Extended Abstract: Neural Networks for Limit Order Books

Justin A. Sirignano
Department of Mathematics, Imperial College London
Mathematical Finance Section

January 2, 2016*

Abstract

We design and test neural networks for modeling the dynamics of the limit order book. In addition to testing traditional neural networks originally designed for classification, we develop a new neural network architecture for modeling spatial distributions (i.e., distributions on $\mathbb{R}^d$) which takes advantage of local spatial structure. Model performance is tested on 140 S&P 500 and NASDAQ-100 stocks. The neural networks are trained using information from deep into the limit order book (i.e., many levels beyond the best bid and best ask). Techniques from deep learning such as dropout are employed to improve performance. Due to the computational challenges associated with the large amount of data, the neural networks are trained using GPU clusters. The neural networks are shown to outperform simpler models such as the naive empirical model and logistic regression, and the new neural network for spatial distributions outperforms the standard neural network.

*The author thanks the Mathematical Finance Section of the Department of Mathematics at Imperial College London for generously providing funds for computations.

1 Introduction

This extended abstract describes the development and testing of neural networks for limit order book modeling. The "limit order book" is a system used by financial exchanges such as NASDAQ and NYSE to match buyers and sellers of stocks. It governs how buy and sell orders can be submitted, when and how orders are executed, and ultimately the stock's price evolution. The complexity and high-dimensionality of limit order books make modeling challenging. A limit order book consists of hundreds of price levels where orders may be submitted and its dynamics are nonlinear. Modeling requires the analysis of large amounts of data, which can be both statistically and computationally challenging.

In spite of the wealth of research on limit order books, there is very little literature adapting machine learning methods to the limit order book setting. In this paper, we design and test neural networks for modeling limit order book dynamics. Neural networks are particularly well-suited for limit order books due to their ability to perform well with high-dimensional data and capture nonlinear relationships. Neural networks also scale favorably with large amounts of data. In addition to investigating the performance of traditional neural networks originally designed for classification tasks, we develop a new neural network architecture for modeling spatial distributions (i.e., distributions on $\mathbb{R}^d$). This new architecture has several advantages over a standard classification architecture for modeling distributions on $\mathbb{R}^d$, including better generalization over space, lower computational expense, and the ability to take advantage of any "local spatial structure". For the dataset considered in this paper, this "spatial neural network" has lower out-of-sample error, much faster training times, and greater interpretability than the standard classification architecture for neural networks.

We train and test models using limit order book data for 140 S&P 500 and NASDAQ-100 stocks. In total, there is over 100 terrabytes of raw data, which is filtered to create training, validation, and test sets for the limit order book at a 1-second time interval. There are serious technical challenges to analyzing the large amounts of data. Data storage and processing is performed using distributed storage and parallel computing on Amazon Web Services (AWS). Model training is also computationally expensive. GPU clusters on AWS are used to train and test deep neural networks. We employ techniques from deep learning such as dropout and inter-layer batch normalization.

We compare several approaches for modeling the joint distribution of the best ask and best bid prices at a future time conditional on the current state of the limit order book. In out-of-sample tests, neural networks strongly outperform simpler approaches such as assuming the naive empirical distribution or logistic regression. Failure to outperform the naive empirical model would imply that the state of the limit order book contains no information on future price movements. The spatial neural network outperforms the standard neural network for classification. In addition, we find that including information from deeper in the limit order book (beyond the best bid and best ask) improves performance.

1.1 Related Literature

Significant research has been conducted with regards to limit order book dynamics and several modeling approaches have been developed. Cont, Stoikov & Talreja (2010), Cont & Larrard (2013), Avellaneda & Stoikov (2008), and Avellaneda, Reed & Stoikov (2011) develop stochastic models for limit order book dynamics. Another related vein of research considers the optimal execution of a buy or sell order. Optimal execution requires understanding the price impact of an executed order given the current state of the limit order book. Cont, Kukanov & Stoikov (2014), Eisler, Bouchaud & Kockelkoren (2012), Maglaras, Moallemi & Zheng (2015), Moallemi, Park & Roy (2012), Alfonsi, Fruth & Schied (2010), Alfonsi & Schied (2010), Predoiu, Shaikhet & Shreve (2011), Schied & Schoneborn (2009), Bayraktar & Ludkovski (2014), and Donier, Bonart, Mastromatteo & Bouchaud (2014) study this problem and develop approaches for optimal execution. Stochastic models have also been developed for limit order books using asymptotic approximations; this is quite natural given the nature of limit order books. Limit order books have large amounts of orders as well as a large number of ticks (i.e., price levels). Cont & Larrard (2012), Blanchet & Chen (2013), and Gao, Dai, Dieker & Deng (2014) develop limiting laws in this spirit. A number of other papers such as Cont (2011), Bouchaud, Mezard & Potters (2002), Potters & Bouchaud (2003), Biais, Hillion & Spatt (1995), Gould & Bonart (2015), and Gould, Porter, Williams, McDonald, Fenn & Howison (2013) empirically study the statistical characteristics of limit order books.

There is relatively little literature on machine learning approaches to limit order books. Kearns & Nevmyvaka (2006) use reinforcement learning for optimal order execution. Kercheval & Zhang (2013) use support vector machines to model limit order books. Kempf & Korn (1999) study the relationship between price changes and net order flow for German index futures with neural networks. Fletcher & Shawe-Taylor (2013) use multiple kernel learning for currency exchange rate prediction.

