Focusing at High Frequency

An Attention-based Neural Network for Limit Order Books

Andrea Barbon*

Job Market Paper

Abstract

Machine learning methods deliver superior forecasting accuracy but can hardly be used to make inference. To overcome this limitation, I propose an encoder-decoder neural network augmented with an attention-based mechanism that can autonomously learn to identify the most critical regions of the input data. I first train the model using high-frequency message data from the NASDAQ and show that it outperforms other state-of-the-art models in forecasting future transaction prices. Then, I develop a methodology that uses the attention mechanism to make inference on the relative share of information content of market orders versus limit orders, concluding that the most informative events are executions of market orders while submission and cancellations of limit orders are less relevant. Finally, I test the model's behavior during the execution of real block orders from institutional investors, showing that it favors liquidity provision rather than front-running strategies.

Keywords: Limit Order Book, Machine Learning, Attention, Inference, Liquidity Provision
JEL classification: C45, C58, G13, G17

*Andrea Barbon (www.abarbon.com, andrea.barbon@usi.ch) is affiliated with USI Lugano and the Swiss Finance Institute. The author is thankful to Joshua D. Coval, Francesco Franzoni, Shimon Kogan, Fabio Pruneri, Jeremy Stein, and Adi Sunderam for fruitful comments and insightful discussions. All mistakes are mine.

1 Introduction

Artificial intelligence and machine learning methods are attracting widespread interest among the scientific community and the industry alike, due to the uncountable successful applications across a variety of different fields. In particular, deep learning methods have proven effective in the context of financial time-series forecasting. On the one hand, these methodologies are of interest for practitioners, since they deliver higher prediction accuracy with respect to traditional techniques. On the other hand, one of the major drawbacks to adopting machine learning models is that they are difficult to interpret and do not allow for inference. The black-boxiness nature of these models seems to preclude the possibility of adding them to the econometric toolkit of empirical researchers in financial economics.

However, the fact the deep neural networks are able to produce accurate out-of-sample forecasts suggests that these models can identify meaningful patterns in the data. This, in turn, opens the possibility that a fitted neural network could be used to make inference on the data on which it was trained. However, despite the intriguing potentials of these data-mining capabilities, to the best of my knowledge the literature has not yet developed any methodology to use machine learning models as tools to answer economic questions in a data-driven fashion.

This paper proposes a novel approach to increase the intelligibility of machine learning models, enabling researches to use them as inferential devices. The idea is based on a recently developed class of neural networks, featuring an attention-based mechanism that can autonomously learn to identify the most informative parts of the input data. In the context of time-series forecasting, in particular, these models learn to allocate a heterogeneous degree of attention to different time steps of the input sequence. This unique feature allows to perform inference from the trained model and delivers a higher level of interpretability relative to previously adopted architectures. Intuitively, one can use the trained attention-layer to identify the temporal regions which are most important for the model to produce its forecasts. The resulting time-series of attention levels can be interpreted as a measure of the informativeness of different events with respect to the assigned prediction task.

A natural way to showcase this idea is to apply attention-based neural networks to a realistic and well-studied forecasting problem in the context of financial markets. With such a motivation in mind, in this paper I focus on the adverse selection problem faced by market makers submitting quotes to a stock exchange. The activity of market makers consists in posting limit orders, that is, providing options for others to buy or sell at specific prices. In doing so, they face the risk of trading with informed investors endowed with more precise forecasts of the future value of the asset. Market makers can effectively limit this risk by extracting (part of) such private information from the orderflow, and updating their quotes accordingly.

My formulation of the prediction problem reproduces the challenge faced by market makers and consists in exploiting past orderflow to produce the best forecast of future transaction prices.

This setting constitutes an ideal laboratory to evaluate the performance and the interpretability of attention-based models vis-a-vis a realistic forecasting problem and, because the issue has been extensively explored by market microstructure theorists, it provides a rich set of testable hypotheses. In particular, I apply the proposed methodology to make inference on the marginal information content of limit orders versus market orders. Microstructure models typically assume that market orders are the most informative, because they are submitted by traders with superior information willing to trade fast and capitalize on their informational advantage. Inspection of the attention allocation of the trained neural network provides a data-driven method to validate or to discard such a hypothesis.

My experimental setup, inspired by the adverse selection problem faced by market makers, is further motivated by a number of additional reasons. First, given that modern market making activity takes place in electronic exchanges at very high frequency, an astonishing amount of data is available. This limits the problem of over-fitting and makes it possible to train deep neural networks featuring a large number of parameters. Another consequence of the increasing speed of market making activity has been the surge of academic interest on the impact of high-frequency traders (HFTs) on the functioning of financial markets. The debate has focused mainly on their effect on liquidity. On the one hand, HFT firms argue that their activity increases market liquidity by reducing bid-ask spreads, a claim that is supported, at least conditional on normal market conditions, by empirical evidence. On the other hand, investors and some observers in the financial press blame HFTs for back-running block trades by large institutions. One of the concerns is that, since large trades need to be split into smaller child executions over a non-trivial time frame, HFTs may quickly extract private information from the first part of the order and trade in the same direction of the institutional investors, thus increasing effective transaction costs and imposing a non-trivial externality to the originator. Alternatively, for liquidity-motivated trades, HFTs may recognize their presence in the orderflow, start trading in the same direction and revert the position before the end of the block, speculating on and adding to the temporary component of the price impact. This predatory behavior is costly for institutions, since it increases their effective cost of trading. My analysis sheds light on this important issue by studying the behavior of my model, a forecasting tool which is closely related to the algorithmic activity of HFTs, during the executions of large block trades by institutional investors.

The empirical analysis is based on high-frequency message traffic data from the NASDAQ electronic exchange. The data contain every single event in the limit order book (LOB) for a given stock, namely trade executions and the submission or cancellation of limit orders,

timestamped with micro-second precision. In particular, I use a large sample of LOB events for ten among the most liquid stocks traded in the exchange to train four distinct machine learning models, including the proposed attention-based model and alternative neural architectures which have been proven useful in this context.

It is self-evident that a necessary condition to make valid inference from the attention layer of my model is that the latter is capable of producing high-quality forecasts. If the model is not able to extract useful patterns from the data, then its attention allocation cannot be used to infer the informational content of different LOB events. Hence, in order to assess the forecasting accuracy of my attention-based network, I compare it to the benchmark models through different measures of out-of-sample performance.

Once the forecasting power of the attention-based model is validated, I proceed by carrying out a systematic analysis of the attention layer. First, I inspect the output of the attention-layer during every example in the test set, consisting of a sequence of LOB events and a target variable representing future transaction prices, to extract the level of attention assigned by the trained model to each step of the input time-series. Then, to systematically asses how the model focuses its temporal attention, I run panel regressions of the resulting attention levels on dummy variables indicating different types of LOB events, namely submissions and cancellations of limit orders and execution of market orders. The estimated coefficients from these regressions can be interpreted as the level of informativeness of each event type, with respect to the forecasting of future execution prices.

For the third part of the analysis, focused on liquidity, I complement order book data with transaction-level data for institutional investors provided by AbelNoser/Ancerno. This database reports transactions in the US equity market from large mutual funds and hedge funds, recording the identity of the funds with unique identifiers. This allows me to identify block orders, searching for multiple sequential executions on a single stock by a single institution summing up to a large volume.

I then train my model on LOB data from time windows preceding each of the identified block-order and use it to produce predictions during the event days. Then, to understand whether my model fosters back-running or predatory trading, I check if the sign of these predictions are systematically related to the side of the block orders.

My first set of results shows that my attention-based model delivers a significantly higher out-of-sample performance relative to the competing models, across different stocks and multiple trading days. Importantly, I provide evidence of a significant marginal contribution of the attention mechanism in improving the forecasting accuracy relative to an alternative specification of the model in which the attention layer is bypassed. This result demonstrates that the attention mechanism can effectively improve the information extraction from LOB events,

thus validating the identification assumption that the attention level assigned by the model to each input time-step can be used as a proxy for the informativeness of the underlying event. This is a necessary condition to justify the use the model to make inference and, therefore, it allows to proceed with the central part of the analysis.

The second set of results, obtained from the above-described panel regression approach, indicates that attention levels peak during trade executions, are mid-range during submission of new limit orders, and are significantly lower on cancellations of outstanding quotes. The coefficient estimates are robust to the inclusion of additional explanatory variables which are expected to be related to information, such as the speed of trading activity and the magnitude of mid-price changes. These findings are consistent with the standard assumption of theoretical microstructure models featuring asymmetric information, that is, that market orders are submitted by informed traders while limit orders come from uninformed market makers.

The third set of results shows that my model, in fact, behaves systematically differently in the presence of block orders. In particular, the sign of its predictions are negatively related to the side of the block, indicating that the model forecasts a higher profit for the market maker if she take the opposite side and provides liquidity to the institutional investor. This evidence is consistent with previous research showing that algorithmic trading improves liquidity and, further, it complements them focusing on the realized trading costs associated to institutional block orders rather then measuring only its effects on bid-ask spreads.

Finally, I demonstrate how the attention layer can be used to visually inspect the model's decisions on a case-by-case basis, adding an extra layer of interpretability to the output of the model. This can be done by superimposing the time-series of attention levels to a plot representing the sequence of input variables. Even though limited by its qualitative nature, this method can provide some insights on the inner workings of the model, both in situations where the forecast is correct and those where it is mistaken.

Related Literature

My attention-based model, which takes as input past orderflow data and produces accurate forecasts of future transaction prices, represents an instance of the broader class of technological innovations in the processing of demand data. More efficient information extraction from the orderflow allows to better distinguish between trades motivated by fundamentals and those driven by exogenous liquidity shocks. Uninformed demand shocks introduce a layer of noise in prices, obstructing the information aggregation role of financial markets (Black, 1986, Shleifer and Vishny, 1997). Information extraction technologies, and my model in particular, could help market participants filtering out the noisy component of the orderflow, leading to a more efficient extraction of fundamental information and increasing price informativeness.

Extraction of orderflow information is also related to market liquidity through the adverse selection channel. The bid-ask spread can be seen as the compensation required by market makers for the risk of trading against agents endowed with superior information on the fundamental value of the asset. For this reason, a standard prediction of microstructure models is that the width of the bid-ask spread is proportional to the level of information asymmetry between market makers and other market participants. But information can leak through prices, thus market makers have an incentive to track the orderflow in real time and try to deduce fundamental information, in an effort to reduce the asymmetry. My model is specifically trained to address this problem and could be of great value for market makers in their quest to reduce the level of information asymmetry. Therefore, to the extent that its usage marginally increases the ability of real-world market makers to forecast future transaction prices, the adoption of my attention-based model could lead to a reduction of bid-ask spreads and an improvement in market liquidity.

The speed in delivering forecasts based on real-time data represents a key factor to determine whether the model is applicable to real exchanges. To understand why this is the case, notice that the submission of limit orders by market makers effectively provides options for others to trade at the posted prices. If liquidity demanders can identify profitable in-the-money options arising from stale limit orders, they can pick them off and profit at the expense of the market maker. Assuming the pick-off risk is non-negligible and market makers require a compensation for it, this has the effect of widening bid-ask spreads and increasing execution costs. Consequently, if algorithms can reduce the cost of free trading options implicit in limit orders, then the level of adverse selection depends on the quality and efficiency of the algorithms employed by market makers (Foucault et al., 2013). Training a deep neural network on historical data takes a non-negligible amount of time and hardware resources, because of the large number of required data points and the computational cost of back-propagation passes. However, once the training process is over, the calculation of forecasts based on real-time data is extremely fast even on a desktop machine. Because they can be employed in high frequency contests, neural network models providing high-quality forecasts of future transaction prices have the potential to help liquidity providers to efficiently update their quotes and reduce the risk of being picked off. My attention-based model can therefore reduce adverse selection and increase market liquidity through this channel, allowing market makers to post narrower spreads in equilibrium.

