

# Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book

Petter N. Kolm<sup>1</sup> | Jeremy Turiel<sup>2</sup> | Nicholas Westray<sup>1</sup>

<sup>1</sup>Courant Institute of Mathematical Sciences, New York University, New York, New York, USA

<sup>2</sup>Department of Computer Science, University College London, London, UK

## Correspondence

Petter N. Kolm, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA.

Email: [petter.kolm@nyu.edu](mailto:petter.kolm@nyu.edu)

## Abstract

We employ deep learning in forecasting high-frequency returns at multiple horizons for 115 stocks traded on Nasdaq using order book information at the most granular level. While raw order book states can be used as input to the forecasting models, we achieve state-of-the-art predictive accuracy by training simpler “off-the-shelf” artificial neural networks on stationary inputs derived from the order book. Specifically, models trained on order flow significantly outperform most models trained directly on order books. Using cross-sectional regressions, we link the forecasting performance of a long short-term memory network to stock characteristics at the market microstructure level, suggesting that “information-rich” stocks can be predicted more accurately. Finally, we demonstrate that the effective horizon of stock specific forecasts is approximately two average price changes.

## KEYWORDS

artificial neural networks, deep learning, financial machine learning, high-frequency trading, limit order books, market microstructure, multiple horizons, order flow, return predictability

## 1 | INTRODUCTION

In this article, we employ deep learning (DL) in forecasting high-frequency returns at multiple horizons for 115 stocks traded on Nasdaq using order book information at the most granular level. In the last decade, DL has experienced enormous success, outperforming more traditional approaches in areas such as image classification, computer vision, and natural language

processing (Devlin et al., 2018; Goodfellow et al., 2016; Krizhevsky et al., 2012; LeCun et al., 2015; Schmidhuber, 2015). A key reason for this success is that DL learns suitable representations directly from the raw data, unlike conventional machine learning (ML) approaches where features are designed by hand and feature engineering frequently involves domain expertise. Artificial neural networks (ANNs) have proven to be particularly good at extracting intricate relationships in complex and high-dimensional settings *without* human input, especially when trained on large amounts of raw data. Although counterintuitive from the perspective of traditional statistical and ML techniques, where the researcher uses handcrafted features and progresses from simpler to more complex models, the essence of DL and related approaches is eloquently summarized in the following quote by Rich Sutton: “The biggest lesson that can be read from 70 years of AI research is that *general methods that leverage computation* are ultimately the most effective, and by a large margin.” (emphasis added) (Sutton, 2019). While results have been mixed because of the lack of enough data, ANNs have been applied to a number of problems in finance (see, for example, the surveys by Wong and Selvi (1998), Li and Ma (2010), Elmsili and Outtaj (2018), Ozbayoglu et al. (2020), Sezer et al. (2020)).

Recently, by leveraging large datasets extracted from *limit order books* (LOBs) (or *order books*, for short) in equity markets, ANNs have shown some success in forecasting high-frequency returns (see, for example, Tsantekidis et al. (2017), Tran et al. (2019), Zhang et al. (2018), Sirignano (2019), Sirignano and Cont (2019), Zhang et al. (2019a), Zhang et al. (2019b), Luo and Yu (2019), Tsantekidis et al. (2020), Briola et al. (2020), Zhang et al. (2021), Zhang and Zohren (2021)). To date, this literature has focused on two key empirical questions. First, to demonstrate that DL models outperform classical statistical and machine learning approaches, such as penalized linear models, decision trees, and kernel-based models. Second, to identify the optimal ANN architecture for forecasting future returns by classification.

While the answer to the first question has been established in favor of ANNs, the choice of the optimal ANN architecture is less clear. The most common ANNs in these studies are convolutional neural networks (CNNs), long short-term memory networks (LSTMs), multilayer perceptrons (MLPs), or a combination thereof.<sup>1</sup> For example, Tsantekidis et al. (2017), Tsantekidis et al. (2020) and Zhang et al. (2019a) advocate for CNNs, in the latter case with an additional *inception module* (Szegedy et al., 2015). In contrast, Sirignano and Cont (2019) provide evidence in favor of stacked LSTMs, and the horse race conducted by Briola et al. (2020) suggests that, as a universal function approximator, a large enough MLP performs as well as the CNN–LSTM of Zhang et al. (2018), an ANN where an LSTM layer is applied to the time series output of a CNN. Additionally, Tsantekidis et al. (2020) compare a CNN, LSTM, and CNN–LSTM. They find that their CNN–LSTM leads to more stable behavior and outperforms their other models. By combining bilinear projection and an attention mechanism (Bahdanau et al., 2014; Mnih et al., 2014), Tran et al. (2019) propose a temporal attention-augmented bilinear network architecture for mid-price return classification and show that it outperforms a CNN, LSTM, and MLP on their dataset. Passalis et al. (2020) develop a temporal logistic neural bag-of-features model that feeds into an MLP, demonstrating that it performs better than their benchmark models (including a CNN, GRU<sup>2</sup>, LSTM, and MLP) in classifying mid-price movement. In the majority of these studies, model inputs are represented as raw or transformed time series of order book states (one major exchange is chosen

<sup>1</sup> See the textbooks by Goodfellow et al. (2016), Friedman et al. (2017) for an introduction to DL and a discussion of many of the standard ANN architectures.

<sup>2</sup> The *gated recurrent unit* (GRU) was introduced by Cho et al. (2014). See also Jozefowicz et al. (2015) who compare the performance of the GRU and LSTM on a number of standard benchmark problems.

in the case of fragmented markets) and the return forecasting problem is formulated as a *classification task*, where a single forecasting horizon is chosen to be either a deterministic time interval such as two seconds, or a stochastic time interval such as until the next price change. Notable exceptions include the work of Mäkinen et al. (2019) and Rahimikia and Poon (2023). Mäkinen et al. (2019) forecast the arrival of jumps in 1-min ahead stock returns. Rahimikia and Poon (2023) examine the performance of forecasting realized volatility with LSTMs using order book data from LOBSTER (Huang & Polak, 2011) and news sentiment derived from Dow Jones Newswires.

Turning to the second empirical question, the objective of the classification frameworks in the existing literature is to predict the sign of the return over the specified horizon (two classes), or whether prices will go up, down, or not change (three classes). The most common approach to create class labels is by smoothing forward prices using moving averages over some prespecified forecast horizon, and then to assign labels based on thresholding. With the exception of special cases, there is no canonical way to perform such labeling. Thus, while these labeling methods provide a form of regularization by removing noise, they introduce ad hoc modeling parameters that are undesirable.

As most studies deploy different datasets, direct comparison of model architectures is challenging. Indeed, it may well be the case that there is no one single “optimal architecture” and different datasets warrant distinct ANNs. Some articles use the FI-2010 benchmark dataset of Ntakaris et al. (2018) that consists of 10 days of LOB events for five companies traded on the Nasdaq Nordic exchange. A caveat with the FI-2010 dataset is that it is preprocessed and downsampled (Kercheval & Zhang, 2015), and, therefore, does not represent the LOBs at the most granular level. In addition, as this dataset is rather small, it is not obvious that results will generalize to other stocks, exchanges, and/or time periods. Sirignano and Cont (2019) employ the most comprehensive dataset to date, assembled from LOBSTER, consisting of order book events of about 1000 stocks traded on the Nasdaq from January 1, 2014 through March 31, 2017. In this study, using a stacked LSTM with three layers, the authors provide compelling evidence for a universal relationship between order book states and price changes. In a related article, with a similar dataset consisting of 489 symbols from the S&P 500 and Nasdaq 100 over the period January 1, 2014 through August 31, 2015, Sirignano (2019) proposes a new spatial neural network architecture and demonstrates that this outperforms logistic regression and an MLP in the return classification task.

The present article makes four main contributions to the literature on high-frequency return forecasting in LOBs. First, in contrast to the previous literature, we formulate the forecasting problem as one of regression to avoid the issues with training classifiers discussed above. Unlike the standard regression setup where the forecast is a scalar, we deploy a *multi-output regression framework* where the model for each stock outputs a vector we refer to as an *alpha term structure*. Each element of the alpha term structure represents a mid-price return forecast at a specified horizon. Alpha term structures render representations of the timescales over which alpha may rise and/or decay that are useful in real-world trading. For instance, they can be used in designing order placement strategies in optimal execution and market making algorithms, and other high-frequency trading strategies. Notably, in their recent work, Zhang and Zohren (2021) deploy a multi-horizon design. However, they use a classification setup and their Seq2Seq architecture (Cho et al., 2014) is different from the ANNs we use.

Second, we perform a large-scale forecasting horse race with common ANNs, including an MLP, LSTM, LSTM-MLP, stacked LSTM, and CNN-LSTM, trained on different inputs including LOB states for the first 10 (nonzero) levels and data derived from the order book. Besides our CNN-LSTM, that is similar to the architecture used by Zhang et al. (2018), all other models

are standard “off-the-shelf” ANNs. Sourced from LOBSTER, our dataset consists of order book events timestamped to nanosecond precision for 115 Nasdaq stocks for the period January 1, 2019 through January 31, 2020. In contrast to many previous studies, we do not downsample our data but work directly with the raw LOB states. We demonstrate that while LOB states, which are a complex *nonstationary* multivariate process, can be used directly as inputs for the ANN models, forecasting performance can be improved significantly by training the models on *stationary* inputs. In particular, we show that ANNs trained on *order flow* (OF), stationary quantities derived from the LOB (Cont et al., 2014), significantly outperform most models that are trained directly on order book states. Perhaps somewhat surprisingly, we find that the exact ANN architecture is of secondary importance for short-term return predictability and models that include an LSTM module exhibit similar performance. In particular, we demonstrate that with OF inputs our simple LSTM model performs as well as the deeper and more complex models. Even the CNN–LSTM, the most complex model in our study, provides no significant improvement over the simple LSTM.

Third, using cross-sectional regressions, we link the forecasting performance of the LSTM model to stock characteristics at the market microstructure level. We show that “information-rich” stocks, defined as stocks that have a higher number of order book updates relative to the number of price changes, can be forecasted more accurately by DL models.

Fourth, by leveraging the multi-output regression setup, we analyze the shape of the alpha term structure from the models and show that stock-level alphas peak at a time scale of about two price changes and decline thereafter.

The outline of the article is as follows. In Section 2, we review the mechanics of LOB markets and specify the forecasting models used in this study. We describe our data, its preprocessing, and our forecasting and model evaluation methodology in Section 3. We present our empirical results and findings in Section 4. Section 5 concludes.

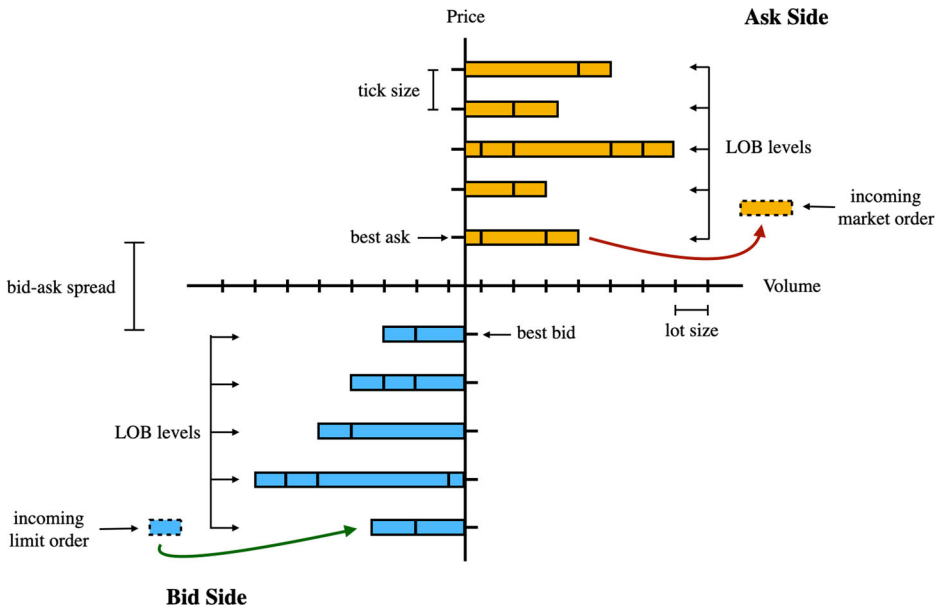
## 2 | PRELIMINARIES

In Section 2.1, we provide an introduction to LOBs, their representation in full and reduced-form, including order book states (LOB), OF, and *order flow imbalance* (OFI). Then in Section 2.2, we describe the forecasting models used in this study, including an autoregression with exogenous inputs (ARX), MLP, LSTM, LSTM–MLP, stacked LSTM, and CNN–LSTM. The first model is a classical linear model whereas the last five are ANN architectures.

### 2.1 | Limit order books and order flow

Modern equity trading is conducted electronically. Major exchanges in the United States and the rest of the world facilitate this via an LOB, one per stock. The order book represents a collection of buyers and sellers, ordered by price and time, bidding and offering stock for purchase or sale. Figure 1 depicts an order book where buyers and sellers are shaded in blue and yellow, respectively.

At a given time  $t$ , the highest price buyers are prepared to buy the stock for is called the *bid price* and is denoted by  $b_t$ . Similarly, the lowest price sellers are prepared to sell the stock for is referred to as the *ask price* and is denoted by  $a_t$ . From the LOB, we can derive the *mid-price*,  $p_t := (b_t + a_t)/2$ , the *bid–ask spread*,  $a_t - b_t$ , and the *tick size*, the smallest price increment between different price levels in the order book, as illustrated in Figure 1. For the Nasdaq exchange, which



**FIGURE 1** An illustration of a limit order book. We observe a market order being filled at the ask side and an incoming limit order being added to the bid side. [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

is the focus of this article, the tick size is \$0.01. Beyond the bid and ask, there are additional *levels* in the order book.

