A Deep Learning Approach to Estimating Fill Probabilities in a Limit Order Book

Costis Maglaras
Graduate School of Business
Columbia University
c.maglaras@gsb.columbia.edu

Ciamac C. Moallemi
Graduate School of Business
Columbia University
ciamac@gsb.columbia.edu

Muye Wang
Graduate School of Business
Columbia University
mw3144@gsb.columbia.edu

Initial Version: June 2019
Current Revision: April 26, 2022

Abstract

Deciding between the use of market orders and limit orders is an important question in practical optimal trading problems. A key ingredient in making this decision is understanding the uncertainty of the execution of a limit order, that is, the fill probability or the probability that an order will be executed within a certain time horizon. Equivalently, one can estimate the distribution of the time-to-fill. We propose a data-driven approach based on a recurrent neural network to estimate the distribution of time-to-fill for a limit order conditional on the current market conditions. Using a historical data set, we demonstrate the superiority of this approach to several benchmark techniques. This approach also leads to significant cost reductions while implementing a trading strategy in a prototypical trading problem.

1. Introduction

Most modern financial exchanges use electronic limit order books (LOBs) as a centralized system to trade and track orders. In such exchanges, resting limit orders await matching to contra-side market orders.[1]

Because exchanges typically offer multiple order types, when traders submit an order, they often face the choices among many order types. The most common choice is between a market order and a limit order. Market orders are orders that execute immediately at the best current price. Such orders are employed by traders whose priority is immediate executions. Limit orders are orders that execute only at a specified price or better. As a result, limit orders typically don't

[1] A market order executes immediately at the current best price. A marketable limit order specifies a limit price as a constraint, but that constraint is bot binding and the order executes immediately. For the purposes of our study, we use these two terms interchangeably.

execute right away, and in some cases, limit orders don't execute at all. The delay between a limit order's submission and its execution is called the "time-to-fill" or "fill time." In order to choose between market orders and limit orders intelligently, it's important for a trader to understand the uncertainty of limit order executions, more specifically, the fill probability with a given time horizon, or equivalently, the distribution of the fill times.

Deep learning is a branch of machine learning that uses neural networks to capture intricate patterns in data. A typical neural network consists of layers of artificial neurons that use activation functions to model nonlinear relationships. A deep neural network with multiple layers effectively represents the function composition of all the layers of these activation functions. As a result, a deep neural network can represent very complicated functions and capture patterns that traditional linear models can not.

In this paper, we apply deep learning technology to study the uncertainty of the limit order executions, that is, the fill probability or equivalently the distribution of the fill times. Accurate fill probability predictions help trader better decide between market orders and limit orders, which we demonstrate via a prototypical trading problem.

The main contributions of this paper are as follows.

We propose a data-driven approach based on a specific recurrent neural network (RNN) architecture to predict limit order executions. Most studies on limit order executions use a model-based approach, which inevitably suffers from various model limitations, such as model misspecification. We propose a data-driven approach that takes advantage of the abundance of exchange market data. In order to model the temporal dynamics of limit order executions, we construct a RNN as opposed to a more traditional feed-forward neural network. In this study, we directly estimate the distribution of the fill times by designing a hazard rate approach. As far as we know, this is the first study that directly predicts limit order executions via the distribution of fill times.

We demonstrate better prediction accuracy against benchmark models. The performance of the RNN are measured using two metrics — fill probability and expectation of the fill times conditioned on execution. We use traditional estimation methods such as logistic regression to establish benchmarks. The RNN method outperforms the benchmarks on both of the metrics over various time horizons.

We demonstrate better performance in a prototypical execution problem. Better limit order execution predictions have important implications in trading strategy implementation. We specify a benchmark trading problem that considers the tradeoff between market orders and limit orders in executing a single share, with the goal of minimizing implementation shortfall. As RNN predicts fill probabilities more accurately, it also improves the trading strategy by reducing implementation shortfall.

1.1. Literature Review

The study of limit order books dates back to the late 1980s. The following is not a comprehensive review, but rather a highlight of a few notable studies that are most relevant to our work. Angel [1994] derives an analytical expression for limit order fill probability, conditional upon an investor's information set. However these results are derived under some rather strong assumptions. Hollifield et al. [2004] build a structural model of a limit order book market and characterize the tradeoff between market orders and limit orders. They compute a semi-parametric estimator of the model primitives using data from the Stockholm Stock Exchange. Another study that compares the use of market orders and limit orders is that of Petersen and Fialkowski [1994]. They conduct an empirical study using data from the NYSE and report a significant difference between the posted spread and the effective spread paid by investors. Lo et al. [2002] develop an econometric model to estimate time-to-first-fill and time-to-completion. They find that execution times are very sensitive to the limit price, but not as sensitive to the order size. They also find that many hypothetical limit order execution models are very poor proxies for actual limit order executions. Cho and Nelling [2000] conduct an empirical study and report that the longer a limit order is outstanding, the less likely it is to be executed.

With the rise of big data and machine learning, researchers have started to apply machine learning to the study of finance. Heaton et al. [2016] outlines the general frameworks of deep learning and many areas in finance that deep learning ideas can be useful. Some more specific deep learning applications include the studies of Xiong et al. [2015], Carr et al. [2019], Ban et al. [2018].

Because deep learning typically requires large data sets, its applications in high frequency domain have also shown to be promising. Sirignano and Cont [2019] use a recurrent neural network to predict next immediate price changes and further argue that there are certain universality in the price formation process across stocks. Zhang et al. [2019] train a deep learning model to predict price movements in the near future. Dixon et al. [2017] use a neural network to predict financial market movement directions and demonstrate its application in a simple trading strategy. Other machine learning applications in this area include the work of Tran et al. [2017,0], Tsantekidis et al. [2017], Passalis et al. [2018], Ntakaris et al. [2018].

1.2. Organization of Paper

The remainder of the paper is organized as follows. Section 2 outlines the limit order book dynamics and demonstrates the tradeoff between market orders and limit orders through a trading problem. The optimal trading strategy of the problem motivates the estimation of limit order fill probabilities. Section 3 describes recurrent neural networks and the hazard rate method for distribution estimation. Section 4 describes the NASDAQ ITCH data source, the simulation procedure of generating synthetic limit orders, and the maximal likelihood estimation of the RNN. Section 5 lists descriptive statistics of these synthetic limit orders and demonstrates a few predictive patterns. Section 6 presents the prediction results. The trading problem from Section 2 is revisited and the economic value of better fill probability predictions is illustrated. Section 7 concludes with a brief

summary and some remarks regarding the limitations of this work.

2. Limit Order Book and Motivation

In this section, we will introduce the mechanics of limit order books and discuss a prototypical trading problem that considers the tradeoff between limit orders and market orders. The optimal trading strategy requires fill probability as an input, which motivates the fill probability estimation problem.

2.1. Limit Order Book Mechanics

Limit order books are responsible for keeping track of all resting limit orders at various price levels. Because investors' preferences and positions change over time, the limit order books also need to be dynamic and change over time. During trading hours, market orders and limit orders are constantly being submitted and traded. These events alter the resting limit orders, and consequently, the shape of the limit order books. Other market events that alter the shape of the limit order books include partial or complete cancellations of resting limit orders.

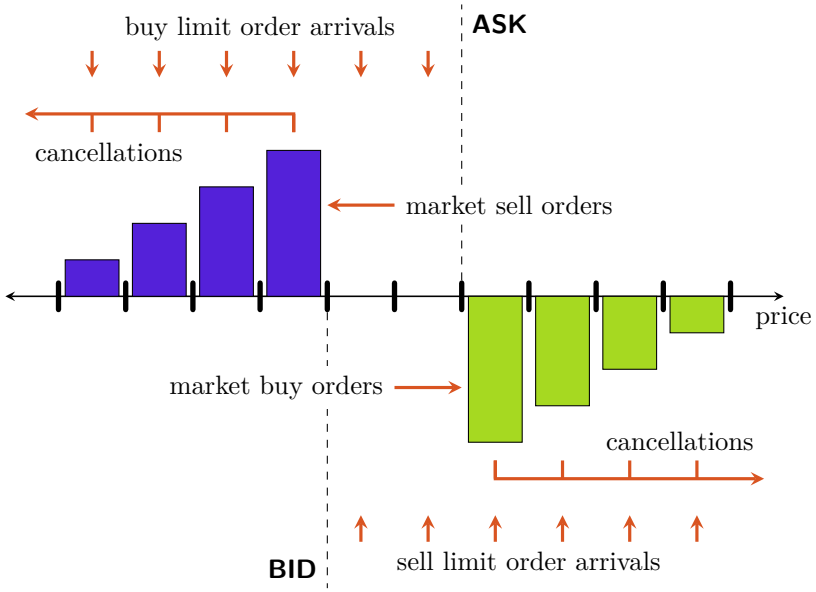

Figure 1: Limit orders are submitted at different price levels. The ask prices are higher than the bid prices. The difference between the lowest ask price and the highest bid price is the "bid-ask spread." Mid-price is the average of the best ask price and the best bid price.

