , U_i , $U_{\tilde{c}}$, and U_o - see Appendix for a detailed derivation of the BPTT process.

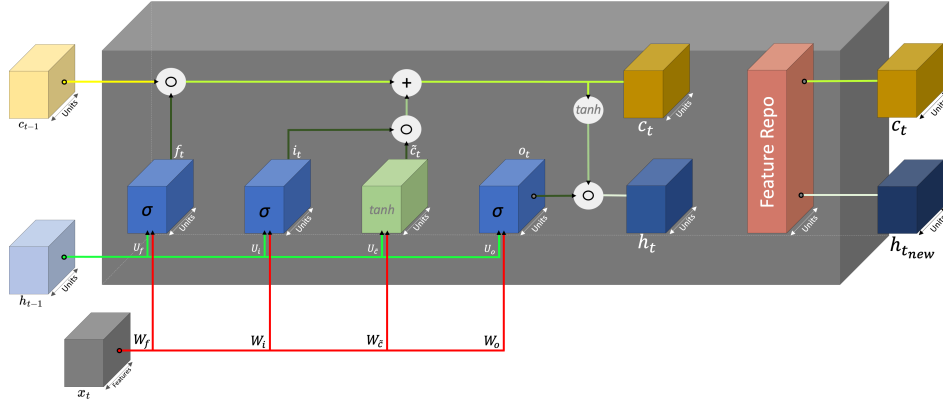


Fig. 2: OPTM-LSTM Cell: The optimizable output LSTM cell mirrors the LSTM prototype in the number of internal gates, states, and outputs. The key difference is that before releasing the hidden and cell state tensors, we group all internal gates and states, perform a feature importance calculation using a non-forecasting supervised regression task, and select the most important feature based on the lowest MSE. The graph omits biases to avoid visual clutter. Note: The OPTM-LSTM is designed in a three-dimensional space, with gates and states as three-dimensional tensors for illustrative purposes, but in practice, they are two-dimensional tensors with a batch size of 1.

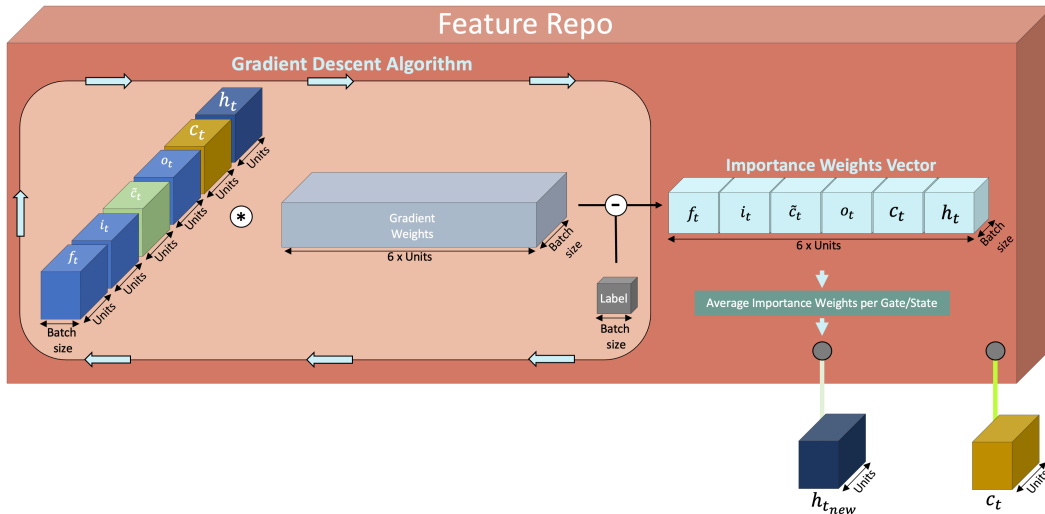


Fig. 3: From left to right: The Feature Repo, a component of the OPTM-LSTM cell, determines which state or gate replaces the default LSTM cell output. It operates in three stages: first, the GD method processes features (LSTM cell states and gates) and the known mid-price as a non-forecasting label, using matrix multiplications and vector differences. After iterations, GD converges, producing an importance vector of learned weights, stored for the next phase. In the second phase, outside the GD algorithm, the importance vector provides weights for each LSTM state and gate, with each having $1 \times U$ weights. By averaging these, we identify the most important state or gate. In the final stage, the OPTM-LSTM replaces the default h_t state with $h_{t_{new}}$, which may be $f_t, i_t, \tilde{c}_t, o_t, c_t,$ or h_t , while the final cell state c_t remains unchanged, as replacing it yielded no significant forecasting improvements. Note: The Feature Repo is designed in a three-dimensional space, with gates, states, gradient weights, labels, and importance vectors as three-dimensional tensors for illustration, but in practice, they are two-dimensional tensors with a batch size of 1.

The mechanics of the Feature Repo block are expressed mathematically based on the following three steps:

- 1) For the input $x_t \in \mathbb{R}^{1 \times D}$, where the first dimension represents the batch size and D the number of input features (e.g., current LOB state) and U the number of hidden units, we concatenate all the internal LSTM gates and states which are $f_t, i_t, \tilde{c}_t, o_t, c_t$, and h_t , in one vector $r_t \in \mathbb{R}^{1 \times C}$, where $C = 6 \times U$ is the total number of the most recently calculated internal gates and states in relation to the number of hidden units, as follows:

$$r_t = [f_t \mid i_t \mid \tilde{c}_t \mid o_t \mid c_t \mid h_t]. \quad (7)$$

- 2) For the non-forecasting label $y_t \in \mathbb{R}^{1 \times 1}$ that represents the present and already known mid-price and the vector r_t the iterative process of the GD algorithm unfolds, as described in Algorithm 1 with the parameter $\theta \in \mathbb{R}^{C \times 1}$.

Algorithm 1 Online Gradient Descent Algorithm

Require: Learning rate α , number of iterations I , initial parameter θ , non-forecasting labels y_t

- 1: **for** $i = 1$ to I **do**
- 2: Compute the predicted labels $\hat{y}_t = \theta^T \cdot r_t$
- 3: Compute the *error* = $\hat{y}_t - y_t$
- 4: Compute the gradient $\nabla J(\theta) = 2 \cdot (r_t^T \cdot \text{error})$
- 5: Update the parameter $\theta \leftarrow \theta - \alpha \cdot \nabla J(\theta)$
- 6: **end for**

Ensure: The optimized parameter vector θ

- 3) The trained parameter θ is the *Importance Weights Vector* or IWV (see Fig. 3) and it will be partitioned based on the six gates and states. Each of the components of this partition represents one of the six gates and states with respect to the number of hidden units U . We then calculate the average importance per component AI_c , as

$$\text{AI}_c = \frac{1}{U} \sum_{j=(c \times U) - (U - 1)}^{c \times U} \text{IWV}_j, \quad (8)$$

where $c = 1, 2, 3, 4, 5, 6$ and then we select the component with the highest AI_c value. This component will be the optimum output (or $h_{t_{new}}$) of the OPTM-LSTM cell.