The idea that technological innovation leads to more informative prices and improves market liquidity is supported by empirical evidence. For instance Hendershott et al. (2011) use a neat identification strategy**,** based on the introduction of automated quote dissemination in the New York Stock Exchange**,** to provide causal evidence supporting this view. The same conclusion arises from the empirical analysis of Chaboud et al. (2014), who exploit a long time

series of high-frequency data from the foreign exchange market. Nevertheless, the argument for liquidity improvements and price efficiency, often used by high frequency traders (HFTs) to justify their increasing presence in markets, has been widely criticized by academics and observers in the financial press. One concern is that the focus on orderflow analysis may lead to a reduction of investment in fundamental research in the long run, which in turn would decrease price informativeness and impede the efficient allocation of capital across firms. The technology growth model proposed by Farboodi and Veldkamp (2018) speaks to this issue, investigating a setting in which agents allocate their budget between fundamental and orderflow information. Their model implies that in the long-run, as information technology increases the total amount of information available to the agents (high-technology limit), fundamental analysis is not crowed-out by investment in orderflow information extraction and that price informativeness should consequently increase over time.

From a theoretical standpoint, the link between orderflow information extraction and liquidity relies on the assumption of information asymmetry between market markers and privately informed investors. Standard microstructure models, starting from the seminal Glosten and Milgrom (1985) and Kyle (1989), assume that market makers post limit orders based on the publicly available information while informed traders, to avoid the risk of losing their informational advantage, submit market orders which guarantee immediate execution. An implication of such an assumption is that market orders are more informative than limit orders in forecasting future price levels. This fact is challenged by the findings of Brogaard et al. (2016), who analyze LOB data from the Canadian equity market and, using a vector autoregression (VAR) model, show that price discovery occurs predominantly through limit orders. The authors argue that the reason why limit orders provide the majority of price discovery is that they are far more numerous than executions (market orders represent less than 5% of messages). In contrast, even though submissions and deletions of limit orders constitute the vast majority of events in the dataset, my analysis reveals that the attention-based model chooses to focus significantly more on execution of market orders. This result suggests that the information content of executed trades is significantly more important as a driver of LOB dynamics, consistent with the view that market orders reveal private information. I conjecture that the discrepancy between the findings and that of the above mentioned work may be due to the limitation of the VAR model.

This view is supported by the work of Sirignano and Cont (2018), who propose a deep neural network model with three LSTM layers, trained on order book data for a large cross-section of US equity stocks, to predict the sign of future price movements. They show that the deep learning approach delivers superior forecast accuracy compared to linear VAR models, uncovering the importance of nonlinear relations between state variables and price changes. Moreover, their results provide evidence of path-dependence in price dynamics and suggest

the existence of a universal and stationary price formation mechanism shared among different stocks.

In a recent stream of literature, machine learning methods have been applied to the problem of forecasting mid-price changes at short horizons using LOB data. A variety of model architectures have been proposed and studied, based on Support Vector Machines (SVMs), on recurrent neural networks (RNNs) like the long short-term memory (LSTM), on convolutional neural networks (CNNs) or a combination of a CNN layer followed by a RNN layer. [1]

More closely related to my approach, a limited number of recent works explore architectures augmented with attention mechanisms. Among these, Tran et al. (2018) propose a bilinear network augmented with an attention-based mechanism to predict mid-price changes for ten stocks traded in the Finnish stock exchange. While the attention layer they implement is similar to the one employed in this work, my model is different because it is based on an encoder-decoder architecture with an RNN as encoder, as described in Section 2. An alternative attention-based model trained on LOB data has been used by Mäkinen et al. (2018) to predict high frequency stock price jumps identified using the non-parametric test proposed by Lee and Mykland (2007). Their results show that an attention-based model achieves a higher accuracy than linear models, convolutional neural networks and plain LSTM in this task.

A different problem has been explored by le Calvez and Cliff (2018), who train a deep neural network on simulated LOB data to learn the trading behavior of a profitable algorithmic trader. They demonstrate that the model can fully reproduce the trading strategy of the trader and can generate even higher profits. Another related work is that of Nevmyvaka et al. (2006), who use millisecond LOB data from NASDAQ and a reinforcement learning model to address the problem of optimized trade execution. Their results show that such an approach to the problem can significantly reduce the transaction costs associated to the liquidation of a large position by an informed trader. My paper addresses a similar problem, but seen from the perspective of market markers providing liquidity to the market and facing adverse selection.

Differently from the objective of the work, however, the research outlined above has tended to focus on forecasting performance rather than interpretability and inference. One key con-

[1] The design of Zhang et al. (2019), for instance, is based on a CNN layer which allows the model to aggregate information from different levels of the LOB, computing weighted-averages with autonomously learned weights. As a second layer they employ an Inception Module (Szegedy et al., 2015) to extract local interactions over different time horizon. The resulting features are finally passed to an LSTM layer, which captures the dynamic temporal behavior. The work by Sirignano (2019) proposes a new neural network architecture for spatial distributions and uses it to model the joint conditional distribution of the future best bid and ask prices. Taking advantage of the local spatial structure of the LOB, the proposed spatial neural network based on bounded hidden units outperforms both a standard neural network and a logistic regression model.

tribution of my work is to provide a novel motivation for the application of machine learning methods in the context of academic research in finance, showing how attention-based models can effectively be used to make data-driven inference.

The concept of *attention* for neural network models was first introduced in the context of neural machine translation by the seminal paper of Bahdanau et al. (2014). The authors augment their network with a novel attention layer to allow the model to dynamically focus, while producing the translation for a specific word, on a subset of related words in the input phrase. Building on this idea, Vaswani et al. (2017) take the attention paradigm to the next level proposing a groundbreaking machine translation approach. Differently from previous state-of-the-art models for that task, their model is based entirely on attention mechanisms and does not include any recurrent nor convolutional layer. They show that the so called Transformer model achieves unprecedented levels of accuracy in English-to-French translation and proves significantly faster to train. Outside the context of machine translation, the attention-based approach has proven useful in a number of different tasks related to textual analysis[2], time-series prediction, and medical diagnosis, achieving high levels of both performance and interpretability[3].

Structure of the Paper

The rest of the paper is organized as follows. Section 2 describes the architecture of the attention-based model and briefly outlines the benchmark models. Section 3 defines the experimental setting, the input features, the target variable, and the training and evaluation procedures. Section 4 compares the model with the competing models with respect to forecasting accuracy. Section 5 uses the attention-layer to make inference on the marginal information content of LOB events and to improve visualizations of the model's decisions during specific examples. Section 6 presents the analysis on the model's behavior in the presence of block orders by institutional investors. Finally, Section 7 contains concluding remarks.

2 Model Architecture

In this section I propose and formalize a neural encoder-decoder model augmented with an attention-based mechanism, which I will apply to the liquidity provision problem described in Section 3.4. More specifically, the proposed model features an intra-attention (or self-

[2] For instance, the attention paradigm has been applied to text comprehension (Cheng et al., 2016, Parikh et al., 2016), textual entailment (Lin et al., 2017), relation classification (Zhou et al., 2016) and text summarization (Paulus et al., 2017).

[3] See for example Qin et al. (2017) and Riemer et al. (2016) for time-series and Choi et al. (2016) for medical diagnosis.

attention) layer, that is, an attention mechanism in which different positions of the same sequence are related to each others with the objective of producing a meaningful representation of the input time-series.

As depicted in Figure 1, the model consists of four layers. The input and the recurrent layers can be though as the *encoder*, while the *decoder* is composed by the attention and the output layers.

The encoder is a recurrent neural network (RNN) which aims to capture long-term temporal dependence in the sequence of LOB events. At each time step, the RNN is trained to produce a representation $\mathbf{h}_t$ of the current state. The so called *hidden state*[4] contains information on the most recent event together with information from past events. To generate the next hidden state $\mathbf{h}_{t+1}$, the RNN decides which part of the information to retain and which to forget. Ideally, the final hidden state $\mathbf{h}_T$ should contain all the information needed to predict the target variable. However, this forces the model to condense all the relevant information in a single vector of fixed length, potentially resulting into a bottleneck.

This problem can be tackled by introducing a decoder featuring an attention layer. This allows the model to additionally perform a soft-search on the entire sequence of the hidden states generated by the encoder, using the final hidden state $\mathbf{h}_T$ as key for the query. Intuitively, the decoder first looks at $\mathbf{h}_T$ to be aware of the current state. Then, before generating the final prediction, it looks *back in time* and extracts the additional information required to put the current state into context.

In the following sections I provide a formal description of the encoder and decoder layers. I implemented the model in python leveraging on the latest version of the TensorFlow framework (Girija, 2016) with eager execution.

2.1 Encoder

Given the input sequence $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_T)$, with $\mathbf{X}_t \in \mathbb{R}^N$ where N is the number of features and T is the number of lags, the encoder learns a representation mapping

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{X}_t) \tag{1}$$

where $\mathbf{h}_s$ is the hidden state of the encoder at time s and f is a nonlinear activation function which could be estimated by any recurrent neural network (RNN). In this work I use the gated recurrent unit (GRU) proposed by Cho et al. (2014). As an alternative, I also experimented with the long short-term memory (LSTM) network of Hochreiter and Schmidhuber (1997), obtaining very similar results.

[4] The term *hidden state* in the context of recurrent neural networks refers to a distinct concept than in the theory of Markov chains and should therefore not being confused with the latter.

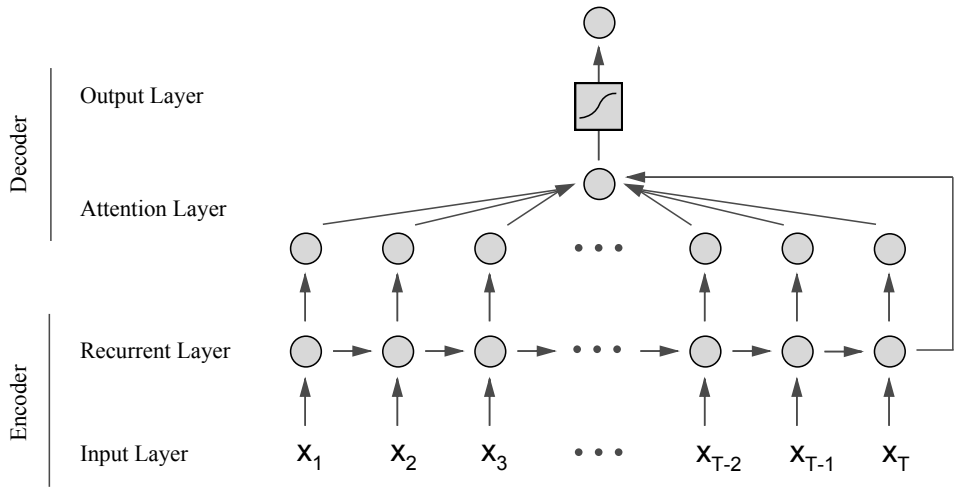


Figure 1: Model Architecture. The Figure provides a schematic for the architecture of my attention-based model, consisting of four layers. The input and the recurrent layers can be though as the *encoder*, while the *decoder* is composed by the attention and the output layers. The encoder is a recurrent neural network, which captures long term temporal dependence in the sequence of LOB events and encodes the input data into a sequence of hidden states. The decoder first employs an attention-mechanism and runs a soft-search on the hidden states generated by the encoder, using the last hidden state as key for the query. Then it uses the resulting attention distribution to construct the *context vector* as the attention-weighted average of the hidden states. Finally, to compute the prediction, the context vector and the last hidden states are fed to a fully-connected layer with hyperbolic tangent activation.

Even though my approach is robust to the choice of the encoder's RNN, both GRU and LSTM are a natural choice since they do not suffer from the vanishing gradient problem described in Hochreiter (1998) and Hochreiter et al. (2001). These RNNs, therefore, allow for the long-term time-series dependencies which are expected to be found in limit order book dynamics.