Limit order books are paired with a matching engine that matches incoming market orders with resting limit orders to fulfill trades. The most common rule that the matching engine operates under is "price-time priority." When a new market order is submitted to buy, sell limit orders at the lowest ask price will be executed; when a new market order is submitted to sell, buy limit orders

at the highest bid price will be executed. For limit orders at the same price, the matching engine follows time priority — whichever order was submitted first gets executed first.

The configuration of limit order books and the matching rule prompt researchers to model limit order books as queuing systems (e.g., Cont et al. [2010], Moallemi and Yuan [2016], Toke [2013]). Market orders correspond to service completion and limit orders correspond to customer arrival. The difficulty of these approaches lies in the complexity of the dynamics of these market events. Empirical evidence suggests that the rates of these market events change based on market conditions. For example, Biais et al. [1995] find evidence that investors are more likely to submit limit orders (rather than hitting the quotes) when the bid-ask spread is large or the order book is thin. Cho and Nelling [2000] report that the longer a limit order is outstanding, the less likely it is to be executed.

2.2. Implementation Shortfall: A Tradeoff Between Market Orders and Limit Orders

The choice between market orders and limit orders can be viewed as a tradeoff between an immediate execution and a price premium. A buy market order executes at the best ask price whereas a buy limit order executes at a lower price. Therefore a limit order gains at least a bid-ask spread over a market order per share. The analogous situation holds for sell orders. However, even though limit orders offer a price premium, the execution isn't guaranteed. Therefore, the price premium is only realized with a certain probability, namely the fill probability.

To better demonstrate this tradeoff, consider the following stylized trading problem. Suppose an agent seeks to buy a share of stock over a fixed time horizon $[0, h]$. (The selling problem is analogous.) The agent seeks to minimize the implementation shortfall

$$\mathrm{IS} = \mathbb{E}[p_E - p_M(0)],$$

where p_E is the execution price which could be a random variable, and $p_M(0)$ is the mid-price at the arrival time 0. This task can be accomplished by using either a market order or a limit order. These two choices would lead to different execution outcomes as follows.

1. **Market Order:** Submit a market order at the arrival time 0 and pay the current best ask price. This leads to an implementation shortfall of

$$\mathrm{IS}_{\mathrm{mkt}} = p_A(0) - p_M(0).$$

2. **Limit Order:** Submit a limit order at the best bid price at the arrival time 0. If it is not filled by time h, place a "clean-up trade" with a market order at time h. The clean-up cost can be expressed as

$$\mathrm{C}_{\text{clean-up}} = p_A(h) - p_M(0).$$

   Let T be a random variable that denotes the fill time for this limit order. Then the expected

implementation shortfall is

$$\mathbb{E}[\mathrm{IS}_{\text{limit}}] = p(T \le h)[p_B(0) - p_M(0)] + p(T > h)\mathbb{E}[\mathrm{C}_{\text{clean-up}}|T > h].$$

For simplicity, assume that the bid-ask spread stays constant and is equal to S in this time horizon, namely

$$p_A(0) - p_M(0) = S/2, \quad p_B(0) - p_M(0) = -S/2.$$

The constant-spread assumption simplifies the above two implementation shortfalls to

$$\mathrm{IS}_{\text{mkt}} = S/2; \qquad \mathbb{E}[\mathrm{IS}_{\text{limit}}] = p(T \le h) \cdot -S/2 + p(T > h) \cdot \mathbb{E}[\mathrm{C}_{\text{clean-up}}|T > h].$$

For a risk-neutral agent, the optimal trading strategy is to pick the order with the smaller cost. In this case, the agent would choose a limit order if and only if $\mathbb{E}[\mathrm{IS}_{\text{limit}}] < \mathrm{IS}_{\text{mkt}}$. This is equivalent to

$$p(T \le h) > \frac{\mathbb{E}[\mathrm{C}_{\text{clean-up}}|T > h] - S/2}{\mathbb{E}[\mathrm{C}_{\text{clean-up}}|T > h] + S/2} \triangleq \Theta.$$

Otherwise the agent would choose a market order. If the market is efficient (the expected future price equals the current price) and the price changes and fills are independent, then

$$\mathbb{E}[\mathrm{C}_{\text{clean-up}}|T > h] = \mathbb{E}[\mathrm{C}_{\text{clean-up}}] = \mathbb{E}[p_A(h) - p_M(0)] = p_A(0) - p_M(0) = S/2.$$

This would lead to $\Theta = 0$ and consequently a market order would always be more preferable. However, in reality, due to the adverse selection of limit order executions, if a buy limit order doesn't get filled, the price has typically moved higher by time h. Therefore, in this case,

$$\mathbb{E}[\mathrm{C}_{\text{clean-up}}|T > h] \ge S/2.$$

This makes the threshold Θ positive, and therefore a meaningful threshold on fill probability.

In practice, because the fill probability $p(T \le h)$ is unknown, it needs to be estimated in order to implement the above trading strategy. This motivates the rest of the paper, where we develop a RNN-based method to estimate the fill probability of particular limit orders.

A better fill probability prediction is valuable in many trading applications where the trade-off between market orders and limit orders is an important consideration. This particular trading strategy proposed above captures this trade-off in the most atomic level and it serves the purpose of demonstrating the economic values of a more accurate fill probability prediction. More complex trading problems can be constructed, however it would likely be a composition or a variant of the atomic problem.

3. Recurrent Neural Networks

For a new limit order an agent is contemplating placing, consider the problem of predicting fill probability over various time horizons. This problem can be formulated as a supervised learning problem. We collect limit orders submitted under various market conditions and record their fill times. Input features are selected to represent the market conditions and a neural network is trained to predict fill probabilities. The rest of this section outlines the input features, the recurrent neural network (RNN) architecture, and the hazard rate method used to predict limit order fill probabilities.

3.1. LSTM Architecture

In a limit order book market, past events can have predictive power of the immediate future. Therefore, we not only collect input features when the limit order is submitted, we also collect features prior to the order's submission. These sets of input features collectively represent the market condition and help predict limit order's fill probability.

To process time series of input features, a special neural network structure is required. Traditional feed-forward neural networks are static models and they are not equipped to model temporal dynamics in data. RNNs are a class of neural networks specially designed to model the temporal dynamics between past observations and future observations. In this study, we use a long short-term memory (LSTM) network, a specific RNN structure that helps establish long term temporal dependency.

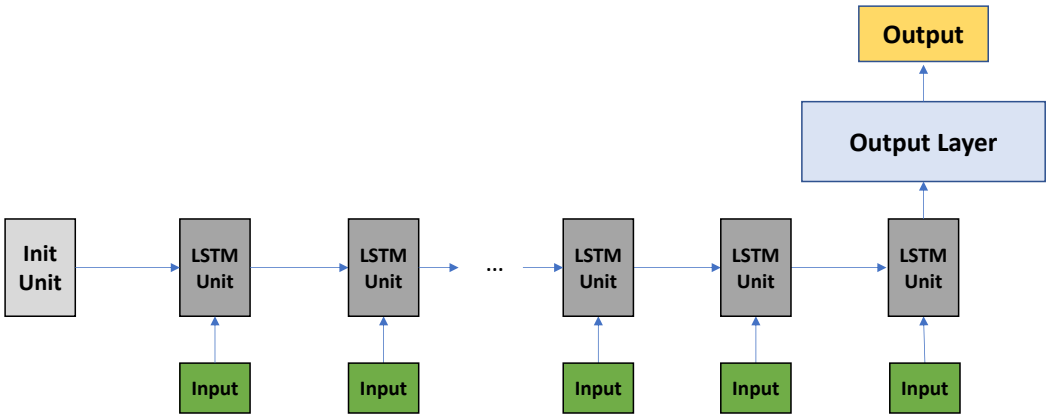

Figure 2: LSTM Recurrent Neural Network

Figure 2 depicts the RNN architecture used in this study. Each input unit represents a set of input features sampled at chronological order. LSTM units are used as hidden states to model temporal dynamics. These LSTM units connect input features from different time instances and capture any effect they have on the prediction output. The output of the neural network is a set of

hazard rates, which collectively represent the distribution of the fill times. For more implementation detail, please see Appendix B .The details of inputs and outputs are described below.