A. Complexity

The LSTM cell is very efficient in terms of time and space complexity. More specifically, the LSTM cell is local in time and space, which means that every update per timestep is independent of the input sequence length. LSTM's cell time and space complexity per timestep is $O(W)$ where W is the number of its parameters and it is equal to $W = 4 \times U^2 + 4 \times U \times I + U \times O + 3 \times U$, where U is the number of hidden cells, I is the number of input

units, and O is the number of the output units. The newly introduced OPTM-LSTM cell contains, in terms of computational complexity, an additional term that represents the GD's online updates. As a result, OPTM-LSTM's time and space complexities are equal to $O(W + N \times D)$ and $O(W + N)$, respectively, where $D = 6 \times U$ and N is the number of input sample points.

The input sample points are based on a two-dimensional tensor with dimensions equal to *the number of hidden cell units* \times *the six gates/states* for the first dimension and *the number of LOB features* for the second dimension. An additional critical point is that GD's complexity analysis is based on a batch size with a single item since the OPTM-LSTM performs online reading of the input data. An additional consideration for the OPTM-LSTM complexity analysis is the complexity (time and space) of the backward pass, which is based on the BPTT (see Appendix) and has $O(N^2 \times L)$ and $O(N \times L)$ time and space complexities, respectively, with N being the number of network units and L being the full timestep length. The BPTT is clear from the GD step since the Feature Repo block (see Fig. 2) is not participating in the backward learning process.

B. Motivation

Now that we have a deeper understanding of what the OPTM-LSTM does, we will provide the motivation behind the development of the optimizable output LSTM cell. Two key components led us to the creation of the OPTM-LSTM, one is theoretical and the other one is practical:

- *Theoretical:* The main advantage of the LSTM architecture is to retain the provided information and delay (or avoid) the vanishing and exploding gradients. If we want to reduce even further the probability of the vanishing (or exploding) gradients we have to pay attention to the derivative of the cell state \tilde{c} at the trading event t (see Eq. 38 in Appendix) which contains the forget gate with a range between 0 and 1. The vanishing and exploding gradient problem refer to the case where the RNN/LSTM during the BPTT process is governed by small or large derivative elements – a process that is directly connected to the length of the input sequence. As a result, the forget gate just slows down [61], but does not eliminate the vanishing of gradients phenomenon, where the rest of the cell state's gradients are responsible for the exploding gradients. There are always methods that can hold the gradients within boundaries (e.g., gradient clipping [62] or restricted initialization [63]), but these methods are not native to the LSTM cell and further experimental calibration is required. The OPTM-LSTM by shuffling strategically LSTM's cell final output delays (or avoids) even further the possibility of uncontrolled gradients.

- *Experimental*: While investigating the information filtering of the internal gates and states of an LSTM cell, we found that they exhibited a remarkable characteristic with respect to their importance. By extracting the internal states and gates from their LSTM cell per trading event and treating them as trained features, we noticed a constant feature importance alternation while trying to forecast LOB’s mid-price target price in an online manner. More specifically, we extracted LSTM’s cell components and we considered them as trading signals. Instead of relying on the full input feature vector length, we performed a feature importance approach based on the GD learning algorithm. The result was that the feature importance frequency alternation was constant and independent of the stock profile. Based on that evidence, we developed the Feature Repo block (see Fig. 3) within the existing LSTM cell.

IV. EXPERIMENTS

In this section, we present details and performance results of the OPTM-LSTM cell based on an online HFT forecasting experimental protocol. In this protocol, we predict the next value of the mid-price. The OPTM-LSTM competes against several key RNN-based and temporal-based developments such as standard LSTM, bidirectional LSTMs, LSTMs with an attention layer², GRUs, a hybrid model combining LSTM and CNN, Temporal Convolutional Networks (TCN) [64] and Bootstrap Fast Ensemble Empirical Mode Decomposition LSTM Twin Support Vector Regression Seeker Optimization Algorithm (BFEEMD-LSTM-TWSVRSOA) [69] adjusted for our financial time series forecasting task. Finally, we use four benchmark models such as naive/baseline regressor, the persistence model that predicts the next mid-price value to be the same as the current value, autoregressive integrated moving average (ARIMA) model and Error, Trend, and Seasonality (ETS) model. Our aim is to compare a broad range of common RNN variants from the literature, applicable to our online forecasting regression task.

The motivation of this extended experimental protocol is to see if other deep learning models, like CNNs, are capable of effectively dealing with the sequential nature of our data. Additionally, the protocol seeks to determine if these models can also understand the hierarchical order of the input features, such as those found in LOB data, if there is any. Note that it includes not only common

²The primary distinction between the Feature Repo Block and the attention layer lies in their roles and applications with respect to the LSTM gates (and states) and the time series data. More specifically, the Feature Repo Block reorganizes the importance of the LSTM gates and states, whereas the attention layer is applied directly on the time series data. The main role of the Feature Repo Block is to select the optimal LSTM gate (or state), while the attention layer is a broader technique for focusing on different parts of input data across various neural network architectures.

RNN architectures but also composite models, such as the hybrid model. Considering the variability and noise that LOB features may introduce, we also incorporated Dropout layers into our experimental setup to mitigate overfitting.

The experimental section unfolds as follows: Initially we introduce some general concepts about the input data based on HFT LOBs, and then we describe details of the online experimental protocol and we conclude the section based on tables with performance measures in terms of MSE scores. We present MSE score results based on raw data and two normalization settings, MinMax and Zscore – two methods that are based on the minimum and maximum values and zero mean and unit standard deviation per input, respectively.

A. Naive baseline models

The naive baseline regressor is based on the idea that the targets in the training set will be a constant value and that value will be utilized for the MSE calculation for the targets in the test set. The inputs to this naive model are the full LOB data but they do not have any contribution to the forecasting process. The second baseline model, so-called persistence algorithm, is developed according to a forecasting function that predicts that the next target price is the same as the current target price $S_{t+1} = S_t$, where S represents the target price, which we call flat prediction. The input values to this model will be just the mid-price that we will formally define in the following section.

B. HFT LOB

LOB is a vital component for the ML trader that wants to extract information about the interplay between liquidity and price for both the ask and bid sides. Bid and ask sides are divided into several price levels that change asynchronously, and in this paper, we use the best 10 LOB levels both side. In terms of the current trading activity, the most aggressive prices are the best ask and bid prices that form the so-called spread, which is the distance between the highest bid and the lowest ask price. Averaging these best ask and best bid prices we form the mid-price (MP), which is an artificial indicator, but can be utilized as a sensitive proxy on a price level for forecasting tasks. The mid-price is extracted for every trading event. Formally, the mid-price at trading event t can be calculated as

$$MP_t = \frac{P_{A_t} + P_{B_t}}{2}, \quad (9)$$

where MP_t is the mid-price at the trading event t , P_{A_t} and P_{B_t} are the best ask and bid prices, respectively. A critical point here is the perception of the trading event t . More specifically, the time perception in the HFT space should be avoided and instead, only the trading events should be considered as measurements of sequential trading progression in an intra-day trading setting. This is due to the time-frequency abnormalities in the HFT universe,

that is trading events that arrive simultaneously or exhibit extensive trading inactivity for several milliseconds. To tackle these time irregularities we perceive every trading event as independent information that is part of the LOB inventory. LOB’s current ask and bid price levels and their corresponding inventory states will be the forces that will trigger the changes in the OPTM-LSTM output information. That information will be utilized to predict the next mid-price at the trading event $t + 1$.

