Delft University of Technology

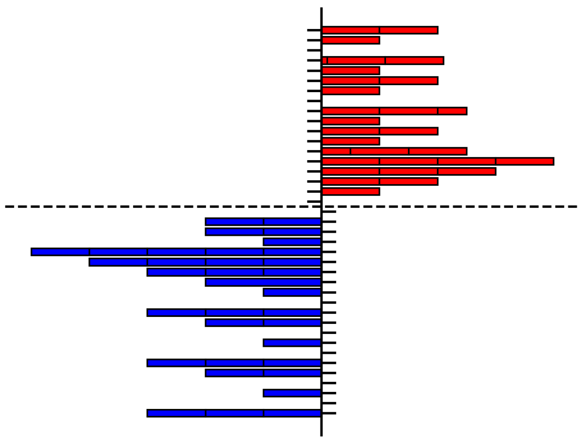
Faculty of EEMCS

Master's Thesis

Applied Mathematics

Specialisation Financial Engineering

WMR prediction using recurrent neural networks on FX limit order book data



Steven Kortekaas
Student number 4453735

supervised by
Prof. Antonis Papapantoleon

June 3rd, 2022

Preface

This thesis investigates the application of machine learning models on foreign exchange data around the WM/R 4pm Closing Spot Rate (colloquially known as the WMR Fix). Due to the nature of the market dynamics around the WMR Fix, inefficiencies can occur and therefore some predictability might be expected. We aim to find these inefficiencies. This is done by applying machine learning models, specifically recurrent neural networks, on limit order book data of foreign exchange (FX). The focus will be on the Euro - US dollar exchange rate.

The thesis is performed as part of the Master Applied Mathematics at Delft University of Technology, in collaboration with PGGM and MN. PGGM and MN are two pension fund service providers and both provide asset management for multiple pension funds with over EUR 400 billion combined assets under management. The thesis is part of the Academic Excellence Program (AXP), which is a collaboration between both companies and universities.

The ideas in this thesis can be useful for both pension funds. Most importantly it demonstrates ideas coming out of academic literature on how to extract information from limit order book data using machine learning models. Furthermore, it provides them with thoughts on how to improve their operations and explore other ideas.

Various "off-the-shelf" recurrent neural networks have been used, which in literature have shown potential in predicting prices on short time scales. An attempt has been made to see if these models can be used on a longer time scale to predict the Fix. The results indicate, however, that the time scales used are too long for the models to make accurate predictions based on the limit order book and a simple estimate performs better.

The thesis committee is composed of Prof. Antonis Papapantoleon, Dr. Fang Fang and Tjerk Methorst, MSc. Prof. Papapantoleon has acted as thesis advisor and Tjerk Methorst as daily supervisor.

I wish to thank Tjerk Methorst (PGGM) and Liakos Papapoulos (MN) for accommodating my graduation and providing all the resources, and I wish to thank Prof. Papapantoleon for his role as thesis advisor.

Steven Kortekaas, May 2022.

Contents

Acronyms

ANN Artificial neural network. 22

CNN Convolutional neural network. 3

EUR Euro. 5

EURUSD Euro-US dollar currency pair. 5, 8, 9, 17

FFNN Feedforward neural network. 23

Fix See WMR. 8

FX Foreign exchange. 1

i.i.d. Independent and identically distributed. 27

LN London time zone. 8, 16

LOB Limit order book. 5–7

LSTM Long short-term memory. ii, 25, 26

MSE Mean-squared error. 26

pip Percentage in point. For EURUSD equal to 0.0001$. 16

RNN Recurrent neural network. 23

USD US dollar. 5

WMR WM/R 4pm Closing Spot Rate. 8

List of Figures

List of Tables

1 Introduction

By investing in assets that are traded in a foreign currency, pension funds run the risk of losing the return on these investments when changes in currency exchange rates occur, a phenomenon called currency risk. The exposure to these risks can be hedged by adding foreign exchange (FX or Forex) derivatives, like FX swaps, to the portfolio. While the size of the currency hedge remains the same, the assets it is protecting drift on the currents of financial markets. This means the hedge needs to be adjusted which is usually done via FX spot transactions.

Moreover, international investors use FX rates to translate asset values denominated in foreign currencies to their domestic (reporting) currency. To do so they usually utilize reference or benchmark rates published by an independent source. One example of a benchmark rate is the WM/R 4pm Closing Spot Rate, commonly known as the 'Fix'. The time at which the Fix is calculated is considered to be the most important moment in the FX spot market [1], which is the largest global market with $6.6 trillion traded daily [2]. It is calculated every day the currency market is open using data from 15:57:30 to 16:02:30 London time, and serves as the FX benchmark rate for that day.

Because of its use as benchmark, hedges are often adjusted according to the value of the Fix, resulting in a need to trade against the Fix. Furthermore the Fix is used by index providers, such as MSCI, which contain assets expressed in various currencies. These indices are widely adopted in the financial industry. In order to reduce tracking errors[1], against for example these indices, many investors wish to execute against the Fix.

As the Fix is used by many investors as benchmark, volume during the window is generally large and much market activity takes place. Previous research has shown that the market dynamics during the window exhibits some predictable behaviour, which might be partly caused by the rebalancing of currency hedges [3], and partly because certain funds need to minimize their tracking error. These effects lead to a large number of transactions during the window which are more driven by the necessity to trade and less by the actual price levels. This could potentially result in inefficiencies which can be exploited.