There are at least two reasons that inspired us to use RNN as the machine learning model in this study. One is that neural networks are very flexible. This enabled us to design the outputs of the neural network to be hazard-rate estimates and train the neural network using maximal likelihood function. This way, the neural network is able to estimate the distribution of the fill time instead of merely providing point estimates. Softplus activation functions are applied to the output layers to enforce the positivities of these outputs. Linear models or tree-based models don't have this kind of flexibility and can't be used to estimate distributions.

The second reason that we chose RNN is that it is designed to capture temporal dynamics in a sequential order in the features, and it is equipped to process time-series of inputs. We could potentially feed lagged features to models such as a feed-forward neural network or tree-based models, but they aren't equipped to process time-series of inputs and the sequential orders of the inputs would be lost.

The usage of RNN in finance aren't without its precedence. In fact, many researchers have used various type of RNN in analyzing financial market data, such as Sirignano [2019], Moghar and Hamiche [2020], Dixon and London [2021].

3.2. Input Features

The input of the RNN is a time series of sets of features sampled at different points in time. This time series can be represented as $X = \{x_{-n}, ..., x_{-2}, x_{-1}, x_0\}$ where x_0 is the set of features collected at the order's submissions, and $x_{-n}, ..., x_{-1}$ are sets of features collected prior to order's submission. This time series of features collectively represent the current state of the limit order book as well as the past evolution.

These input features contain raw limit order book information such as queue depth at each price levels, as well as hand-crafted features such as intensity measures. For a complete set of input features, please see Appendix A.

These input features treat the market symmetrically. In other words, buy limit orders and sell limit orders are not distinguished by the features. "Near side" is the side of the market that the limit order is submitted to — if the limit order is to buy, the bid side of the market is the near side. The "far side" is the opposite side of the market that the limit order is submitted to — if the limit order is to buy, the ask side of the market is the far side.

3.3. Hazard Rates Outputs

The limit order fill times can be conceptualized as a type of "survival time" from survival analysis, a branch of statistics for analyzing the duration of time until a certain event occurs, such as death in biological organisms or failure in mechanical systems. The event in our study that defines the analysis is the execution of limit orders. In these type of analyses, hazard function is commonly used to model the distribution of survival time. The simplest survival time distribution, the exponential

distribution, is modeled by a constant hazard rate. In this study, in order to model the distribution of limit order fill times, we estimate constant hazard rates on multiple time intervals. This method can model distributions more general than the exponential distribution, while also maintaining tractability. Our approach is oriented to accurately representing the entire conditional distribution of the interarrival time. This is in contrast to the simpler parameterization of Xiao et al. [2017], that seeks to predict only the conditional mean.

Specifically, for a particular limit order, let f_T and F_T be the density and the cumulative distribution of its fill time T. A supervised learning problem can be formulated to capture the mapping from the input features to the fill time density function.

In order to parameterize f_T concretely, we partition the time axis into a set of pre-determined intervals and estimate a constant hazard rate for each interval. More specifically, the time axis $\mathbb{R}_+$ can be partitioned into $M+1$ intervals: $[0, \tau_1), [\tau_1, \tau_2), ..., [\tau_M, +\infty)$. For each interval, a constant hazard rate can be estimated. Let $\lambda \in \mathbb{R}^{M+1}$ be the set of hazard rates on these intervals, as illustrated in Figure 3.

Figure 3: Hazard Rate Estimation on Pre-determined Intervals

This set of pre-determined intervals and the corresponding hazard rates uniquely determine a distribution on $\mathbb{R}_+$. This distribution also provides explicit expressions. For $t \in \mathbb{R}_+$, let $i^*(t) = \max\{i : \tau_i < t\}$. If $t \geq \tau_M$, then let $i^*(t) = M$. Then the cumulative hazard rate up to t can be expressed as

$$\Lambda(t) = \sum_{i=1}^{i^*(t)} \lambda_i \cdot (\tau_i - \tau_{i-1}) + \lambda_{i^*(t)+1} \cdot (t - \tau_{i^*(t)}).$$

This leads to the cumulative and the density distribution functions as follows

$$F_T(t) = 1 - e^{-\Lambda(t)}; \qquad f_T(t) = \lambda_{i^*(t)+1} \cdot e^{-\Lambda(t)}.$$

Now that the distribution of the fill times are expressed collectively as a set of hazard rates, the problem becomes estimating the hazard rates. Therefore, we design the output of the RNN to be a set of estimated hazard rates $\hat{\lambda} = (\hat{\lambda}_1, \hat{\lambda}_2, ..., \hat{\lambda}_{M+1})$, corresponding to each pre-determined time interval. This model can be estimated through maximum likelihood estimation, which we will discuss in the next section.

4. Model Estimation

This section outlines the estimation procedure of the RNN model. This includes discussions about the data source, the simulation approach to generate synthetic limit orders and the maximum

likelihood estimation.

4.1. NASDAQ ITCH

Our approach is tested using NASDAQ ITCH data (see NASDAQ [2010]), which provides Level III market data. Level III market data contains message feeds for events that have transpired in the market. Common market events that change the limit order book include "Add Order", "Order Executed", "Order Cancel" and "Order replaced." These market events occur throughout the trading hours and constantly change the limit order books. A sample of an "Add Order" event message is shown in Table 1.

time	ticker	side	shares	price	Type
9:30:00.4704337	BAC	B	2000	12.02	"A"

Table 1: This event reads: A bid limit order of 2000 shares of BAC stock is added to the LOB at price level $12.02 at 9:30:00.4704337.

From these event messages, a limit order book can be constructed to compute the total number of resting shares (depth) at each price level. The limit order book is dynamic and updates each time a new market event occurs.

time	b.prc5	b.prc4	b.prc3	b.prc2	b.prc1	a.prc1	a.prc2	a.prc3	a.prc4	a.prc5
9:30:00.4704337	12.01	12.02	12.03	12.04	12.05	12.06	12.07	12.08	12.09	12.10
9:30:00.8582938	12.01	12.02	12.03	12.04	12.05	12.06	12.07	12.08	12.09	12.10

Table 2: The above table is a snapshot of the limit order book displaying the prices of the top 5 price levels on both sides of the market at two timestamps. The event from Table 1 doesn't change the prices at each level.

time	b.vol5	b.vol4	b.vol3	b.vol2	b.vol1	a.vol1	a.vol2	a.vol3	a.vol4	a.vol5
9:30:00.4704337	10000	43700	13100	12100	7500	5200	15300	15900	17000	22200
9:30:00.8582938	10000	41700	13100	12100	7500	5200	15300	15900	17000	22200

Table 3: The above table is a snapshot of the limit order book displaying the number of shares on the top 5 price levels on both sides of the market at two timestamps. The event from Table 1 reduces 2000 shares at price $12.02.

4.2. Synthetic Limit Orders

In order to estimate fill time of a new limit order, we need a data set of new limit orders, input features, and associated fill times that are submitted under various market conditions. One might seek to use real limit orders from the market, however there are some immediate issues:

- Censoring: Most limit orders are canceled before they are executed. This is particularly true for the limit orders are submitted at the best bid or ask prices. These limit orders at the best

prices are typically cancelled whenever the price changes. For example, in our dataset, among limit orders for stock BAC submitted to the best prices, more than 80% of the orders are canceled completely or partially within the first minute. This makes the fill time observations highly censored and reduces the number of effective observations.