My GRU implementation is based on the first version of Cho et al. (2014), which has the advantage of being compatible with the cuDNN library proposed by Chetlur et al. (2014) and can thus be trained on a GPU. The activation of the step-t hidden state $\mathbf{h}_t \in \mathbb{R}^H$ is described as follows. First, the *reset* gates $\mathbf{r}_t = (r_t^1, \dots, r_t^H)$ are computed as

$$\mathbf{r}_t = \sigma\left(\mathbf{W}^r \mathbf{X}_t + \mathbf{U}^r \mathbf{h}_{t-1} + \mathbf{b}^r\right) \tag{2}$$

where σ denotes the sigmoid function applied component-wise, while $\mathbf{W}^r$, $\mathbf{U}^r$ and $\mathbf{b}^r$ are parameters to be learned. Similarly, the *update* gates $\mathbf{z}_t = (z_t^1, \dots, z_t^H)$ are computed as

$$\mathbf{z}_t = \sigma\left(\mathbf{W}^z \mathbf{X}_t + \mathbf{U}^z \mathbf{h}_{t-1} + \mathbf{b}^z\right) \tag{3}$$

where the matrices $\mathbf{W}^z$, $\mathbf{U}^z$ and the vector $\mathbf{b}^z$ are parameters to be learned. Finally, the

hidden state $\mathbf{h}_t$ of the GRU unit is computed as

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \circ \widetilde{\mathbf{h}}_t \tag{4}$$

$$\widetilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}^h \mathbf{X}_t + r_t \circ \mathbf{U}^h \mathbf{h}_{t-1} + \mathbf{b}^h\right) \tag{5}$$

where $\circ$ denotes the Hadamard product and $\mathbf{W}^h$, $\mathbf{U}^h$ and $\mathbf{b}^h$ are parameters to be learned.

2.2 Decoder

The decoder takes as input the last cell state $\mathbf{s}_T$ and the entire sequence of hidden states $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_T)$ generated by the encoder[5]. These are processed through a temporal attention mechanism, defined as follows, which selects the most relevant encoder hidden states across time. The vector $\mathbf{e}$, containing the temporal attention weights associated to each time step, is a non-linear transformation of the last encoder cell state $\mathbf{s}_T$ and the full sequence of hidden states. The resulting vector is then standardized using the component-wise softmax function, obtaining the attention-distribution $\boldsymbol{\alpha}$

$$\mathbf{e} = \mathbf{v}^\top \tanh(\mathbf{W}^a \mathbf{s}_T + \mathbf{U}^a \mathbf{h} + \mathbf{b}^a)\,, \tag{6}$$

$$\boldsymbol{\alpha}_t = \frac{\exp(\mathbf{e}_t)}{\sum_{s=1}^{T} \exp(\mathbf{e}_s)}, \quad t = 1, \ldots, T\,, \tag{7}$$

where the vectors $\mathbf{v}$ and $\mathbf{b}^a$ and the matrices $\mathbf{W}^a$ and $\mathbf{U}^a$ are parameters to be learned.

Next, the context vector $\mathbf{c}$ is calculated as the attention-weighted average of the hidden state vectors, where each state $\mathbf{h}_s$ is weighted by the corresponding attention level $\boldsymbol{\alpha}_s$

$$\mathbf{c} = \sum_{s=1}^{T} \boldsymbol{\alpha}_s \mathbf{h}_s\,. \tag{8}$$

The attention weight $\boldsymbol{\alpha}_t$ can thus be interpreted as the importance, in determining the prediction, of the information content of the encoder hidden state at time t. Finally, the prediction is computed feeding the context vector $\mathbf{c}$ and the last hidden state $\mathbf{h}_T$ to a fully-connected network with one layer and hyperbolic tangent activation function

$$\hat{y} = \tanh(\mathbf{F}(\mathbf{c}, \mathbf{h}_T) + \mathbf{b}), \tag{9}$$

where the matrix $\mathbf{F}$ and the vector $\mathbf{b}$ are parameters to be learned.

The attention mechanism is useful in this context because it does not force the model to encode a whole sequence of LOB events into a single fixed-length vector. Rather, it encodes the input

[5]Notice that in the case, since I use a GRU as the RNN layer, I have $\mathbf{s}_T = \mathbf{h}_T$.

into a sequence of vectors and adaptively over-weights a subset of these vectors to produce the prediction. This frees the model from having to squash all the information contained in the order book dynamics into a fixed-length vector. In a sense, the models learns to choose what parts of the input sequence are most meaningful as a context to the current state, and selectively predicts an output based mostly on those parts. Moreover, once the model is trained, the attention mechanism allows for a post-hoc analysis of the model's decision on each particular situation. In Section 5 I exploit this feature to systematically asses what types of LOB events are most relevant for the price formation process and to qualitatively analyze the model's predictions on specific examples.

2.3 Benchmark Models

I compare the proposed encoder-decoder attention-based model with alternative architectures proposed by the literature. These alternative models have been applied to the task of predicting future mid-price movements using sequences of LOB events, a slightly different problem from the one I study in this work, where the target variable includes also information about liquidity. Nevertheless, since previous research proved these architectures effective at extracting information from past order book dynamics, they constitute a natural benchmark to evaluate the performance of my model.

Plain-Vanilla Neural Network

A fully-connected neural network with a single layer and hyperbolic tangent activation function. The input is a $(25 \times T)$-dimensional array, containing the entire panel of lagged features over the previous T events flattened into a single vector. This model is the simplest among those I consider, as it does not feature a recurrent layer and thus is not expected to handle time-series dependence efficiently.

CNN-LSTM (DeepLOB)

This model, proposed by Zhang et al. (2019), features a convolutional layer aimed at capturing the spatial structure of the LOB followed by a recurrent layer for long-run time dependency. The convolutional layer is composed by a number of sub-layers, each containing 16 filters of different sizes and strides, followed by an inception module with blocks of 32 filters. The output is then passed to the recurrent layer, consisting of a LSTM network with 64 hidden units. I replicate the model using the TensorFlow backend and replace the softmax activation of the final fully-connected layer with a tanh function, since the task analyzed in this paper is a regression and not a classification problem.

Encoder-Decoder With No Attention

This model is nested into my encoder-decoder attention-based model, the difference being that the attention layer is completely bypassed. Hence the encoder is exactly the same, while the

decoder computes the final activation simply as $\hat{y} = \tanh(\mathbf{F}\mathbf{h}_T + \mathbf{b})$ rather than using equation (9). Including this models to the benchmark allows to infer the marginal improvement in performance due to the attention layer.

3 Experimental Design

This section describes the experimental setting underlying this work. I provide information on the employed dataset, outline the construction of the feature and the target variables, provide a formal statement of the prediction problem and finally describe the training and evaluation procedures. The experimental results on the out-of-sample performance the four models described in Section 2.3 are reported in Section 4.

3.1 Dataset

I collected data for 10 of the most liquid stocks traded in the NASDAQ exchange[6], for the period from February 2019 to March 2019. The data is provided by Lobster Data and is based on the NASDAQ Historical TotalView-ITCH database. Even though the sample spans only a relatively period in calendar time, given the high-frequency nature of the LOB dynamics it contains more than 47 million observations.

Each entry of the dataset represents an event (or *message*) regarding the first five levels of the LOB, on both the ask and the bid side, time-stamped with microsecond precision and classified into three main types: *deletion* events are recorded when a limit order is (fully or partially) canceled by the originator and removed from the LOB; *execution* describe trades, that is, when a limit order is matched with an incoming market order and executed; *submission* events indicate that a new limit order is posted or an existing limit order is updated. Each event represents a change in the LOB and, accordingly, the dataset provides a snapshot of the new state implied by that event, including the price levels and the aggregate sizes of the limit orders outstanding in the first five levels of the LOB.

Table 1 presents the average number of events per minute for each of the sample stocks, broken-down by event type. Microsoft is the most active stock, with more than 3000 events fired on average for each minute of the trading day. Other stocks, e.g. Tesla or Adobe, are significantly less active but nevertheless present more than 500 events per minute. I note that only a small fraction of events represents executions, while the vast majority refers to submission and cancellation of new orders.

[6] American Airlines (AAL), Apple (AAPL), Adobe (ADBE), Comcast (CMCSA), Ebay (EBAY), Facebook (FB), Microsoft (MSFT), Nvidia (NVDA), Pepsico (PEP) and Tesla (TSLA).

	AAL	AAPL	ADBE	CMCSA	EBAY	FB	MSFT	NVDA	PEP	TSLA
Submissions per Minute	438.2	1074.9	264.2	583.4	499.6	643.8	1472.8	378.5	304.0	256.5
Deletions per Minute	449.7	1053.8	245.2	621.0	525.4	616.5	1532.8	362.7	306.0	214.4
Executions per Minute	16.1	62.1	24.0	14.2	13.7	65.4	53.4	49.4	20.2	75.8

Table 1: Frequency of Events by Type. The table presents the average number of LOB events per minute, by event type. In *deletion* events, a limit order is (fully or partially) canceled by the originator and removed from the LOB; in *execution* events, a limit order is matched with a market order and executed; in *submission* events, a new limit order is posted in the LOB or an existing limit order is updated.

3.2 Features

In this section I describe the procedure to construct the 25 features variable from the raw LOB data fed to the benchmark supervised learning models. My objective is to avoid polluting the features set with ratios between variables, moving averages or other arbitrarily constructed quantities. On the contrary, I select the minimum set of variables which in principle allows a full description of the LOB dynamics. I make sure that each feature has a straightforward interpretation, with the objective to improve the intelligibility of the considered models, in the sense of Lipton (2016). Moreover, this choice allows me to perform a post-hoc visual analysis of the proposed attention-based model in Section 5.

As in most machine learning settings involving time-series data, an important aspect of pre-processing is to ensure the stationary of the employed features. This is a necessary condition to allow models to generalize the patterns learned in the training set and to achieve a satisfactory out-of-sample performance. The issue is particularly severe when the objective is to forecast financial data, given the high level of non-stationarity and the multi-modal nature of the variables involved.

I develop a novel approach to guarantee the informational quality of the resulting features. First of all, following Tsantekidis et al. (2018), I avoid using z-score standardization for price and size variables. This choice prevents me to run into forward-looking bias (if full-sample statistics are used) or to be forced to choose arbitrary windows (if backward-looking statistics are computed from rolling-windows). Moreover using past observations to standardize future quantities may lead to a distortion of the latter. For instance, suppose the outstanding liquidity is particularly high on day $d-1$. If one uses the previous day average to standardize order sizes on the next day, this would result into under-stated levels and variability of the features representing order sizes for day d.

An alternative approach to standardization is proposed by Passalis et al. (2019), who develop a novel neural layer with the specific objective of dealing with this problem. Their adaptive layer learns to efficiently normalize the raw LOB data to be passed as input to a plain

or convolutional neural network. My approach is different in that I do not build an input normalization layer into the model.

I construct features representing the relative distribution of available liquidity across the first 5 levels of the LOB[7]. For each event t, let $as(\ell, t)$ and $bs(\ell, t)$ denote the number of shares on offer in level ℓ of the ask and the bid side of the LOB, respectively. I define the corresponding features as the ratio to the total amount of outstanding liquidity

$$AS(\ell, t) = \frac{as(\ell, t)}{\sum_{l=1}^{5} as(l, t) + bs(l, t)} \tag{10}$$

$$BS(\ell, t) = \frac{bs(\ell, t)}{\sum_{l=1}^{5} as(l, t) + bs(l, t)} \tag{11}$$

This results into 10 features with unitary sum, describing the shape of the distribution of available liquidity in the LOB at each point in time.