C. Experimental Protocol and Datasets

Our experimental protocol aims to forecast the next LOB mid-price using tick-level HFT data without information lag, treating it as an online regression task that processes every trading event in the LOB without data sampling, relying solely on the latest LOB state. Within the OPTM-LSTM cell, an additional automated non-forecasting regression task ranks the importance of internal gates and states, while the ML trader focuses on the forecasting task. The protocol uses HFT LOB datasets from NASDAQ’s ITCH protocol, which ensures ultra-low latency and advanced visibility, covering two high-liquid US stocks (Amazon, Google) for the first two trading months of 2015 and two less-liquid Nordic stocks (Kesko, Wartsila) from June 1 to July 31, 2010. To prevent memory loss from decimal points, LOB price levels were multiplied by 10,000.

We implement a progressive training scenario using up to 20,000,000 trading events for high-liquidity US stocks and 5,000,000 for less-liquid Nordic stocks, each representing about two months of trading data, as model performance typically plateaus or declines beyond these sizes with ultra-high frequency data. For testing, we use 1,000 trading events, evaluated progressively, as their duration varies from simultaneous arrivals to events up to eight minutes apart. In this framework, each of the 1,000 testing events becomes the latest training event for the next forecasting step, with the training set sequentially absorbing each new testing event. Testing performance is assessed by averaging a sequentially-stored MSE score at the end of the testing period.

The performance results of the models stem from a comprehensive grid search of topology and hyperparameters to optimize MSE scores. The search was capped at four hidden layers for the prototype LSTM, LSTM with attention, bidirectional LSTM, GRU, and LSTM-CNN hybrid models, and two hidden layers for the OPTM-LSTM model, due to the extensive topological grid search required in an HFT setting for the five RNN competitors. The grid search included various combinations of hidden units, optimizers, Dropout levels, look-back periods, and batch sizes. For OPTM-LSTM, limitations were based on achieving a lower MSE score than the five RNNs, using an early stopping mechanism. We report in detail the optimal topologies in Section IV-D.

The results reported progressively based on several data size scenarios. The motivation for this type of performance reporting is that HFT requires ultra-fast data digestion/learning and we need to know what is the optimal minimum data size per model. Based on the concept of ultra-fast data digestion/learning we provide an additional layer of reporting which is related to the number of training epochs and we name them Long Training, which corresponds training up to 60 epochs, and Short Training, which corresponds training up to five epochs. The Long Training and the Short Training settings will give us an insight into every model’s behaviour with respect to data requirements and learning time.

Finally, to evaluate the effectiveness of the OPTM-LSTM model, we present MSE scores derived from raw data across various data size scenarios and two normalization settings, MinMax and Z-score. These are tailored to specific and crucial data size scenarios for MSE performance. The evaluation considers two distinct input feature sets: the first set includes the complete LOB data comprising 40 features, which encompasses both ask and bid price levels along with their corresponding volume levels; the second set focuses solely on the mid-price. These two different features groups are suitable for employing two naive regressors such as the naive/baseline regressor based on the full LOB data input and the persistence algorithm based only on the mid-price for a flat tick-by-tick forecasting comparison against the five RNN competitors and the OPTM-LSTM model. We conclude our experimental setting with a self-comparison of the OPTM-LSTM performance. That means that we compare the Long Training with the Short Training MSE scores for our model and check its ability to learn fast and based on smaller datasets. We complement our result tables with plots for a visual interpretation of our findings.

The motivation for these two progressive training settings is to investigate whether the OPTM-LSTM cell is robust across higher and lower trading volume stock examples and also to check whether it provides better performance compared to the other RNN developments. Additionally, this two-phased experimental protocol aims to address a critical question for the HFT ML trader: can a small dataset, when trained over a high number of epochs, compete with a larger dataset that has undergone less training? The restrictions in the previous questions arise because the HFT ML trader needs the fastest forecasting ability, which corresponds to the point where the model’s performance plateaus, along with the shortest possible training time, which involves using the minimum optimal data size. In simple terms, we want to cover the minimum distance between the model’s performance, input data size, and the number of epochs.

D. Hyperparameters

We employ several competitors against our new cell and we curate them based on a wide and fully-automated topological grid search range. That means that we run an extensive topological grid search for these models and report the best candidates in terms of the lowest MSE score. The reported models consider the following as hyper-parameters: the depth which is limited to four hidden layers, the number of hidden units that varies from 8 to 512, the type of optimizer (e.g., Adam, Nadam, RMSProp, and Stochastic Gradient Descent or SGD), the batch size which varies from 1 to 64, the use (or not) of Dropout percentage (e.g., 0.20 or 0.50), and a look-back period of 1, 5 or 10 past trading LOB states. Our OPTM-LSTM model hyper-parametrization considers the following, as hyperparameters: the depth is limited to two hidden layers, the number of hidden units varies from 4 to 512, and the type of optimizer (e.g., Adam, Nadam, RMSProp, and SGD). Our model considers a batch size and a look-back period of 1. We need to mention that the optimal models' topologies are stock specific and that can be seen in Table I below and Table IV in Appendix. The five RNN models and the baseline/naive models can be found in the tables as: 'LSTM' for the prototype LSTM, 'Attention' for the LSTM with the attention layer, 'Bidirectional' for the bidirectional LSTM, 'GRU' for the GRU RNN, 'Hybrid' for the hybrid LSTM-CNN model, 'Baseline' for the naive/baseline model based on the LOB data input, and 'Persistence' for the persistence model based on the mid-price as input.

We see that the OPTM-LSTM requires fewer building blocks compared to its well-established competitors. The effectiveness of the OPTM-LSTM cell is highlighted based on the development of a two-phased experimental setting that unfolds, as follows: the first phase, based on Short Training with five epochs, our optimizable output LSTM cell competes with the best candidate models (see Table I below) via a progressive training process which incrementally goes from 1,000 trading events up to 20,000,000 trading events for the two high-liquid stocks, Amazon and Google, and from 1,000 trading events up to 2,000,000 trading events for the less-liquid Nordic stocks. The second phase is based on Long Training with 60 epochs equipped with an active Early Stopping mechanism, the same models, and the same progressive training process that goes from 1,000 to 15,000 trading events for both sets of stocks.