While censoring can be corrected for in the fitting process (and we describe how to do so in Section 4.3), at the end of the day high cancellation rates of real limit orders limit the amount of information that can be extracted from the data set.

- Selection Bias: Informed traders may have strategies that influence the submission of limit orders. These strategies can be based on factors such as short-term price predictions. Orders such as these may have very different fill time distributions than the orders of uninformed traders. In order to predict fill times for uninformed traders, we need unbiased fill time observations of uninformed orders.

Due to these issues, we choose to simulate synthetic limit orders to generate data. These synthetic limit orders are assumed to be infinitesimal and devoid of any market impact. We randomize these order to buy and sell, and their submission times are uniformly sampled throughout the trading hours. These orders are then submitted to the best price level in the same side of the limit order book. As the limit order book evolves over time, the queue positions of these synthetic limit orders also change in the order book. We keep track of these positions and continuously check fill conditions. If the fill conditions are met, we then regard the limit order as executed; if the fill conditions are never met and the market closes, we regard the limit order as unexecuted.

Fill conditions are meant to track the progress of the limit order and identify its fill time had it actually been submitted. Fill conditions are defined as follows.

1. New Limit Order:
    - If a new buy limit order comes in at a higher price than that of a synthetic sell order, then the synthetic sell limit order is filled.
    - If a new sell limit order comes in at a lower price than that of a synthetic buy order, then the synthetic buy limit order is filled.
2. New Market Order:
    - If a market order comes in at a different price as that of a synthetic limit order, then the same logic above applies.
    - If a market order comes in at the same price as a synthetic limit order, then the synthetic limit order is executed if the size of the market order is larger than the share of limit orders in front of the synthetic limit order. Otherwise the queue position of the synthetic limit order advances.
3. Cancellations:

- If a cancellation occurs in front of a synthetic limit order, then the queue position of the limit order advances.

This simulation approach avoids the aforementioned two issues of real limit orders. First, because synthetic limit orders are submitted uniformly over time, it avoids selection biases introduced by trading strategies. Second, because synthetic limit orders aren't canceled until the end of the day, the censoring of fill-time observations is vastly alleviated.

Figure 4 gives a graphic depiction of synthetic limit orders using historical Bank of America data over a particular trading day.

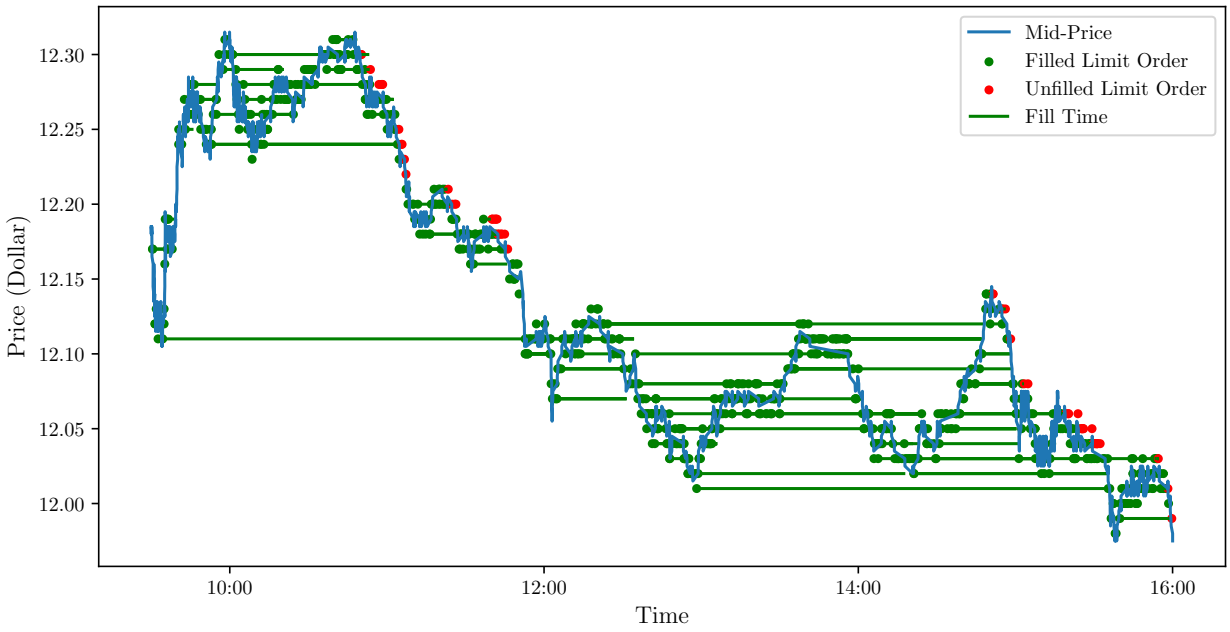

Figure 4: The blue line is the mid-price of BOA stock over the course of a day. Each synthetic limit order is represented by a dot at the time of submission. The dot is colored according to its execution outcome: If a limit order is filled by the end of the day, it is colored green; otherwise it is colored red. For any particular order that is filled, a horizontal line connects its submission time and its execution time — the length of the line represents the time-to-fill.

We record a limit order execution outcome Y as follows:

- A synthetic limit order is filled after time t: $Y = (\text{FILLED}, t)$.
- A synthetic limit order is not filled by time t and was cancelled automatically due to market close: $Y = (\text{UNFILLED}, t)$.

4.3. Maximum Likelihood Estimation

From the hazard rate setup, density and cumulative distribution can be explicitly derived. Therefore log-likelihood can be calculated for each limit order and maximum likelihood estimation can be used for training the RNN. The log-likelihood function can be expressed as follows:

- For an executed limit order: $\mathcal{L}(\lambda; (\text{FILLED}, t)) = \log(f_T(t;\lambda)) = -\Lambda(t) + \log\left(\lambda_{i^*(t)+1}\right)$.
- For an unexecuted limit order: $\mathcal{L}(\lambda; (\text{UNFILLED}, t)) = \log(F(t;\lambda)) = \log\left(1 - e^{\Lambda(t)}\right)$.

The hazard rates λ's are functions of RNN parameters θ, and therefore the log-likelihood function is ultimately a function of the RNN parameters θ. The RNN can be trained by maximizing the average log-likelihood across all synthetic limit orders

$$\max_{\theta} \mathbb{E}\left[\mathcal{L}(\theta; Y)\right].$$

5. Numerical Experiment Setup

This section outlines the details of the data used in the numerical experiments, including stock selection, limit order simulation, and train-test split procedure. Descriptive statistics are presented and some predictive patterns are discussed as well.

5.1. Stock Selection and Experiment Setup

The data we use is from the 502 trading days in the interval from October 1st 2012 to September 30th 2014. A set of large-tick U.S. stocks with high liquidity are selected for this study (see Table 4). Our emipirical study is restricted to large-tick stocks because, at it's core, our method is investigating queueing effects. As is well known in the literature, large- and small-tick stocks trade differently. Queueing effects are more pronounced in large-tick stocks, and features such as queue imbalance matter more.

For each trading day, 1000 synthetic limit orders are simulated at times chosen uniformly throughout the trading hours. For each synthetic limit order, the set of input features is collected and its execution outcome is recorded. These are used as inputs and outputs of the supervised learning algorithm.

For the purpose of fill time distribution estimation, we divide the time axis into 10 intervals (9 closed intervals and 1 half-open half-closed interval). The boundaries of these intervals are set to the deciles of the synthetic fill times.

We train and test a RNN model using 14 months of the data — first 12 months as training data, a subsequent month for validation, and the final month as testing data. The model is regularized by early stopping on the validation data set — the performance on the validation data set is monitored during the training process and the training is stopped once the performance stops improving. This procedure is repeated 10 times, with training, validation and testing data each advancing a month, until reaching the end of the data period (Sept. 30th 2014). Once the RNN models are trained, the performances are computed on the testing data sets. For more details on hyper-parameters tuning and the final neural netowrk architecture, please see Appendix.