1.2 Advantages of Neural Networks over other Machine Learning methods

Neural networks are particularly well-suited to limit order book modeling due to their highly flexible and nonlinear characteristics, successful performance for high-dimensional inputs, and favorable scaling with large amounts of data. There are certainly other methods which could be applied. Decision trees, boosted trees, and random forests are also able to learn nonlinear functions. However, their disadvantage is that they divide the input space into rectangular cells while neural networks can learn arbitrary functions of the input space. The ability to learn arbitrary functions (i.e., to *generalize*) is essential for high-dimensional inputs; dividing a high-dimensional space into rectangular cells quickly suffers from the curse of dimensionality. Decision trees are also not optimized for online learning; upon the arrival of new data, typically the complete structure of the decision tree will change and it must be re-formed from scratch. In contrast, neural networks are easily trained online: the parameters can be simply be updated with minibatch gradient descent. Gaussian process regression is another method which rivals neural networks in accuracy for small data sets. Unfortunately, Gaussian process regression does not scale well and quickly becomes intractable for larger data sets.

Neural networks have achieved major success on many classification tasks, such as image classification. Neural networks have 99% accuracy on the MNIST dataset and over 90% accuracy on the CIFAR 10 dataset.[1]

1.3 Organization of this Extended Abstract

Section 2 describes the dataset. In Section 3 different neural network architectures are analyzed for modeling spatial distributions. The deep learning approaches and the GPU computational framework used to train neural networks are explained in Section 4. Out-of-sample results for predicting the distribution of the best bid and best ask are reported in Section 5.

2 The Data

We use Level III limit order book data from the NASDAQ stock exchange. For each stock, there is event-by-event data recording the current state of the limit order book. Every event (order submission, cancellation, and transactions) in the limit order book is recorded as well as the state of the limit order book at the time of each event. The times of events are reported at either the millisecond or nanosecond granularity, depending upon the time period. Between events, the limit order book state does not change. The limit order book data includes the first 400 levels in the limit order book (200 on the ask side and 200 on the bid side). At each level, it reports the price and the volume. Figure 1 shows an example of the state of Microsoft's limit order book at one particular time for the first 15 ask prices and bid prices.

The data includes trading halts. During the trading halts, the limit order book is reported as unchanging. These samples are removed from the dataset for model training and testing.

The data used in this paper comes from the time period January 1, 2014 until August 31, 2015 and covers 140 stocks from the S&P 500 and NASDAQ-100. Notable stocks in the dataset include Facebook, Apple, Netflix, Amazon, Amgen, Bank of America, Microsoft, Boeing, Berkshire Hathaway (Class B shares), Broadcom, and Caterpillar. In total, the raw data is over 100 terrabytes, which is filtered to create training, validation, and test sets for the limit order book at a 1-second time interval. Processing and storage of the raw dataset is challenging due its large size. We use distributed storage and parallel computing to store and process this data,. Training complex models with many parameters (such as neural networks with multiple layers) on such a large data is computationally expensive. We train and test models using GPU clusters on Amazon Web Services (AWS); see Section 4 for more details.

[1]The MNIST dataset contains images of handwritten digits. The goal is to classify correctly the handwritten digits. The CIFAR 10 dataset is composed of images from ten different classes (e.g., automobiles, birds, dogs). The goal is to classify correctly an image as one of the ten classes.

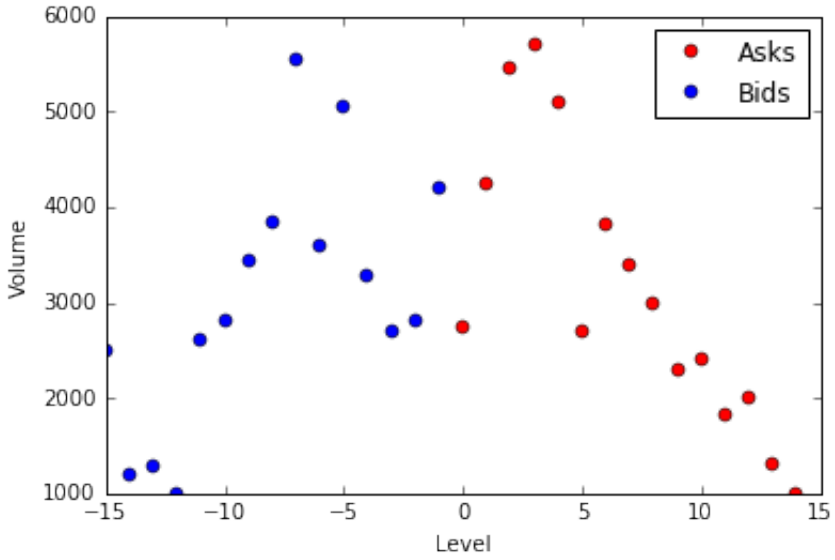

Figure 1: Limit order volumes for first 15 ask and bid prices for Microsoft. Level 0 is the best ask price.

2.1 Some summary statistics of the dataset

Statistics for the spread and mid-price are given in Table 1.[2] We highlight that the spread is typically more than one tick ($.01); therefore, the movements of the best ask and best bid must both be modeled. One cannot simply model the mid-price and assume that the spread is a constant one tick.

Feature	Mean	Median	10 % Quantile	25 % Quantile	75 % Quantile	90 % Quantile
Spread	$.044	$.02	$ .01	$ .01	$ .04	$ .09
Mid-price	77.12	52.38	34.51	38.15	100.55	131.91

Table 1: Summary statistics of dataset.