I complement the above information with 10 additional features describing the offer price for each order book level. Let $ap(\ell, t)$ and $bp(\ell, t)$ be the raw offer prices associated to level ℓ of the ask and bid sides of the LOB, respectively. Further, let $m(t) = (ap(1, t) + bp(1, t))/2$ denote the prevailing mid-price at time t. I define the price-level features as the relative distance of each offer price from the mid price, expressed in percentage points

$$AP(\ell, t) = \frac{ap(\ell, t) - m(t)}{m(t)} \times 10^2 \tag{12}$$

$$BP(\ell, t) = \frac{m(t) - bp(\ell, t)}{m(t)} \times 10^2 \tag{13}$$

The resulting set of 10 features convey information on the level-ℓ *spreads* posted by market makers in the LOB at each point in time. Hence, their first differences can be used to infer the widening or shrinking of the spreads over time. However, these features fail to capture information about the price level. I address this issue by adding an additional feature describing the mid-price movement relative to the previous message, expressed in basis points

$$\text{PriceChange}(t) = \frac{m(t) - m(t-1)}{m(t-1)} \times 10^4 \tag{14}$$

Next, I construct two dummy variables categorizing message types. The first dummy, 'Execution', indicates a transaction being executed, that is, when (part of) an incoming market order is matched with an outstanding limit order. As in Zhang et al. (2019), I do not distinguish between the execution of *visible* and *hidden* limit orders. The dummy 'Deletion' indicates the

[7] It turns out that augmenting the depth to 10 or more levels does not significantly increase predictive power, consistent to the results of Zhang et al. (2019).

removal or the cancellation (partial removal) of a limit order from the outstanding LOB. The complementary set of messages corresponds to the submission of new limit orders.[8]

I further add to the set of features the binary variable 'Direction', taking values in $\{-1, 1\}$ and providing the directionally of the LOB message, i.e positive (negative) values indicate a change in the bid (ask) side of the order book. In particular, for executions, positive (negative) values of 'Direction' indicate seller-initiated (buyer-initiated) trades.

Finally, to provide information on the frequency of the trading activity in real time, which are expected to be relevant given the significant clustering of trading activity over time observed in the data and reported by Menkveld (2018). I hence define the feature 'Elapsed Time' as $\log(1+\tau)$, where τ measures the seconds in calendar time which have passed from the previous message.

Summary statistics on the above described features are reported in Table 6, separately for each of the sample stocks. I notice that the distribution of time intervals between consecutive events is significantly right-skewed. This signals the above mentioned clusters of high-frequency trading activity in the data. Even if not visible from the table, I report here that the median time interval is less than 1 millisecond for every stock and that the fastest events unfold at the order of micro-seconds. As a final remark, I highlight the fact that only a small fraction of the messages are triggered by executions. In fact, the share of executed transactions ranges between 1% and 6% across all sample stocks but TSLA, exhibiting a significantly higher fraction of executions.

3.3 Target Variable: Market Maker's Expected Profit

In electronic limit order markets, like the NASDAQ, market makers post ask and bid quotes in the LOB. These can later be crossed with incoming market orders, giving rise to transactions. Doing so the market maker provides liquidity to other market participants and she is compensated by earning the bid/ask spread. However she faces the adverse selection problem arising from orders submitted by agents trading on private information (Glosten and Milgrom, 1985, Kyle, 1989). Intuitively, posting a sell (buy) limit order is similar to underwriting a call (put) option with strike price equal to the quote. Informed speculators may exercise these options and "pick off limit orders" when these become stale after the arrival of new information (Copeland and Galai, 1983). It is therefore key for market makers to efficiently extract information from LOB events to quickly update their quotes.

The target variable I define below is tailored to capture the payoff of a strategic market maker

[8]This is the case because I drop observations with positive 'trading halt' indicator. I do not add any indicator for submission events to avoid redundancy.

posting buy market orders and facing adverse selection. More specifically, it measures the profit earned (or the loss suffered) by a market maker whose bid is matched to an incoming sell market order, posing that she wants to minimize her risk exposure by closing the long position on the stock as fast as possible. I assume the market maker (passively) buys the stock at the prevailing bid price and, immediately afterwards, she tries to unload the position by posting sell limit orders at the prevailing ask price. In case she finds a counterpart buying the stock at a reasonably high ask price, she makes a profit equal on average to the bid/ask spread. Otherwise, if she does not find a buyer within some pre-determined time range, I assume she actively unloads the position submitting a market order at the bid price.

Formally, for each ordered sequence of LOB events $[t_0, t_1]$ let $\mathcal{B}(t_0, t_1)$ denote the subset of buyer-initiated executions during that interval, that is,

$$\mathcal{B}(t_0, t_1) = \{\tau \in [t_0, t_1] \text{ s.t. } \text{Direction}(\tau) \times \text{Execution}(\tau) = -1\} \tag{15}$$

Next recall that $ap(1, t)$ and $bp(1, t)$ denote the best offer prices on the ask and bid sides of the LOB, respectively. I define the *minimal execution price*, that is, a proxy for the price at which the market maker can actively close a long position, as

$$\mathcal{P}_1(t_0, t_1) = \begin{cases} \min_{\tau \in \mathcal{B}(t_0, t_1)} & ap(1, \tau) \quad \text{if } \mathcal{B}(t_0, t_1) \neq \varnothing \\ & bp(1, t_1) \quad \text{otherwise} \end{cases} \tag{16}$$

Finally, setting the prediction horizon to $H > 0$, I define the time-t target variable as

$$\mathscr{T}(t, H) = \frac{\mathcal{P}_1(t, t + H) - bp(1, t)}{bp(1, t)} \times 10^2 \tag{17}$$

Notice that $\mathscr{T}(t, H)$ provides a lower-bound for the percentage return earned by a market maker who provides liquidity to an incoming sell order at time t and closes the resulting long position within the H following LOB events. In reality, a sophisticated market maker could increase her profit by selling at a higher price, e.g. at the *maximum* executed sell order rather that at the *minimum* as in equation (16). I nevertheless decide to be conservative, so that $\mathscr{T}(t, H)$ is more effective at measuring the down-side risk of the market-making strategy.

3.4 Problem Statement

The prediction problem I analyze in this paper is motivated by the situation faced by a human market maker providing liquidity in a limit order market. More specifically I focus on one side of the decision-making process, assuming that the market maker acts as a counterpart to *sell* market orders. Even though the symmetrical problem of liquidity provision to *buy* market orders is equally interesting, I choose to focus on the sell side because of its connection with the issue of flash crashes (Kirilenko et al., 2017, Easley et al., 2011).

As noted in the above section it is crucial for a market maker to extract information from LOB events in a fast and efficient fashion, to reduce the risk of being "picked off" by speculators in case of the arrival of new information. My objective is to apply supervised learning models to help dealing with such a task. The input to the model is the time-series describing the most recent order book dynamics, represented by the features defined in Section 3.2. The models are trained to use this information set to predict the future return earned by the market maker, as defined in Section 3.3.

I now formally state the prediction problem studied in the paper. Let $F(\tau) \in \mathbb{R}^{25}$ denote the vector of features described in Section 3.2, at time τ. Setting the number of lagged features to $T > 0$, the sequence of features that will be used as input to produce a prediction for time t is given by $\mathscr{F}(t, T) = [F(t - T), \ldots, F(t - 1)]$. The task of the candidate machine learning model is predict the target variable $\mathscr{T}$ using the backward-looking information set $\mathscr{F}$. In other words I want the model to best approximate the function $\mathscr{F}(t, T) \mapsto \mathscr{T}(t, H)$ for each example indexed by t in the dataset. For the experimental results described in Section 4, I set the number of lags to $T = 50$ and the prediction horizon to $H = 100$.

Notice that the setting is fundamentally different from the problems explored by the recent literature on machine learning methods applied to LOB data, in two dimensions. First the target variable I try to predict is not based only on the future change in the mid-price as in Lee and Mykland (2007), Sirignano (2019), Passalis et al. (2019). In particular, the target variable contains information on both price levels and executions, taking into account the future orderflow hitting the LOB. This aspect is key for any predictive model which aims to be helpful for real-world market makers. Second the target variable is continuous, which allows to state the prediction task as a regression problem rather than a classification problem as in the above cited works. In this regard, the approach has the advantage to deliver a more informative evaluation of the models' performance and, further, it does not require to choose arbitrary thresholds to define categorical variables.

3.5 Training Procedure

This section describes the training and evaluation procedures, employed for each of the four supervised learning models considered in the study.

The training period ranges from the 22nd to the 27th of February 2019, while the 28th of February is used as validation. The test period goes from the 1st to the 7th of March 2019. I use only data recorded during trading hours, from 9:30 AM to 16:00 PM. To be conservative, I further remove the first and the last minute of each trading day. This ensures that lagged features do not belong to the previous trading day.

All models are trained separately for each sample stock using stochastic gradient descend (SGD) and the Adam optimizer proposed by Kingma and Ba (2014). A training epoch consists of 50 batches, with each batch containing 1024 examples. I use the same random seed to generate the training batches, ensuring that differences in the models' performance are not driven by random differences in the training process.

Since all the models are smooth and differentiable, parameters can be learned by standard back propagation with mean squared error (MSE) as the loss function

$$\mathscr{L} = \frac{1}{N}\sum_{t}(y_t - \hat{y}_t)^2 \tag{18}$$

where N is the sample size. To limit over-fitting, the end of the training is triggered by an early stopping rule (Prechelt, 1998) based on the validation MSE, with patience parameter set to 5 epochs. The training and prediction procedures are run on a machine featuring a *NVIDIA Tesla P4* GPU unit, which allows a considerable speed-up relative to CPUs. The process for the 10 sample stocks took about 5 hours to run.

4 Experimental Results

In this section I report experimental results on the prediction problem described in Section 3.4. As explained above, the target variable is a proxy for the profits earned by a market maker who provides liquidity to incoming sell market orders. I compare the out-of-sample performance of the four models outlined in Section 2, using data for 10 stocks traded in the NASDAQ exchange. The test period spans one trading week, from the 1st to the 7th of March 2019, and comprises roughly 26 million test samples.

I compare each model's performance using two standard evaluation metrics for regression problems. The first metric is the Root Mean Squared Error, defined as

$$RMSE = \left(\frac{1}{N}\sum_{i=i}^{N}(y_i - \hat{y}_i)^2\right)^{1/2} \tag{19}$$

where y_i is the target variable described in Section 3.3, expressed in basis points, $\hat{y}_i$ is the model's prediction based on backward-looking features and i runs over the samples of the test set for a given stock. The second metric is the Fraction of Variance Unexplained $FVU = 1 - R^2$, where R^2 is the coefficient of determination of a linear regression of the realized target variable y_i on the model's prediction $\hat{y}_i$. Regressions are run at the stock-level over the test data using a standard OLS estimator.

Results are reported in Table 2, where the two panels display the RMSE the FVU, respectively. Panel (a) shows that my attention-based model achieves, on average, lower out-of-sample

	Attention (1)	No Attention (2)	DeepLOB (3)	Neural Net (4)
AAL	**3.334**	3.350	3.346	3.464
AAPL	**0.986**	0.988	0.989	1.063
ADBE	**3.044**	3.123	3.095	3.111
CMCSA	1.661	**1.658**	1.668	1.767
EBAY	**2.007**	2.055	2.018	2.151
FB	**1.599**	1.607	1.604	1.621
MSFT	**0.970**	0.976	0.979	1.045
NVDA	**3.009**	3.038	3.068	3.078
PEP	**1.456**	1.460	1.466	1.509
TSLA	**4.454**	4.590	4.594	4.623
Average	**2.252**	2.284	2.283	2.343

(a) Root Mean Squared Error

	Attention (1)	No Attention (2)	DeepLOB (3)	Neural Net (4)
AAL	**93.85%**	94.83%	94.45%	97.57%
AAPL	**97.49%**	97.82%	98.15%	99.41%
ADBE	**93.62%**	98.94%	96.81%	97.04%
CMCSA	79.30%	**79.14%**	79.72%	88.07%
EBAY	**83.78%**	87.89%	84.73%	92.51%
FB	**96.73%**	97.77%	97.30%	96.85%
MSFT	**91.76%**	93.15%	93.73%	97.23%
NVDA	**95.89%**	97.13%	98.52%	97.70%
PEP	97.67%	98.21%	99.02%	**97.04%**
TSLA	**91.90%**	97.59%	97.71%	95.92%
Average	**92.20%**	94.25%	94.01%	95.93%

(b) Fraction of Variance Unexplained

Table 2: Experimental Results. The table reports out-of-sample experimental results for 10 stocks traded in the NASDAQ exchange. Four predictive models are compared: (1) 'Attention' is the attention-based encoder-decoder model proposed in this paper; (2) 'No'Attention' is similar to the encoder-decoder model (i), but with the attention layer bypassed; (3) 'DeepLOB' is the model proposed in Zhang et al. (2019), based on a convolutional layer followed by a LSTM layer; (4) 'Neural Network' is a plain-vanilla fully-connected neural network. The root mean squared error computed as $RMSE = (\sum_i (y_i - \hat{y}_i)^2/N)^{1/2}$, where y_i is the target variable described in Section 3.3, expressed in basis points, and i runs over the samples of the test set. The fraction of variance unexplained is computed as $FVU = 1 - R^2$, where R^2 is the coefficient of the determination from a linear regression of the target variable on the model's predictions, estimated on the test set.

prediction errors relative to the benchmark models. This is the case for the vast majority of the sample stocks, with the exception of CMCSA for which the lowest RMSE is obtained with the plain encoder-decoder model. I notice that the CNN-LSTM model (DeepLOB) achieves an accuracy which is virtually identical to that of the encoder-decoder model without attention layer, while the plain-vanilla neural network performs significantly worse.