Ticker	Avg. Price($)	Vol.(%)	Volume($m)	One tick(%)	T-Size(s)	T-Size(%)
BAC	13.99	24.3	97.6	98.6	51548	0.73
GE	24.40	16.4	70.7	98.7	12047	0.42
VZ	47.94	16.3	85.4	99.1	2225	0.15
F	15.09	22.9	41.9	98.9	20248	0.73
INTC	24.97	20.9	137.1	98.4	13654	0.25
MSFT	34.56	22.1	236.6	97.7	10057	0.15
CSCO	22.52	21.6	131.8	99.1	17090	0.29
WFC	42.85	15.4	97.5	96.9	3433	0.15

Table 4: Descriptive statistics for the above 8 stocks over the two-year period. Average price and (annualized) volatility are calculated using daily closing price. Volume($m) is the average daily trading volume in the unit of a million dollars. One tick(%) is the percentage of time during the trading hours that the spread is one tick. Touch size(s) is the time average of the shares on the top price levels, averaged across the bid and ask. Touch size(%) is normalized using average daily volume, reflecting the percentage of daily liquidity that is available at the best prices.

5.2. Execution Statistics

The following statistics are average values across all 8 stocks over the two-year period. In the following discussion, we will focus on the two quantities below.

- Fill Probability: The probability that a limit order gets filled within a given time threshold h, in other words, $P(T < h)$.
- Conditional Fill Time: The expected fill time given an execution within the time threshold, mathematically expressed as $\mathbb{E}[T|T < h]$.

	Time Horizon		
Statistics	1 Min	5 Min	10 Min
Fill Probability	45%	76%	84%
Average Fill Time (sec)	22.5	70.0	103.4

Table 5: Descriptive Statistics

5.3. Predictable Patterns

Even though limit order executions are inherently random events, there are some features that exhibit strong predictable patterns. These patterns motivate the selection of input features and the construction of benchmark models.

Time of Day:

Trading intensity exhibits intraday patterns. It is most intense around market open and market close, and slowest around noon. This general pattern has strong implication to limit order executions as well. Limit orders are executed faster and with higher fill probabilities around market open

and close and are executed slower and with lower fill probabilities around noon. To demonstrate these patterns, trading hours are broken into 5-minute intervals, and for each interval, average conditional fill times and fill probabilities for all synthetic limit orders given a one-minute time horizon ($h = 1$ min), are plotted on Figure 5. This pattern persists for different time horizons.

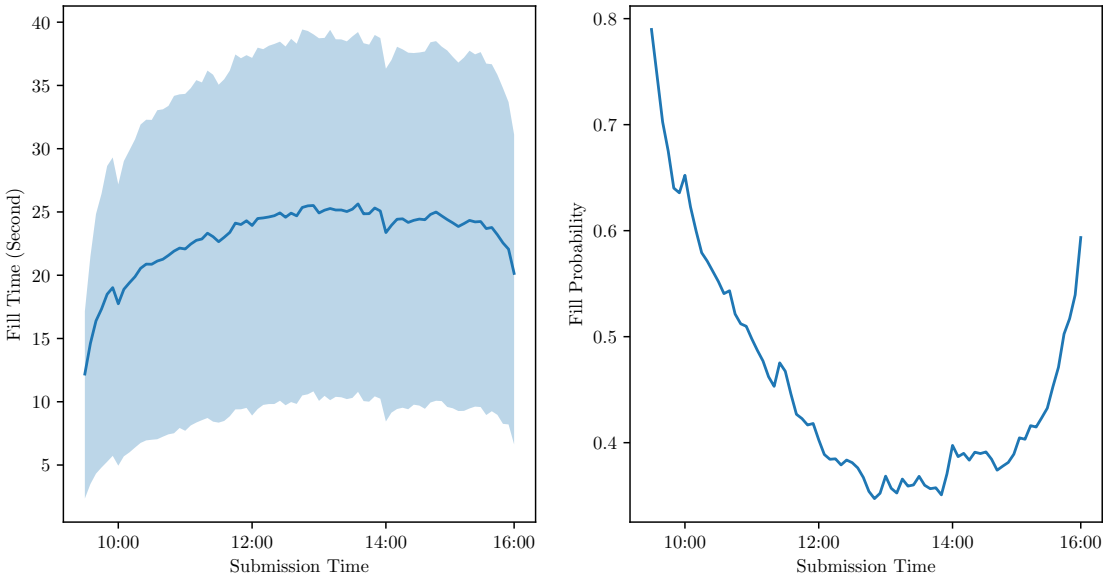

Figure 5: With $h = 1$ min, left sub-graph depicts average conditional fill time as a function of submission time; right sub-graph depicts fill probability as a function of submission time.

Queue Imbalance:

Queue imbalance (QI) is the percentage difference between queue lengths at the top price levels. Queue imbalance can be expressed mathematically as follows

$$\text{QI} = \frac{Q_{\text{near}} - Q_{\text{far}}}{Q_{\text{near}} + Q_{\text{far}}},$$

where Q_{near} and Q_{far} are the queue length at top price levels on the near side and the far side respectively. Queue imbalance reflects the instantaneous imbalance between the supply and the demand for the stock at the current price level. A negative queue imbalance signifies a stronger far side, and the price is more likely to move towards the near side to rebalance the supply and demand. This leads to a higher fill probability and a faster execution for orders submitted to the near side. Conversely, a positive queue imbalance signifies a stronger near side, and the price is more likely to move towards the far side. This leads to a lower fill percentage and a slower execution on average.

Figure 6 shows these patterns. The queue imbalance is recorded at the submission time for each synthetic limit order. These queue imbalance values are then divided into 10 deciles. The average fill times and fill probabilities of the synthetic limit orders submitted with queue imbalance within each deciles are computed and plotted in Figure 6.

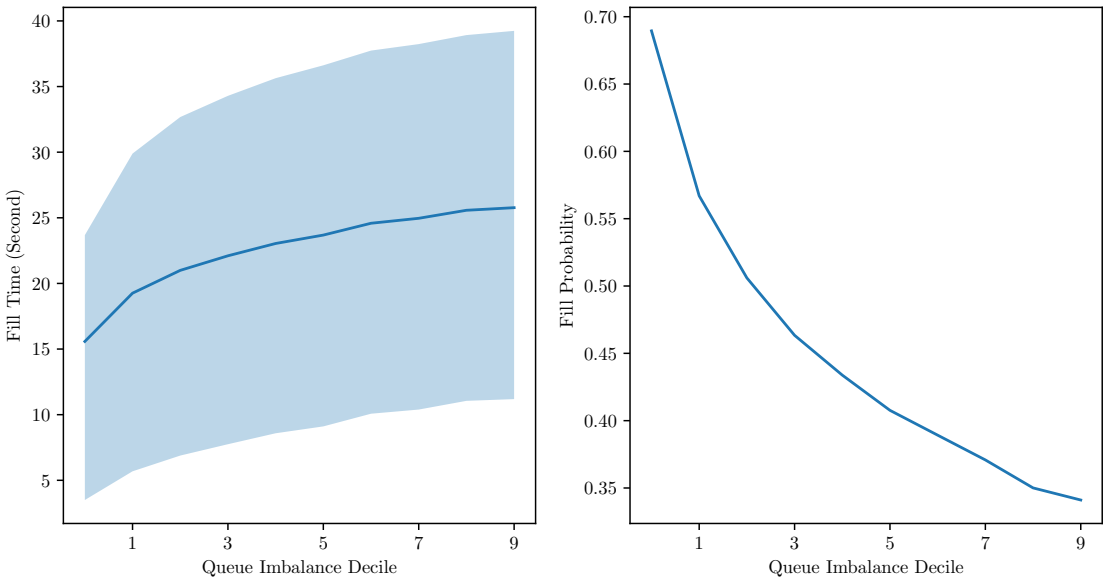

Figure 6: With $h = 1$ min, left sub-graph depicts average conditional fill time as a function of queue imbalance; right sub-graph depicts fill probability as a function of queue imbalance.

Another way to see the impact of queue imbalance is as follows. Trades don't occur uniformly over time. Rather, they occur more often when queue imbalance is at extreme values. Figure 7 illustrates this fact. The left histogram is of queue imbalance sampled at uniformly random time throughout trading days in our data period. The near/far side is also chosen randomly. Clearly, the histogram has a symmetric bell shape centered at 0. This implies that the supply and demand in the market are nearly balanced most of the time, and extreme imbalance occurs very rarely. The right histogram is of queue imbalance sampled only at moments of trades. The near/far side is also chosen randomly. The histogram is still symmetric and centered at 0, but it has much higher concentration at extreme values (close to -1 and 1).