2.2 Nonlinearity of Limit Order Books

The goal is to use neural networks to capture nonlinear relationships between the state of the limit order book and the distribution of future best bid and ask prices. It is well-known that the limit order book has a nonlinear relationship with future price movements.[3] An example of nonlinear behavior for Bank of America is shown below in Figure 2, where one can see that the probability the best ask price increases has a strong nonlinear dependance on the volume at the best ask.

We show later that neural networks have consistently lower error than logistic regression across many different stocks. A logistic regression is the softmax of a linear function of the data features while a neural

[2] Statistics are calculated using a sub-sample.
[3] For instance, see Figure 4 in Cont & Larrard (2013).

network is the softmax of a nonlinear function of the data features. The neural networks' outperformance indicates the existence of significant nonlinearity in limit order book dynamics.

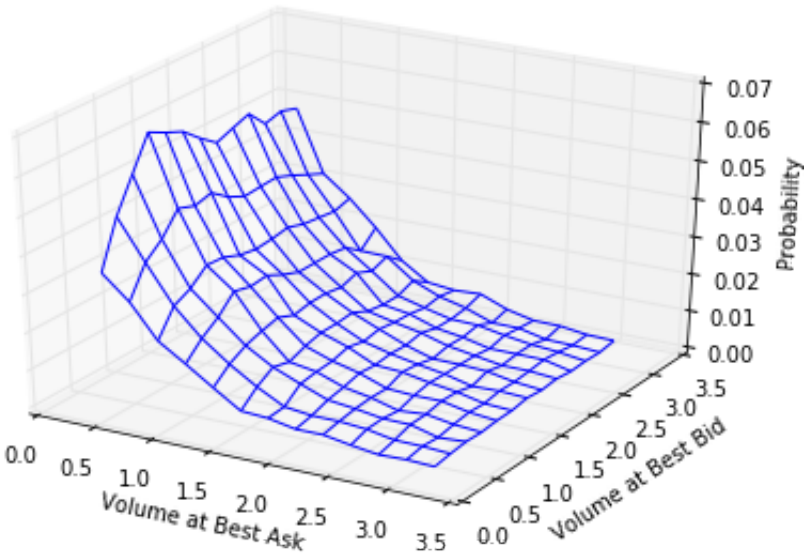

Figure 2: Probability that the best ask price for Bank of America increases for a 1-second horizon. The units for volume are 10^4 shares.

2.3 Local spatial structure in limit order books

Limit order books exhibit some local spatial structure. Without loss of generality, let the current best ask price be level 0. Conditional on the future best ask price Y being greater or equal to level y where $y > 0$, the probability that $Y > y$ appears to strongly depend upon the volume of the limit order book at the level y. Figure 3 demonstrates this phenomenon. The conditional probability that $Y > y$ given $Y \geq y$ decreases with the volume at y.

There is some intuition regarding why the relationship in Figure 3 may hold. To reach a level $y' > y$, the sell limit orders at level y must first be consumed by buy orders. The larger the sell limit order volume at level y, the less likely the future best ask price will reach a level $y' > y$. Since we have already conditioned on $Y \geq y$, the limit orders at levels $y' < y$ are less relevant. Similarly, the event $Y > y$ requires only that the buy orders consume the sell limit orders at y, so the limit order volumes at levels $y' > y$ are less important.

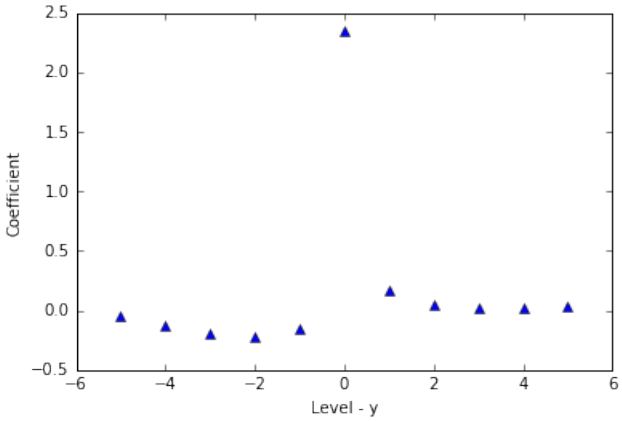

Figure 3: Coefficients from logistic regression for the probability that the future best ask price Y is greater than y conditional on $Y \geq y$ where $y > 0$; i.e., $\mathbb{P}[Y > y | Y \geq y] = \left(1 + \exp(\theta \cdot \text{factors})\right)^{-1}$. The current best ask price has been centered at 0. The plotted coefficients are coefficients for the limit order volumes at price levels minus y. The conditional probability that $Y > y$ given $Y \geq y$ decreases with the volume at y. The reported coefficients were fitted on the stock Amazon.

3 Neural Network Architectures for Modeling Distributions on $\mathbb{R}^d$

The goal of this paper is to model a distribution on $\mathbb{R}^d$ via neural networks. Although there are many applications, we are particularly motivated by modeling the joint distribution of the best ask and best bid prices at a future time conditional on the current state of the limit order book.

In order to model a distribution on $\mathbb{R}^d$, we first discretize $\mathbb{R}^d$ into the grid $\mathcal{R}^d$ where $\mathcal{R} = \ldots, r_{-2}, r_{-1}, 0, r_1, r_2, \ldots$ and then model a distribution on the discrete space $\mathcal{R}^d$. In the context of limit order books, this discretization is exact since price levels are discrete multiples of the tick-size.