A similar pattern is observed in Panel (b) for the FVU metric. The attention-based model is is able to explain a larger fraction of the variability in the target variable for most of the sample stocks, relative to the benchmark models. As for the RMSE the average performance of the DeepLOB model is comparable to that of the encoder-decoder model with no attention layer, while the plain-vanilla neural network performs significantly worse.

For some of the sample stocks in particular, the increase in prediction accuracy is substantial. Considering e.g. the ADBE stock, the rightmost panel of Table 2 shows that the attention-based model is able to explain more than 6% of the variation in the target variable. Remarkably, this translates roughly into a two-times increase relative to the amount of variation explained by the second-best model, that is the CNN-LSTM network by Zhang et al. (2019).

These results show that my attention-based model is superior, in terms of prediction accuracy, to the benchmark models with respect to the regression problem considered in this paper. In particular, they demonstrate that the addition of the attention-layer to the encoder-decoder network generates a significant improvement. This result supports the intuition that the attention layer provides more flexibility to the model, freeing it from the need to condensate the relevant information on past LOB events in a single vector of fixed dimension. Moreover it implies that the model has learned how to allocate its attention to the most relevant events of the input time-series, thus justifying the identification assumption at the base of the inference exercise described in the next section.

5 Making Use of Attention

This section explores the intelligibility improvements resulting from the inclusion of an attention layer to deep neural network models. I first present a novel methodology which, in principle, allows to make data-driven inference from any neural network trained on time-series data and augmented with an attention-layer. I then provide a showcase application in the context of market microstructure, in which the trained model is used to make inference on the relative information content of market orders and limit orders. Finally, I further exploit the attention layer to improve the visualization of the neural network's decision-making process.

5.1 Inference Methodology

In this Section I describe my novel inferential methodology and apply it to the context of information dissemination though orderflow. The idea is based on the following observation on the functioning of neural networks augmented with an attention layer. In order to forecast a given target variable using time-series data, the model has to *jointly* learn two distinct tasks. The first one is to identify in the input data the most relevant information with respect to the forecasting problem, while the second one is to process such information to actually produce the forecast.

Standard deep neural networks[9] perform these two tasks by first finding a sequence of non-linear transformations of the input space, one for each of the layers, and then using the transformed data to generate the final prediction. Deep networks have been empirically proven to be highly efficient and, in fact, the ability to transform input data over multiple steps is one of the most prominent rationalization for the adoption of a deep architecture. The drawback

[9] Neural network are considered *deep* if they contain at least four hidden layers of neurons between input and output.

is that the large number of required neurons and the fully connected network linking them together generate a high level of complexity, making it very difficult for humans to comprehend and make sense of the transformations chosen by the network. The situation becomes even more complicated when dealing with time-series data, where recurrent neural networks have been showed to be the best-performing architecture.

In an attempt to tackle this intelligibility problem, I argue that adding an attention layer to neural networks makes it possible to extract at least some information on their inner configuration. The reason is that attention-based neural networks applied to time-series forecasting are designed to highlight the most relevant parts of the input sequence by assigning to them higher levels of attention. Even though it does not help us to shed light on the transformations performed by the deep part of the network, the attention layer provides insights on the relative importance of each time-step in a clean and intelligible fashion.

Such a feature of attention-based models provides a straightforward way to make inference on the relative information content of different events in the context of time series forecasting. First, the researcher recovers the attention vector $\boldsymbol{\alpha}$ defined in equation (6) and generated by the trained model for each sample e of the test set. This gives a set of attention levels $y_{e,t}$ associated to time-step t of the input of e. Second, she constructs a set of categorical (or continuous) variables $X_{e,t}$ describing any characteristic of interest associated to time-step t of the input sequence of e. Finally, the researcher pools the resulting data from the entire test set and runs a panel regression of the attention levels $y_{e,t}$ on the characteristics $X_{e,t}$. The estimated beta coefficients can be interpreted as the relative importance of each of the characteristics of interest, thus making it possible to make inference on the value of their informational content.

One of the advantages of this approach is that the inference is based on linear regression estimators. First of all, results from a panel regression are relatively easy to interpret and communicate. Moreover, regressions provide out-of-the-box confidence intervals and p-values (potentially based on clustered standard errors) and a number of other standard metrics. Further, it allows to easily perform multi-variate analysis and include specifications with fixed effects, thus enabling the researcher to properly perform inference.

5.2 Market Orders versus Limit Orders

I now apply the inference methodology described in Section 5.1 to test if certain kinds of LOB event are more informative than others for price discovery, that is if they are more useful to construct a forecast of future prices. The list of LOB event types I consider is exhaustive and includes: (i) the execution of a market order; (ii) the submission of a new limit order or the update of an outstanding limit order; (iii) the full or partial cancellation of an outstanding limit

order. As argued in the introduction, standard microstructure models typically assume that market orders are the most informative events for price discovery, because they are submitted by traders holding private information and, therefore, willing to trade fast and capitalize on their informational advantage. The analysis of this section uses the the attention allocation of my attention-based model to provide a data-driven method to validate or reject such an assumption. Formally, I want to test validity of the following hypothesis:

H0: *Market-order executions are more informative than the submission or cancellation of limit orders with respect to price discovery, that is to the forecasting of future prices.*

In the empirical microstructure literature, a standard approach to test the above hypothesis is to fit a Vector Auto Regression (VAR) model to LOB data and perform a variance-decomposition exercise (Hasbrouck, 1991a,b, Brogaard et al., 2014, 2016). A limitation of such a methodology arises from the fact that the VAR is a linear model and, most importantly, that it has been shown by Sirignano and Cont (2018) to under-perform the forecasting ability of non-linear neural network models. My methodology provides an alternative approach based on neural networks, thus overcoming the limitations of the VAR model and allowing for an arbitrary data-driven parametrization of the relationship between LOB dynamics and future prices.

To test the null hypothesis **H0**, I first recover the attention levels produced by the trained model for each of the 51200 examples in the test set. I then construct three dummy variables indicating the three event types, thus categorizing each of the 50 input time-steps of each example as *execution*, *submission* or *cancellation*. Per-minute summary statistics of such a classification are reported in Table 1. Finally, I run the panel regressions described by

$$\begin{aligned} \text{Attention}_{e,t} \;=\; \alpha \;&+\; \beta_0 \,\text{Event Type}_{e,t} \\ &+\; \beta_1 \,\text{Elapsed Time}_{e,t} \\ &+\; \beta_2 \,|\,\text{Price Change}_{e,t}\,| \;+\; \varepsilon_{e,t} \end{aligned}$$

where $\text{Attention}_{e,t}$ is the attention level associated to time-step t of the input sequence of example e and $\text{Event Type}_{e,t}$ is one the above described dummies, depending on the specification. Controls include $\text{Elapsed Time}_{e,t}$ and $|\,\text{Price Change}_{e,t}\,|$, measuring the time elapsed and the absolute mid-price change, respectively, from the previous LOB event. Standard errors are conservatively clustered by Day $\times$ Stock to account for potential serial correlation in residuals.

Results, reported in Table 3, show that the coefficient on the dummy indicating executions is positive and highly statistically significant. This implies that attention levels are significantly higher during the executions of market orders (+6%) and, in turn, that the model over-weights

	(1)	(2)	(3)	(4)	(5)	(6)
Dependent Variable	Attention Level	Attention Level	Attention Level	Attention Level	Attention Level	Attention Level
Execution	6.00***			5.93***		
	(5.63)			(5.59)		
Cancellation		-1.46***			-1.45***	
		(-7.96)			(-7.98)	
Submission			0.88***			0.88***
			(5.72)			(5.71)
Elapsed Time				1.81***	2.06***	2.08***
				(3.62)	(4.22)	(4.26)
|Price Change|				0.24***	0.56***	0.72***
				(3.35)	(4.17)	(4.82)
Constant	1.85***	2.71***	1.57***	1.78***	2.62***	1.47***
	(66.26)	(30.11)	(20.93)	(58.83)	(30.01)	(18.14)
Observations	2,560,000	2,560,000	2,560,000	2,560,000	2,560,000	2,560,000
R-squared	0.03	0.02	0.01	0.03	0.02	0.01
SEs Clustered By	Date×Stock	Date×Stock	Date×Stock	Date×Stock	Date×Stock	Date×Stock

Table 3: Attention Drivers. The table reports results from a panel regressions at the example-event level of attention levels onto dummy variables indicating three types of LOB events (executions, submissions and cancellations). For each example in the test dataset, attention levels for each time-step of the input sequence are extracted from the attention-based model described in Section 2 trained on the training set as described in Section 6.2. In columns (4), (5) and (6) controls for the time elapsed and the absolute price change from the previous LOB event are added to the regression. t-stats are reported in parentheses, based on standard errors clustered at the Day × Stock to account for potential serial correlation in residuals. Asterisks denote significance levels (***= 1%, **= 5%, *= 10%).

the corresponding time-steps to produce the final predictions. On the contrary, the negative coefficient on the cancellation dummy implies that the model allocates less attention to cancellation events. Submission events are somewhat in between, with a slight but significant increase in attention of less than 1 percentage points relative to the average.

Importantly, these coefficients are stable when controls are added to regression in specifications (4), (5) and (6). These results ensure that the heterogeneity in attention levels is not solely motivated by bid-ask bounce effects or by illiquidity. I can therefore rule out an alternative explanation based on the fact that executions may mechanically induce predictability because of the bid-ask bounce.

All in all, the results of this analysis cannot reject the null hypothesis **H0**. In other words, executions of market orders are by far the most informative events in LOB dynamics to forecast future transaction prices. Even though submission of limit orders are also deemed

informative, the model assigns a level of attention that is an order of magnitude lower with respect to executions. Cancellations of outstanding limit orders, instead, seem to play a limited informational role.

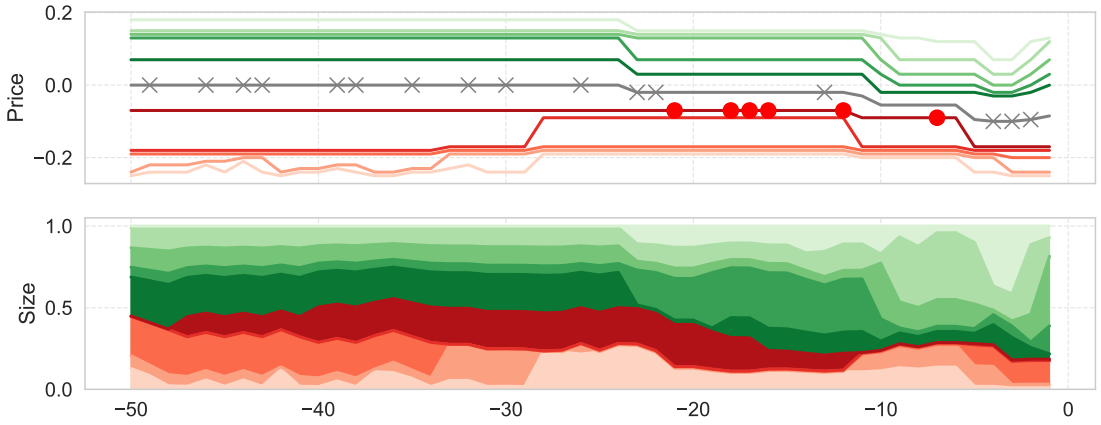
5.3 Visualization

Machine learning models are heavily criticized for their black-boxiness nature and their lack of interpretability. However, as argued by Lipton (2016), the interpretability of a supervised learning model is far from being a well-defined concept, as it refers to several distinct ideas and tries to address the need for desiderata from the model distinct from prediction accuracy. One characteristic to expect form an *ideal* model is a high level of informativeness, that is, its ability to convey information to a human decision-maker through its outputs or additional procedures. I argue that my attention-based model is preferable to alternative model architectures in this dimension, because its attention layer allows to produce visualizations that convey more intuition about the decisions taken by the model.