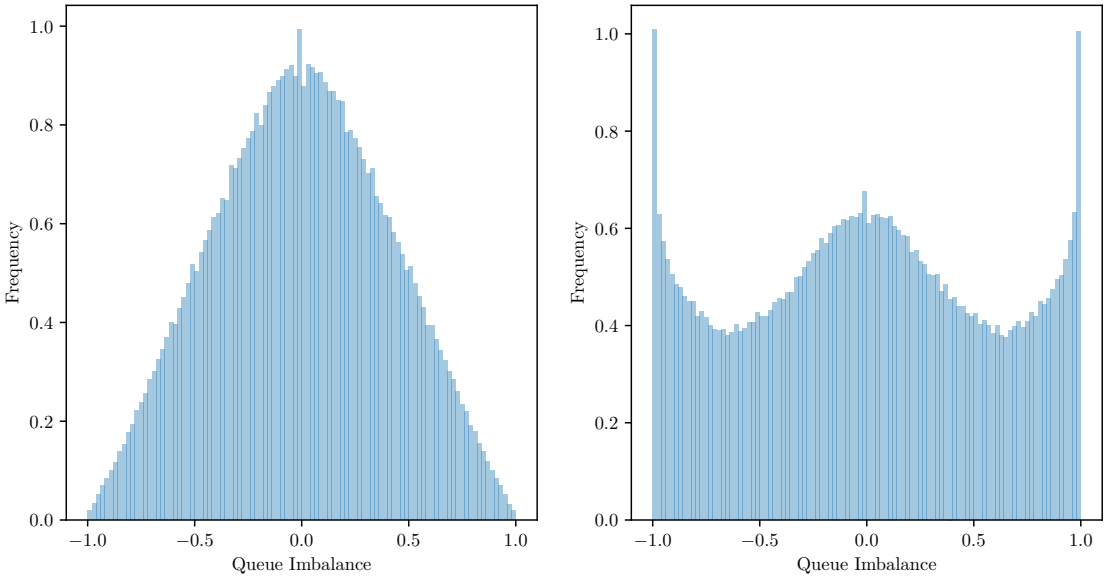

Figure 7: Left sub-graph is a histogram of queue imbalance sampled at random time, the right sub-graph is a histogram of queue imbalance sampled at trades.

6. Numerical Experiment Results

This section outlines the results of the numerical experiments. The performance of the RNN models are compared to benchmark models, and their applications in the trading problem of Section 2.2 is revisited.

6.1. Benchmark Models

We compare the performance of RNN against benchmark models using the same two metrics from Section 5.2, namely fill probability and conditional expected fill time. The following benchmark models are used for comparison purposes.

Linear/Logistic Regression:

Predicting whether an order will be filled is a binary classification problem and logistic regression is a natural linear benchmark. Predicting conditional expected fill time is a continuous value prediction problem and linear regression is a natural benchmark. Only the input features collected at the time of submission are used in these two models.

Bucket Prediction:

Regressions only capture linear patterns in the data. To construct a non-parametric benchmark, we use bucketed empirical means as estimators. Based on the discussion in Section 5.3, we have chosen time of day and queue imbalance as features for bucketing.

Time of day is divided into 15-minute intervals and queue imbalance is divided into quintiles. Each bucket is the intersection of a time-of-day interval and a queue imbalance quintile. Within each bucket, simple empirical means of whether orders are filled and their fill times are used as the predictions.

Point Estimator:

For the problem of estimating fill time, the simplest estimation method wold be to make completely unconditional prediction with respect to market conditions. We call this the point estimator, it is computed by averaging fill times across all orders filled within a target horizon.

6.2. Fill Probability

To evaluate the accuracy of fill probability predictions using various models, the area under the curve (AUC) of a receiver operating characteristic (ROC) curve is used as a metric.

	Time Horizon		
AUC	1 min	5 min	10 min
Logistic Regression	0.62	0.60	0.60
Bucket Prediction	0.63	0.60	0.60
RNN	0.72	0.67	0.66

Table 6: Fill Probability Prediction Results

As we see in Table 6, RNN outperforms both benchmark models in all three time horizons. As the time horizon lengthens, the prediction accuracy of all three models decreases, suggesting that the fill probability is more predictable within a shorter time horizons.

6.3. Conditional Expectation of Fill Time

To evaluate the accuracy of the predictions of conditional expectation of fill time, the root mean square error (RMSE) between actual fill times and estimated fill times is used as a metric.

	Time Horizon					
	1 min		5 min		10 min	
	RMSE	reduct.%	RMSE	reduct.%	RMSE	reduct.%
Point Estimator	16.6	-	70.2	-	123.1	-
Linear Regression	16.3	1.8%	68.9	1.9%	121.2	1.5%
Bucket Prediction	16.1	3.0%	68.3	2.7%	119.1	3.2%
RNN	15.1	9.0%	63.9	9.0%	112.8	8.4%

Table 7: Conditional Expected Fill-Time Prediction Results: The percentage reduction (reduct.%) is relative to the point estimator RMSE.

As we can see in Table 7, RNN outperforms all benchmark models in all three time horizons. Similar to fill probability prediction, as the time horizon lengthens, the prediction accuracy of all models decreases, suggesting that the fill-time is more predictable within a shorter time horizon.

6.4. Implementation Shortfall

In this section, we revisit the trading problem of Section 2.2 and demonstrate that a more accurate fill probability prediction leads to a better trading strategy in terms of reducing implementation shortfall.

As discussed in Section 2.2, a trader is seeking to buy a share of the stock with either a market order or a limit order. In order to minimize implementation shortfall, the optimal trading decision will be based on a threshold policy. More specifically, a limit order should be used if and only if

$$P(T \leq h) > \Theta,$$

where h is the target horizon, T is the random variable representing the fill time, $P(T \leq h)$ is the fill probability, and Θ is the unknown threshold. The fill probability $P(T \leq h)$ can be estimated using various predictive models, and Θ will be optimized according to a method described shortly.

For a particular trade, the choice of limit order versus market order is a direct consequence of the fill probability prediction. This choice impacts the resulting implementation shortfall. Let $\hat{p}$ be the predicted fill probability within time horizon h. The implementation shortfall for the particular order can be expressed as a function of fill probability $\hat{p}$, horizon h, and the threshold Θ by

$$\text{IS} = \mathbb{1}\{\hat{p} \leq \Theta\}\text{IS}_{\text{mkt}} + \mathbb{1}\{\hat{p} > \Theta\}\text{IS}_{\text{limit}}, \tag{1}$$

where IS_{mkt} and IS_{limit} are the implementation shortfall of using a market order and a limit order respectively, given by

$$\text{IS}_{\text{mkt}} = p_A(0) - p_M(0); \quad \text{IS}_{\text{limit}} = \mathbb{1}\{T \leq h\}[p_B(0) - P_M(0)] + \mathbb{1}\{T > h\}\text{C}_{\text{clean-up}}. \tag{2}$$

At the limit order submission time, the mid-price is $p_M(0)$, the best ask price is $P_A(0)$, and the best bid price is $p_B(0)$. The clean-up trade occurs when the limit order is predicted to be executed within the target horizon, but doesn't actually execute. This condition can be expressed as $\mathbb{1}\{\hat{p} > \Theta\}\mathbb{1}\{T > h\} = 1$. Under this condition, the trader is required to fill the order with a market order at time h. The cost of this market order is the clean-up cost, given by

$$\text{C}_{\text{clean-up}} = p_A(h) - p_M(0), \tag{3}$$

where $p_A(h)$ is the best ask price at time h, which is the cost of a market order at the end of the target horizon.

To optimally set the threshold Θ, for each model, we choose the threshold that minimizes

implementation shortfall empirically over the test data set.

Numerical Results:

Table 8 displays the average implementation shortfall per trade using various strategies. The market order strategy is when the trader submits market orders for all trades at the beginning of the time horizon. Because every trade yields half-spread as implementation shortfall, the length of the time horizon doesn't affect this strategy at all. The limit order strategy is when the trader submits limit orders for all traders at the beginning of the time horizon and places clean-up traders if the limit orders aren't executed by the end of the horizon. For longer time horizons, limit orders are executed with higher probability and this leads to a lower implementation shortfall. These two strategies do not require any predictive modeling.