³

³We conducted a grid search with over 2 million simulations per model for four stocks, testing hyperparameters including up to four hidden layers, seven neuron sizes, four optimizers, three dropout rates, three look-back periods, and five batch sizes. Using the best topology for each model, we performed further experiments under Long and Short protocols, each repeated 10 times to calculate average performance, totaling over 7,000 experiments for both protocols. This resulted in more than 10 million experiments overall.

E. Results

We present our findings in terms of MSE scores for different stocks in Table II – Table XIV (see Appendix). For space economy, we present the case of Long experimental protocol for Google in Table II in the main body (apart from the Short experimental protocol and benchmark results that can be found in Appendix) of the text and the rest of the stocks can be found in Appendix. The tables are reported in the following order:

- Two MSE score tables for Short and Long protocols (5 and 60 epochs) show data size scenarios using raw LOB data for OPTM-LSTM and five RNN competitors, highlighting the minimal sample length needed for effective HFT model performance.
- Two multidimensional MSE score tables for Benchmark Training and Testing compare OPTM-LSTM, two baseline models, and five RNN competitors, using optimal minimal sample length, with LOB data and mid-price inputs under MinMax and Zscore normalization settings.

The new cell outperforms others across selected stock examples, maintaining stable performance for most data sizes in high-liquid and less-liquid stocks. For Amazon stock, the Hybrid model occasionally neared OPTM-LSTM NN's performance despite significant MSE score fluctuations, likely due to CNNs' non-linear nature causing output sensitivity to input changes (see Table IV in Appendix). In testing, OPTM-LSTM cell surpassed other models in both low- and high-epoch protocols across all stocks (see Tables II - XII in Appendix). Most models plateaued before 20,000,000 trading events for high-liquid stocks and 2,000,000 for less-liquid stocks.

OPTM-LSTM achieves lower MSE scores than competitors by instantly detecting price or volume changes via its optimizable output cell. Its simple topology, with one hidden layer, few units, minimal look-back history, and small batch size, supports efficient HFT ML trading. Despite higher per-cell computational complexity than prototype LSTM, OPTM-LSTM processes data as fast or faster due to fewer training parameters from its look-back period, layers, and units. Its internal optimization and gate/state selection occur only in the forward step, avoiding extra backpropagation costs as these parameters are trained locally, bypassing the chained partial derivation rule. Additionally, OPTM-LSTM's lower MSE scores stem from its ability to leverage LOB structure compared to other RNNs. Each LOB time instance includes current and past price and volume levels, with our experiments using LOBs of 10 price levels (10 bid and ask price/volume levels). The best levels, forming the mid-price, are more informative than deeper levels, which may hold past data and serve as a look-back period for OPTM-LSTM. While other models access the same data, their fixed, pretrained topologies may incorporate noise, limiting their effectiveness.

TABLE I: Best-performing candidates based on a topological grid search for the US stocks.

Stock	Model	Topology	Stock	Model	Topology
Amazon	LSTM	<ul style="list-style-type: none"> • LSTM layer with 32 units • Dropout 50% • Dense layer with 1 unit • RMSProp optimizer • Batch size of 32 samples 	Google	LSTM	<ul style="list-style-type: none"> • LSTM layer with 32 units • Dropout 50% • Dense layer with 1 unit • Adam optimizer • Batch size of 32 samples
	LSTM with Attention	<ul style="list-style-type: none"> • LSTM layer with 40 units • PReLU • Attention layer • Dense layer with 40 units • Dense layer with 1 unit • Nadam optimizer • Batch size of 64 samples 		LSTM with Attention	<ul style="list-style-type: none"> • LSTM layer with 64 units • PReLU • Attention layer • Dense layer with 32 units • Dense layer with 1 unit • Nadam optimizer • Batch size of 64 samples
	Bidirectional RNN	<ul style="list-style-type: none"> • Bidirectional LSTM layer with 32 units • Dense layer with 32 units • Dropout 50% • Dense layer with 4 units • Dense layer with 1 unit • Adam optimizer • Batch size of 32 samples 		Bidirectional RNN	<ul style="list-style-type: none"> • Bidirectional LSTM layer with 32 units • Dense layer with 32 units • Dropout 50% • Dense layer with 4 units • Dense layer with 1 unit • Adam optimizer • Batch size of 64 samples
	GRU	<ul style="list-style-type: none"> • GRU with 32 units • Dense layer with 32 units • Dense layer with 1 unit • Adam Optimizer • Batch size of 32 samples 		GRU	<ul style="list-style-type: none"> • GRU with 32 units • GRU with 32 units • Dense layer with 32 units • Dense layer with 1 unit • Adam Optimizer • Batch size of 32 samples
	Hybrid	<ul style="list-style-type: none"> • 1D Convolution layer with with 60 filters, 6 as kernel size • ReLU activation function • LSTM layer with 64 units with Tanh activation function • LSTM layer with 64 units with Tanh activation function • Dense layer with 30 units with ReLU activation function • Dense layer with 10 units with ReLU activation function • Dense layer with 1 unit • Nadam Optimizer • Batch size of 32 samples 		Hybrid	<ul style="list-style-type: none"> • 1D Convolution layer with with 60 filters, 6 as kernel size • MaxPooling1D • LSTM layer with 64 units with Tanh activation function • LSTM layer with 64 units with Tanh activation function • Dense layer with 30 units with ReLU activation function • Dense layer with 10 units with ReLU activation function • Dense layer with 1 unit • Nadam Optimizer • Batch size of 64 samples
OPTM-LSTM	<ul style="list-style-type: none"> • OPTM-LSTM layer with 64 units • Dense layer with 4 units • Dense layer with 1 unit • Adam optimizer • Batch size of 1 sample 	OPTM-LSTM	<ul style="list-style-type: none"> • OPTM-LSTM layer with 8 units • Dense layer with 4 units • Dense layer with 1 unit • Adam optimizer • Batch size of 1 sample 		

TABLE II: Google MSE scores under the Long experimental protocol.