An *order* is defined as the four-tuple (side, price, quantity, time), representing the side of the order book the order is posted to, the price at which the order is submitted, the desired amount to be traded and the time of submission. Orders may be entered, and if active, cancelled at any time. When an order is submitted the matching engine of the exchange attempts to match it with existing orders in the book. Matched orders are called *market orders*. Orders which do not match, or only match partially, are referred to as a *limit orders*. When there are multiple limit orders with the same side and price present, they are queued chronologically at that price level according to the first-in-first-out (FIFO) principle. For an in-depth description of modern equity trading and the order book, we refer to Gould et al. (2013), Abergel et al. (2016), Bouchaud et al. (2018).

### 2.1.1 | Order book states

Throughout the rest of the article, we consider the first 10 levels of the order book for each stock. We define the *state of the order book at time  $t$*  as the vector of price and volume information for the top 10 (nonempty) bid and ask levels

$$\mathbf{s}_t^{\text{LOB}} := \left( a_t^1, v_t^{1,a}, b_t^1, v_t^{1,b}, \dots, a_t^{10}, v_t^{10,a}, b_t^{10}, v_t^{10,b} \right)^T \in \mathbb{R}^{40}, \quad (1)$$

where  $b_t^i, a_t^i$  are the bid and ask prices at the  $i$ th level at time  $t$  and  $v_t^{i,b}, v_t^{i,a}$  are the corresponding share volumes. With a slight abuse of terminology, in the following, we refer to time series of order book states as the LOB. Note that the LOB for each stock is an irregularly spaced time series.

### 2.1.2 | Bid–ask order flow and order flow imbalance

For each stock and date, we let  $t \in \{1, \dots, T\}$  denote an enumeration of all order book updates on that day. Given two consecutive order book states for the stock at  $t - 1$  and  $t$  of the form (1), we define the *bid order flows* (bOF) and *ask order flows* (aOF) at time  $t$  as the vectors  $\mathbf{bOF}_t, \mathbf{aOF}_t \in \mathbb{R}^{10}$ , where each component is given by

$$\mathbf{bOF}_{t,i} := \begin{cases} v_t^{i,b}, & \text{if } b_t^i > b_{t-1}^i, \\ v_t^{i,b} - v_{t-1}^{i,b}, & \text{if } b_t^i = b_{t-1}^i, \\ -v_{t-1}^{i,b}, & \text{if } b_t^i < b_{t-1}^i, \end{cases} \quad (2)$$

$$\mathbf{aOF}_{t,i} := \begin{cases} -v_{t-1}^{i,a}, & \text{if } a_t^i > a_{t-1}^i, \\ v_t^{i,a} - v_{t-1}^{i,a}, & \text{if } a_t^i = a_{t-1}^i, \\ v_t^{i,a}, & \text{if } a_t^i < a_{t-1}^i, \end{cases} \quad (3)$$

for  $i = 1, \dots, 10$ . By concatenating and subtracting the bOF and aOF at time  $t$ , we obtain the OF

$$\mathbf{OF}_t := \begin{pmatrix} \mathbf{bOF}_t \\ \mathbf{aOF}_t \end{pmatrix} \in \mathbb{R}^{20}, \quad (4)$$

and the OFI

$$\mathbf{OFI}_t := \mathbf{bOF}_t - \mathbf{aOF}_t \in \mathbb{R}^{10}, \quad (5)$$

respectively. The nonlinear transformations represented by formulas (2), (3), and (5) have become a common approach to map the nonstationary time series of order book states to a stationary time series (Cont et al., 2014). Cont et al. (2014) establish a linear relationship between price changes and OFI at the first level of the order book, and Xu et al. (2018), Cont et al. (2021), Kolm and Westray (2023) extend this result to multilevel and cross-sectional OFIs. OF as defined in Equation (4), which keeps the bid and ask sides separate, represents a modest generalization of OFI and offers any forecasting exercise greater flexibility. In particular, unlike OFI that weights bid and ask OFs equally, by using the nondifferenced OFs directly, forecasting models have the possibility of combining OFs from bid and ask sides asymmetrically.

## 2.2 | Forecasting models

In this article, we consider an MLP, single and stacked LSTM, LSTM–MLP, and CNN–LSTM. Besides the CNN–LSTM that is similar to the architecture used by Zhang et al. (2018), all other models are standard “off-the-shelf” ANNs and we keep their descriptions brief. For an introduction to DL and a discussion of many of the standard ANN architectures, we refer to Goodfellow et al. (2016), Friedman et al. (2017). For comparative purposes, we also include a classical linear ARX model in our empirical work.

### 2.2.1 | ARX

An *autoregressive with exogenous inputs* (ARX) model, based on a traditional autoregressive (AR) model (Hamilton, 1994), is a time series model where the dependent variable,  $y_t \in \mathbb{R}$ , is regressed onto lagged versions of itself and vectors of exogenous inputs,  $\mathbf{x}_t \in \mathbb{R}^m$ , that is

$$y_t = w_0 + \sum_{i=1}^{n_y} w_i y_{t-i} + \sum_{i=1}^{n_x} \mathbf{v}_i^\top \mathbf{x}_{t-i} + \varepsilon_t, \quad (6)$$

where  $w_0, w_i \in \mathbb{R}$ , and  $\mathbf{v}_i \in \mathbb{R}^m$  are parameters,  $\varepsilon_t$  is a random white noise term with zero mean, and  $n_y, n_x$  denote the orders of the autoregressive and exogenous inputs, respectively. We refer to Equation (6) as an ARX( $n_y, n_x$ ) model.

In our empirical work, we explore two formulations using 100 lags. In the first, we regress mid-price returns on lagged versions of itself and OFs for the top 10 levels of the LOB. In the second, we regress mid-price returns only on lagged OFs up to ten levels. Effectively, the second formulation is a standard linear regression of the forward returns on (lagged) OFs. We find that the results from these two formulations are similar. Therefore, in the empirical section of this article, we only report the results of the latter.

### 2.2.2 | Multilayer perceptron

An MLP is a *feedforward artificial neural network* (FFNN) that consists of  $L \geq 1$  hidden layers, each with  $N_l \in \mathbb{N}$  neurons. Given an input vector  $\mathbf{x} \equiv \mathbf{h}_0 \in \mathbb{R}^{N_0}$ , an MLP with  $L$  hidden layers is a mapping of the form

$$\mathbf{h}^{(l)} = g^{(l)}(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, 2, \dots, L, \quad (7)$$

$$\mathbf{y} = \mathbf{W}^{(L+1)} \mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}, \quad (8)$$

where  $\mathbf{h}^{(l)} \in \mathbb{R}^{N_l}$  is the hidden state of the  $l$ th layer,  $\mathbf{W}^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$  is a weight matrix,  $\mathbf{b}^{(l)} \in \mathbb{R}^{N_l}$  is a bias vector, and  $\mathbf{y} \in \mathbb{R}^{N_{L+1}}$  is the output. Here,  $g^{(l)} : \mathbb{R} \rightarrow \mathbb{R}$  is an activation function that acts component-wise on its argument. The MLP we employ in our empirical work is summarized in the second column of Table 1. We use the *exponential linear unit* (ELU) activation function of Clevert et al. (2015), that is,  $g^{(l)}(x) := \max(0, x) + \alpha \min(0, \exp(x) - 1)$  for all  $l$ , where we set  $\alpha = 1$ .

We emphasize that while an MLP is a universal approximator (Hornik, 1991), it is not inherently designed to model time dependencies as it lacks spatiotemporal invariance. Specifically, as each timestamp is treated independently by an MLP, the temporal information is lost. In addition, the number of parameters in each layer of an MLP is not invariant across time series of different lengths, making transferability of the network challenging (Fawaz et al., 2019).<sup>3</sup> Briola et al. (2020) conduct a horse race comparing classical ML approaches and DL in forecasting high-frequency returns from LOB data using classification. Their results suggest, as a universal function approximator, a large enough MLP performs as well as the CNN-LSTM of Zhang et al. (2018).

<sup>3</sup> Transfer learning is the ability to reuse a pretrained neural network on a new problem. Besides the compute and time savings transfer learning can provide, it is particularly useful in many financial application as many real-world problems do not have large enough training datasets to train models from scratch (Bengio, 2012; Pan & Yang, 2009).



TABLE 1 The MLP- and LSTM-based architectures for LOB inputs. The input sizes for OFI and OF inputs are listed in the footnote.

Model	MLP	LSTM	LSTM-MLP	LSTM (3)
Layer input <sup>a</sup>	100 × 40	100 × 40	100 × 40	100 × 40
Hidden layers	Dense @ 512 units (ELU)	LSTM @ 128 units (Tanh)	LSTM @ 128 units (Tanh)	LSTM @ 150 units (Tanh)
	Dense @ 512 units (ELU)		Dense @ 64 units (ELU)	LSTM @ 150 units (Tanh)
	Dense @ 32 units (ELU)			LSTM @ 150 units (Tanh)
Layer output	Dense @ 10 units	Dense @ 10 units	Dense @ 10 units	Dense @ 10 units

<sup>a</sup>The input sizes shown in the table are for LOB inputs. Input sizes are 100 × 10 and 100 × 20 for OFI and OF inputs, respectively.



### 2.2.3 | Long short-term memory and extensions

First introduced by Hochreiter and Schmidhuber (1997), LSTM is an ANN explicitly designed to handle temporal dependencies in sequential data. Belonging to the family of *recurrent neural networks* (RNNs), LSTMs address the vanishing gradient problem that is common in training traditional RNNs (Rumelhart et al., 1986). The key elements of the LSTM are (i) a memory cell, representing its state over time, and (ii) nonlinear gates that determine the information flow in and out of the memory cell. Commonly, three gates are used, referred to as *input*, *output*, and *forget gates* (Gers et al., 2000). Letting  $\mathbf{x}_t \in \mathbb{R}^N$  denote an input vector, the LSTM unit takes the form<sup>4</sup>

$$\mathbf{f}_t = \sigma(\mathbf{U}^f \mathbf{x}_t + \mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{b}^f) \quad (9)$$

$$\mathbf{i}_t = \sigma(\mathbf{U}^i \mathbf{x}_t + \mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{b}^i) \quad (10)$$

$$\mathbf{o}_t = \sigma(\mathbf{U}^o \mathbf{x}_t + \mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{b}^o) \quad (11)$$

$$\mathbf{s}_t = \mathbf{f}_t \circ \mathbf{s}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{U}^s \mathbf{x}_t + \mathbf{W}^s \mathbf{h}_{t-1} + \mathbf{b}^s) \quad (12)$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{s}_t) \quad (13)$$

where  $\sigma := (1 + e^{-x})^{-1}$  is the sigmoid activation function,  $\mathbf{f}_t \in \mathbb{R}^M$  is the forget gate's activation vector,  $\mathbf{i}_t \in \mathbb{R}^M$  is the input gate's activation vector,  $\mathbf{o}_t \in \mathbb{R}^M$  is the output gate's activation vector,  $\mathbf{h}_t \in \mathbb{R}^M$  is the output vector of the LSTM unit,  $\mathbf{s}_t \in \mathbb{R}^M$  is the unit's hidden state vector, and  $\mathbf{W} \in \mathbb{R}^{M \times N}$ ,  $\mathbf{U} \in \mathbb{R}^{M \times M}$ , and  $\mathbf{b} \in \mathbb{R}^M$  (superscripts dropped) are weight matrices and bias vector parameters that are learned during training. Here,  $\circ$  and  $\tanh$  denote the element-wise product and hyperbolic tangent, respectively. To avoid the vanishing gradient problem, we initialize  $\mathbf{b}^f$  to one when training our networks (Gers et al., 2000). The simple LSTM we employ in our empirical work is summarized in the third column of Table 1.

In applications, the output from an LSTM frequently feeds into an MLP, referred to as an *LSTM-MLP* architecture. In this model setup, temporal dependencies of the input are preserved by an LSTM, and the extracted embedding (i.e., the transformed input) is then passed to an MLP. The LSTM-MLP we use in our empirical work is summarized in the fourth column of Table 1.

By combining multiple LSTMs, in such a way that the output of one LSTM becomes the input to another LSTM, we obtain a *stacked LSTM* architecture, a deeper version of the LSTM (Graves, 2013; Graves et al., 2013). A generic stacked LSTM consists of several hidden LSTM layers, each containing multiple memory cells. Each layer provides a sequential output rather than a single value to the next layer, allowing the hidden state at each level to operate at different timescales. In our empirical work, we use a deep LSTM that consists of three stacked LSTMs (see, the fifth column of Table 1). We refer to this architecture as an *LSTM* (3).

In a large-scale comparison of eight shallow LSTM variants, Greff et al. (2016) demonstrate that they perform well on a range of tasks, including speech and handwriting recognition, and polyphonic music modeling. Pascanu et al. (2013) confirm that deeper architectures outperform

<sup>4</sup> This formulation follows Keras' default implementation of the LSTM, available at [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/LSTM](https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM). We use this implementation in our empirical work.

conventional shallow networks in polyphonic music prediction and language modeling tasks. Recent work by Sirignano (2019) and Tsantekidis et al. (2020) provide support for LSTMs being able to forecast high-frequency returns using order book data.

## 2.2.4 | CNN–LSTM

While standard ANNs such as CNNs<sup>5</sup>, LSTMs, and MLPs can be used individually, it is well-known that they are complementary in their modeling capabilities and therefore each can be found as building blocks in combined network architectures. For instance, CNNs are effective in reducing frequency variations, LSTMs excel in learning temporal structures, and MLPs are universal approximators. Therefore, used together in combined networks, each extracts information from the input at different spatial and temporal scales (Sermanet & LeCun, 2011; Sainath et al., 2015). Notably, the work of Sainath et al. (2015) is one of the first to combine CNNs, LSTMs, and MLPs into a unified architecture that can be trained jointly. CNN–LSTMs have been used successfully in a number of applications, including image and video recognition (Donahue et al., 2015; Vinyals et al., 2015), language modeling (Kim et al., 2016), and nowcasting (Shi et al., 2015).

The CNN–LSTM we employ in this article is similar to the architecture used by Zhang et al. (2018). While they forecast high-frequency returns as a classification task, we use the multihorizon regression framework described in Section 3.2.<sup>6</sup> Because the CNN–LSTM for LOB inputs is the most general, we describe it first and later comment on how we modify it for OF inputs. The specific details of the architecture, including descriptions of each layer, is elucidated in Table 2. To keep this table and following description concise, we assume that the reader is familiar with convolutions and CNNs as described in the textbook by Goodfellow et al. (2016).