Section 3.1 reviews the standard neural network architecture for classification, which has no generalization over the space $\mathcal{R}^d$. Section 3.2 discusses a straightforward modification which allows the neural network to generalize over $\mathcal{R}^d$. However, training this neural network architecture is computationally expensive. We develop a new architecture for modeling distributions on $\mathcal{R}^d$ in Section 3.3. The proposed neural network architecture is computationally efficient, can take advantage of local spatial structure, and can be more interpretable than the aforementioned architectures.

3.1 Standard Neural Network Architecture for Classification

The basic neural network for classification is a highly nonlinear parameterized function that takes an input $x \in \mathcal{X}$ and produces a probability distribution on the finite discrete space $\mathcal{Y}$.

Given an input x, the output $f_{\theta,l}(x) \in \mathbb{R}^{d_l}$ of the l-th layer of a neural network is

$$f_{\theta,l}(x) = g^l(W_l^\top f_{\theta,l-1}(x) + b_l), \quad l = 1, \ldots, L, \tag{1}$$

where $W_l \in \mathbb{R}^{d_l} \times \mathbb{R}^{d_{l-1}}$, $b_l \in \mathbb{R}^{d_l}$, $f_{\theta,0}(x) = x$, and $d_L = |\mathcal{Y}|$. For $l = 1, \ldots, L-1$, the nonlinear transformation $g^l(z) = \big(\sigma(z_1), \ldots, \sigma(z_{d_l})\big)$ for $z \in \mathbb{R}^{d_l}$ and $z_1, \ldots, z_{d_l} \in \mathbb{R}$. The function σ is nonlinear;

typical choices are sigmoidal functions, tanh, rectified linear units (ReLU), and clipped rectified linear units. The function g^L for the final layer L is the softmax function:

$$g(z) = \Big(\frac{e^{z_1}}{\sum_{i=1}^{d_L} e^{z_i}}, \ldots, \frac{e^{z_{d_L}}}{\sum_{i=1}^{d_L} e^{z_i}}\Big), \quad z \in \mathbb{R}^{d_L}. \tag{2}$$

The final output of the neural network $f_{\theta,L}(x)$ is a probability distribution on $\mathcal{Y}$ conditional on the features x. The parameters collectively are $\theta = (W_1, \ldots, W_L, b_1, \ldots, b_L)$, where L is the number of layers in the neural network. The objective is to choose the parameters θ such that the log-likelihood $\mathcal{L}$ of the neural network's output $f_{\theta,L}$ is maximized for the data.

Let the data be $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $(x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$. Then, the normalized log-likelihood of the data for the neural network model is

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{y \in \mathcal{Y}} \mathbf{1}_{y=y_n} \log f^y_{\theta,L}(x_n), \tag{3}$$

where $f^y_{\theta,L}$ is the y-th element of the vector $f_{\theta,L}$.

The complexity of the model is determined by both the number of layers ("deepness") and the number of neurons $(d_1, \ldots, d_L)$ in each layer. Although the dividing line is somewhat arbitrary, neural networks are typically considered deep if there are three or more hidden layers ($L \geq 4$). Equation (1) describes the basic neural network achitecture, and there are several popular modifications to the architecture of the layers $1, \ldots, L-1$ (e.g., convolution neural networks). The discussion below is also applicable to these other architectures.

A potential drawback of the standard neural network for classification is that, although it generalizes over the input space $\mathcal{X}$, it does not allow for generalization over the output space $\mathcal{Y}$. As mentioned earlier, one approach to modeling a distribution on $\mathbb{R}^d$ is to discretize $\mathbb{R}^d$ into the grid $\mathcal{R}^d$ where $\mathcal{R} = \ldots, r_{-2}, r_{-1}, 0, r_1, r_2, \ldots$ and then model a distribution on the discrete space $\mathcal{R}^d$. Many problems may have some spatial structure where the event $r \in \mathcal{R}^d$ will be strongly related to the event $r' \in \mathcal{R}^d$ if r and r' are close in distance. A training sample at r should then allow one to learn about both r and r'. However, the standard neural network architecture for classification would regard r and r' as completely separate events, failing to take advantage of any available spatial structure since it has no generalization over space.

Generalizing over space is especially useful in the tails of the distribution where less data is available. The tails of the distribution, although associated with less frequent events, are important for risk analysis since they represent extreme events which can have a disproportionate impact. Generalization over space also helps to combat the curse of dimensionality. The number of grid points grows exponentially with the dimension d, meaning there is less data per grid point (and less data per state $y \in \mathcal{Y}$). This can be a source of overfitting.

There are other disadvantages. Since $\mathcal{Y}$ is a finite discrete space, $\mathcal{R}$ must be truncated to cover only a

finite region of space, which may not be desirable. Secondly, even if the bulk of the events occur in a small region of space, probabilities may still be needed at a large number of spatial points, incurring significant computational cost. For instance, even if 99% of events occur in $[0, 1]$, and the rest are uniformly spread across $[-1000, 1000]$, probabilities at all (discretized) spatial points must be calculated for each data sample during training and prediction. This incurs significant computational cost and thus slower training rates, especially in higher dimensions $d > 1$ where the number of grid points grows rapidly with d.