Stock	Size	Model	MSE - Train	Stock	Size	Model	MSE - Train	Stock	Size	Model	MSE - Train
Google	1,000	OPTM-LSTM	7.13393E+12	Google	2,000	OPTM-LSTM	1.95355E+13	Google	3,000	OPTM-LSTM	3.62966E+14
		LSTM	1.92849E+13			LSTM	1.95598E+13			LSTM	3.69272E+14
		Attention	1.94585E+13			Attention	1.95592E+13			Attention	3.69269E+14
		Bidirectional	1.94585E+13			Bidirectional	1.97603E+13			Bidirectional	3.69270E+14
		GRU	1.94583E+13			GRU	1.96591E+13			GRU	3.69269E+14
	4,000	Hybrid	5.15419E+13	5,000	Hybrid	6.22681E+13	6,000	Hybrid	6.43307E+14		
		OPTM-LSTM	2.18488E+14		OPTM-LSTM	2.70888E+14		OPTM-LSTM	1.80689E+14		
		LSTM	2.86598E+14		LSTM	2.37052E+14		LSTM	2.04031E+14		
		Attention	2.86591E+14		Attention	2.37038E+14		Attention	2.03984E+14		
		Bidirectional	2.86589E+14		Bidirectional	2.37032E+14		Bidirectional	2.03993E+14		
	7,000	GRU	2.86549E+14	10,000	GRU	2.37033E+14	15,000	GRU	2.03987E+14		
		Hybrid	7.24462E+14		Hybrid	1.12468E+15		Hybrid	1.36507E+15		
		OPTM-LSTM	1.67232E+14		OPTM-LSTM	1.13227E+14		OPTM-LSTM	5.04511E+13		
		LSTM	1.80438E+14		LSTM	1.37966E+14		LSTM	1.16907E+14		
		Attention	1.80375E+14		Attention	1.37812E+14		Attention	8.07426E+13		
1,000	Bidirectional	1.80392E+14	Google	2,000	Bidirectional	1.37815E+14	Google	3,000	Bidirectional	1.10559E+14	
	GRU	1.80364E+14			GRU	1.37761E+14			GRU	1.70163E+14	
	Hybrid	1.89508E+15			Hybrid	1.99974E+15			Hybrid	2.19058E+15	
	OPTM-LSTM	1.02565E+15			OPTM-LSTM	2.21936E+13			OPTM-LSTM	2.42504E+13	
	LSTM	1.04581E+15			LSTM	1.05493E+15			LSTM	3.93302E+13	
4,000	Attention	1.05495E+15	5,000	Attention	1.05495E+15	6,000	Attention	3.93206E+13			
	Bidirectional	1.05495E+15		Bidirectional	1.05495E+15		Bidirectional	8.93262E+13			
	GRU	1.05494E+15		GRU	1.05493E+15		GRU	3.69269E+14			
	Hybrid	1.03878E+15		Hybrid	1.14449E+15		Hybrid	1.33994E+15			
	OPTM-LSTM	6.48726E+12		OPTM-LSTM	7.15408E+12		OPTM-LSTM	1.67019E+12			
7,000	LSTM	3.93236E+13	10,000	LSTM	3.92020E+13	15,000	LSTM	3.91175E+13			
	Attention	3.93125E+13		Attention	3.91765E+13		Attention	3.90502E+13			
	Bidirectional	3.93107E+13		Bidirectional	3.91813E+13		Bidirectional	3.90623E+13			
	GRU	3.93068E+13		GRU	3.91734E+13		GRU	3.90501E+13			
	Hybrid	1.33515E+15		Hybrid	4.39233E+14		Hybrid	9.10956E+14			
1,000	OPTM-LSTM	2.64051E+12	Google	2,000	OPTM-LSTM	3.48733E+11	Google	3,000	OPTM-LSTM	1.74703E+09	
	LSTM	3.90955E+13			LSTM	3.90843E+13			LSTM	3.90295E+13	
	Attention	3.90155E+13			Attention	3.88844E+13			Attention	2.17723E+13	
	Bidirectional	3.90497E+13			Bidirectional	3.88841E+13			Bidirectional	3.01027E+13	
	GRU	3.90018E+13			GRU	3.88263E+13			GRU	3.94140E+13	
4,000	Hybrid	2.82755E+14	5,000	Hybrid	1.38452E+15	6,000	Hybrid	1.46894E+15			

We also noticed that while LSTMs mitigate rapid gradient vanishing, they remain at risk (see Eq. 5). Determining the optimal look-back period is challenging, but OPTM-LSTM addresses this by processing the input space in multiple dimensions, independent of batch size. It focuses on the current LOB state, discarding prior timesteps, updating each batch (size 1) based on present LOB supply and demand, thus reducing vanishing gradient risks. Consequently, backpropagation treats each OPTM-LSTM cell as an independent neural network, not part of a chain rule calculation. Another observation is that all RNN models, including OPTM-LSTM, are sensitive to higher mid-price variance. For Google stock, Figures 1 and 2 (Appendix) show higher MSE scores for 2,000–3,000 data samples compared to Amazon, where Figures 3 and 4 (Appendix) indicate better performance for the same sample size. This results from short data size and mid-price variability. Google and Amazon have mid-price variances of $7.37\text{E}+14$ and $5.78\text{E}+12$, respectively, for 3,000 samples without transformation. For 5,000 samples, variances are $2.03\text{E}+14$ (Google) and $5.66\text{E}+12$ (Amazon), reflecting a 72% and 2% variance drop, respectively. Variance impacts model performance during training, particularly for shorter trading horizons and smaller samples, with Google’s 72% lower mid-price variability (and similar LOB price level variance) affecting model testing.

Next, Tables III, VI, VIII, IX, and XII (Appendix) show MSE scores for OPTM-LSTM NN versus two baseline methods and five RNN models under two normalization settings and raw data. OPTM-LSTM maintained stable performance, as seen in Short (five epochs) and Long (60 epochs) training protocols, and outperformed the baseline models. A smaller-scale topological grid search was conducted for all RNN and OPTM-LSTM models. Among Raw, MinMax, and Zscore input settings, Zscore was the most challenging, with models showing close performance, but OPTM-LSTM excelled. Its adaptability, driven by an internal gradient descent method with a fixed, adjustable learning rate (e.g., 0.0001 for Google), enabled superior results through hyperparameter tuning.

To confirm robustness, we ran all models 10 times with random initial weights, reported in Table XIII (Appendix) for four stocks, including TCN and BFEEMD-LSTM-TWSVRSOA performance. BFEEMD-LSTM-TWSVRSOA underperformed in online forecasting, likely due to TWSVR’s fixed regression and SOA’s non-adaptive tuning, unlike the event-updating LSTM (Table XIII, Appendix). Two additional baselines, ARIMA and ETS, were included (Table XV, Appendix) to highlight OPTM-LSTM’s forecasting strength. Experiments used MinMax normalization to prevent feature dominance. To assess performance consistency, we applied the Kruskal-Wallis test on MSE scores across the four stocks, followed by Dunn’s post hoc analysis for significant differences, as per [65]. False Discovery Rate (FDR) control at a 0.005

p-value mitigated false positives. OPTM-LSTM showed statistically significant MSE performance differences compared to other models across all stocks, especially against Hybrid and TCN (Table XIV, Appendix). The Kruskal-Wallis test excluded BFEEMD-LSTM-TWSVRSOA due to its consistently poor performance.

In the last part of our experimental analysis we compared Short Training (five epochs) and Long Training (60 epochs) protocols, focusing on Google’s MSE scores by data size (Fig. 9, Appendix). Higher epochs consistently yielded lower MSE scores, with Amazon, Kesko, and Wartsila showing similar trends. For ML traders with capacity for 15,000 trading events and time for more epochs, the Long Training protocol (60 epochs) is better for forecasting. If time is limited but more data is available, the Short Training protocol (five epochs) is preferable. Training took approximately 0.4 seconds per epoch for 15,000 events on an NVIDIA Ampere A100 GPU. Fewer epochs are vital in HFT for rapid decisions, and our results show minimal performance loss with reduced epochs, maintaining practicality for low-latency trading.