We compare these to policies based on fill probabilities prediction using three different models — logistic regression, bucket prediction, and the RNN model. These three strategies follow the threshold policy with their respective fill probability estimates. Table 8 displays the implementation shortfall of these five trading strategies averaged over all synthetic limit orders.

	Time Horizon								
	1 min			5 min			10 min		
IS (ticks)	mean	s.e.	reduct.%	mean	s.e.	reduct.%	mean	s.e.	reduct.%
Market Order Strat.	0.508	-	-14%	0.508	-	-51%	0.508	-	-95%
Limit Order Strat.	0.443	0.02	-	0.337	0.04	-	0.260	0.07	-
Logistic Reg.	0.321	0.02	28%	0.198	0.04	41%	0.163	0.07	37%
Bucket Pred.	0.317	0.02	28%	0.194	0.04	42%	0.159	0.06	39%
RNN	0.279	0.02	37%	0.172	0.03	49%	0.148	0.06	43%

Table 8: Percentage reduction (reduct.%) is relative to the limit order strategy. The implementation shortfall displayed above is in ticks.

As can be seen in Table 8, using any predictive models significantly reduces implementation shortfall, with RNN yielding the best performance across all three time horizons. These reductions is partly due to a more accurate fill probability prediction, but also partly due to a smaller clean-up cost. Empirically, RNN consistently produces a lower clean-up cost compared to other models (see Table 9). This suggests that when RNN incorrectly predicts a limit order execution, the price doesn't move as far to the other direction as when the benchmark models make the same incorrect prediction. For longer time horizons, because the price has more time to move away even further, the clean-up cost becomes much higher.

	Time Horizon					
	1 min		5 min		10 min	
Clean-up Cost (ticks)	mean	s.e.	mean	s.e.	mean	s.e.
Limit Order Strategy	1.22	0.01	3.01	0.02	4.29	0.05
Logistic Regression	1.17	0.01	2.95	0.02	4.25	0.05
Bucket Estimator	1.14	0.01	2.94	0.03	4.12	0.06
RNN	1.09	0.01	2.89	0.03	4.08	0.05

Table 9: Average Clean-up Cost (ticks) Conditioned on Clean-up

7. Conclusion

The choice between market orders and limit orders can be viewed as a tradeoff between immediate executions and price premium. To make this choice intelligently, one must consider the uncertainty of limit order executions. In this study, we develop a data-driven approach to predict fill probabilities via estimating the distribution of limit order fill times.

In order to generate an unbiased data set of limit order fill times, we use historical NASDAQ ITCH dataset to simulate synthetic limit orders, track their positions, and record their fill times. To estimate the distribution of fill times, we construct a RNN to predict hazard rates on pre-determined intervals on the time axis. The RNN produces significant predictabilities, more accuracy than benchmark models. This prediction improvement has economic values as well. In a prototypical trading problem, when the trading strategy is implemented by RNN, it results the lowest implementation shortfall.

This study differs from many other studies in the following ways:

1. As far as we know, this is the first study to predict the distribution of limit order fill times.
2. By using a data-driven approach, we operate under minimal model assumptions.
3. We use a RNN to incorporate past order flow information for prediction.

The following are some remarks regarding limitations of this study:

1. Our current method only provides estimates for limit orders submitted to the best bid/ask price. Previous studies have found that the execution outcomes are sensitive to limit prices, and therefore it's inappropriate to use our current model to provide estimates for limit orders submitted at other price levels. A further study can be conducted by extending this paper to multiple price levels. This would help evaluate a further tradeoff between limit prices and fill probabilities.
2. The synthetic limit orders are assumed to be infinitesimal and devoid of any market impact. This also implies that the synthetic limit orders can't be partially filled. However, previous studies such as Lo et al. [2002] have suggested that the size of the limit order doesn't impact the execution outcomes significantly.

References

J. J. Angel. Who gets price improvement on the NYSE. *Working Paper*, 1994.

G. Y. Ban, N. E. Karoui, and A. E. B. Lim. Machine learning and portfolio optimization. *Management Science*, 64(3):1136–1154, March 2018.

B. Biais, P. Hillion, and C. Spatt. An empirical analysis of the limit order book and the order flow in the Paris Bourse. *The Journal of Finance*, 50(5):1655–1689, December 1995.

P. Carr, L. Wu, and Z. Zhang. Using machine learning to predict realized variance. *Working Paper*, 2019.

J. W. Cho and E. Nelling. The probability of limit-order execution. *Financial Analysts Journal*, 56(5):28–33, September 2000.

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

M. Dixon and J. London. Financial forecasting with α-rnns: A time series modeling approach. *Front. Appl. Math. Stat.*, February 2021.

M. Dixon, D. Klabjan, and J. H. Bang. Classification-based financial markets prediction using deep neural networks. *Working Paper*, 2017.

J. B. Heaton, N. G. Polson, and J. H. Witte. Deep learning in finance. *Working Paper*, 2016.

B. Hollifield, R. A. Miller, and P. Sandas. Econometric analysis of limit-order executions. *The Review of Economic Studies*, 71(4):1027–1063, October 2004.

A. W. Lo, A. C. MacKinlay, and J. Zhang. Econometric models of limit-order executions. *Journal of Financial Economics*, 65(1):31–71, July 2002.

C. C. Moallemi and K. Yuan. A model for queue position valuation in a limit order book. *Working Paper*, 2016.

A. Moghar and Mhamed Hamiche. Stock market prediction using lstm recurrent neural network. 170:1168–1173, 2020.

NASDAQ. Nasdaq totalview-itch 4.1, 2010. URL `http://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/nqtv-itch-v4_1.pdf`.

A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Working Paper*, 2018.

N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data. *IEEE Transactions on Emerging Topics in Computational Intelligence*, October 2018.

M. A. Petersen and D. Fialkowski. Posted versus effective spreads: Good prices or bad quotes. *Journal of Financial Economics*, 35(3):269–292, June 1994.

J. Sirignano and R. Cont. Universal features of price formation in financial makrets: Persepectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, July 2019.

J. A. Sirignano. Deep learning for limit order books. *Quantitative Finance*, 19(4):549–570, 2019. 10.1080/14697688.2018.1546053. URL `https://doi.org/10.1080/14697688.2018.1546053`.

I. M. Toke. The order book as a queueing system: average depth and influence of the size of limit orders. *Quantitative Finance*, 15(5):795–808, November 2013.

D. T. Tran, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Tensor representation in high-frequency financial data for price change prediction. *IEEE Symposium Series*, November 2017.

D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE transactions on neural networks and learning systems*, 30(5):1407–1418, May 2019.

A. Tsantekidis, N. Passalis, J. Kanniainen A. Tefas, M. Gabbouj, and A. Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. *IEEE Business Informatics (CBI)*, 2017.

Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen Chu. Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

R. Xiong, E. P. Nichols, and Y. Shen. Deep learning stock volatility with Google domestic trends. *Working Paper*, 2015.

Z. Zhang, S. Zohren, and S. Roberts. Deeplob: Deep convolutional neural networks for limit order books. *Working Paper*, 2019.

A. Input Features

The set of features that make up x_t at any given time t is listed in Table 10. These features are designed to be symmetric between buying and selling. In order words, the side of the market (bid or ask) is only distinguishable relative to the intended trade direction as near-side and far-side. Near-side is the side at which the execution seeks to fulfill an order, and the far-side is the opposite side. Namely, for a buying order, the near-side is the bid-side and the far-side is the ask-side. For a selling order, the near-side is the ask-side and the far-side is the bid-side.

Category	**Features**
General Information	Time of day, time since last trade, bid-ask spread
Depth (top 5 price levels)	Queue imbalances, near/far depths in shares
Flow Information	Number of shares added to the top price levels since last trade, number of shares cancelled at the top price levels since last trade, number of shares executed last trade.
Intensity Measures	Intensity measure for trades at near-side and far-side, price changes at near-side and far-side

Table 10: Input features.

These features are designed as follows:

- Near depths and far depths refer to the number of outstanding shares at any given price level at respective sides. These values are normalized by the trailing 21-day average daily trading volumes in shares.