3.2 Straightforward modification to allow generalization

There is a straightforward modification to create a neural network which generalizes over space. This modification has been studied before; for instance, see Likas (2001). Let $f_\theta(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be the unnormalized log-probability of the event y conditional on the feature x, where $f_\theta(x, y)$ is a neural network with inputs (x, y). The probability of y conditional on the feature x is

$$\frac{e^{f_\theta(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{f_\theta(x,y')}}. \tag{4}$$

(4) can be computationally expensive. For each training sample, $f_\theta(x, y)$ and its gradient must be evaluated at every $y \in \mathcal{Y}$. If the number of training samples is N, this is comparable to training a standard neural network for binary classification on $N \times |\mathcal{Y}|$ training samples. For instance, if $\mathcal{Y}$ is a Cartesian grid covering $\mathbb{R}^3$ with $1,000$ grid points in each direction, $|\mathcal{Y}| = 1$ billion.

A second disadvantage is that (4) cannot model distributions on $\mathcal{R}^d$ but instead must truncate the space in order to form a finite grid.

3.3 A computationally efficient architecture for modeling spatial distributions

First consider a distribution on $\mathbb{R}_+ = (0, \infty)$, which is discretized into $\mathcal{R}_+ = r_1, r_2, \ldots$. Later this is extended to the more general case of $\mathcal{R}^d$. Let $f_\theta(x, y) : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ be a neural network. Provided certain technical conditions [4], the distribution of a random variable $Y \in \mathcal{R}_+$ conditional on the random variable $X \in \mathcal{X}$ is completely specified by the following model:

$$\mathbb{P}[Y = y \big| Y \geq y, X = x] = \frac{e^{f_\theta(x,y)}}{1 + e^{f_\theta(x,y)}}. \tag{5}$$

[4] It is required that $\sum_{y \in \mathcal{R}_+} \frac{e^{f_\theta(x,y)}}{1+e^{f_\theta(x,y)}} = 1$. A sufficient condition is that the hidden units for the neural network $f_\theta(x, y)$ are bounded (e.g., sigmoidal, tanh, or clipped ReLU). Proving that this technical condition holds for ReLU (where $f_\theta(x, y)$ is unbounded in y) requires further analysis.

(5) is analogous to the first arrival time of a nonhomogenous Poisson process in continuous time. The log-likelihood for a training sample (x, y) is:

$$\mathcal{L}(\{(x,y)\}) = \log\left(\frac{1}{1+e^{f_\theta(x,y)}}\right) + \sum_{y' \in \mathcal{R}_+ : y' < y} \log\left(\frac{e^{f_\theta(x,y')}}{1+e^{f_\theta(x,y')}}\right) \tag{6}$$

The architecture (5) has two advantages over (4). The first is that the neural network $f_\theta(x, y)$ and its gradient need to be evaluated at far fewer grid points. For each sample (x, y), (5) only needs to be evaluated up until y whileas (4) needs to be evaluated on the entire grid. Secondly, (5) can model the entire space $\mathcal{R}_+$; there is no need to form a truncated grid as in (4).

3.3.1 Extension to $\mathcal{R}^d$

(5) can be extended to model distributions on $\mathcal{R}^d$. Let $Y = (Y_1, \ldots, Y_d) \in \mathcal{R}^d$ and have the conditional distribution:

$$\begin{aligned}
\mathbb{P}[Y = (y_1, \ldots, y_d) \big| X = x] &= \mathbb{P}[Y_1 = y_1 \big| X = x] \prod_{i=2}^{d} \mathbb{P}[Y_i = y_i \big| Y_{0:i-1} = y_{0:i-1}, X = x], \\
\mathbb{P}[Y_1 = y_1 \big| X = x] &= g_\theta^1(x, y_1), \\
\mathbb{P}[Y_i = y_i \big| Y_{0:i-1} = y_{0:i-1}, X = x] &= g_\theta^i(x, y_{0:i-1}, y_i),
\end{aligned} \tag{7}$$

The conditional distributions $g^1, \ldots, g^d$ will be functions of neural networks, which will be specified shortly. Note that the framework (7) avoids the curse of dimensionality for large d since the computational expense of the log-likelihood grows linearly with d:

$$\mathcal{L}(\{(x,y)\}) = \log\left(g_\theta^1(x, y_1)\right) + \sum_{i=2}^{d} \log\left(g_\theta^i(x, y_{0:i-1}, y_i)\right). \tag{8}$$

The conditional distribution of Y_1 conditional on X is completely specified by:

$$\begin{cases}
\mathbb{P}[Y_1 = y_1 \big| Y_1 \geq y_1, X = x] = \frac{e^{f_\theta^{1,+}(x,y_1)}}{1+e^{f_\theta^{1,+}(x,y_1)}} & y_1 \geq 1 \\
\mathbb{P}[Y_1 = z | X = x] = h_\theta^{1,z}(x) & z \in \{y_1 > 0\}, \{y_1 = 0\}, \{y_1 < 0\} \\
\mathbb{P}[Y_1 = y_1 \big| Y_1 \leq y_1, X = x] = \frac{e^{f_\theta^{1,-}(x,y_1)}}{1+e^{f_\theta^{1,-}(x,y_1)}} & y_1 \leq -1
\end{cases}$$

$f_\theta^{1,-} : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ and $f_\theta^{1,+} : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ are neural networks. The neural network $h_\theta^1(x)$ is a standard neural network for classification (as described in Section 3) which produces a vector of three probabilities for the events $\{y_1 > 0\}, \{y_1 = 0\}, \{y_1 < 0\}$, and $h_\theta^{1,z}(x)$ is the z-th vector element. The standard neural network for classification h_θ^1 is required to "stitch" together $\mathcal{R}_+$ and $\mathcal{R}_- = \ldots, r_{-2}, r_{-1}$.