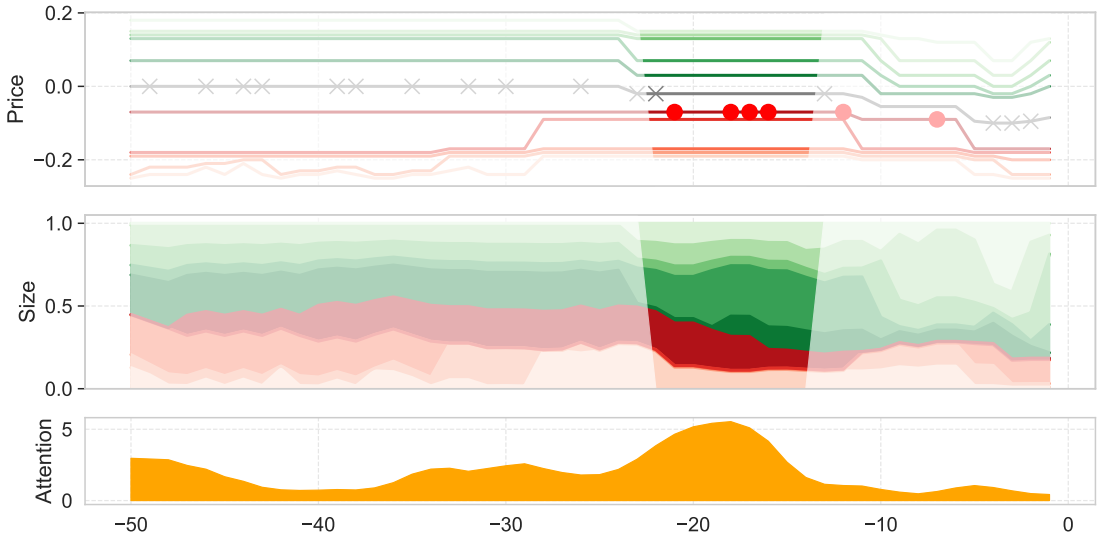
The computer science literature provide a large number of studies in which attention has been used to improve visualization, leading to useful insights of the inner working of neural network models (Bahdanau et al., 2014, Choi et al., 2016). A widely cited work by Xu et al. (2015), for instance, devise a neural model with visual attention and train it to generate textual captions from images. The attention layer helps the model to achieve higher accuracy relative to previous architectures and, moreover, it allows to visually inspect on which part of the image the model focuses to generate each word of the caption. Interestingly, the authors analyze a sample of model's mistakes, and use attention to gain intuition into what the model erroneously saw.

To showcase how the attention layer can be helpful in producing insightful and informative visualizations in the context of this work, consider Figure 2, representing the time-series of input features for a specific example in the test set of the Tesla stock, for which the predicted return is highly positive. The top panel displays the evolution of the mid-price (in gray) and five price levels of outstanding limit orders on the ask side (green) and the bid side (red), in percentage distance from the mid-price. Green (red) dots mark buyer (seller) initiated executions, while gray crosses represent order deletions. The second panel displays the evolution of the ask (in green) and bid (in red) sizes, presented as a percentage of the total liquidity available in the first five levels of the LOB. These two panels surely contain a significant amount of information but, unfortunately, a human decision maker cannot easily understand what is the rationale behind the model's decision to produce a highly positive forecast for future price movements.

The situation can be improved using the attention distribution $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_T$ associated to this



(a) Visualization without Attention



(b) Visualization with Attention

Figure 2: Visualization The figure provides a graphical representation of the time-series of features for a specific event in the test set of the Tesla stock, for which the predicted return is highly positive. The first plot of both panels (a) and (b) display the evolution of the mid-price (in gray) and five levels on the ask side (green) and the bid side (red), in percentage distance from the mid-price. Green (red) dots mark buyer (seller) initiated executions, while gray crosses represent order deletions. The second plot of both panels (a) and (b) display the evolution of the relative ask (in green) and bid (in red) sizes, presented as a percentage of the total liquidity in shares available in the first five levels of the LOB. Panel (b) includes an additional third plot, displaying the temporal attention distribution (in percentage) employed by the model to make the prediction, that is, the attention vector $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_T$ associated to this particular event. Using this distribution, the time steps in the other two plots corresponding to a level of attention below 2% are shaded, while the region where the model focuses the most are highlighted.

example, which is displayed in the bottom panel. I highlighted the time steps on which the model has decided to focus, shading away those for which the level of attention is below the 2% level. This adjustment makes the plot more informative for a human viewer, allowing her to focus on the most relevant part of the image. In this case, the positive forecast of the model seems to depend on a series of four large sell executions that consume a significant portion of liquidity on the bid side of the book.

6 Liquidity Provision to Institutional Block Orders

The rise of electronic trading platforms resulted into an increased presence of high-frequency traders (HFTs) in equity markets. Members of this new category of traders are described by the Securities and Exchange Commission (SEC) as "professional traders" using "extraordinarily high-speed and sophisticated computer programs for generating, routing, and executing orders". These new market participants and their impact on market functioning have been investigated by a number of academic studies, the vast majority of which finds that HFTs activity results into reduced bid-ask spreads and improved price efficiency (Hendershott et al., 2011, Chaboud et al., 2014, Boehmer et al., 2014, Brogaard et al., 2015).

Observers in the financial press and practitioners in the industry, however, claim that algorithmic traders may harm market liquidity by front-running block orders of institutional investors[10]. Block orders are large trades which are split over time into multiple executions (child orders) in an attempt to minimize their market impact. One of the main concerns is that algorithms may identify such orders recognizing patterns in the initial part of their execution and, then, front-run the remainder of the block – a strategy usually referred to as *back-running*(Van Kervel and Menkveld, 2019). This issue is of first importance to institutional investors, since their effective trading costs do not depend only on bid-ask spreads but rather on *implementation shortfall*, that is, the cumulative price impact of their block orders.

In principle, the neural network model described in this paper could learn from order book data to recognize blocks before they are fully executed and autonomously engage in back-running strategies, imposing externalities on institutions. Assuming my model – or a similar one – is used by real-world market makers and HFTs, such an automatic tendency to front-run may result in a coordinated action during block trades and induce a systematic reduction of liquidity exactly when institutions need it the most. This dynamics could increase trading costs for institutional investors and, moreover, it may pose significant threats to market stability and

[10] See, for instance, *Wealth Fund Cautions against Costs Exacted by High-Speed Trading*, The New York Times, October 2013; *High-Frequency Trading is Basically Evil* Berkshire Munger, May 2013; and *Institutional Investors Air HFT Concerns* Financial Times, September 2011.

Aggregate Statistics		Event-level Statistics	
Number of events	716	Average block duration (minutes)	39.91
Number of institutions	51	Average number of trades	22.89
Number of stocks	78	Average dollar volume (million $)	6.61
Number of days	234	Average volume ratio wrt CRSP	2.36%

Table 4: Block-Trade Events Summary Statistics. The table reports summary statistics on the identified block-trade events, including aggregate information and event-level averages by minute. The events are defined as collections of executions by an institutional investor such that: (i) the collection consists of at least 5 distinct trades; (ii) the trades are about a single stock j; (iii) the trades are on the same side (buy or sell); (iv) the trades are executed on a single trading day t; (v) the trades are executed during a period of at least 5 minutes and at most 1 hour; (vi) the aggregate volume generated by the trades is at least 1% of the trading volume for stock j reported in CRSP on day t.

lead to excess volatility and even to flash crashes. Understanding how these kind of models behave in the presence of block orders is therefore a relevant issue for both institutional investors and regulatory agencies.

To shed light on this question, I test the behavior of my model during real block orders by institutions. The empirical strategy is based on the ex-post identification of block orders in the AbelNoser/Ancerno dataset, containing transaction-level data from mutual funds and other large institutional investors. In particular, I use data from the year 2014 and impose the conditions outlined in Section 6.1, resulting into a large number of distinct block-order events. Once these large transactions are identified, for each event I train my model in the preceding days and then produce out-of-sample predictions during the actual block order execution and during placebo periods. To conclude the analysis, I run a panel regression of the out-of-sample predictions of the model onto a categorical variable indicating the presence and the side of the block orders. Regression results provide evidence both on the ability of the model to detect block orders from the orderflow and, most importantly, on its relative tendency to engage in front-running or liquidity provision strategies.

6.1 Block Trades Identification

The AbelNoser/Ancerno database contains execution-level data for a large number of institutional investors in the US equity market. Among other information, each row in the dataset reports the institution, the traded stock, the share volume, and the execution price and timing. This structure makes it possible to identify block orders, by searching for multiple repeated executions by a single institution for the same stock. To give the model enough time to rec-

ognize the block, I impose that the sequence of executions is spread over at least 5 minutes[11]. However, I also impose a limit of 1 hour on the duration of each block, to guarantee enough precision of the categorical variable indicating the presence of a block, used in the regression analysis of Section 6.3. Formally, I define a block order $B(i, j, t)$ by institution i on stock j during day t as a collection of trades satisfying the following conditions:

(i) The collection consists of at least 5 distinct trades;

(ii) The trades are about a single stock j;

(iii) The trades are on the same side (buy or sell);

(iv) The trades are executed on a single trading day t;

(v) The trades are executed during a period of at least 5 minutes and at most 1 hour;

(vi) The aggregate volume generated by the trades is at least 1% of the trading volume for stock j reported in CRSP on day t.

The universe of stocks potentially subject to a block order is restricted to the 100 stocks with the highest trading volume in the Nasdaq Stock Exchange as of August 2019. This ensures that the corresponding LOB data is available in the LOBster dataset.

The procedure results in the identification of 716 block trades by 51 different institutions on 78 distinct stocks[12]. Summary statistics, reported in Table 4, show that the identified block-trade events are sizeable in dollar terms and as a fraction of the stock daily volume. Moreover the blocks are split significantly and carried over long periods of time, as they comprise on average more than 20 child executions and last for about 40 minutes on average. Finally, the fact that they are spread overs 234 trading days ensures that they do not arise from specific calendar time events but, rather, these kind of block trades are a systematic feature of US equity market.

6.2 Training Procedure and Predictions

For each block-trade event $B(i, j, t)$ on stock j during day t, I collect data describing the LOB events of that stock during days $t-3$, $t-2$, $t-1$ and t (in business-day units). I use these data to re-train the attention-based model from scratch for each event on the prediction task described in 3.4, that is, to forecast the target variable described in Section 3.3, representing

[11] Even though this may seem a short time frame, Table 1 shows that a window of 5 minutes corresponds on average to more than 1000 LOB events.

[12] The list of stocks subject to a least one block order is the following: AAL, ADBE, ADI, ADSK, ALGN, ALXN, AMAT, AMGN, ATVI, BIIB, BMRN, CDNS, CELG, CERN, CHTR, CMCSA, COST, CSCO, CSX, CTAS, CTSH, DLTR, EA, EBAY, EXPE, FISV, FOX, FOXA, GILD, GOOG, GOOGL, HAS, HSIC, IDXX, ILMN, INCY, INTC, INTU, ISRG, JBHT, KLAC, LRCX, LULU, MAR, MCHP, MDLZ, MELI, MNST, MSFT, MXIM, MYL, NFLX, NTAP, NVDA, ORLY, PAYX, PCAR, PEP, QCOM, REGN, ROST, SBUX, SIRI, SNPS, SWKS, SYMC, TMUS, TSLA, TTWO, TXN, UAL, ULTA, VRSK, VRTX, WDAY, WDC, XEL, XLNX

the expected return of a market maker providing liquidity to the market. In particular, the three business days preceding each block trade are used as training set ($t-3$ and $t-2$) and validation set ($t-1$), using the same procedure as that described in Section 3.5.