From a high-level perspective, the architecture consists of (i) a CNN, (ii) an inception module, and (iii) an LSTM, each consisting of one or more blocks and subblocks (see Table 2). The CNN is made up of three blocks. Block 1 reduces the spatial dimension of the input from 40 to 20 by combining prices and volumes. Block 2 further reduces the spatial dimension from 20 to 10 by combining information across bid and ask sides at the same level of the order book. Then, by aggregating information across all levels in the order book, block 3 reduces the spatial dimension down to one. All activation functions in the CNN blocks are *leaky ReLUs*, that is,  $g(x) := \max(0, x) + 0.01 \min(0, x)$ . The inception module is a standard architectural element in CNNs used for extracting features at different temporal frequencies (Szegedy et al., 2015). Finally, after concatenating and reshaping the outputs of the inception module, an LSTM with 64 hidden units is used to extract temporal features, returning an alpha term structure.

The different sizes of the OF inputs necessitate small adjustments of the CNN–LSTM architecture. In particular, for OF inputs, it consists of block 2 through block 5 in Table 2, where the input to the first layer is of size  $100 \times 20 \times 1$ . Otherwise, the CNN–LSTM architectures for OF inputs are the same as the one with LOB inputs.

<sup>5</sup> *Convolutional neural networks* (CNNs) are multilayer FFNNs first popularized in image processing, obtaining their name from having layers consisting of convolutional filters. CNNs are translation equivariant and can be viewed as regularized versions of MLPs. See, Goodfellow et al. (2016) for an introduction to CNNs.

<sup>6</sup> The implementation of Zhang et al. (2018) is available at <https://github.com/zcakhha/DeepLOB-Deep-Convolutional-Neural-Networks-for-Limit-Order-Books>.

**TABLE 2** The CNN–LSTM architecture for LOB inputs. The architectures for OF inputs involve just block 2 through block 5.

Block	Description	Details	Layer input	Layer output
1	<b>Convolutions of prices and volumes</b>			
	(i) Combine price and volume information for each side and level of the order book	One spatial convolution ( $1 \times 2$ ) (stride = $1 \times 2$ ) with 32 filters	$100 \times 40 \times 1$	$100 \times 20 \times 32$
	(ii) Combine price–volume information across time for each side and level of the order book	Two temporal convolutions ( $4 \times 1$ ) (stride = $1 \times 1$ ) with 32 filters and padding	$100 \times 20 \times 32$	$100 \times 20 \times 32$
2	<b>Convolutions across LOB levels<sup>a</sup></b>			
	(i) Combine imbalance information across sides for each level of the order book	One spatial convolution ( $1 \times 2$ ) (stride = $1 \times 2$ ) with 32 filters	$100 \times 20 \times 32^b$	$100 \times 10 \times 32$
	(ii) Combine imbalance information across time for each side and level of the order book	Two temporal convolutions ( $4 \times 1$ ) (stride = $1 \times 1$ ) with 32 filters and padding	$100 \times 10 \times 32$	$100 \times 10 \times 32$
3	<b>Convolutions of imbalance information</b>			
	Combine imbalance information across all levels of the book	One spatial convolution ( $10 \times 1$ ) (stride = $1 \times 10$ ) with 32 filters	$100 \times 10 \times 32^c$	$100 \times 1 \times 32$
4	<b>Inception module</b>	The inception module consists of three subblocks, each acting on the output from block 3 above. Each subblock uses 64 filters. Outputs are then concatenated and reshaped before being passed to the LSTM		
	Subblock 1	Spatial convolution ( $1 \times 1$ ) and temporal convolution ( $3 \times 1$ ) (stride = $1 \times 1$ ) with 64 filters and padding	$100 \times 1 \times 32$	$100 \times 1 \times 64$
	Subblock 2	Spatial convolution ( $1 \times 1$ ) and temporal convolution ( $5 \times 1$ ) (stride = $1 \times 1$ ) with 64 filters and padding	$100 \times 1 \times 32$	$100 \times 1 \times 64$
	Subblock 3	Spatial max pooling and temporal convolution ( $1 \times 1$ ) (stride = $1 \times 1$ ) with 64 filters and padding	$100 \times 1 \times 32$	$100 \times 1 \times 64$
	Concatenation and reshaping		$100 \times 1 \times 64 \times 3$	$100 \times 192$

(Continues)

TABLE 2 (Continued)

Block	Description	Details	Layer input	Layer output
5	<b>LSTM and dense layer</b>	LSTM with 64 hidden units	$100 \times 192$	$64 \times 1$
	The output of this block is an alpha term structure	Dense layer	$64 \times 1$	$10 \times 1$

<sup>a</sup>For LOB inputs these are convolutions of notional features (in dollars), while for OF inputs these are convolutions of volumes (number of shares).

<sup>b</sup>This is the size for LOB inputs. For OF inputs the size is  $100 \times 20 \times 1$ .

<sup>c</sup>This is the size for LOB and OF inputs. For OFI inputs the size is  $100 \times 10 \times 1$ .

### 3 | DATA AND METHODOLOGY

#### 3.1 | Data

We source our data from LOBSTER (Huang & Polak, 2011). This dataset provides complete order book state information from Nasdaq, except for submissions and cancellations of hidden orders. We include all stocks from January 1, 2019 through January 31, 2020 that meet the following three criteria: (i) their being a member of the Nasdaq 100 index at some point during the time period, (ii) their having data for at least 260 trading days, and (iii) their having no corporate actions during the period of study. Due to issues with the underlying data, January 9, 2019 is excluded. This results in a universe of 115 stocks, containing many household names across a variety of industries, including Amazon (AMZN), American Airlines (AAL), Facebook (FB), Google (GOOGL), Microsoft (MSFT), and Netflix (NFLX). The larger number of stocks and longer time period than most previous studies provide greater reliability in results from our out-of-sample testing and allow us to explore stock specific properties.

LOBSTER data must be preprocessed before use (Bouchaud et al., 2018; Bugaenko, 2020). The preprocessing addresses situations such as that of a single market order executing against multiple limit orders, or a modification to a limit order being implemented as a cancellation followed by a new submission. We describe our preprocessing steps in Appendix A.

Our preprocessed dataset, summarized in Table A.1 in Appendix A, is identical to that used in Kolm and Westray (2023) with a size of about 10TB uncompressed. As a result, storage and efficient processing become challenging tasks. Our dataset is one to two orders of magnitude larger than similar studies that apply DL to LOBs (Briola et al., 2020; Tsantekidis et al., 2020, 2017; Tran et al., 2019; Zhang et al., 2018, 2019a, 2019b, 2021; Zhang & Zohren, 2021). The datasets used by Sirignano (2019), Sirignano and Cont (2019) are exceptions. Their universe is drawn from the S&P 500 and is larger than ours; however, their focus is on a single architecture. Sirignano (2019) uses a spatial neural network architecture and Sirignano and Cont (2019) focuses on a stacked LSTM, both in a classification setting. In contrast, we consider MLP, LSTM, LSTM-MLP, stacked LSTM, and CNN-LSTM architectures in a multihorizon regression setting, allowing us to compare and shed light on which aspects of the architectures are most useful for predicting high-frequency returns while ensuring the results apply to a broad universe of stocks.

#### 3.2 | Methodology

We focus on simultaneously forecasting future mid-price returns across multiple horizons. For each stock and time  $t$ , our multihorizon forecasts are represented as the vector

$$\mathbf{r}_t := (r_{t,1}, \dots, r_{t,H})^\top \in \mathbb{R}^H, \text{ where } H \geq 1, \tag{14}$$

where  $H$  denotes the number of horizons. We refer to Equation (14) as an *alpha term structure* at time  $t$ . Our forecasting models are trained at the single stock level and take the form

$$\mathbf{r}_t = \mathbf{g}(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-W}) + \varepsilon_t, \quad (15)$$

where the vector-valued function  $\mathbf{g}$  is approximated by an ANN,  $\varepsilon_t \in \mathbb{R}^H$  is the residual,  $W$  denotes the length of the lookback window, and the input,  $\{\mathbf{x}_\tau\}_{\tau=t-W}^{t-1}$ , is either LOBs or OFs at the most granular level. As is common in the literature, in our empirical work, we set  $W = 100$  for all symbols, (see, for example, Zhang et al. (2019a)). The robustness check we perform in Appendix B.2, using stock specific lookback windows, suggests that the length of the lookback window is a less important driver of predictive performance. To avoid issues around market open and close, we drop the first and last 10 min of each day in our empirical work.

In contrast to our work, Sirignano and Cont (2019) investigate whether the function  $\mathbf{g}$  in Equation (15) is *universal*, that is, if the same function is *shared across stocks*. To estimate their model, the authors downsample the time series of updates to the order book to only those times where price changes occur. It is well-known in the market microstructure literature that not just trades but all changes to the order book are important in understanding price formation (see, for example, Cont et al. (2014), Bouchaud et al. (2018)). Therefore, as our focus in this article is return forecasting rather than questions of universality, and downsampling risks removing potentially important information, we use all order book updates and train an ANN for each symbol in the dataset.

### 3.2.1 | Dependent variables

Our stock universe contains a diverse set of symbols with significantly different order book activity. For instance, EBAY and MSFT differ by three orders of magnitude in terms of number of order book updates per day (see, Table A.1 in Appendix A). The wide-ranging update frequencies are related to the notion of what the practitioner community refers to as “stock characteristic time” and can be formalized by intermittent processes such as Hawkes processes (see, for example, Jaisson & Rosenbaum, 2016; Bouchaud et al., 2018). Using single stock models in our work, we make the horizons of the alpha term structure in Equation (14) stock specific. Specifically, we define a stock specific time increment

$$\Delta t := \frac{2.34 \cdot 10^7}{N}, \quad (16)$$

where the numerator is the number of milliseconds in a trading day, and the denominator,  $N$ , denotes the average number of nonzero tick-by-tick mid-price returns. This ratio captures how long on average we have to wait for a change in the mid-price of a given stock.

We choose an alpha term structure (14) of dimension  $H = 10$  and define our forecast horizons,  $\{h_k\}_{k=1}^{10}$ , as the following multiples of  $\Delta t$

$$h_k := \frac{1}{5} k \Delta t, \quad k = 1, \dots, 10. \quad (17)$$

As dependent variables in our single stock forecasting models (15), we use the  $h_k$ -horizon mid-price returns  $r_{t,k} := p_{t+h_k} - p_{t+\delta\tau}$  (in dollars), where  $p_t$  denotes the mid-price at time  $t$  and  $\delta\tau$  represents a 10 ms latency buffer. Analogously, we evaluate model performance by computing

out-of-sample returns in the same way. It is common practice to use latency buffers in developing and testing alphas to account for delays associated with the time needed to compute model forecasts and the round-trip communication time between the exchange and the trading algorithm. Using Nasdaq-sourced data for 1 month in 2007 and 2008, respectively, Hasbrouck and Saar (2013) estimate the fastest traders in their sample had an effective latency of 2–3 ms. Zhang et al. (2019a) report that the average time needed to produce a return forecast from a number of state-of-the-art ANNs, including that of a CNN and LSTM, ranges from 0.03 to 0.97 ms, depending on the model used. Hence, the 10 ms latency buffer we use in this study is conservative.

Using stock-specific forecast horizons ensures that the time scale adjusts appropriately for each stock. Naturally, due to nonconstant volume and volatility profiles throughout a typical trading day, the average time for a price change is not uniform. Therefore, the specification of the stock-specific forecast horizons can certainly be improved to take various intraday and trading-related effects into account. Needless to say, the ones we use here are parsimonious and allow us to sensibly compare results across stocks in our empirical analysis. While generalizations that incorporate intraday profiles are possible within our framework (see, for example, Mertens et al. (2022)), we leave this for future work.

### 3.2.2 | Independent variables

We consider two choices of independent variables as inputs to our models, LOBs and OFs, defined in Equations (1) and (4), respectively. LOBs provide a complete representation of the order book. In general, while return series are stationary, price series tend to be nonstationary (Cont, 2001). Therefore, we expect order book states to be nonstationary, and the resulting regression problem in Equation (15) will have stationary-dependent variables and nonstationary independent variables. In contrast, as emphasized in Section 2.1.2, OFs are reduced stationary representations of order book states. Unlike OFIs (defined in Equation 5), OFs keep bid and ask sides separate, thereby enabling models to incorporate them asymmetrically.

We winsorize all dependent and independent variables in our training sets at the 0.5% and 99.5% levels and then perform Z-score normalization of the resulting data. The parameters from the in-sample normalization are stored to scale test and validation sets that are used for out-of-sample evaluation.

### 3.2.3 | Training and evaluation

We design the training and evaluation procedure to be similar to a real-world production setting. In particular, we train our models in a rolling-window backtesting fashion across 48 weeks using a (1 week, 4 weeks, 1 week) structure, where the first week is used for validation, the following 4 weeks are used for training, and the last week is reserved for out-of-sample testing. To keep the training data for consecutive windows nonoverlapping, we move our rolling window forward by 3 weeks each time. With 115 symbols, 12 model and input combinations, and 18 rolling windows, we train and evaluate a total of about 25K individual model fits. To the best of our knowledge, previous studies in this area do not perform out-of-sample testing on a rolling basis. While rolling comes at a significantly higher computational cost, as many models need to be trained and retrained, empirical findings will be more reliable as out-of-sample forecasting performance is evaluated over longer time periods.



We train our models by minimizing mean-squared error (MSE) with *stochastic gradient descent* (SGD) using the Adam optimizer (Kingma & Ba, 2014). Table 3 summarizes all input and hyperparameter settings. The batch size is kept small as there is evidence that smaller batch sizes lead to local minima, which generalize better (Keskar et al., 2017). If the time series of order book updates and returns are naively concatenated, elements in a batch can contain data from two separate trading days. To avoid this, we implement a custom Keras generator. We apply early stopping for all models to avoid overfitting, terminating training when validation loss has not decreased for five consecutive epochs. Our code is written in Python and ANNs are implemented with Tensorflow 2.3.1 (Abadi et al., 2015) and Keras (Chollet et al., 2015).