Similarly, the conditional distribution of Y_i conditional on $(Y_{0:i-1}, X)$ for $i \geq 2$ is completely specified by:

$$\begin{cases} \mathbb{P}[Y_i = y_i \big| Y_i \geq y_i, Y_{0:i-1} = y_{0:i-1}, X = x] = \frac{e^{f_\theta^{i,+}(x, y_{0:i-1}, y_i)}}{1+e^{f_\theta^{i,+}(x, y_{0:i-1}, y_i)}} & y_i \geq 1 \\ \mathbb{P}[Y_i = z | Y_{0:i-1} = y_{0:i-1}, X = x] = h_\theta^{i,z}(x, y_{0:i-1}) & z \in \{y_i > 0\}, \{y_i = 0\}, \{y_i < 0\} \\ \mathbb{P}[Y_i = y_i \big| Y_i \leq y_i, Y_{0:i-1} = y_{0:i-1}, X = x] = \frac{e^{f_\theta^{i,-}(x, y_{0:i-1}, y_i)}}{1+e^{f_\theta^{i,-}(x, y_{0:i-1}, y_i)}} & y_i \leq -1 \end{cases}$$

3.3.2 Advantages of the spatial neural network

There are several potential advantages to this proposed architecture for the spatial neural network. The model and its gradient can be evaluated at far fewer grid points in the computationally efficient architecture. Secondly, the proposed architecture can model the entire space $\mathcal{R}^d$; there is no need to form a truncated grid as in standard architectures. Thirdly, the compoutational cost only grows linearly with the dimension d. One disadvantage is that the architecture is composed of several neural networks instead of a single neural network. The number of neural networks grows linearly with d.

The proposed architecture can also take advantage of "local spatial structure", if it exists in the application setting. The spatial neural network (5) is local in nature; it models the local dynamics within a small region in space. The spatial neural network (5) can leverage a priori knowledge that conditional on Y being in some region of space, the local dynamics of Y in that region only depend a particular subset of the values in the vector X. This can improve performance and increase interpretability since it reduces the dimension of the input space. For example, if X is a vector containing information at locations in $\mathbb{R}^d$ and $Y \in \mathbb{R}^d$, Y's local dynamics in some small region of $\mathbb{R}^d$ may only depend upon information at locations close to that small region. This is arguably the case for limit order books; see Section 2.3. Such local behavior can be naturally modeled by $f_\theta(x, y)$ in (5). Let $f_\theta(x, y) = g_\theta(m(x, y), y)$ where g_θ is a neural network and $m(x, y)$ is a map taking the vector of limit order volumes at all levels and outputting a smaller vector of limit order volumes at only levels close to y. Although the local distribution of Y conditional on Y being in a particular region depends only on a subset of X, the global distribution of Y can depend upon the entire variable X.

4 Model Training

The neural networks are trained using approaches from deep learning, which we describe in Section 4.1. The computational implementation using GPU clusters is outlined in Section 4.2. The division of the dataset into training, validation, and test sets is specified in Section 4.3.

4.1 Deep Learning

We use 4 layers for the neural networks. Neural networks with more layers have lower bias, but are also more difficult to train and more prone to overfitting. Recent research has developed many new methods for training deep neural networks, and we employ several of these techniques. We use dropout to prevent overfitting (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov 2014). Batch normalization is used between each hidden layer to prevent internal covariate shift (Ioffe & Szegedy 2015). Stochastic gradient descent with momentum is used for training (Sutskever, Martens, Dahl & Hinton 2013). The learning rate is adaptive, decreasing by a constant factor whenever the error increases over a training epoch. Early stopping via a validation set is imposed to reduce overfitting (Bengio 2012). Although ReLU units have often produced the best performance for deep neural networks, we find that it is preferable in the limit order book setting to use hidden units which are bounded (e.g., clipped ReLU, sigmoidal, or tanh). The limit order volumes are unbounded, and a small fraction have very large values. These outlier values can cause undesirably large gradient steps.

4.2 Computational Approach

Due to the size of the dataset and the large number of parameters in the neural networks, model training is computationally expensive. To address this, we use a GPU cluster on Amazon Web Services (AWS) to train models. Distributed storage is used to facilitate accessing and processing of data. Pre-processing of data is performed using parallelization over 30 CPUs on high-performance instances.

4.3 Division of Data into Training, Validation, and Test Sets

The data is divided into three sets. The test set is all data from June 1, 2015 to August 31, 2015. The training data is composed of 95% of the data from January 1, 2014 to May 31, 2015 (drawn at random). The validation set is the remaining 5% of the data from January 1, 2014 to May 31, 2015. Models are trained and tested separately on each stock; i.e., a new randomly initialized model is trained for each stock.

5 Out-of-sample Results

Out-of-sample results for the distribution of the best ask price are reported in Section 5.1. Results for the joint distribution of the best ask and best bid prices (a distribution on a two-dimensional space) are described in Section 5.2. We provide some analysis of the numerical results in Section 5.3. The spatial neural network outperforms the standard neural network for classification.

Neural network results are also compared against baseline models. The first baseline model is the naive empirical model, which is simply the naive empirical distribution from the training set. If models do not have lower errors than the naive empirical model, this would indicate that the limit order book contains no information on future movements of the best ask and best bid prices. The second baseline model is a logistic

regression. If a neural network does not have a lower error than a logistic regression, this would indicate that the nonlinearity of the limit order book dynamics is not significant (since logistic regression is the softmax of a linear function). Both of the neural networks outperform these baseline models.