Once the model is trained, I use it to produce predictions for each event of the block-trade day. These predictions are then averaged at the minute-frequency and standardized using the mean and standard deviation of the distribution of predictions for each stock.

6.3 Regression Results

Once the event-minute level predictions described in the previous Section are linked to the panel of block-trade events, I construct the categorical variable "Block Order Side", defined at the event-minute level and taking values in $\{-1, 0, +1\}$. In particular, "Block Order Side(B, m)" takes the value of -1 (respectively $+1$) when the corresponding block order B is executing during minute m and it represents a *sell* (respectively *buy*) order by the institutional investor. In those minutes in which no block order was identified by the procedure described in Section 6.1, instead, the variable is set to 0. Notice that by construction the "Block Order Side" variable is equal to zero in the majority of the data points, since block orders are required to last at most for one trading hour. Indeed, in the final panel data, it takes non-trivial values in about 8% of the observations.

To test whether my attention-based model has a tendency to front-run or back-run institutional investors' block orders, I run a regression of the model's predictions on the "Block Order Side". Formally, we estimate the baseline regression model

$$\text{Model Prediction}(B, m) = \alpha + \beta_0 \, \text{Block Order Side}(B, m) + \varepsilon(B, m), \tag{20}$$

and more stringent specifications with stock and day fixed effects, to control for stock-specific and date-specific events that could bias the model's prediction in a specific direction. The last specification includes day×stock fixed effects, which allows to compare the predictions of the model within each stock-day, focusing on the differences between different minutes conditional on the presence of a block order.

Results reported in Table 5 show that the coefficient on β_0 highly significantly different from zero across all specifications, proving that the model's predictions are strongly correlated with the presence of executing block orders. This suggests that the model is able to identify block-trades solely from anonymous orderflow information provided by LOB dynamics. This is a striking result, given that the model has no direct information on institutional investors trades and has not been trained specifically to detect them. Moreover, the fact that the coefficient is negative indicates that the model is significantly more likely to forecast a profit by taking the opposite side of the block order. In other words, the attention-based model suggests the

	(1)	(2)	(3)	(4)
Dependent Variable	Model Prediction	Model Prediction	Model Prediction	Model Prediction
Block Order Side	-0.08***	-0.08***	-0.06***	-0.05**
	(-3.39)	(-3.30)	(-2.74)	(-2.57)
Constant	-0.01			
	(-0.64)			
Observations	277,112	277,112	277,112	277,112
R-squared	0.01	0.01	0.13	0.14
Stock Fixed Effects		Yes		Yes
Day Fixed Effects			Yes	Yes
SEs Clustered By	Stock-Day	Stock-Day	Stock-Day	Stock-Day

Table 5: Model's Predictions during Block Orders. The table reports results from panel regressions of minute-level average predictions of the attention-based model's onto the categorical variable Block Order Side(B, m), taking values in $\{-1, 0, +1\}$ and indicating the presence of a sell block order (-1), of a buy block order $(+1)$ or of no block order during that minute. Block orders are identified imposing the conditions described in Section 6.1 and predictions are generated by the attention-based model trained from scratch and validated on the three business days preceding each block order event. t-statistics are reported in parentheses, based on two-ways standard errors clustered by stock and by day to account for potential serial correlation in residuals. Asterisks denote significance levels (***= 1%, **= 5%, *= 10%).

market maker to buy the stock during liquidations by an institutional investor and, on the contrary, to sell it (or simply not to buy it) when the institution is building up a position on the stock.

All in all these results suggest that the attention-based neural network model studied in this paper, if employed by real world market makers, could potentially improve available liquidity for institutional investors and reduce their trading costs. It is worth remarking that this positive externality for institutions does not imply that market makers are worse off. On the contrary, assuming that the model produces a valuable forecast of their expected profits, market makers should benefit by using its predictions as a signal to adjust their limit orders. This result is consistent with the view that market makers can profit from block orders by engaging in liquidity provision strategies and being compensated for immediacy.

Is liquidity provision the optimal strategy available to market makers? This is an open question, whose answer crucially depends on the information set they have at their disposal. A recent work by Barbon et al. (2019) provides empirical evidence of orderflow leakage by brokers during fire sales of their institutional clients. The authors also show that the market players

who receive the information engage in predatory strategies and earn significant profits. A potential interpretation of the results from this section is that the attention-based model is simply not able to perfectly re-construct the first-hand information available to brokers and, consequently, it cannot forecast the total size and duration of block orders.

7 Concluding Remarks

In this paper I argue that machine learning models can be used to make inference on economic questions, providing a showcase example based on an attention-based neural network. In the first step of the analysis I use LOB data to show that my model is more effective than other architectures in providing accurate forecasts of future transaction prices. I then leverage on this result to justify the use of the attention distributions generated by the model as a proxy for the informational content of different LOB events with respect to price discovery. My results provide empirical evidence of the importance of market order executions for price discovery, while showing that limit orders dynamics play a limited role.

The idea of using attention-based neural network models as inferential device to explore economic issues is novel and could be further explored by future research. For instance, the same method can be applied to different settings to answer different economic questions. Moreover, the attention mechanism could be applied to the cross-sectional rather then the time-series dimension, providing a non-linear way of assessing the relative importance of different factors.

In the second part of the paper, I study how the model behave when faced with the execution of block orders from institutional investors. Results from this exercise suggest that the model favors liquidity provision rather than front-running strategies. However, to better understand the impact that algorithms of this kind can have on market liquidity, further research is needed to analyze the behavior of alternative model architectures and alternative training strategies. For example, one could think of adopting a reinforcement-learning approach and, further, to train the model to directly identify block orders. This, in principle, could be done by constructing a training set with examples of ex-post identified block-order events.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Barbon, A., Di Maggio, M., Franzoni, F., and Landier, A. (2019). Brokers and order flow leakage: Evidence from fire sales. *The Journal of Finance*, Forthcoming.

Black, F. (1986). Noise. *The journal of finance*, 41(3):528–543.

Boehmer, E., Fong, K., and Wu, J. (2014). International evidence on algorithmic trading, singapore management university. Technical report, unpublished working paper.

Brogaard, J., Hagströmer, B., Nordén, L., and Riordan, R. (2015). Trading fast and slow: Colocation and liquidity. *The Review of Financial Studies*, 28(12):3407–3443.

Brogaard, J., Hendershott, T., and Riordan, R. (2014). High-frequency trading and price discovery. *The Review of Financial Studies*, 27(8):2267–2306.

Brogaard, J., Hendershott, T., and Riordan, R. (2016). Price discovery without trading: Evidence from limit orders. *The Journal of Finance*.

Chaboud, A. P., Chiquoine, B., Hjalmarsson, E., and Vega, C. (2014). Rise of the machines: Algorithmic trading in the foreign exchange market. *The Journal of Finance*, 69(5):2045–2084.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choi, E., Bahadori, M. T., Sun, J., Kulas, J., Schuetz, A., and Stewart, W. (2016). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512.

Copeland, T. E. and Galai, D. (1983). Information effects on the bid-ask spread. *the Journal of Finance*, 38(5):1457–1469.

Easley, D., De Prado, M. L., and O'Hara, M. (2011). The microstructure of the flash crash: Flow toxicity, liquidity crashes and the probability of informed trading. *Journal of Portfolio Management*, 37(2):118–128.

Farboodi, M. and Veldkamp, L. (2018). *Long Run Growth of Financial Data Technology*. Centre for Economic Policy Research.

Foucault, T., Kadan, O., and Kandel, E. (2013). Liquidity cycles and make/take fees in electronic markets. *The Journal of Finance*, 68(1):299–341.

Girija, S. S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

Glosten, L. R. and Milgrom, P. R. (1985). Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, 14(1):71–100.

Hasbrouck, J. (1991a). Measuring the information content of stock trades. *The Journal of Finance*, 46(1):179–207.

Hasbrouck, J. (1991b). The summary informativeness of stock trades: An econometric analysis. *The Review of Financial Studies*, 4(3):571–595.

Hendershott, T., Jones, C. M., and Menkveld, A. J. (2011). Does algorithmic trading improve liquidity? *The Journal of Finance*, 66(1):1–33.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Kirilenko, A., Kyle, A. S., Samadi, M., and Tuzun, T. (2017). The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, 72(3):967–998.

Kyle, A. S. (1989). Informed speculation with imperfect competition. *The Review of Economic Studies*, 56(3):317–355.

le Calvez, A. and Cliff, D. (2018). Deep learning can replicate adaptive traders in a limit-order-book financial market. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1876–1883. IEEE.

Lee, S. S. and Mykland, P. A. (2007). Jumps in financial markets: A new nonparametric test and jump dynamics. *The Review of Financial Studies*, 21(6):2535–2563.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Lipton, Z. C. (2016). The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Mäkinen, M., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2018). Forecasting of jump arrivals in stock prices: New attention-based network architecture using limit order book data. *arXiv preprint arXiv:1810.10845*.

Menkveld, A. J. (2018). High-frequency trading as viewed through an electron microscope. *Financial Analysts Journal*, 74(2):24–31.

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680. ACM.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2019). Deep adaptive input normalization for price forecasting using limit order book data. *arXiv preprint arXiv:1902.07892*.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*.

Riemer, M., Vempaty, A., Calmon, F., Heath, F., Hull, R., and Khabiri, E. (2016). Correcting forecasts with multifactor neural attention. In *International Conference on Machine Learning*, pages 3010–3019.

Shleifer, A. and Vishny, R. W. (1997). The limits of arbitrage. *The Journal of finance*, 52(1):35–55.

Sirignano, J. and Cont, R. (2018). Universal features of price formation in financial markets: perspectives from deep learning. *Available at SSRN 3141294*.

Sirignano, J. A. (2019). Deep learning for limit order books. *Quantitative Finance*, 19(4):549–570.

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

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

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2018). Using deep learning for price prediction by exploiting stationary limit order book features. *arXiv preprint arXiv:1810.09965*.

Van Kervel, V. and Menkveld, A. J. (2019). High-frequency trading around large institutional orders. *The Journal of Finance*, 74(3):1091–1137.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, pages 5998–6008.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Zhang, Z., Zohren, S., and Roberts, S. (2019). Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2:207–212.