Simple models like MLP/ARX can be trained directly using CPUs. However, training a large number of DL models that incorporate CNNs and LSTMs would most likely be impossible without the use of GPUs. When fitting our LSTM and CNN-LSTM-based models, we use single GPUs, which are either an NVIDIA Quadro RTX8000 (48GB), an NVIDIA V100 (32GB), or an AMD MI-50 (32GB). All of our computations are performed on NYU's Greene<sup>7</sup> and Hudson<sup>8</sup> high performance computing environments. This infrastructure allows us to massively parallelize the data processing and training of the models. The time to train a single model varies in the range of 10–60 min, depending on model and stock.

We evaluate the performance of each forecasting model at the individual stock level using *out-of-sample*  $R^2$  at horizon  $h$  ( $R_{OS,h}^2$ ) for each test period, defined as

$$R_{OS,h}^2 := 1 - \text{MSE}_{m,h} / \text{MSE}_{\text{bmk},h}, \quad (18)$$

where  $\text{MSE}_{m,h}$  and  $\text{MSE}_{\text{bmk},h}$  are the mean-squared errors of the model forecasts and benchmark at the  $h$ th horizon, respectively. To be conservative, we use the average out-of-sample return as our benchmark. If  $R_{OS,h}^2 > 0$ , then the forecasting model outperforms the benchmark represented by the average return on the test set. Importantly, a forecast relies only on data available up to the time at which it is computed. When it is clear what the horizon is, in the remainder of the article, we sometimes drop the subscript  $h$  and just write  $R_{OS}^2$ .

To statistically examine questions such as the relative forecast accuracy of model and input combinations, what kind of input works better for particular models, and whether particular models exhibit superior forecasting ability, we use the standard methodology of Patton (2011), Liu et al. (2015).

## 4 | EMPIRICAL RESULTS

In this section, we present our empirical divided into three parts. First, in Section 4.1, we investigate short-term return predictability (up to two average price changes) at the individual stock level. We evaluate which model and input combination deliver the best performance, and examine statistically the question of whether there is superior forecasting ability at short horizons. Using cross-sectional regressions, in Section 4.2, we link the forecasting performance of the LSTM trained on OF inputs to stock characteristics at the market microstructure level. Finally,

<sup>7</sup> Greene has 32K CPU cores, 332 NVIDIA GPUs, and 145TB of RAM distributed over 568 nodes. See <https://www.nyu.edu/about/news-publications/news/2020/november/GreeneSupercomputer.html>.

<sup>8</sup> Hudson has 960 CPU cores, 160 AMD GPUs, and 10TB of RAM distributed across 20 nodes. See <https://wp.nyu.edu/connect/2020/06/08/nyu-expands-supercomputing-amd>.

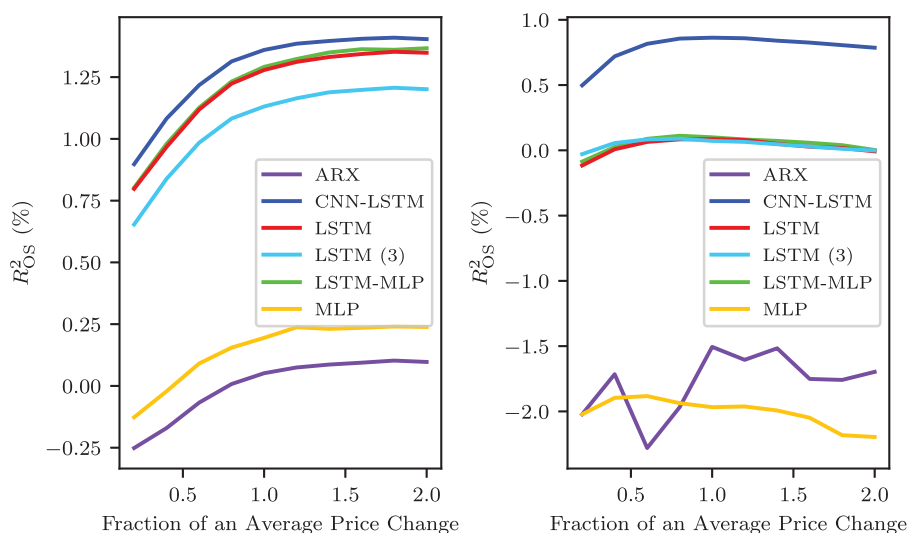


TABLE 3 Summary of inputs and hyperparameters for the models in this article.

Model	Input	Number of layers	Number of parameters <sup>a</sup>	Learning rate	Batch size	Training epochs	Early stopping
ARX	OF	1	$2.0 \times 10^3$	$10^{-4}$	256	50	Yes
CNN-LSTM	OF	27	$1.3 \times 10^5$	$10^{-3}$	256	50	Yes
LSTM	OF	2	$1.0 \times 10^5$	$10^{-5}$	256	50	Yes
LSTM (3)	OF	4	$4.6 \times 10^5$	$10^{-5}$	256	50	Yes
LSTM-MLP	OF	3	$8.4 \times 10^4$	$10^{-5}$	256	50	Yes
MLP	OF	4	$1.3 \times 10^6$	$10^{-5}$	256	50	Yes
ARX	LOB	1	$4.0 \times 10^3$	$10^{-4}$	256	50	Yes
CNN-LSTM	LOB	27	$1.4 \times 10^5$	$10^{-3}$	256	50	Yes
LSTM	LOB	2	$1.1 \times 10^5$	$10^{-5}$	256	50	Yes
LSTM (3)	LOB	4	$4.7 \times 10^5$	$10^{-5}$	256	50	Yes
LSTM-MLP	LOB	3	$9.4 \times 10^4$	$10^{-5}$	256	50	Yes
MLP	LOB	4	$2.3 \times 10^6$	$10^{-5}$	256	50	Yes

<sup>a</sup>The number of parameters are approximated to the nearest order of magnitude and truncated for readability.

## DEEP ORDER FLOW IMBALANCE



**FIGURE 2** Out-of-sample forecasting performance of the models at different horizons from 10 levels of order flows (left panel) and order book states (right panel). Model performance is measured as the average  $R^2_{OS}$  across days and stocks. Horizons are given as the fraction of an average price change for each stock. Each model is trained using a rolling-window out-of-sample methodology across 11 months of data. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1111/jmfi.12414)]

in Section 4.3, we analyze return predictability of the CNN-LSTM, LSTM, and LSTM-MLP, all trained on OF inputs, on longer time scales (up to 10 average price changes). In our empirical work,  $t$ - and  $F$ -statistics are calculated using heteroskedasticity-consistent standard errors (White, 1980).

#### 4.1 | Short-term return predictability

Figure 2 presents the out-of-sample forecasting performance of the models at different horizons using OF inputs (left panel) and LOB inputs (right panel). We train each model using the rolling-window out-of-sample methodology across 48 weeks as described in Section 3.2.3, measuring performance as the average  $R^2_{OS}$  across days and stocks. To produce the curve for each model, we first compute the daily average  $R^2_{OS}$  for each stock and horizon. The resulting performance across stocks is then averaged to obtain the average  $R^2_{OS}$  for each model. Recall that horizons are determined as the fraction of an average price change for each stock (see Section 3.2.1).

Except for the CNN-LSTM, it is apparent that the models are not able to learn from LOBs.<sup>9</sup> As OFs represent a particular transformation of the LOBs, given enough data, ANNs trained on LOBs should in principle be able to learn this (or perhaps an even better) transformation. This was the main motivation for the CNN-LSTM architecture of Zhang et al. (2019b). However, comparing

<sup>9</sup>Of course, an ARX is not sensible in the case of LOBs as the regressors are nonstationary. We include it for completeness only.

the left and right panels in Figure 2, OFs deliver better predictive performance for all the models, independent of model structure or complexity.

The left panel of Figure 2 suggests that the predictive power, as measured by average  $R_{OS}^2$ , increases up to two average price changes for most models. This is in line with expectations. As our models are trained on data at the highest frequency with a relatively short lookback window, we do not expect to generate alphas with a long horizon. In Section 4.3, we examine return predictability beyond two average price changes of the CNN-LSTM, LSTM, LSTM-MLP trained on OF inputs.

The left panel of Figure 2 also shows that ANNs with at least one LSTM module perform significantly better than the ARX and MLP models. In the case of the MLP, a standard explanation for its underperformance is that, despite early stopping, the large number of parameters leads to overfitting the training data and results in poor generalization. More recently though, Wilson and Izmailov (2020) emphasize that “parameter counting is a poor proxy for understanding generalization behavior.” The authors highlight the notion of the *inductive biases* of a model, which in a Bayesian framework describes the datasets that are likely under the model in question. While our approach is not Bayesian, we believe that this reasoning applies here. Specifically, as order book data has a natural ordering, we expect the underlying data generating process to exhibit temporal dependence. An LSTM is intrinsically a sequential model, which naturally captures the temporal dependence, thereby leading to better generalization than an MLP.

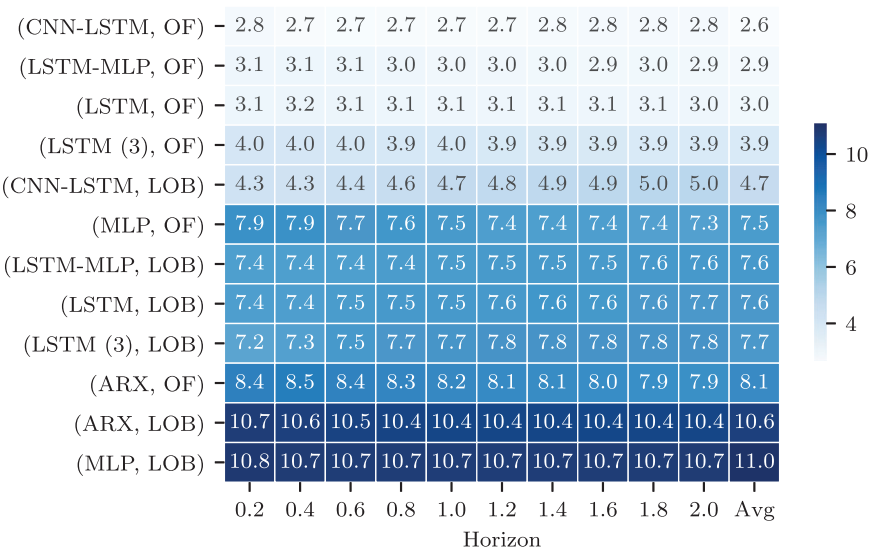
#### 4.1.1 | Rankings of average forecast accuracy

Training the six models on either LOB or OF inputs results in 12 model and input combinations. To compare their out-of-sample performance, we rank them by their average forecast accuracy as follows. For each day, stock, and horizon, we rank the combinations from best to worst based on their  $R_{OS}^2$ , such that the best combination receives a rank of 1 and the worst receives a rank 12. Then for each day, we average the ranks for each combination across stocks, arriving at a daily average rank for each model, input, and horizon. Then, by averaging the daily average ranks over the whole test set, we obtain a global average rank for each model, input, and horizon. These ranks are displayed in the first 10 columns of Figure 3. Similarly, by first averaging the daily average ranks of all the horizons and then averaging over the whole test set, we obtain a global average rank for all models, denoted by “Avg” in Figure 3.

There are two key practical take-aways from the figure. First, ANN models with OF input consistently outperform their LOB counterparts. Second, once an ANN has an LSTM as part of its architecture, its predictive performance increases significantly over the ARX and MLP models.

#### 4.1.2 | Which inputs provide better predictive performance?

To assess which inputs yield better predictive performance, following Patton (2011), Liu et al. (2015), we perform pairwise comparisons of each model, trained on either OF or LOB inputs. The comparisons are done as follows. For each model, forecast horizon, and stock, we first compute the  $t$ -statistic of the average difference in  $R_{OS}^2$  that result from using OF and LOB inputs. Then for each horizon, we calculate the difference between the number of stocks with a significantly positive and negative  $t$ -statistics at a significance level of 1%. The results are displayed in the first 10 columns of Table 4. Here, a positive difference of  $X$  implies that the model with OF inputs



**FIGURE 3** Average ranks of model and input combinations by horizon. The best performing combination for each symbol and day receives a rank of 1, and the worst receives a ranks 12. The last column, denoted by “Avg,” displays rankings based first averaging the daily average ranks of all horizons and then averaging over the whole test set to obtain a global average rank for all models. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1111/jmfi.12413)]

**TABLE 4** Summary results of pairwise comparisons of the  $R^2_{OS}$  from models using either LOB or OF inputs. For each model, forecasting horizon and stock, we first compute a  $t$ -statistic of the average loss difference that result from using OF and LOB inputs. We obtain each table entry by calculating the difference between the number of significantly positive and negative  $t$ -statistics (at a significance level of 1%). To produce the rightmost column, at each point in time, we first average the  $R^2_{OS}$  across the alpha term structure. Then we calculate the difference between the number of significantly positive and negative  $t$ -statistics (at a significance level of 1%). A positive entry  $X$  implies that the model with OF inputs outperformed the same model with LOB inputs for  $X$  stocks in our sample.  $t$ -statistics are calculated using heteroskedasticity-consistent standard errors (White, 1980).

Model	Horizon (fraction of Avg price change)										
	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	Avg
LSTM	115	115	115	115	115	114	115	115	115	115	115
LSTM (3)	114	115	115	115	115	115	115	114	115	114	115
LSTM-MLP	115	115	114	114	115	114	115	115	114	115	115
MLP	111	110	109	110	111	110	110	110	110	110	110
ARX	101	94	89	90	91	90	92	91	93	94	93
CNN-LSTM	70	76	81	86	89	91	92	91	93	92	89

outperforms the same model with LOB inputs for  $X$  stocks in our sample. We repeat the same analysis after first averaging  $R^2_{OS}$  across the alpha term structure for each model, stock, and day. Then as before, we calculate the difference between the number of stocks with a significantly positive and negative  $t$ -statistics at a significance level of 1%, presenting these results in the column denoted by “Avg” in Table 4.