For the purposes of the model comparison in Sections 5.1 and 5.2, $\mathcal{R}$ is truncated to $-25, -24, \ldots, 24, 25$. The "error" reported is the cross-entropy error, which is equivalent to the negative log-likelihood.

5.1 Distribution of Best Ask

Tables 2 and 3 compare the out-of-sample performance of the different models. The neural networks consistently have lower error than the naive empirical model and the logistic regression. The spatial neural network consistently has lower error than the standard neural network.

Model 1/Model 2	Naive empirical model	Logistic Reg.	Neural Net.	Spatial Neural Net.
Naive empirical model	NA	3/140	1/140	0/140
Logistic Reg.	137/140	NA	0/140	0/140
Neural Net.	139/140	140/140	NA	12/140
Spatial Neural Net.	140/140	140/140	128/140	NA

Table 2: Number of stocks out of 140 total stocks where Model 1 has a lower out-of-sample error than Model 2. "Neural Net." is the standard neural network architecture described in Section 3.1. "Spatial Neural Net." is the computationally efficient neural network architecture for spatial distributions developed in Section 3.3.

Model 1/Model 2	Naive empirical model	Logistic Reg.	Neural Net.	Spatial Neural Net.
Naive empirical model	NA	-4.25	-7.21	-8.80
Logistic Reg.	3.97	NA	-2.84	-4.39
Neural Net.	6.61	2.75	NA	-1.51
Spatial Neural Net.	8.00	4.18	1.47	NA

Table 3: Average percent decrease in error for Model 1 versus Model 2: $\frac{\text{Model 2 error} - \text{Model 1 error}}{\text{Model 2 error}} \times 100\%$. "Neural Net." is the standard neural network architecture described in Section 3.1. "Spatial Neural Net." is the computationally efficient neural network architecture for spatial distributions developed in Section 3.3.

5.2 Joint Distribution of Best Ask and Best Bid

Tables 4 and 5 compare the out-of-sample performance of the different models. The spatial neural network consistently has lower error than the standard neural network.

Model 1/Model 2	Naive empirical model	Neural Net.	Spatial Neural Net.
Naive empirical model	NA	0/140	1/140
Neural Net.	140/140	NA	10/140
Spatial Neural Net.	139/140	130/140	NA

Table 4: Number of stocks out of 140 total stocks where Model 1 has a lower out-of-sample error than Model 2. "Neural Net." is the standard neural network architecture described in Section 3.1. "Spatial Neural Net." is the computationally efficient neural network architecture for spatial distributions developed in Section 3.3.

Model 1/Model 2	Naive empirical model	Neural Net.	Spatial Neural Net.
Naive empirical model	NA	-9.26	-11.92
Neural Net.	8.11	NA	-2.53
Spatial Neural Net.	10.42	2.18	NA

Table 5: Average percent decrease in error for Model 1 versus Model 2: $\frac{\text{Model 2 error} - \text{Model 1 error}}{\text{Model 2 error}} \times 100\%$. "Neural Net." is the standard neural network architecture described in Section 3.1. "Spatial Neural Net." is the computationally efficient neural network architecture for spatial distributions developed in Section 3.3.

5.3 Some analysis of numerical results

Figure 4 shows that the spatial neural network performs better in comparison with the standard neural network for more volatile stocks. The Spearman correlation between the decrease in error and the standard deviation is 68%. This observation matches the initial intuition motivating the spatial neural network: distributions spread over wider ranges of $\mathbb{R}^d$ will be more challenging for the traditional neural network designed for classification.

Besides having lower error, the spatial neural network converges much more rapidly than the standard neural network for classification. Tables 6 and 7 report the error versus computational time for the two neural network architectures when trained on the stock Amazon. The spatial neural network converges in several minutes while the standard neural network takes several hours to converge.

Time (**hours**)	.5	1	1.5	2	2.5	3	3.5	4
Error	1.961	1.893	1.885	1.882	1.880	1.879	1.880	1.880

Table 6: Test error versus computational time for Amazon using standard neural network for classification. $\mathcal{R} = -50, -49, \ldots, 49, 50$.

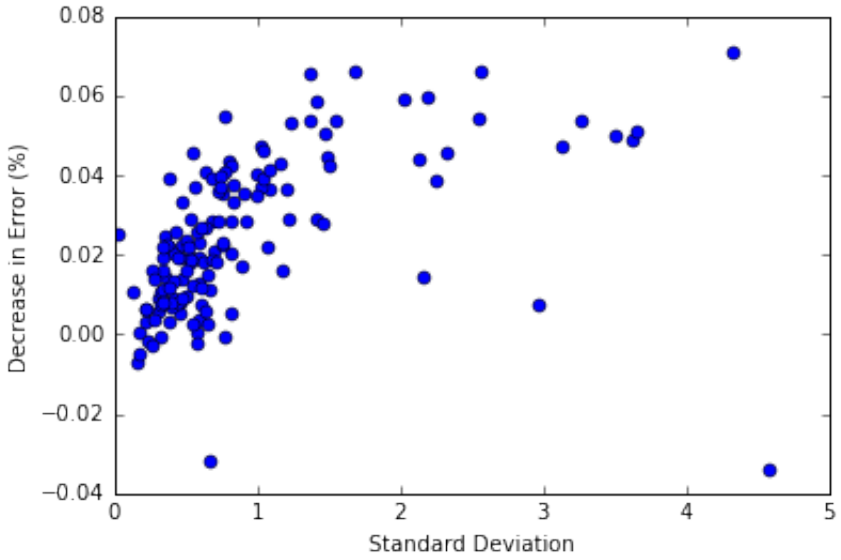

Figure 4: Percent decrease in error for spatial neural network versus standard neural network plotted against the standard deviation of the best ask price. The errors compared here are from the case for the joint prediction of the best ask and best bid prices.