Finally, the forecasting performance of competitor models and two baseline models aligns with existing literature, except for OPTM-LSTM, which outperformed them. Per [42], the prototype LSTM is robust across diverse datasets, and altering its topology (e.g., removing gates) does not enhance performance - a trend seen in the five RNN competitors. These RNNs showed performance patterns similar to the baseline models across data sizes, normalization techniques, and feature scenarios. The persistence algorithm was notably reliable, mirroring findings in [66] where it performed comparably to the prototype LSTM. Similarly, [67] showed the prototype LSTM outperforming the persistence algorithm.

F. Interpretations and Limitations

The OPTM-LSTM cell demonstrates superior forecasting performance in HFT LOB mid-price prediction, consistently achieving the lowest MSE scores across diverse stocks and more than 33 training scenarios. For highly liquid stocks like Amazon and Google, OPTM-LSTM recorded significantly better MSE scores under MinMax normalization, outperforming all models, including the latest BFEEMD-LSTM-TWSVRSOA (see Table XIII - Appendix). For less-liquid stocks like Kesko and Wartsila, OPTM-LSTM again maintained robust performance with the lowest MSE scores across the entire lineup of competitors. Specifically, OPTM-LSTM achieves MSE reductions of 40.40% for Amazon (against Bidirectional LSTM), 39.17% for Google (against GRU), 49.03% for Kesko (against LSTM), and 12.38% for Wartsila (against Attention LSTM) compared to the next best models (see Table XIII - Appendix). These differences, statistically significant in the majority of comparisons (see Table XIV - Appendix), highlight OPTM-LSTM’s adaptability across

data scales. Performance across training cycles further underscores OPTM-LSTM's superiority. On both shorter (five epochs) training cycles under 18 different sample lengths and longer (60 epochs) training cycles under nine different sample lengths, OPTM-LSTM stabilizes faster while achieving better results with lower MSE scores for all four stocks. This efficiency stems from OPTM-LSTM's online learning protocol, which dynamically recalibrates gates and states using a non-forecasting regression task optimized via gradient descent. Unlike static architectures (e.g., Attention LSTM's fixed attention or BFEEMD-LSTM-TWSVRSOA's ensemble complexity), OPTM-LSTM adapts in real-time to LOB volatility, minimizing latency-critical for HFT. Economically, these improvements enhance bid-ask spread capture, yielding significant profits in high-volume tick-by-tick trading - a critical edge in HFT's nanosecond-driven environment.

Although our OPTM-LSTM cell offers better forecasting performance in the online HFT universe there are a few limitations that we need to mention. One limitation is the restricted number of stocks and trading horizons. Despite the fact that we utilized indicative examples of high-liquid and less-liquid stocks we believe that a wider selection of stocks will provide more insights into the behaviour of the OPTM-LSTM model. Another limitation of our study is the selected trading horizon. We limited the selected number of trading events only when we realize a steady decrease in the MSE score performance. Potentially, an even lower MSE score could have been achieved by incorporating a larger trading horizon. Additionally, we aim to model the high-dimensional LOB feature space and its long-range dependencies between price levels, following the Copula Variational LSTM architecture [70]. Building on this, future work could incorporate graph-based approaches like Graph-Based Autocorrelation Preserving (GAP-LSTM) [71] to capture spatial-temporal patterns. Lastly, we believe that a more advanced optimization method could be integrated as part of the OPTM-LSTM cell.

V. DISCUSSION AND CONCLUSION

HFT LOB forecasting demands models that adapt instantly to volatile market dynamics - a challenge not fully met by existing architectures. We introduce the OPTM-LSTM cell, a novel solution for online mid-price forecasting. Unlike traditional LSTM models and its recent variants, OPTM-LSTM incorporates a unique internal non-forecasting supervised regression task, optimized via gradient descent. This dynamically recalibrates gates and states in real time. Given these OPTM-LSTM's rapid dynamics, its advantages in high-frequency environments are significant. Its online learning-based protocol enables faster performance stabilization in both shorter (five epochs) and longer (60 epochs) training scenarios, minimizing latency and allowing traders to exploit fleeting

arbitrage opportunities. Compared to existing models, such as BFEEMD-LSTM-TWSVRSOA with its computationally intensive ensemble structure, Attention LSTM with a static attention mechanism, and Hybrid LSTM-CNN with its reliance on convolutional layers, OPTM-LSTM is particularly well-suited for the event-driven dynamics of high-frequency trading. The model is especially advantageous in financial domains where reaction times are measured in milliseconds, but it could also be applicable in other time-critical fields such as telecommunications. On top of its suitability for HFT environments, experimental results show that OPTM-LSTM consistently achieves the lowest MSE scores across all stocks compared to other models, with statistical significance in the majority of comparisons (Table XIV - Appendix).

The development of the OPTM-LSTM cell also opens additional research avenues. For instance, the same internal cell architecture can be extended to classification tasks. Currently, the internal supervised regression problem is directly connected to the final forecasting objective, which is also a regression task. It would be particularly interesting to explore classification scenarios as the cell's internal supervised problem and connect them to a final classification objective. This work is classified as a narrow artificial intelligence (AI) application. However, the same cell architecture could be utilized in other forecasting challenges, such as those in the M5 Competition [68], which involves hierarchical and large-scale retail sales prediction. The competition's emphasis on hierarchical time series forecasting, daily sales data, and the need to account for external factors (e.g., promotions and holidays) aligns well with the strengths of our model. We also believe that our developments here can be applied to other online tasks, including computer vision and fully autonomous driving systems.

ACKNOWLEDGMENTS

The authors would like to thank CSC-IT Center for Science, Finland, for the generous computational resources.

REFERENCES

- [1] A. Ntakaris, M. Magris, J. Kannianen, M. Gabbouj, A. and Iosifidis. "Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods", *Journal of Forecasting*, vol. 37, issue 8, pp.852-866, Dec 2018.
- [2] A. Ntakaris, G. Mirone, J. Kannianen, M. Gabbouj and A. Iosifidis, "Feature Engineering for Mid-Price Prediction With Deep Learning", *IEEE Access*, vol. 7, pp. 82390-82412, Jun 2019.
- [3] J. A. Sirignano, "Deep learning for limit order books", *Quantitative Finance*, vol. 19, issue 4, pp. 549-570, Apr 2019.
- [4] D. Tran, J. Kannianen, M. Gabbouj, and A. Iosifidis. *Data-driven architecture learning for financial time-series forecasting*. *arXiv preprint arXiv:1903.06751*, 2019.
- [5] S. Hochreiter, and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, issue 8, pp. 1735-1780, Nov 1997.
- [6] T. Zia, and U. Zahid (2019), "Long short-term memory recurrent neural network architectures for Urdu acoustic modeling", *International Journal of Speech Technology*, vol. 22, issue 1, pp. 21-30, Mar 2019.