**TABLE 5** Summary of the number of models trained on LOB and OF inputs that have an  $R^2_{OS}$  that is significantly greater than 0. For each model, forecasting horizon and stock, we first compute a  $t$ -statistic of the average  $R^2_{OS}$ . We obtain each table entry by calculating the difference between the number of significantly positive and negative  $t$ -statistics (at a significance level of 1%). To produce the rightmost column, at each point in time, we first average the  $R^2_{OS}$  across the alpha term structure. Then we calculate the difference between the number of significantly positive and negative  $t$ -statistics (at a significance level of 1%). A positive entry  $X$  implies that the model has an  $R^2_{OS}$  that is significantly greater than 0 for  $X$  stocks in our sample.  $t$ -statistics are calculated using heteroskedasticity-consistent standard errors (White, 1980).

Model	Horizon (fraction of Avg price change)										
	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	Avg
(CNN-LSTM, OF)	97	103	109	109	109	108	109	109	109	108	110
(LSTM, OF)	90	98	103	105	109	108	108	107	107	109	109
(LSTM-MLP, OF)	93	96	105	108	110	110	108	107	107	107	109
(LSTM (3), OF)	75	89	98	101	106	106	105	106	106	105	107
(CNN-LSTM, LOB)	47	61	66	68	73	72	74	73	74	74	69

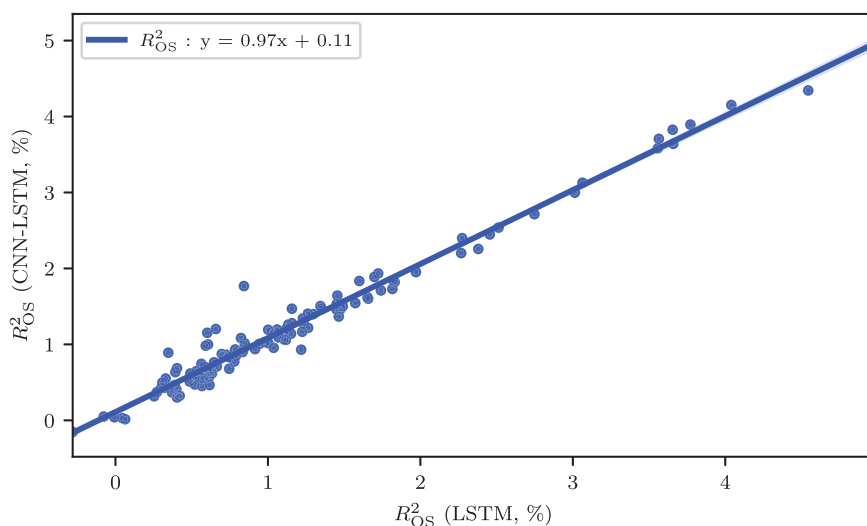
We observe that all models with OF inputs outperform the same models with LOB inputs. This result is particularly strong for the LSTM, LSTM (3), and LSTM-MLP models, where for almost all horizons, OF inputs outperform across the 115 stocks. Interestingly, at the shorter forecasting horizons, there are about 40 stocks for which the CNN-LSTM with LOB inputs perform better, whereas beyond one average price change OF inputs dominate.

4.1.3 | Is there superior forecasting ability?

Above we demonstrated that the best performing model and input combinations are (i) ANNs with at least one LSTM module trained on OFs inputs, and (ii) the CNN-LSTM trained on LOBs. Next, following Patton (2011), Liu et al. (2015), we test whether these models have an  $R^2_{OS}$  that is significantly greater than 0. We perform these tests as follows. For each model, forecasting horizon and stock, we first compute a  $t$ -statistic of their average  $R^2_{OS}$ . Then for each horizon, we calculate the difference between the number of significantly positive and negative  $t$ -statistics (at a significance level of 1%). The results are displayed in the first 10 columns of Table 5, where a positive entry  $X$  implies that the model has an  $R^2_{OS}$  that is significantly greater than 0 for  $X$  stocks in our sample. We repeat the same analysis after first averaging  $R^2_{OS}$  across the alpha term structure for each model, stock, and day, calculating the difference between the number of stocks with a significantly positive and negative  $t$ -statistics at a significance level of 1%. The column denoted by “Avg” in Table 5 summarizes the results. Together, these tests provide strong support for that models trained on OFs have superior out-of-sample forecasting ability, with the CNN-LSTM and LSTM coming out slightly ahead of the rest.

4.1.4 | Discussion

The results above provide support for the fact that ANNs trained on features from the order book can be used to forecast mid-price returns at high frequencies. We demonstrated that the LSTM-based models trained on OFs inputs perform better than all other model and input



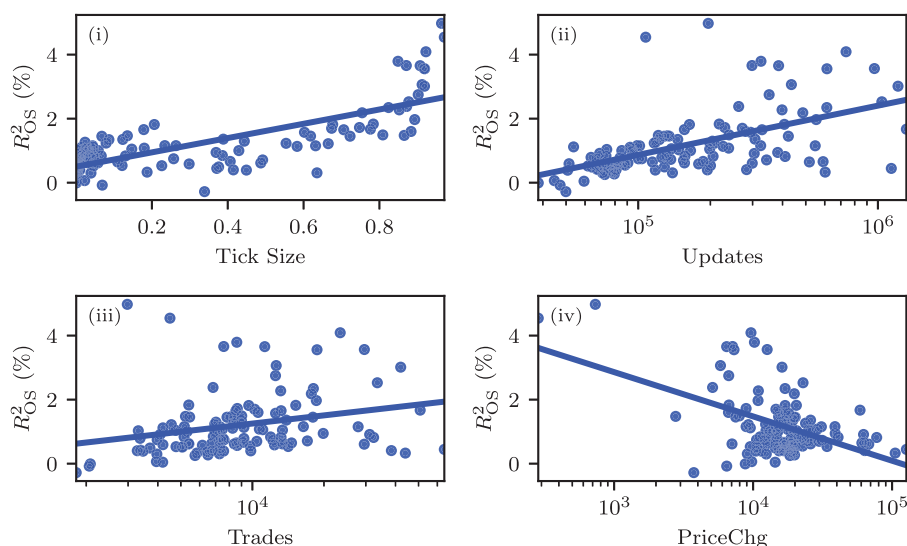
**FIGURE 4** Scatter plot of the  $R^2_{OS}$ 's of the LSTM and CNN-LSTM models. Performance is almost identical across all stocks, suggesting that the models extract similar features. CNN, convolutional neural network; LSTM, long short-term memory. [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

combinations we considered. Perhaps this is not surprising given that the time series of LOBs are nonstationary. We believe that more sophisticated training techniques are likely to be needed to train ANNs directly on LOBs and achieve superior forecasting performance (Arjovsky et al., 2020).

With OF inputs, our results show that the simple shallow LSTM model performs as well as the deeper and more complex models. Surprisingly, even the CNN-LSTM, that uses several convolutional layers on top of an inception module, provides no significant improvement over the simple shallow LSTM. By way of illustration, in Figure 4, we plot the average  $R^2_{OS}$  of the LSTM versus CNN-LSTM models. We observe that the average  $R^2_{OS}$ 's are remarkably similar across all stocks, suggesting that the models extract similar features.

In Appendix B.1, we perform several robustness checks and examine the out-of-sample performance of the models trained on two alternate inputs; in particular, standard OFIs, and LOBs with share volumes only (prices removed). The results show that the out-of-sample performance of all LSTM-based models are qualitatively similar whether they are trained on OFI or OF inputs. Furthermore, we show that the out-of-sample performance of the models trained on LOBs, and LOBs with only share volumes are quite similar, suggesting that bid and ask prices at the first 10 levels of the order book do not add predictive power.

Naturally, as the majority of models in this article are based on standard “off-the-shelf” ANN architectures, one would expect that more specialized architectures may deliver better performance. Our results imply that the question of what is the optimal network for return prediction is still an open question. Notably, the more sophisticated CNN-LSTM architecture shows some promise in being able to work directly with LOB inputs. In particular, across all the forecast horizons, for 23–45 stocks (out of a total of 115), the CNN-LSTM with LOB inputs exhibits improvements over OF inputs (see Table 4). Needless to say, from Table 5, we observe that the CNN-LSTM with OF inputs achieves a statistically significant positive  $R^2_{OS}$  across a significantly larger number of symbols than any of the other models in our study (including that of the CNN-LSTM with LOB inputs).



**FIGURE 5** Scatter plots of stock characteristics and  $R^2_{OS}$ 's from the LSTM model. The  $x$ -axes in the panels represents (i) tick size, (ii) number of updates, (iii) number of trades, and (iv) number of price changes, respectively. Note that number of updates, number of price changes, and number of trades are shown on a logarithmic scale. Tick size represents the fraction of time that the bid–ask spread is equal to one tick for each stock (Curato & Lillo, 2015). LSTM, long short-term memory. [Color figure can be viewed at wileyonlinelibrary.com]

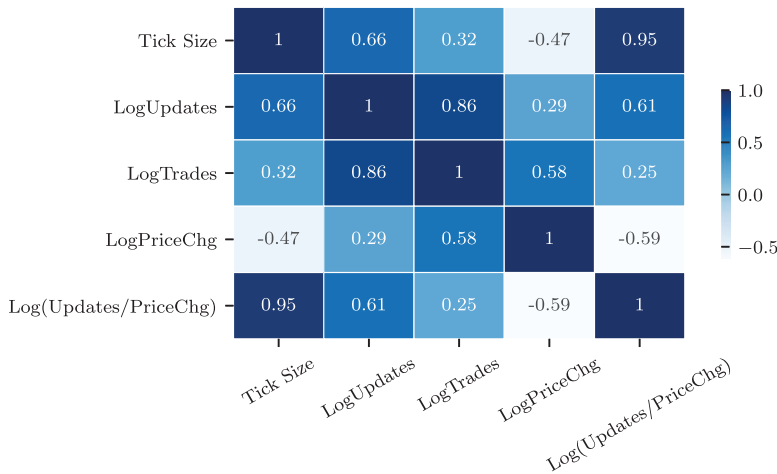
## 4.2 | Forecasting performance of the LSTM and stock characteristics

We turn to analyzing how model performance depends on individual stock characteristics at the microstructure level. Based on the results above, where models with an LSTM module trained on OF inputs showed superior performance, we limit our analysis to the simple LSTM architecture with OF inputs.

We represent the forecasting performance of the LSTM for each stock by its average  $R^2_{OS}$  over the test set. Specifically, for each stock and day, first we average  $R^2_{OS}$  across the alpha term structure. Using the daily averages of  $R^2_{OS}$ 's for each stock, we then determine the grand average over the test set, resulting in a single  $R^2_{OS}$  per symbol.

Following Curato and Lillo (2015), for the purposes of the analysis in this section, we measure tick size for each stock as the fraction of time that the bid–ask spread is equal to one tick. Figure 5 presents scatter plots of forecasting performance versus tick size, number of updates (calendar time), number of trades (calendar time), and number of price changes (calendar time). In panel (i), we observe that a significant proportion of stocks are small-tick and that their  $R^2_{OS}$ 's are clustered in the left-hand side of the plot, while the remaining symbols are distributed fairly uniformly. Notably, there is a sharp rise in forecasting performance for large-tick stocks. As can be seen in panel (ii), the number of updates exhibit a log-linear relationship with  $R^2_{OS}$ , with only minor clustering along the  $x$ -axis. Stocks with a large number of updates show greater variation in forecasting performance. Similarly, in panel (iii), we see that the number of trades and forecasting performance have a log-linear relationship. Lastly, the logarithm of the number of price changes, a proxy for volatility, is negatively correlated with forecasting performance, suggesting that stocks with lower update frequency are less efficient and hence more predictable.





**FIGURE 6** Correlation matrix of the stock characteristics. LogUpdates, LogTrades, and LogPriceChg denote the logarithm of the number of updates, trades, and price changes, respectively. Observe that tick size is highly correlated with the logarithm of updates over price changes. Tick size represents the fraction of time that the bid-ask spread is equal to one tick for each stock (Curato & Lillo, 2015). [Color figure can be viewed at wileyonlinelibrary.com]

Figure 6 shows that the stock characteristics have high pairwise correlations, where LogUpdates, LogTrades, and LogPriceChg denote the logarithm of the number of updates, trades, and price changes, respectively. Notably, tick size and the logarithm of updates over price changes have the remarkably high correlation of 0.95. In the next section, using cross-sectional regressions, we investigate how these stock characteristics relate to out-of-sample performance.

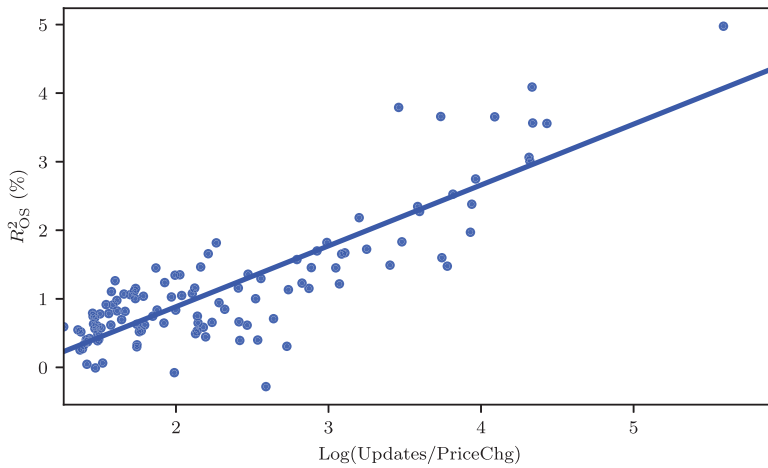
#### 4.2.1 | Cross-sectional regressions

The first eight columns in Table 6 present the results from univariate regressions with tick size, LogUpdates, LogTrades, and LogPriceChg, where  $t$ - and  $F$ -statistics are calculated using heteroskedasticity-consistent standard errors (White, 1980). As we already discussed the sign of the partial correlations above, here we comment only on the  $t$ -statistics. Noticeably, the  $t$ -statistics are significant at the 1% for all stock characteristics except LogTrades, where the  $t$ -statistic of 2.28 corresponds to a  $p$ -value of 2.3%.