Appendix

	AAL, N=3,486,402					AAPL, N=8,444,983					ADBE, N=2,047,355					CMCSA, N=4,696,675					EBAY, N=4,004,569				
	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%
AS(1)	0.07	0.05	0.00	0.06	0.25	0.05	0.05	0.00	0.04	0.20	0.09	0.07	0.00	0.08	0.31	0.09	0.06	0.00	0.08	0.27	0.10	0.07	0.01	0.08	0.33
AS(2)	0.10	0.05	0.02	0.09	0.30	0.09	0.05	0.01	0.08	0.25	0.10	0.07	0.00	0.09	0.32	0.11	0.04	0.05	0.10	0.25	0.11	0.05	0.03	0.10	0.31
AS(3)	0.11	0.06	0.02	0.09	0.38	0.11	0.05	0.03	0.11	0.29	0.10	0.06	0.00	0.09	0.32	0.11	0.04	0.05	0.10	0.22	0.10	0.05	0.03	0.09	0.26
AS(4)	0.11	0.06	0.03	0.10	0.35	0.13	0.05	0.04	0.12	0.32	0.11	0.06	0.00	0.09	0.32	0.10	0.04	0.05	0.10	0.21	0.10	0.05	0.04	0.10	0.27
AS(5)	0.11	0.06	0.03	0.10	0.32	0.13	0.06	0.04	0.12	0.34	0.11	0.07	0.00	0.09	0.34	0.10	0.04	0.04	0.09	0.21	0.10	0.05	0.03	0.09	0.27
BS(1)	0.07	0.06	0.00	0.06	0.26	0.05	0.04	0.00	0.04	0.18	0.08	0.07	0.00	0.07	0.29	0.09	0.06	0.00	0.08	0.27	0.10	0.06	0.00	0.08	0.30
BS(2)	0.11	0.06	0.02	0.09	0.33	0.08	0.04	0.01	0.08	0.21	0.09	0.07	0.00	0.08	0.30	0.11	0.04	0.05	0.10	0.24	0.10	0.04	0.03	0.09	0.26
BS(3)	0.11	0.07	0.02	0.10	0.37	0.11	0.04	0.03	0.10	0.24	0.10	0.06	0.00	0.09	0.31	0.11	0.03	0.05	0.10	0.22	0.09	0.04	0.03	0.09	0.22
BS(4)	0.11	0.07	0.03	0.10	0.37	0.12	0.04	0.03	0.12	0.26	0.11	0.06	0.00	0.09	0.32	0.10	0.03	0.05	0.10	0.18	0.10	0.04	0.03	0.09	0.23
BS(5)	0.11	0.07	0.03	0.10	0.38	0.13	0.05	0.03	0.12	0.27	0.11	0.07	0.00	0.10	0.33	0.09	0.03	0.04	0.09	0.17	0.10	0.04	0.03	0.09	0.24
AP(1)	0.02	0.01	0.01	0.01	0.04	0.00	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.02	0.06	0.01	0.00	0.01	0.01	0.03	0.01	0.00	0.01	0.01	0.03
AP(2)	0.05	0.01	0.04	0.04	0.07	0.01	0.00	0.01	0.01	0.02	0.03	0.01	0.01	0.02	0.08	0.04	0.00	0.04	0.04	0.05	0.04	0.00	0.04	0.04	0.05
AP(3)	0.08	0.01	0.07	0.07	0.10	0.02	0.00	0.01	0.01	0.02	0.03	0.02	0.01	0.03	0.10	0.07	0.00	0.06	0.06	0.08	0.07	0.00	0.06	0.07	0.08
AP(4)	0.10	0.01	0.10	0.10	0.13	0.02	0.00	0.02	0.02	0.03	0.04	0.02	0.02	0.03	0.11	0.09	0.00	0.09	0.09	0.10	0.10	0.00	0.09	0.09	0.11
AP(5)	0.13	0.01	0.12	0.13	0.16	0.03	0.00	0.03	0.03	0.03	0.04	0.02	0.02	0.04	0.12	0.12	0.00	0.11	0.12	0.13	0.12	0.00	0.12	0.12	0.14
BP(1)	0.02	0.01	0.01	0.01	0.04	0.00	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.02	0.06	0.01	0.00	0.01	0.01	0.03	0.01	0.00	0.01	0.01	0.03
BP(2)	0.05	0.01	0.04	0.04	0.07	0.01	0.00	0.01	0.01	0.02	0.03	0.01	0.01	0.02	0.08	0.04	0.00	0.04	0.04	0.05	0.04	0.00	0.04	0.04	0.05
BP(3)	0.08	0.01	0.07	0.07	0.10	0.02	0.00	0.01	0.01	0.02	0.03	0.02	0.01	0.03	0.10	0.07	0.00	0.06	0.06	0.08	0.07	0.00	0.06	0.07	0.08
BP(4)	0.10	0.01	0.10	0.10	0.13	0.02	0.00	0.02	0.02	0.03	0.04	0.02	0.02	0.03	0.11	0.09	0.00	0.09	0.09	0.10	0.10	0.00	0.09	0.09	0.11
BP(5)	0.13	0.01	0.12	0.13	0.16	0.03	0.00	0.03	0.03	0.03	0.04	0.02	0.02	0.04	0.12	0.12	0.00	0.11	0.12	0.13	0.12	0.00	0.12	0.12	0.14
Execution	0.02	0.13	0.00	0.00	1.00	0.03	0.16	0.00	0.00	1.00	0.04	0.21	0.00	0.00	1.00	0.01	0.11	0.00	0.00	1.00	0.01	0.11	0.00	0.00	1.00
Deletion	0.50	0.50	0.00	0.00	1.00	0.48	0.50	0.00	0.00	1.00	0.46	0.50	0.00	0.00	1.00	0.51	0.50	0.00	1.00	1.00	0.51	0.50	0.00	1.00	1.00
Direction	-0.04	1.00	-1.00	-1.00	1.00	-0.02	1.00	-1.00	-1.00	1.00	-0.01	1.00	-1.00	-1.00	1.00	-0.00	1.00	-1.00	-1.00	1.00	0.04	1.00	-1.00	1.00	1.00
Elapsed Time	0.07	0.26	0.00	0.00	1.14	0.03	0.09	0.00	0.00	0.41	0.11	0.43	0.00	0.00	2.02	0.05	0.20	0.00	0.00	0.92	0.06	0.24	0.00	0.00	1.09
Price Change	-0.00	0.27	-1.41	0.00	1.41	-0.00	0.09	-0.29	0.00	0.29	-0.00	0.27	-0.96	0.00	0.95	-0.00	0.13	0.00	0.00	0.00	-0.00	0.16	0.00	0.00	0.00

	FB, N=5,090,807					MSFT, N=11,800,393					NVDA, N=3,031,962					PEP, N=2,415,737					TSLA, N=2,097,592				
	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%	Mean	Std	1%	50%	99%
AS(1)	0.06	0.06	0.00	0.04	0.27	0.06	0.05	0.00	0.06	0.22	0.08	0.08	0.00	0.06	0.39	0.07	0.05	0.00	0.06	0.25	0.10	0.11	0.00	0.08	0.52
AS(2)	0.09	0.07	0.00	0.08	0.31	0.09	0.04	0.03	0.09	0.24	0.09	0.08	0.00	0.07	0.40	0.10	0.05	0.02	0.09	0.25	0.10	0.11	0.00	0.08	0.56
AS(3)	0.11	0.07	0.01	0.10	0.34	0.10	0.05	0.03	0.10	0.26	0.10	0.08	0.00	0.08	0.41	0.10	0.05	0.02	0.10	0.26	0.10	0.11	0.00	0.08	0.57
AS(4)	0.12	0.07	0.02	0.11	0.37	0.11	0.05	0.04	0.11	0.29	0.11	0.08	0.00	0.09	0.43	0.12	0.05	0.02	0.11	0.28	0.10	0.11	0.00	0.07	0.57
AS(5)	0.13	0.07	0.02	0.12	0.39	0.12	0.06	0.04	0.11	0.32	0.12	0.08	0.01	0.11	0.44	0.12	0.06	0.03	0.11	0.30	0.10	0.11	0.00	0.07	0.58
BS(1)	0.06	0.06	0.00	0.04	0.26	0.06	0.05	0.00	0.06	0.21	0.08	0.08	0.00	0.06	0.37	0.07	0.05	0.00	0.06	0.25	0.10	0.11	0.00	0.07	0.59
BS(2)	0.09	0.06	0.00	0.08	0.29	0.10	0.04	0.03	0.09	0.23	0.09	0.08	0.00	0.07	0.40	0.09	0.05	0.02	0.09	0.24	0.10	0.12	0.00	0.07	0.61
BS(3)	0.11	0.06	0.01	0.10	0.32	0.11	0.04	0.04	0.10	0.24	0.10	0.08	0.00	0.08	0.41	0.10	0.05	0.02	0.09	0.24	0.10	0.12	0.00	0.07	0.63
BS(4)	0.12	0.06	0.01	0.11	0.34	0.12	0.05	0.04	0.11	0.26	0.11	0.08	0.00	0.10	0.43	0.11	0.05	0.02	0.10	0.27	0.10	0.12	0.00	0.07	0.64
BS(5)	0.13	0.06	0.02	0.12	0.36	0.12	0.05	0.04	0.12	0.28	0.12	0.08	0.00	0.11	0.44	0.12	0.06	0.03	0.11	0.30	0.10	0.12	0.00	0.07	0.66
AP(1)	0.01	0.00	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.01	0.04	0.01	0.00	0.00	0.00	0.03	0.02	0.01	0.00	0.02	0.07
AP(2)	0.02	0.01	0.01	0.01	0.04	0.01	0.00	0.01	0.01	0.02	0.02	0.01	0.01	0.02	0.06	0.02	0.01	0.01	0.01	0.04	0.04	0.02	0.01	0.03	0.11
AP(3)	0.02	0.01	0.01	0.02	0.04	0.02	0.00	0.02	0.02	0.03	0.03	0.01	0.02	0.03	0.07	0.02	0.01	0.02	0.02	0.05	0.05	0.03	0.01	0.05	0.14
AP(4)	0.03	0.01	0.02	0.03	0.05	0.03	0.00	0.03	0.03	0.04	0.04	0.01	0.02	0.04	0.08	0.03	0.01	0.03	0.03	0.06	0.06	0.03	0.02	0.06	0.16
AP(5)	0.03	0.01	0.03	0.03	0.06	0.04	0.00	0.04	0.04	0.05	0.05	0.01	0.03	0.04	0.09	0.04	0.01	0.04	0.04	0.07	0.07	0.03	0.02	0.07	0.19
BP(1)	0.01	0.00	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.01	0.04	0.01	0.00	0.00	0.00	0.03	0.02	0.01	0.00	0.02	0.07
BP(2)	0.02	0.01	0.01	0.01	0.04	0.01	0.00	0.01	0.01	0.02	0.02	0.01	0.01	0.02	0.06	0.02	0.01	0.01	0.01	0.04	0.04	0.02	0.01	0.03	0.11
BP(3)	0.02	0.01	0.01	0.02	0.04	0.02	0.00	0.02	0.02	0.03	0.03	0.01	0.02	0.03	0.07	0.02	0.01	0.02	0.02	0.05	0.05	0.03	0.01	0.04	0.14
BP(4)	0.03	0.01	0.02	0.03	0.05	0.03	0.00	0.03	0.03	0.04	0.04	0.01	0.02	0.04	0.09	0.03	0.01	0.03	0.03	0.07	0.06	0.03	0.02	0.05	0.16
BP(5)	0.03	0.01	0.03	0.03	0.06	0.04	0.00	0.04	0.04	0.05	0.05	0.01	0.03	0.04	0.09	0.04	0.01	0.04	0.04	0.08	0.07	0.03	0.02	0.07	0.18
Execution	0.05	0.21	0.00	0.00	1.00	0.02	0.13	0.00	0.00	1.00	0.06	0.24	0.00	0.00	1.00	0.03	0.17	0.00	0.00	1.00	0.14	0.34	0.00	0.00	1.00
Deletion	0.47	0.50	0.00	0.00	1.00	0.50	0.50	0.00	1.00	1.00	0.46	0.50	0.00	0.00	1.00	0.49	0.50	0.00	0.00	1.00	0.39	0.49	0.00	0.00	1.00
Direction	0.01	1.00	-1.00	1.00	1.00	0.01	1.00	-1.00	1.00	1.00	0.02	1.00	-1.00	1.00	1.00	0.01	1.00	-1.00	1.00	1.00	0.06	1.00	-1.00	1.00	1.00
Elapsed Time	0.05	0.17	0.00	0.00	0.77	0.02	0.08	0.00	0.00	0.36	0.08	0.26	0.00	0.00	1.23	0.10	0.32	0.00	0.00	1.48	0.11	0.37	0.00	0.00	1.72
Price Change	0.00	0.17	-0.60	0.00	0.60	0.00	0.09	-0.45	0.00	0.45	-0.00	0.28	-0.96	0.00	0.96	-0.00	0.15	-0.43	0.00	0.43	-0.00	0.64	-2.17	0.00	2.19

Table 6: Features Summary Statistics. The table presents stock-level summary statistics for the feature variables used as input to the machine learning models. The 'Ask Price', 'Bid Price', 'Ask Size' and 'Bid Size' variables are generated from the raw dataset by following the standardization procedure outlined in Section 3.2. The 'Execution' and 'Deletion' features are dummy variables for LOB messages indicating executions and order cancellations, respectively (the complementary set of observations constitutes submission of new limit orders). 'Direction' is a binary variable taking values in $\{-1, 1\}$ and providing the directionally of the LOB message, i.e positive (negative) values indicate a change in the bid (ask) side of the order book. In particular, for executions, positive (negative) values of 'Direction' indicate seller-initiated (buyer-initiated) trades. 'Elapsed Time' measures the time from the previous message, in seconds. 'Price Change' denotes the percentage mid-price change from the previous message.