- [7] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "Comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition", In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 2462-2466, Mar 2017.
- [8] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker diarization with LSTM", In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5239-5243, Calgary, 2018.
- [9] 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 *Proceedings of the 54th annual meeting of the association for computational linguistics*, vol. 2 *Short papers*, pp. 207-212, Aug 2016.
- [10] O. Melamud, J. Goldberg, and I. Dagan, (2016, August). "context2vec: Learning generic context embedding with bidirectional lstm", In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pp. 51-61, Berlin, 2016.
- [11] G. Liu, and J. Guo, (2019), "Bidirectional LSTM with attention mechanism and convolutional layer for text classification" *Neuro-computing*, vol. 337, pp. 325-338, Apr 2019.
- [12] K. M. Tsiouris, V. C. Pezoulas, M. Zervakis, S. Konitsiotis, D. D. Koutsouris, and D. I. Fotiadis, "A long short-term memory deep learning network for the prediction of epileptic seizures using EEG signals", *Computers in biology and medicine*, vol. 99, pp. 24-37, Aug 2018.
- [13] Ö. Yildirim, "A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification", *Computers in biology and medicine*, vol 96, pp. 189-202, May 2018.
- [14] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, "Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats", *Computers in biology and medicine*, vol. 102, pp. 278-287, Nov 2018.
- [15] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang, "A siamese long short-term memory architecture for human re-identification", In *European conference on computer vision*, pp. 135-153, Cham, 2016.
- [16] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, (2016, October), "Semantic object parsing with graph lstm", In *European Conference on Computer Vision*, Springer, pp. 125-143, Cham, 2016.
- [17] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction", In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Nevada, 2018, pp. 1186-1194.
- [18] T. Fischer, and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions", *European Journal of Operational Research*, vol. 270, issue 2, pp. 654-669, Oct 2018.
- [19] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory", *PLoS One*, vol. 12, issue 7, Jul 2017.
- [20] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, "Deep learning for stock prediction using numerical and textual information", In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, IEEE, pp. 1-6, Okayama, 2016.
- [21] D. M. Nelson, A. C. Pereira, and R. A. De Oliveira, "Stock market's price movement prediction with LSTM neural networks", In *2017 International joint conference on neural networks (IJCNN)*, IEEE, pp. 1419-1426, Anchorage, 2017.
- [22] M. Fabbri, and G. Moro, "Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks", *Data*, pp. 142-153, Jul 2018.
- [23] J. Sirignano, and R. Cont, "Universal features of price formation in financial markets: perspectives from deep learning", *Quantitative Finance*, vol 19, issue 9, pp. 1449-1459, Sep 2019.
- [24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, S. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, and S. Ghemawat, 2016. *Tensorflow*: "Large-scale machine learning on heterogeneous distributed systems". arXiv preprint arXiv:1603.04467.
- [25] F. Chollet, 2015, "keras", Github, GitHub repository, <https://github.com/fchollet/keras>, commit: 5bcac37
- [26] A. Tsantekidis, N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 2511-2515, 2017.
- [27] A. Tsantekidis, N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Using deep learning for price prediction by exploiting stationary limit order book features", *Applied Soft Computing*, vol. 93, Aug 2020.
- [28] I. N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis. *Deep adaptive input normalization for time series forecasting*. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9), pp.3760-3765, Dec 2019.
- [29] D. T. Tran, A. Iosifidis, J. Kannianen and M. Gabbouj, "Temporal Attention-Augmented Bilinear Network for Financial Time-Series Data Analysis", *Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1407-1418, May 2019.
- [30] Z. Zhang, S. Zohren, and S. Roberts, "Deeplob: Deep convolutional neural networks for limit order books", *IEEE Transactions on Signal Processing*, vol 67, issue 11, 3001-3012, Mar 2019.
- [31] Y. Li, L. Li, X. Zhao, T. Ma, Y. Zou, and M. Chen, "An Attention-Based LSTM Model for Stock Price Trend Prediction Using Limit Order Books", In *Journal of Physics: Conference Series*, vol. 1575, issue. 1, pp. 012124, Jun 2020.
- [32] Y. Mäkinen, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data", *Quantitative Finance*, vol. 19, issue 12, pp. 2033-2050, Dec 2019.
- [33] T. Sun, D. Huang, and J. Yu, "Market Making Strategy Optimization via Deep Reinforcement Learning", *IEEE Access*, vol. 10, pp. 9085-9093, Jan 2022.
- [34] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures", In *International conference on machine learning*, PMLR, pp. 2342-2350, 2015.
- [35] Y. Lu, and F. M. Salem, "Simplified gating in long short-term memory (lstm) recurrent neural networks", In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1601-1604, Medford, 2017.
- [36] C. Tallec, and Y. Ollivier, "Can recurrent neural networks warp time?", arXiv preprint arXiv:1804.11188, Mar 2013.
- [37] A. Gu, C. Gulcehre, T. Paine, M. Hoffman, and R. Pascanu, "Improving the gating mechanism of recurrent neural networks", In *International Conference on Machine Learning*, PMLR, pp. 3800-3809, 2020.
- [38] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. R. Salakhutdinov, and Y. Bengio, "Architectural complexity measures of recurrent neural networks", *Advances in neural information processing systems*, vol. 29, 2016.
- [39] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks", In *International Conference on Machine Learning*, PMLR, pp. 1120-1128, New York, 2016.
- [40] A. G. Ororbia II, T. Mikolov and D. Reitter, "Learning Simpler Language Models with the Differential State Framework", in *Neural Computation*, vol. 29, no. 12, pp. 3327-3352, Dec. 2017.
- [41] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units", arXiv preprint arXiv:1504.00941, Apr 2015
- [42] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, J., "LSTM: A search space odyssey", *Transactions on neural networks and learning systems*, IEEE, vol. 28, issue 10, pp. 2222-2232, Jul 2016.
- [43] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM", *Neural computation*, vol 12, issue 10, pp. 2451-2471, Oct 2000.
- [44] Z. He, S. Gao, L. Xiao, D. Liu, H. He, and D. Barber, "Wider and deeper, cheaper and faster: Tensorized lstms for sequence learning", in *Advances in neural information processing systems*, NIPS, California, 2017.
- [45] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, N.R., A. Goyal, Y. Bengio, A. Courville, A. and C. Pal, "Zone-out: Regularizing rnns by randomly preserving hidden activations", arXiv preprint arXiv:1606.01305, June 2016.
- [46] D. Neil, M. Pfeiffer, and S. C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences", *Advances in neural information processing systems*, NIPS, Barcelona, 2016.