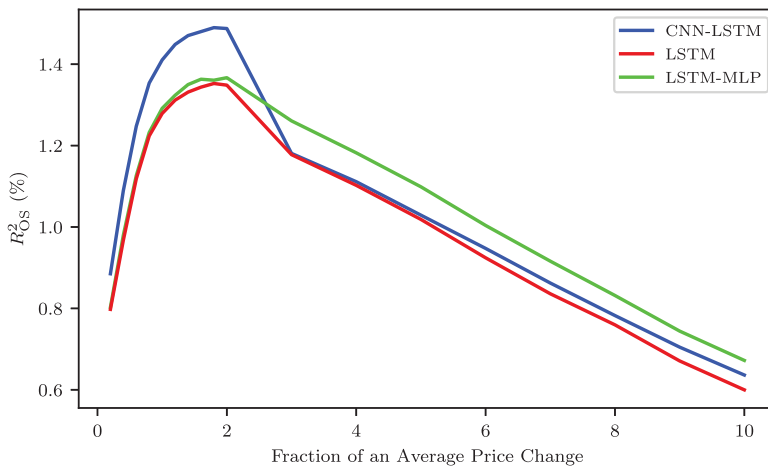
Columns 9 and 10 display the results from multivariate regressions on all the stock characteristic but tick size. We leave out tick size as it is highly correlated with LogUpdates and LogPriceChg. In this setting, LogTrades is no longer significant at the 5% level ( $p$ -value = 26.5%). The number of updates and trades are of course closely related. Particularly, some order book updates result in trades, yet it is well-known in the market microstructure literature that the information contained in order book updates is much richer than trades alone (see, for example, Cont et al., 2014; Bouchaud et al., 2018). Notice that the coefficients for LogUpdates and LogPriceChg are of opposite sign but of similar size, suggesting the regression of  $R_{OS}^2$  on the logarithm of their ratio, Log(Updates/PriceChg); see the 11th and 12th columns in Table 6 and Figure 7. While tick size and the logarithm of this ratio are highly correlated, the latter is superior in explaining  $R_{OS}^2$ .

**TABLE 6** Results from multivariate regressions of stock characteristics on forecasting performance ( $R^2_{OS}$ ) from the LSTM model. LogUpdates, LogTrades, LogPriceChg, and Log(Updates/PriceChg) denote the logarithm of the number of updates, trades, price changes, and number of updates per price change, respectively. Tick size represents the fraction of time that the bid–ask spread is equal to one tick for each stock (Curato & Lillo, 2015).  $t$ - and  $F$ -statistics are calculated using heteroskedasticity-consistent standard errors (White, 1980).

Variable	Coeff	t-stat	Coeff	t-stat	Coeff	t-stat	Coeff	t-stat	Coeff	t-stat	Coeff	t-stat
Intercept	0.005	8.963	−0.068	−5.515	0.070	5.505	−0.021	−1.453	−0.0105	−1.304	−0.009	−7.748
Tick Size	0.022	9.837										
LogUpdates			0.007	6.288					0.008	6.188		
LogPriceChg					−0.006	−4.717			−0.009	−12.140		
LogTrades							0.004	2.277	0.002	1.116		
Log(Updates/PriceChg)											0.009	16.289
Adj. $R^2$	0.586		0.294		0.229		0.057		0.745		0.746	
$F$ -stat	96.76		39.54		22.25		5.19		94.90		265.30	



**FIGURE 7** Scatter plot of  $\text{Log}(\text{Updates}/\text{PriceChg})$  and  $R^2_{OS}$ 's of the LSTM model trained on OF inputs. LSTM, long short-term memory; OF, order flow. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1111/jmfi.12413)]



**FIGURE 8** Predictive power of the models on horizons up to 10 average price changes. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1111/jmfi.12413)]

### 4.3 | Return predictability at longer horizons

Above we demonstrated that there is predictability in mid-price stock returns at short horizons corresponding to the stock-specific time scale of up to two average price changes. In this section, we explore predictability over longer time spans by extending the forecast horizons in Equation (17) up to 10 average price changes. For this analysis, we restrict ourselves to the CNN-LSTM, LSTM, and LSTM-MLP architectures with OF inputs.

Figure 8 depicts the out-of-sample performance, where the curve for each model is produced just as in Section 4.1. We observe that average  $R^2_{OS}$  plateaus at around two average price changes for all models. With a small margin, the CNN-LSTM outperforms the two other models up to about

2.3 average price changes. Thereafter, the LSTM–MLP is slightly better, although the difference is minimal.

#### 4.4 | Discussion: Why is there predictability?

This article contributes to the emerging literature that uses DL in extracting alpha from the most granular information of the order book. While a natural question is *why* this form of predictability exists, to the best of our knowledge, there is little supporting theoretical evidence. Although it is generally accepted that order book events (OF, in particular) impact prices, *specific* details of their interplay is still an open question. In the following, we summarize some of the key findings in the literature and provide some of our conjectures for this phenomenon.

Classical microstructure theories of inventory (Amihud & Mendelson, 1980; Garman, 1976; Ho & Stoll, 1981; Stoll, 1978) and asymmetric-information (Easley & O'Hara, 1987; Glosten & Milgrom, 1985; Kyle, 1985) predict that OF has an impact on prices. For instance, standard inventory models establish that traders acting as market makers adjust their quotes in response to incoming orders, and information-based models predict that trades from informed market participants affect prices. These and related results are also well-established in the empirical literature. For example, Huang and Stoll (1994) find that differences in quoted depth between the bid and ask sides predict returns, especially over short horizons. Hasbrouck (1988), Hasbrouck (1991) apply VAR models to separate permanent information effects from transitory inventory effects, showing that information has a significant effect on prices and OF affects daily returns. Chordia et al. (2002), Chordia and Subrahmanyam (2004) establish that trade-based order imbalances predict daily returns for individual stocks as well as the overall market. Cao et al. (2009) extend these results up to 10 levels of the order book, showing that multilevel imbalances affect returns positively. Needless to say, most of the earlier microstructure literature focus on quote-driven markets, where market makers match orders or fill them from their inventory. In today's electronic markets where trading is conducted via LOBs without any intermediaries, the distinction between market makers and informed market participants is no longer obvious.

Based on their empirical analysis of trade and quote data (best bid and ask at the first level of the order book) of French stocks, Bouchaud et al. (2004), Bouchaud et al. (2006) suggest that markets are statistically efficient in the sense that mid-price return processes are close to white noise. The authors emphasize two main opposing effects contributing to this result. First, OF exhibits long-range correlations due to supply and demand imbalances that take time for the market to digest. Second, while these imbalances alone would lead to price predictability over longer horizons, high-frequency liquidity providers ("market makers"), competing for OF, cause these mispricings to mean-revert at short horizons.<sup>10</sup> Our empirical results in Sections 4.1 and 4.3 support that in the intersection of these two counteracting effects, there is a short-term price predictability.

More recent microstructure literature establishes that not just trades but all updates to the order book are important in understanding price formation; see, for example, the empirical studies of Eisler et al. (2012), Cont et al. (2014), Bouchaud et al. (2018), Brogaard et al. (2019), and theoretical models of Kaniel and Liu (2006), Goettler et al. (2009), Hoffmann (2014). For instance, Eisler et al.

<sup>10</sup> Lillo and Farmer (2004) provide a different explanation, suggesting that the compensating mechanism that leads to uncorrelated returns is that long-range liquidity fluctuations that are correlated with the order flow dampen the otherwise permanent effect of market orders and make the price diffusive.

(2012) conduct an empirical study measuring the individual price impact of all order book events, including market orders, limit orders, and cancellations. For large-tick stocks, they show that their results are consistent with a model where the impact of all order book events is permanent and nonfluctuating. However, they provide compelling evidence that for small-tick stocks price impact must contain a history-dependent part that represents the internal fluctuations of the order book. Toth et al. (2012) argue that, while liquidity providers are only marginally influenced by their past actions, their current action depends strongly on the recent actions of other market participants. The impact of market orders on prices results from a complex interplay of the broker who pushes the price in the direction of their OF, as others may trade in the same or different direction. For instance, the authors show that the autocorrelation of OF in response to a nonprice-changing market order is positive on average, suggesting herding amongst brokers. However, in the case of price-changing market orders, Toth et al. (2012) demonstrate that this autocorrelation tends to be negative as other brokers respond with activity in the opposite direction.

An important question is why the return predictability of our models have higher  $R_{OS}^2$ 's for large-tick stocks than those of small-tick stocks. The cross-sectional regression analysis in Section 4.2 sheds some light about how the return predictability varies across stocks. Higher  $R_{OS}^2$  is in part explained by a greater number of order book updates *relative* to that of the number of price changes. A possible explanation might be the following. The number of order book updates over some *characteristic time interval* is a measure of the rate of activity amongst market participants in the order book. Should this characteristic time interval be wall-clock time, or perhaps stock-specific? Our results above suggest that the time scale of a stock's average price change is a sensible choice, and that stocks with a greater number of updates per price change are characterized by a more granular and information-rich price discovery. Additionally, we conjecture that in the case of information-rich stocks (defined as the logarithm of the ratio of updates to price changes), less information is exploited by "traditional" quantitative models as they frequently aggregate information over a fixed time interval that is not stock-specific. Therefore, using ANNs to extract "unexploited" information from the most granular data leads to improved predictability.

While using a different set of models, we believe that the market microstructure-based explanation of Gould and Bonart (2016) also applies to our setting, providing a possible explanation for the difference in return predictability of large- and small-tick stocks.<sup>11</sup> Gould and Bonart (2016) argue that there are two main ways in which mid-price can change. First, a new limit order can arrive inside the bid–ask spread. Second, either the first-level bid or ask volume,  $v_t^{1,a}$  or  $v_t^{1,b}$ , of the order book is fully consumed. Of course, for large-tick stocks, the average bid–ask spread will be close to the minimum tick size of \$0.01, diminishing the likelihood of new limit orders arriving inside the bid–ask spread. In this case, positive (negative) mid-price returns are governed only by whether the first-level ask (bid) volume is consumed before the first-level bid (ask) volume. Therefore, it is reasonable to expect that the bid and ask volumes provide stronger predictive power for large-tick stocks. Gould and Bonart (2016) also provide an argument based on Nasdaq's price-time priority rule for why large- and small-tick stocks behave differently, even when the bid–ask spread is not at its minimum. Specifically, traders can place limit orders inside the bid–ask spread to achieve higher priority in the order book. For large-tick stocks, the cost associated with this priority is higher (on a relative basis) than that of small-tick stocks. This incentivizes traders to submit limit orders to the first levels of the order book, resulting in average bid and ask volumes,  $v_t^{1,a}$ ,  $v_t^{1,b}$ ,

<sup>11</sup> Gould and Bonart (2016) use logistic regression classifiers to determine the direction of the subsequent mid-price movement based on OFI for 10 stocks on Nasdaq. They find that their model provides a considerable improvement (50%–60%) over their benchmark model for large-tick stocks but only a moderate improvement (10%–30%) for small-tick stocks.

being greater. In contrast, for small-tick stocks, this cost is lower, which make traders more likely to submit limit orders inside the bid–ask spread that decrease the average bid and ask volumes. As before, the greater bid and ask volumes of large-tick stocks (relative to that of small-tick stocks) lead to improved predictability.

## 5 | CONCLUSIONS

In this article, we employed DL in forecasting high-frequency returns at multiple horizons for 115 stocks traded on Nasdaq using order book information at the most granular level. While raw order book data can be used as an input to the forecasting models, we achieve state-of-the-art predictive accuracy by training simpler “off-the-shelf” ANNs on stationary inputs derived from the order book. Specifically, models trained on OF significantly outperform most models trained directly on the order book. Using cross-sectional regressions, we linked the forecasting performance of the LSTM model to stock characteristics at the market microstructure level, suggesting that “information-rich” stocks can be predicted more accurately by deep learning. Finally, we demonstrated that stock-specific forecasts peak at a time scale of about two price changes and declines thereafter.

There are several practical applications for the findings in this article. First, the short-term forecasts can be incorporated into execution algorithms, either as alphas or indirectly in the order submission logic that decides whether to place limit or market orders. Second, the alpha forecasts will be useful in market making systems. In addition, the alphas can be used to design high-frequency trading strategies. Of course, for the latter use case, proper modeling of transaction costs and application of an appropriate execution strategy is critical to their profitability.

Finally, our proposed approach is similar to that of a real-world production setting where the models are updated on a rolling basis. Based on our experience from training the deep learning models in this article across a large number of stocks, we conclude that deploying these models at large scale in practice is fully feasible and no longer a pipe-dream.

## ACKNOWLEDGMENTS

We very much appreciate the research assistance of Antonio Briola for this project, especially for creating the illustration in Figure 1. We thank Joan Bruna for insightful suggestions on transfer learning and out-of-distribution generalization, and Will Whitney for thoughtful discussions on casual inference and confounders. This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. In particular, we are grateful to Shenglong Wang for his assistance when developing our data processing and computing environment on NYU’s Greene and Hudson high-performance computing clusters.

## DATA AVAILABILITY STATEMENT

The data used in this study cannot be shared. It can be obtained from LOBSTER (Huang & Polak, 2011).