- Queue imbalance is defined in Section 5.3. This is a value between -1 and 1 and represents the imbalance of the supply and demand of the stock at the current price level. This can be calculated using depths at the top price levels and the aggregated depth at the top 5 price levels. More specifically, we compute queue imbalance for the top k levels as

$$QI^{(k)} = \frac{Q^{(k)}_{near} - Q^{(k)}_{far}}{Q^{(k)}_{near} + Q^{(k)}_{far}}$$

  where $Q^{(k)}_{near}$ and $Q^{(k)}_{far}$ are the total number of shares on the top k price levels at the near and far side. We compute this quantity for $k = 1, ..., 5$.

- We define the intensity measure of any event as an exponentially decaying function with increments only at occurrences of such an event. Let S_t be the magnitude of the event at any given time t, with $S_t = 0$ if there is no occurrence of such event at time t. The intensity measure $X(t)$ is defined as

$$X(t + \Delta t) = X(t) \cdot \exp(-\Delta t/t_c) + S_{t+\Delta t},$$

At any time t and for any duration Δt, if there is no event occurrence between t and $t + \Delta t$, then the intensity measure decays exponentially. The time constant t_c controls the rate of the decay.

In our implementation, we compute intensity measure for trades and price changes on the near-side and far-side. The magnitude of a trade is the size of the trade in dollars, normalized by average daily volume. The magnitude of a price change is normalized by the average spread. t_c is chosen so that the intensity of any event (trades or price changes) would only have 10% remaining after 10 minutes. In other words, $t_c = -10/\ln 0.1$ minutes.

B. Neural Network Implementation

In order to capture the temporal dynamic of the financial market, the same set of features are collected not only at the moment of order submission, but also prior to the order's submission. This time series of features can be represented as $X = \{x_{-n}, ..., x_{-2}, x_{-1}, x_0\}$ where x_0 is the set of features collected at the order's submissions, and $x_{-n}, ..., x_{-1}$ are sets of features collected prior to order's submission. In this particular implementation, prior features are collected at the moment of trades. In other words, x_{-1} is is collected at the trade that occurred immediately prior to the submission of the limit order, and in general, x_{-k} is collected at the kth trade immediately prior to order's submission.

This time series of features is used as input features into the RNN. The RNN units are implemented as LSTM units with dimension 64. The output of the LSTM units go through another 5 layers of fully-connected network, which is denoted as "Output Layer" in Figure 2. The output layer uses ReLu as activations functions in the hidden layers. The last layer of the output layer uses softplus activation function to enforce the positivity of the output. The final output of the RNN is a vector of hazard rates $\lambda \in \mathbb{R}^{10}$ on 10 pre-determined intervals. These intervals are set individually for each stock so that the boundaries of the intervals are the deciles of fill times. A Python Tensorflow Implementation of the neural network is provided in Table 11

```python
import tensorflow as tf
import tensorflow.contrib.layers as layers

def rnn_model(inpt, features_dim, rnn_seq, output_dim, scope, phase):
    # inpt: an array that contains the data within each batch in the SGD algorithm
    # inpt.shape = [batch_size, rnn_seq, features_dim]
    # features_dim: the dimensions of each set of feature
    # rnn_seq: the length of the input series
    # output_dim: the dimensions of the outout

    def dense_linear(x, size):
        return layers.fully_connected(x, size, activation_fn=None)

    def dense_batch_relu(x, size):
        h1 = layers.fully_connected(x, num_outputs = size, activation_fn=None)
        h2 = layers.batch_norm(h1, center=True, scale=True, is_training=phase)
        return tf.nn.relu(h2)

    inpt_rnn = tf.reshape(inpt, [tf.shape(inpt)[0], rnn_seq, rnn_states])

    with tf.variable_scope(scope):
        inputs_series = tf.split(inpt_rnn, rnn_seq, 1)
        inputs_series = [tf.squeeze(ts,axis = 1) for ts in inputs_series]
        cell = tf.contrib.rnn.LSTMCell(64)
        states_series, current_state = tf.contrib.rnn \
            .static_rnn(cell, inputs_series, dtype =tf.float32 )

        out = current_state
        for i in range(5):
            out = dense_batch_relu(out, 64)
        out = dense_linear(out, output_dim)
        out =  tf.nn.softplus(out)

        return out

```

Table 11: The Python Tensorflow implementation of the RNN architecture.

Hyperparameter Tuning:

A few neural network architectures have been tried including more a traditional feed-forward neural network and LSTM with 2 layers. The reported architecture above had the best performance on the validation datasets. For each of the following hyper-parameters, a few common values were tried and the best values are reported below:

- Length of the input series: $n = 50$
- Learning rate: 3×10^{-5}
- Batch size: 1024

C. Supplementary Numerical Experiments

In order further test the robustness of the RNN model and the choice of the input features, we have also conducted a limited numerical experiments only on BAC. The purpose of this numerical experiment is to mainly test the following comparisons.

C.1. Feed-forward Neural Network vs. RNN

The main advantage of a RNN over a more traditional feed-forward neural network is that it naturally process a time series of input features and captures temporal dynamics. In order to numerical test the additional explanatory power of the temporal dynamic, we have tested a MLP (multilayer perceptron) or a feed-forward neural network. Specifically, it has the same architecture as the feed-forward neural network portion of the RNN, in other words, it is the "Output Layer" in Figure 2. Because MLP can't process time-series of input features, the input of the MLP consists of solely the features sampled at limit order submission. Because this MLP has the same input and the same structure as the feed-forward portion of the RNN, the performance difference between the MLP and the RNN can be attributed to the lagged features and the additional RNN architecture.

The numerical results are displayed in Table 12, 13, and 14, and the comparisons are between MLP and RNN. As can be seen in the numerical results, MLP incorporates non-linear patterns and outperforms linear models, and RNN still outperforms the MLP by incorporating temporal dynamic in the data.

C.2. Volatility Feature

The current feature set doesn't include a measurement of volatility, however, many of existing features in the linear model also captures the "rate of the market events" such as "time since last trade" and all the flow features. As a result, including volatility in the linear models doesn't improve the performance significantly.

A numerical experiment has been conducted on BAC to demonstrate the benefit of adding volatility to linear model (see Table 12, 13, and 14). The volatility is taken to be the volatility of the preceding 10-minute interval of a limit order submission, in other words, the standard deviation of the returns of each price changes within the time interval. The numerical results below compares the results of the linear models with and without the volatility feature.

	Time Horizon		
AUC	1 min	5 min	10 min
Logistic Regression	0.60	0.58	0.58
Logistic Regression with Vol.	0.61	0.58	0.58
MLP	0.66	0.62	0.61
RNN	0.71	0.65	0.63

Table 12: Fill Probability Prediction Results on BAC

	Time Horizon					
	1 min		5 min		10 min	
	RMSE	reduct.%	RMSE	reduct.%	RMSE	reduct.%
Point Estimator	12.1	-	45.3	-	92.5	-
Linear Regression	11.8	2.5%	44.3	2.2%	90.4	2.27%
Linear Regression with Vol.	11.7	3.1%	44.2	2.43%	90.3	2.38%
MLP	11.2	7.4%	42.5	6.2%	87.9	4.9%
RNN	10.9	9.9%	41.2	9.1%	87.0	5.9%

Table 13: Conditional Expected Fill-Time Prediction Results on BAC

	Time Horizon								
	1 min			5 min			10 min		
IS (ticks)	mean	s.e.	reduct.%	mean	s.e.	reduct.%	mean	s.e.	reduct.%
Market Order Strat.	0.508	-	-14%	0.508	-	-51%	0.508	-	-95%
Limit Order Strat.	0.424	0.06	-	0.316	0.11	-	0.201	0.21	-
Logistic Reg.	0.318	0.05	25%	0.196	0.08	38%	0.133	0.19	34%
Logistic Reg. with Vol.	0.305	0.05	28%	0.192	0.08	39%	0.133	0.19	34%
MLP	0.291	0.05	31%	0.185	0.08	41%	0.129	0.19	36%
RNN	0.279	0.05	34%	0.175	0.08	45%	0.123	0.19	39%

Table 14: Implementation Shortfall for BAC