- [47] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers", *Advances in neural information processing systems*, NIPS, Barcelona, 2016.
- [48] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated recurrent neural networks", *arXiv preprint arXiv:1508.03790*, Aug. 2015.
- [49] R. Dangovski, L. Jing, P. Nakov, M. Tatalović, and M. Soljačić, "Rotational unit of memory: a novel representation unit for RNNs with scalable applications", *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 121-138, Aug 2019.
- [50] S. Mootha, S. Sridhar, R. Seetharaman, and S. Chitrakala, 2020. "Stock price prediction using bi-directional LSTM based sequence to sequence modeling and multitask learning". In *11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, UEMCON*, New York, 2020.
- [51] M.A.I. Sumy, M.M.S.Maswood, and A.G., Alharbi, 2020, October. "Deep learning-based stock price prediction using LSTM and bi-directional LSTM model". In *Novel Intelligent and Leading Emerging Sciences Conference*, (NILES), Egypt, 2020.
- [52] K.A. Althelaya, E.S.M. El-Alfy, and S. Mohammed. "Evaluation of bidirectional LSTM for short-and long-term stock market prediction". In *9th international conference on information and communication systems, (ICICS)*, France, 2018.
- [53] W. Lu., J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-based model to forecast stock prices. *Complexity*, 2020.
- [54] S., Mehtab, and J. Sen. "Stock price prediction using CNN and LSTM-based deep learning models. In *International Conference on Decision Aid Sciences and Application (DASA)*, IEEE, Bahrain, 2020.
- [55] J.M.T. Wu, Z. Li, N. Herencsar, B. Vo, and J.C.W. Lin. "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimedia Systems*, pp.1-20, Feb. 2021.
- [56] Z. Nourbakhsh, and N. Habibi. "Combining LSTM and CNN methods and fundamental analysis for stock price trend prediction. *Multimedia Tools and Applications*, 2022.
- [57] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, Dec. 2014.
- [58] T. Cansu, E. Kolemen, Ö. Karahasan, E. Bas, and E. Egrioglu. A new training algorithm for long short-term memory artificial neural network based on particle swarm optimization. *Granular Computing*, pp.1-14, Jun. 2023.
- [59] A.P. Engelbrecht, and F. van den Bergh. Effects of swarm size on cooperative particle swarm optimisers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 892-902, Jan. 2001.
- [60] T.M. Shami, A.A. El-Saleh, M. Alswaiti, Q. Al-Tashi, M.A. Summakieh, and S. Mirjalili. Particle swarm optimization: A comprehensive survey. *IEEE Access*, 10, pp.10031-10061, Jan. 2022.
- [61] H.Y.S. Chien, J.S. Turek, N. Beckage, V.A. Vo, C.J. Honey, and T.L. Willke. "Slower is Better: Revisiting the Forgetting Mechanism in LSTM for Slower Information Decay". *arXiv preprint arXiv:2105.05944*, May 2021.
- [62] T. Mikolov, "Statistical Language Models Based on Neural Networks", PhD Thesis, Department of Computer Graphics and Multimedia, BRNO University of Technology, Brno, Czechia, 2012
- [63] M. Mehdipour Ghazi, M. Nielsen, A. Pai, M. Modat, M.J. Cardoso, S. Ourselin, S. and L. Sorensen. "On the initialization of long short-term memory networks". In *International Conference on Neural Information Processing* (pp. 275-286). Springer, Cham, Dec. 2019
- [64] Y. He, and J. Zhao. Temporal convolutional networks for anomaly detection in time series. In *Journal of Physics: Conference Series* (Vol. 1213, No. 4, p. 042050), IOP Publishing, Jun. 2019.
- [65] J. Tuominen E. Pulkkinen, J. Peltonen, J. Kannianen, N. Oksala, A. Palomäki, and A. Roine. Forecasting emergency department occupancy with advanced machine learning models and multivariable input. *International Journal of Forecasting*, Dec. 2023.
- [66] B. Laperre, J. Amaya, and G. Lapenta. "Dynamic time warping as a new evaluation for dst forecast with machine learning". *Frontiers in Astronomy and Space Sciences*, p.39, Jul. 2020.
- [67] G. Gürses-Tran, and A. Monti. "Advances in Time Series Forecasting Development for Power Systems" *Operation with MLOps. Forecasting*, 4(2), pp.501-524, May 2022.
- [68] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), pp.1346-1364, Oct. 2022.
- [69] Z. Zhang, Y. Dong, and W.C. Hong. Long Short-Term Memory-Based Twin Support Vector Regression for Probabilistic Load Forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, Vol 36, No. 1, Jan. 2025.
- [70] J. Xu and L. Cao. Copula variational LSTM for high-dimensional cross-market multivariate dependence modeling. *IEEE Transactions on Neural Networks and Learning Systems*. Vol. 35, Jul. 2023.
- [71] M. Altieri, R. Corizzo, and M. Ceci. GAP-LSTM: Graph-Based Autocorrelation Preserving Networks for Geo-Distributed Forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 35, May 2024.

VI. BIOGRAPHY SECTION



Adamantios Ntakaris is a Lecturer in Financial Technology and Programme Director at the University of Edinburgh Business School with extensive experience in consulting and managing industrial quant projects. He received his PhD from Tampere University under the Big-DataFinance Marie Skłodowska-Curie training network in 2019, the MSc degree in Financial Modeling and Optimization from the University of Edinburgh in 2014, and the BA degree in Mathematics from the Aristotle University of Thessaloniki, in 2009. He also completed a quant placement at Abrdn in 2014. From 2014 to 2016 he was an Effective Interest Rate Analyst with CitiGroup Investment Bank, and from 2010 to 2013 a Math Olympiad Coach at Systima.



MONCEF GABBOUJ (F'11) received his MS and PhD degrees in electrical engineering from Purdue University, in 1986 and 1989, respectively. Dr. Gabbouj is Professor of Information Technology at the Department of Computing Sciences, Tampere University, Finland. He was Academy of Finland Professor (2011-2015). His research interests include Big Data analytics, multimedia analysis, artificial intelligence, machine learning, pattern recognition, nonlinear signal processing, video processing and coding.

Dr. Gabbouj is a Fellow of the IEEE and Asia-Pacific Artificial Intelligence Association. He is member of the Academia Europaea, the Finnish Academy of Science and Letters and the Finnish Academy of Engineering Sciences. He served as associate editor and guest editor of many IEEE, and international journals.



Juho Kannianen Dr Juho Kannianen is a Professor in Computing Sciences at Tampere University, Finland, where he is heading a research group Financial Computing and Data Analytics. He has previously coordinated two EU projects HPCFinance (GA 289032) and BigDataFinance (GA 675044, www.bigdatafinance.eu). Totally, these two networks secured 7.5 Meur of EU funding and trained 26 PhD students and 2 post-docs. He has organized several conferences and he has served as a co-editor for a book "High-

Performance Computing in Finance" (Chapman & Hall). His papers on financial market research with different methods have been published in top-tier journals, including IEEE Transactions on Neural Networks and Learning Systems, Pattern Recognition, and Review of Finance . He has supervised and co-supervised 10 PhD students and 2 are ongoing. His main research focus lies on financial data science with methods on machine learning and network theory.