## ORCID

Petter N. Kolm  <https://orcid.org/0000-0002-6794-4335>

## REFERENCES

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. <https://www.tensorflow.org/>
- Abergel, F., Anane, M., Chakraborti, A., Jedidi, A., & Toke, I. M. (2016). *Limit order books*. Cambridge University Press.
- Amihud, Y., & Mendelson, H. (1980). Dealership market: Market-making with inventory. *Journal of Financial Economics*, 8(1), 31–53.
- Arjovsky, M., Bottou, L., Gulrajani, I., & Lopez-Paz, D. (2020). Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning, JMLR Workshop and Conference Proceedings* (pp. 17–36).
- Bouchaud, J.-P., Bonart, J., Donier, J., & Gould, M. (2018). *Trades, quotes and prices: Financial markets under the microscope*. Cambridge University Press.
- Bouchaud, J.-P., Gefen, Y., Potters, M., & Wyart, M. (2004). Fluctuations and response in financial markets: The subtle nature of 'random' price changes. *Quantitative Finance*, 4(2), 176–190.
- Bouchaud, J.-P., Kockelkoren, J., & Potters, M. (2006). Random walks, liquidity molasses and critical response in financial markets. *Quantitative Finance*, 6(2), 115–123.
- Briola, A., Turiel, J., & Aste, T. (2020). Deep learning modeling of limit order book: A comparative perspective. *arXiv preprint arXiv:2007.07319*.
- Brogaard, J., Hendershott, T., & Riordan, R. (2019). Price discovery without trading: Evidence from limit orders. *The Journal of Finance*, 74(4), 1621–1658.
- Bugaenko, A. (2020). Empirical study of market impact conditional on order-flow imbalance.
- Cao, C., Hansch, O., & Wang, X. (2009). The information content of an open limit-order book. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 29(1), 16–41.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Chollet, F. (2015). *Keras*. <https://github.com/fchollet/keras>
- Chordia, T., Roll, R., & Subrahmanyam, A. (2002). Order imbalance, liquidity, and market returns. *Journal of Financial Economics*, 65(1), 111–130.
- Chordia, T., & Subrahmanyam, A. (2004). Order imbalance and individual stock returns: Theory and evidence. *Journal of Financial Economics*, 72(3), 485–518.
- Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.
- Cont, R. (2001). Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1(2), 223–236.
- Cont, R., Cucuringu, M., & Zhang, C. (2021). Price impact of order flow imbalance: Multi-level, cross-sectional and forecasting. *Quantitative Finance*. *arXiv preprint arXiv:2112.13213, to appear*.
- Cont, R., Kukanov, A., & Stoikov, S. (2014). The price impact of order book events. *Journal of Financial Econometrics*, 12(1), 47–88.
- Curato, G., & Lillo, F. (2015). How tick size affects the high frequency scaling of stock return distributions. In *Financial econometrics and empirical market microstructure* (pp. 55–76). Springer.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition* (pp. 2625–2634).



- Easley, D., & O'Hara, M. (1987). Price, trade size, and information in securities markets. *Journal of Financial Economics*, 19(1), 69–90.
- Eisler, Z., Bouchaud, J.-P., & Kockelkoren, J. (2012). The price impact of order book events: Market orders, limit orders and cancellations. *Quantitative Finance*, 12(9), 1395–1419.
- Elmsili, B., & Outtaj, B. (2018). Artificial neural networks applications in economics and management research: An exploratory literature review. In *2018 4th International Conference on Optimization and Applications (ICOA)* (pp. 1–6). IEEE.
- Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, 33(4), 917–963.
- Friedman, J., Hastie, T., & Tibshirani, R. (2017). *The elements of statistical learning. Series in Statistics New York* (12th Printing ed.). Springer.
- Garman, M. B. (1976). Market microstructure. *Journal of Financial Economics*, 3(3), 257–275.
- Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10), 2451–2471.
- Glosten, L. R., & 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.
- Goettler, R. L., Parlour, C. A., & Rajan, U. (2009). Informed traders and limit order markets. *Journal of Financial Economics*, 93(1), 67–87.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
- Gould, M. D., & Bonart, J. (2016). Queue imbalance as a one-tick-ahead price predictor in a limit order book. *Market Microstructure and Liquidity*, 2(2), 1650006.
- Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance*, 13(11), 1709–1742.
- Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.
- Graves, A., Mohamed, A.-R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). IEEE.
- Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks And Learning Systems*, 28(10), 2222–2232.
- Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press.
- Hasbrouck, J. (1988). Trades, quotes, inventories, and information. *Journal of Financial Economics*, 22(2), 229–252.
- Hasbrouck, J. (1991). Measuring the information content of stock trades. *The Journal of Finance*, 46(1), 179–207.
- Hasbrouck, J., & Saar, G. (2013). Low-latency trading. *Journal of Financial Markets*, 16(4), 646–679.
- Ho, T., & Stoll, H. R. (1981). Optimal dealer pricing under transactions and return uncertainty. *Journal of Financial Economics*, 9(1), 47–73.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- Hoffmann, P. (2014). A dynamic limit order market with fast and slow traders. *Journal of Financial Economics*, 113(1), 156–169.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2), 251–257.
- Huang, R., & Polak, T. (2011). LOBSTER: Limit order book reconstruction system. *SSRN 1977207*.
- Huang, R. D., & Stoll, H. R. (1994). Market microstructure and stock return predictions. *The Review of Financial Studies*, 7(1), 179–213.
- Jaisson, T., & Rosenbaum, M. (2016). Rough fractional diffusions as scaling limits of nearly unstable heavy tailed Hawkes processes. *The Annals of Applied Probability*, 26(5), 2860–2882.
- Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning* (pp. 2342–2350). PMLR.
- Kaniel, R., & Liu, H. (2006). So what orders do informed traders use? *The Journal of Business*, 79(4), 1867–1913.
- Kercheval, A. N., & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15(8), 1315–1329.
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). on large-batch training for deep learning: Generalization gap and sharp minima.
- Kim, Y., Jernite, Y., Sontag, D., & Rush, A. (2016). Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 30).
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

- Kolm, P. N., & Westray, N. (2023). Information content of cross-sectional and multilevel order flow imbalances: A Bayesian approach. In preparation.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.
- Kyle, A. S. (1985). Continuous auctions and insider trading. *Econometrica*, 53(6), 1315–1335.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Li, Y., & Ma, W. (2010). Applications of artificial neural networks in financial economics: A survey. In *2010 International Symposium on Computational Intelligence and Design* (Vol. 1), pp. 211–214. IEEE.
- Lillo, F., & Farmer, J. D. (2004). The long memory of the efficient market. *Studies in Nonlinear Dynamics & Econometrics*, 8(3), 1.
- Liu, L. Y., Patton, A. J., & Sheppard, K. (2015). Does anything beat 5-minute RV? A comparison of realized measures across multiple asset classes. *Journal of Econometrics*, 187(1), 293–311.
- Luo, W., & Yu, F. (2019). Recurrent highway networks with grouped auxiliary memory. *IEEE Access*, 7, 182037–182049.
- Mäkinen, Y., Kannianen, J., Gabbouj, M., & Iosifidis, A. (2019). Forecasting jump arrivals in stock prices: New attention-based network architecture using limit order book data. *Quantitative Finance*, 19(12), 2033–2050.
- Mertens, L., Ciacchi, A., Lillo, F., & Livieri, G. (2022). Liquidity fluctuations and the latent dynamics of price impact. *Quantitative Finance*, 22(1), 149–169.
- Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent models of visual attention. *arXiv preprint arXiv:1406.6247*.
- Ntakaris, A., Magris, M., Kannianen, J., Gabbouj, M., & Iosifidis, A. (2018). Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*, 37(8), 852–866.
- Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*, 93, 106384.
- Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359.
- Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.
- Passalis, N., Tefas, A., Kannianen, J., Gabbouj, M., & Iosifidis, A. (2020). Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data. *Pattern Recognition Letters*, 136, 183–189.
- Patton, A. J. (2011). Data-based ranking of realised volatility estimators. *Journal of Econometrics*, 161(2), 284–303.
- Pearl, J. (2009). *Causality*. Cambridge University Press.
- Rahimikia, E., & Poon, S.-H. (2023). Machine learning for realised volatility forecasting. *SSRN 3707796*.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536.
- Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4580–4584). IEEE.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117.
- Sermanet, P., & LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of the 2011 International Joint Conference on Neural Networks* (pp. 2809–2813). IEEE.
- Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90, 106181.
- Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proceedings of the Advances in Neural Information Processing Systems*, 28.
- Sirignano, J. A. (2019). Deep learning for limit order books. *Quantitative Finance*, 19(4), 549–570.
- Sirignano, J. A., & Cont, R. (2019). Universal features of price formation in financial markets: Perspectives from deep learning. *Quantitative Finance*, 19(9), 1449–1459.
- Stoll, H. R. (1978). The supply of dealer services in securities markets. *The Journal of Finance*, 33(4), 1133–1151.
- Sutton, R. (2019). *The bitter lesson*. <http://www.incompleteideas.net/IncIdeas/BitterLesson.html>

- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9).
- Toth, B., Eisler, Z., Lillo, F., Kockelkoren, J., Bouchaud, J.-P., & Farmer, J. D. (2012). How does the market react to your order flow? *Quantitative Finance*, 12(7), 1015–1024.
- Tran, D. T., Iosifidis, A., Kannianen, J., & Gabbouj, M. (2019). Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Transactions on Neural Networks And Learning Systems*, 30(5), 1407–1418.
- Tsantekidis, A., Passalis, N., Tefas, A., Kannianen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)* (Vol. 1, pp. 7–12). IEEE.
- Tsantekidis, A., Passalis, N., Tefas, A., Kannianen, J., Gabbouj, M., & Iosifidis, A. (2020). Using deep learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing*, 93, 106401.
- Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3156–3164).
- White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4), 817–838.
- Wilson, A. G., & Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*.
- Wong, B. K., & Selvi, Y. (1998). Neural network applications in finance: A review and analysis of literature (1990–1996). *Information & Management*, 34(3), 129–139.
- Xu, K., Gould, M. D., & Howison, S. D. (2018). Multi-level order-flow imbalance in a limit order book. *Market Microstructure and Liquidity*, 04(03n04), 1950011.
- Zhang, Z., Lim, B., & Zohren, S. (2021). Deep learning for market by order data. *Applied Mathematical Finance*, 28(1), 79–95.
- Zhang, Z., & Zohren, S. (2021). Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units. *arXiv preprint arXiv:2105.10430*.
- Zhang, Z., Zohren, S., & Roberts, S. (2018). BDLOB: Bayesian deep convolutional neural networks for limit order books. *arXiv preprint arXiv:1811.10041*.
- Zhang, Z., Zohren, S., & Roberts, S. (2019a). DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11), 3001–3012.
- Zhang, Z., Zohren, S., & Roberts, S. (2019b). Extending deep learning models for limit order books to quantile regression. *arXiv preprint arXiv:1906.04404*.

**How to cite this article:** Kolm, P. N., Turiel, J., & Westray, N. (2023). Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. *Mathematical Finance*, 33, 1044–1081. <https://doi.org/10.1111/mafi.12413>

## APPENDIX A: PREPROCESSING LOBSTER DATA

In this section, we discuss the format of the LOBSTER data and summarize the preprocessing steps we performed for this article. For additional information, we refer to Bouchaud et al. (2018, Appendix A) and Bugaenko (2020, Appendix C).

LOBSTER provides two files for each date and symbol, the message and order book files. The number of rows in each file is identical and the timestamps are given to nanosecond precision. The message file lists every market order arrival, limit order arrival, and cancellation. We note that the arrival of hidden orders is not provided. The order book file lists the states of the LOB

with each row corresponding to the state after the corresponding event from the message file. It is the order book file which is the primary object of study in this paper.

Inspection of the message file reveals that there are certain clusters of events that require special care. In particular, consider the following three situations:

- (1) Limit order modification, which is implemented as a cancellation followed by an immediate new arrival.
- (2) Single market order executing against multiple resting limit orders.
- (3) Auction prints: The auction trade is printed and all the remaining resting limit orders are moved into the LOB.

Common amongst these situations is that we have one conceptual event which appears as multiple rows in the message file, all with the same timestamp. Consequently, it is critical for modeling purposes that these events are collapsed to avoid artificially inflated prediction accuracy. Therefore, we group the order book data by unique timestamps and take the last entry (they are ordered sequentially within a timestamp by the process, which generates this data). This type of processing is standard in the literature using LOBSTER data (see, for example, Bouchaud et al., 2018; Bugaenko, 2020). In addition, we remove all rows for which the quotes are crossed. As noted in Bouchaud et al. (2018), the occurrence of such events is extremely rare.

Table A.1 provides summary statistics after preprocessing the LOBSTER data for the 115 stocks used in this study.

**TABLE A.1** Summary statistics of the 115 stocks from Nasdaq used in this study. The data are sourced from LOBSTER, covering the time period January 1, 2019 through January 31, 2020. Updates, trades, and price changes represent the average number of order book updates, trades, and price changes in thousands per day. Price, spread, and volume denote the average price in dollars, bid–ask spread in basis points, and daily volume in dollars, respectively.