Time (**minutes**)	1	2	5	10	20
Error	1.812	1.809	1.802	1.796	1.799

Table 7: Test error versus computational time for Amazon using spatial neural network. $\mathcal{R} = -50, -49, \ldots, 49, 50$.

References

Alfonsi, A., A. Fruth & A. Schied (2010), 'Optimal execution strategies in limit order books with general shape functions', *Quantitative Finance* **10**(2), 143–157.

Alfonsi, A. & A. Schied (2010), 'Optimal trade execution and absence of price manipulations in limit order book models', *SIAM Journal on Financial Mathematics* **1**(1), 490–522.

Avellaneda, M., J. Reed & S. Stoikov (2011), 'Forecasting prices from level-i quotes in the presence of hidden liquidity', *Algorithmic Finance* **1**(1).

Avellaneda, M. & S. Stoikov (2008), 'High-frequency trading in a limit order book', *Quantitative Finance* **8**(3), 217–224.

Bayraktar, E. & M. Ludkovski (2014), 'Liquidation in limit order books with controlled intensity', *Mathematical Finance* **24**(4), 627–650.

Bengio, Y. (2012), 'Practical recommendations for gradient-based training of deep architectures', *Neural Networks: Tricks of the Trade* pp. 437–478.

Biais, B., P. Hillion & C. Spatt (1995), 'An empirical analysis of the limit order book and the order flow in the paris bourse', *Journal of Finance* pp. 1655–1689.

Blanchet, J. & X. Chen (2013), Continuous-time modeling of bid-ask spread and price dynamics in limit order books.

Bouchaud, J., M. Mezard & M. Potters (2002), 'Statistical properties of stock order books: empirical results and models', *Quantitative Finance* **2**(4), 251–256.

Cont, R. (2011), 'Statistical modeling of high-frequency financial data', *Signal Processing Magazine, IEEE* **28**(5), 16–25.

Cont, R. & A. De Larrard (2012), Order book dynamics in liquid markets: limit theorems and diffusion approximations.

Cont, R. & A. De Larrard (2013), 'Price dynamics in a markovian limit order market', *SIAM Journal on Financial Mathematics* **4**(1), 1–25.

Cont, R., A. Kukanov & S. Stoikov (2014), 'The price impact of order book events', *Journal of financial econometrics* **12**(1), 47–88.

Cont, R., S. Stoikov & R. Talreja (2010), 'A stochastic model for order book dynamics', *Operations Research* **58**(3), 549–563.

Donier, J., J. Bonart, I. Mastromatteo & J. Bouchaud (2014), A fully consistent, minimal model for non-linear market impact.

Eisler, Z., J. Bouchaud & J. Kockelkoren (2012), 'The price impact of order book events: market orders, limit orders and cancellations', *Quantitative Finance* **12**(9), 1395–1419.

Fletcher, T. & J. Shawe-Taylor (2013), 'Multiple kernel learning with fisher kernels for high-frequency currency prediction', *Computational Economics* **42**(2), 217–240.

Gao, X., J. Dai, T. Dieker & S. Deng (2014), Hydrodynamic limit of order book dynamics.

Gould, M. & J. Bonart (2015), Queue imbalance as a one-tick-ahead price predictor in a limit order book.

Gould, M., M. Porter, S. Williams, M. McDonald, D. Fenn & S. Howison (2013), 'Limit order books', *Quantitative Finance* **13**(11), 1709–1742.

Ioffe, S. & C. Szegedy (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Kearns, M. & Y. Nevmyvaka (2006), Reinforcement learning for optimized trade exectuion, 23rd international conference on Machine Learning, ACM.

Kempf, A. & O. Korn (1999), 'Market depth and order size', *Journal of Financial Markets* **2**(1), 29–48.

Kercheval, A. & Y. Zhang (2013), Modeling high-frequency limit order book dynamics with support vector machines.

Likas, A. (2001), 'Probability density estimation using artificial neural networks', *Computer Physics Communications* **135**, 167–175.

Maglaras, C., C. Moallemi & H. Zheng (2015), Optimal execution in a limit order book and an associated microstructure market impact model.

Moallemi, C., B. Park & B. Van Roy (2012), 'Strategic execution in the presence of an uninformed arbitrageur', *Journal of Financial Markets* **15**(4), 361–391.

Potters, M. & J. Bouchaud (2003), 'More statistical properties of order books and price impact', *Physica A: Statistical Mechanics and its Applications* **324**(1), 133–140.

Predoiu, S., G. Shaikhet & S. Shreve (2011), 'Optimal execution in a general one-sided limit-order book', *SIAM Journal on Financial Mathematics* **2**(1), 183–212.

Schied, A. & T. Schoneborn (2009), 'Risk aversion and the dynamics of optimal liquidation strategies in illiquid markets', *Finance and Stochastics* **13**(2), 181–204.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research* **15**(1), 1929–1958.

Sutskever, I., J. Martens, G. Dahl & G. Hinton (2013), 'On the importance of initialization and momentum in deep learning', *Proceedings of the 30th international conference on machine learning (ICML-13)* pp. 1139–1147.