<b>Ticker</b>	<b>Updates (000)</b>	<b>Trades (000)</b>	<b>Price changes (000)</b>	<b>Price (USD)</b>	<b>Spread (bps)</b>	<b>Volume (USD MM)</b>
AAL	398.13	13.15	10.99	30.65	3.88	50.29
AAPL	1137.36	64.19	127.86	215.80	0.99	1156.30
ADBE	205.29	13.20	41.76	283.83	4.08	194.36
ADI	238.35	10.70	29.23	109.36	3.52	73.59
ADP	128.11	7.82	20.02	160.65	3.62	72.97
ADSK	118.22	8.34	24.27	161.21	5.48	74.96
ALGN	80.86	6.78	20.01	251.14	9.68	81.68
ALXN	93.19	7.47	18.08	117.26	7.45	52.07
AMAT	611.97	18.05	17.15	47.36	2.62	91.72
AMD	1213.04	42.06	16.26	31.33	3.64	343.63
AMGN	137.56	11.06	25.61	198.36	3.85	138.33
AMZN	304.76	31.14	64.23	1795.51	2.59	1531.40
ANSS	63.55	3.37	15.22	206.76	9.39	25.86
ASML	147.63	3.31	21.21	223.92	7.06	46.42
ATVI	423.01	17.97	20.25	49.87	2.81	91.82

(Continues)

TABLE A.1 (Continued)

Ticker	Updates (000)	Trades (000)	Price changes (000)	Price (USD)	Spread (bps)	Volume (USD MM)
AVGO	136.68	12.20	32.10	289.33	4.62	182.39
BIDU	155.90	13.37	26.50	132.58	4.79	129.79
BIIB	88.20	9.80	23.17	265.36	6.01	129.65
BKNG	51.44	4.20	15.00	1881.52	8.68	169.08
BMRN	61.69	5.08	10.81	82.87	10.20	25.70
CDNS	128.56	7.32	15.40	64.99	4.28	32.49
CDW	53.73	4.20	9.73	112.53	7.93	28.02
CERN	117.85	7.57	10.59	66.41	3.53	38.58
CHKP	67.19	3.98	10.82	114.42	6.51	27.00
CHTR	80.97	6.75	19.21	400.68	6.22	111.71
CMCSA	612.78	18.70	7.44	42.40	2.58	128.95
COST	125.41	9.21	26.87	264.93	3.26	120.40
CPRT	80.99	4.77	9.66	73.53	5.03	23.75
CSCO	736.33	23.41	9.81	50.37	2.17	191.94
CSGP	38.27	2.08	9.03	536.05	23.06	34.75
CSX	294.56	12.52	18.20	72.07	2.35	78.23
CTAS	64.05	3.71	14.75	237.20	7.29	34.05
CTSH	271.85	10.22	14.73	64.84	2.39	58.60
CTXS	103.49	7.60	12.32	102.47	3.55	52.46
DISH	138.93	7.27	9.16	34.30	4.95	21.35
DLTR	132.57	8.14	15.42	102.30	4.23	57.93
EA	198.67	13.38	26.25	96.82	4.19	95.90
EBAY	435.54	12.59	5.91	37.05	2.98	66.41
EXC	298.32	7.58	7.21	47.35	2.45	37.55
EXPE	122.44	8.33	16.93	122.62	4.45	62.15
FAST	229.64	8.64	8.99	44.16	3.47	34.60
FB	602.54	44.01	106.38	184.08	1.71	622.29
FISV	178.66	12.38	20.43	97.41	3.14	90.26
FOX	199.41	4.56	9.31	37.81	4.33	19.34
FOXA	239.37	9.02	8.03	38.22	3.39	44.54
GILD	407.04	15.22	20.67	65.53	2.22	96.45
GLIBA	47.21	2.06	6.55	61.12	15.97	7.51
GOOG	590.59	13.38	64.96	1206.29	4.71	468.31
GOOGL	517.61	14.61	62.70	1208.90	4.40	445.79
HAS	73.34	5.16	11.42	101.28	6.61	30.45
HOLX	134.48	5.45	7.55	48.02	3.85	20.87
HSIC	99.05	5.18	9.01	66.23	5.41	23.73
IDXX	67.22	3.39	16.85	251.26	9.72	30.08
ILMN	79.51	6.50	20.37	310.12	7.77	84.54

(Continues)

TABLE A.1 (Continued)

Ticker	Updates (000)	Trades (000)	Price changes (000)	Price (USD)	Spread (bps)	Volume (USD MM)
INCY	72.65	5.39	12.91	81.55	8.74	26.16
INTC	966.81	29.58	12.86	52.13	2.10	231.35
INTU	100.24	7.37	23.84	257.26	5.30	88.90
ISRG	78.95	5.84	20.09	535.71	7.83	94.48
JBHT	73.90	4.76	12.57	104.88	7.74	28.07
JD	552.74	18.58	10.95	29.94	3.79	95.11
KHC	351.72	12.51	6.75	32.58	3.48	58.06
KLAC	94.44	7.33	19.12	136.17	6.19	53.08
LBTYA	217.71	5.40	6.73	24.95	5.39	13.11
LBTYK	262.49	6.83	5.15	24.16	4.99	19.60
LILAK	49.89	1.81	3.77	18.09	15.92	2.63
LRCX	154.50	10.43	30.42	215.34	5.48	111.39
LULU	92.51	8.25	19.44	184.49	6.34	87.43
MAR	116.61	6.94	17.30	130.38	4.51	54.67
MCHP	146.21	8.99	19.50	91.11	4.95	51.90
MDLZ	385.19	11.27	6.57	51.78	2.16	68.01
MELI	68.15	3.28	16.99	538.46	19.44	72.08
MNST	207.65	9.02	12.41	59.63	3.14	45.60
MSFT	1314.51	50.80	59.44	132.70	0.96	674.33
MU	1038.48	33.55	23.06	43.55	2.64	224.59
MXIM	165.32	6.96	13.37	57.14	4.29	28.77
MYL	280.72	10.56	6.72	22.25	5.33	30.69
NFLX	299.09	29.56	69.40	329.25	3.92	554.70
NTAP	186.58	9.14	16.83	60.89	4.14	40.97
NTES	59.40	4.19	14.83	268.10	13.15	55.21
NVDA	410.14	32.36	78.34	179.73	3.12	388.83
NXPI	156.79	11.93	21.49	103.13	5.01	88.60
ORLY	71.82	5.06	18.57	393.07	7.90	69.74
PAYX	204.00	7.03	15.93	81.79	2.86	38.72
PCAR	131.38	6.85	13.03	70.63	3.82	34.43
PEP	307.10	13.89	26.13	128.67	1.73	125.57
PYPL	381.85	19.86	39.32	105.65	2.21	164.14
QCOM	547.61	27.86	31.30	73.04	2.23	213.91
QRTEA	121.32	5.23	2.79	13.54	9.75	9.53
REGN	72.53	5.74	18.86	344.93	9.32	70.71
ROST	155.81	8.53	17.27	103.03	3.85	55.34
SBUX	485.99	17.70	19.99	82.04	1.63	139.81
SGEN	44.67	3.95	9.96	82.34	15.15	19.56

(Continues)

TABLE A.1 (Continued)

Ticker	Updates (000)	Trades (000)	Price changes (000)	Price (USD)	Spread (bps)	Volume (USD MM)
SIRI	107.42	4.50	0.29	6.21	16.69	25.20
SNPS	85.39	5.32	15.31	125.01	5.77	35.22
SPLK	84.00	7.54	19.09	129.00	8.36	57.97
STX	226.02	9.21	14.74	50.18	3.67	38.19
SWKS	114.57	8.55	16.98	86.36	5.71	46.25
TMUS	194.31	9.84	15.56	75.70	3.08	60.40
TSLA	283.15	38.43	65.03	292.35	4.66	724.21
TTWO	101.33	7.31	18.13	112.77	6.62	51.14
TXN	324.28	16.38	39.16	116.63	2.43	136.22
UAL	163.59	9.65	17.19	86.65	3.98	59.90
ULTA	77.25	6.45	19.19	297.27	8.64	79.56
VOD	195.93	2.98	0.74	18.56	5.62	14.53
VRSK	71.10	3.96	12.71	142.97	5.62	28.14
VRSN	69.03	4.10	15.10	191.87	6.80	34.27
VRTX	76.14	6.16	17.19	187.81	7.44	58.90
WBA	328.50	13.20	15.18	57.81	2.54	75.19
WDAY	87.49	8.63	20.57	185.12	8.02	86.66
WDC	343.08	17.03	24.69	51.91	3.89	77.10
WLTW	50.77	3.47	11.74	185.22	8.16	27.11
WYNN	90.56	8.39	19.15	123.37	7.61	64.33
XEL	325.18	8.60	10.32	59.35	2.06	44.25
XLNX	200.22	14.14	34.38	107.30	4.31	103.57
XRAY	83.07	5.41	7.11	52.32	4.35	21.67



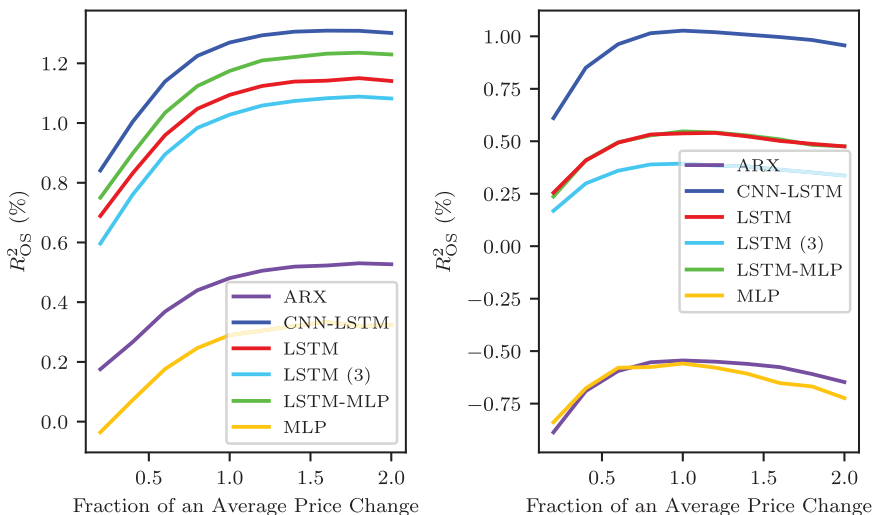
## APPENDIX B: SUPPLEMENTARY RESULTS

### B.1 | Alternate inputs

We report on the out-of-sample performance of our models trained on two alternate inputs: (i) standard OFIs (see Equations 5), and (ii) LOBs with share volumes only (i.e., inputs of the form of Equation 1 where bid and ask prices have been removed). Just as for the OFs in the main article, the different sizes of these inputs necessitate small adjustments of the CNN-LSTM architecture. For instance, the CNN-LSTM with OFI inputs is made up of block 3 through block 5 only, with the input to the first layer being of size  $100 \times 10 \times 1$ .

Figure B.1 depicts the out-of-sample performance for the models with OFIs (left panel) and LOBs without prices (right panel). Comparing the left panels of Figures 2 and B.1, we observe that the out-of-sample performance of all LSTM-based models are qualitatively similar whether trained on OFI or OF inputs. However, when the ARX and MLP are trained on OFI instead of OF inputs, their  $R^2_{OS}$  improve from about 0.1 to 0.5 and 0.25 to 0.3, respectively. This improvement is likely due to the reduction of model parameters when going from OF to OFI inputs. Additional tuning of the MLP hyperparameters and customized feature engineering for the ARX model are interesting topics that will likely further improve their performance. However, as they are outside the scope of this article, we do not explore them here.

Examining the right panels of Figures 2 and B.1, it is clear that the out-of-sample performance of the models trained on full LOBs or LOBs with share volumes only are qualitatively similar. This suggests that bid and ask prices are not confounding factors for returns and volumes (Pearl, 2009).



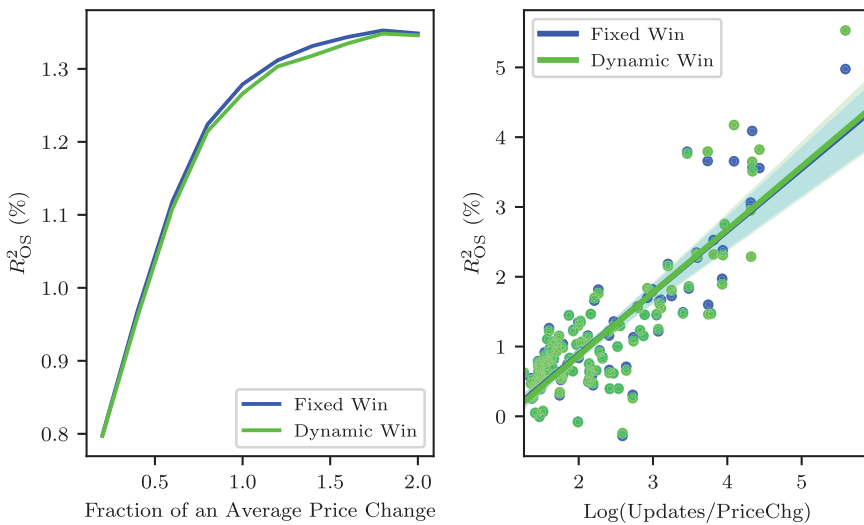
**FIGURE B.1** Results from robustness checks of the out-of-sample forecasting performance of models at different horizons from 10 levels of standard OFIs (left panel) and order book states with bid and ask prices removed (right panel). Model performance is measured as the average  $R^2_{OS}$  across days and stocks. Horizons are given as the fraction of an average price change for each stock. Each model is trained using a rolling-window out-of-sample methodology across 11 months of data. OFIs, order flow imbalances. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

## B.2 | Short-term return predictability and its dependence on the training window

In the training of the models in the main article, we use a fixed lookback window of  $W = 100$  for all stocks (see Section 3.2 for details). Here we revisit this assumption, examining the performance of dynamic lookback windows that are specific to each stock.

We choose the stock specific windows as follows. First, for each stock, we determine the average number of updates that occur during two price changes and multiply this number by three. If the result is less than or equal to 300, we set this to be the stock-specific window size; if it is greater than 300, we cap the stock specific window size to 300. This results in a median stock-specific window length of 50 updates, and 25th and 75th percentiles of 29.5 and 107 updates, respectively.

Figure B.2 compares fixed and dynamic lookback windows for the LSTM model trained on OF inputs for all stocks. We train and report the out-of-sample performance in the same way as in the main article. The left panel illustrates that the model performance of the two different kinds of training windows are quite similar. The right panel provides a scatter plot of  $R_{OS}^2$ 's and  $\text{Log}(\text{Updates}/\text{PriceChg})$  of the two approaches. These results suggest that window length is less important as a driver of out-of-sample predictive performance.



**FIGURE B.2** The left panel shows the out-of-sample forecasting performance of the LSTM model trained OF inputs with a fixed window length of 100 updates (Fixed Win, in blue), as in the main article, and a stock-specific window lengths (Dynamic Win, in green). Model performance is measured as the average  $R_{OS}^2$  across days and stocks. Horizons are given as the fraction of an average price change for each stock. Each model is trained using a rolling-window out-of-sample methodology across 11 months of data. The right panel provides a scatter plot of  $\text{Log}(\text{Updates}/\text{PriceChg})$  and  $R_{OS}^2$ 's of the LSTM model trained on OF inputs with a fixed window length of 100 updates (in blue) and a stock-specific window length (in green). LSTM, long short-term memory; OF, order flow. [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]