However, in practice it is hard to get a better rate than the Fix. Since traded volumes during the window are large, small improvements against the Fix can result in reasonable savings for the pension funds. We aim to use insights from a data-driven approach, with the anticipation that these insights can lead to an improvement of the FX execution strategies with respect to the Fix benchmark.

Modelling of financial time series and forecasting[2] is a vastly studied area [4], including its application to FX. However, in recent years the application of machine learning on financial time series has emerged [5], and is showing promising results compared to traditional approaches to time series. Specifically the application of machine learning models on limit order books has increased. Modern day electronic market places, including those used in the FX market, use limit order books to keep track of orders placed by market participants and to match the orders if possible.

Traditional approaches in financial time-series often require extensive domain knowledge. Furthermore in order to be comprehensible they are limited in predictive performance. Machine learning does not assume an underlying stochastic process generating the time series, which allows for greater performance but at the cost of interpretability.

Few academic studies on the use of machine learning in finance investigate FX specifically, which is surprising given the non-linear, non-stationary nature of FX [6]. Moreover, none of these methods have been specifically applied to FX data at times around the Fix. We aim to use the ideas in academic literature on predicting prices using machine learning methods together with order book data of FX markets to forecast the Fix, where the focus will be specifically on recurrent neural networks (and derived models).

[1]The tracking error is the difference between a fund and the benchmark it follows.
[2]The term forecasting and predicting are used interchangeably.

Broadly speaking, forecasting models can be split in two groups: price prediction and prediction of price movement (e.g. up or down). In machine learning terms, the former constitutes regression while the latter constitutes classification. While regression might be appealing as it produces not only the price movement, but also the magnitude, it can be harder than classification. Furthermore, the forecasts from regression do not yield a distribution of the uncertainty as it is assumed that the data generating process is unknown. On the other hand, classification is dependent on chosen thresholds, which regression does not suffer from. In this thesis, both regression and classification are considered.

With the previous in mind, the main thesis problem is given by: Is it possible to accurately predict the WMR by using machine learning models on FX limit order book data? As stated the focus of the machine learning models is on recurrent neural networks, and is based on results in academic literature.

In Section 2 the related work is described and the ideas are presented in which way the information can be extracted from the data. Then in Section 3 a description of FX markets and limit order books is given. After this a general description is given of the WMR in Section 4. The raw data used is described in Section 5. The models which are used on the data are described in Section 6. Lastly Section 7 shows the results and Section 8 concludes.

2 Related work

In recent years, more evidence has been found which support the predictability of financial markets [7] and the efficient market hypothesis has been increasingly questioned [8]. In case of FX, working papers have shown that currency markets around the Fix exhibit some predictability [9, 10]. Multiple sources attribute this predictability to the hedging of currency exchange rates by investors and models have been developed which in the past have accurately predicted the direction of the Fix using equity returns [1, 3, 11]. It is assumed in this thesis that these effects manifest themselves in the limit order book data.

In general there are two approaches in predicting financial time-series: statistical (parametric) models and data-driven machine learning models [12, 13]. The "traditional" statistical methods assume that the time-series is generated from a stochastic process [12, 14]. However there is agreement that financial time-series behave more complex, typically due to their non-stationarity and non-linearity [12]. Machine learning techniques are able to overcome this deficit. In essence the need of domain knowledge has been replaced by complexity of the models and computational power [15]. Machine learning models learn suitable representations from raw data, without human input.

Specifically in the case of limit order books (LOB), stochastic models have been developed to simulate their dynamics, e.g. [16, 17]. These models are motivated by the desire to bridge the gap between the microscopic description of price formation and the stochastic processes used to describe price evolution at macroscopic time scales.

Although they have advanced the economic and mathematical understanding of limit order books, the assumed simplifications limit their practical scope [18], for instance because of their non-linearity [19]. Because of this practitioners in finance rely on statistical methods to model their dynamics. Machine learning methods are able to capture arbitrary non-linear relationships without information on the data [20]. Arguably, machine learning with neural networks is the best approach to data-driven modelling of limit order books [18].

In recent literature there has been an increase in interest regarding the use of machine learning in finance on LOB data [21]. The models have been applied on LOBs of different asset classes, such as equities. There has been an indication, however, that the models exhibit some form of universality in the sense that they also perform well on data of other financial instruments [22]. In the broad class of machine learning models, artificial neural networks (ANN) have shown promising results on the prediction of price using LOB data. Examples include Convolutional Neural Networks (CNN) [23], Recurrent Neural Networks () [24], Long Short-Term Memory (LSTM) [15], and combinations of CNN and LSTM [12, 21]. Even though all these papers assume there is predicitve value in the limit order book, there is little supporting evidence why that is [15]. Thorough literature reviews of machine learning on limit order books can be found in [5], [25], and [26].

In the majority of these studies, the inputs are represented as raw or transformed times-series of order book states, and the return forecasting problem is formulated as a classification task, where a single time horizon is chosen which is either deterministic, e.g. fixed time horizon, or stochastic, e.g. the time interval until the next price change. Usually, the time scales of the prediction is short, e.g. seconds or even milliseconds. The application of these models is mostly relevant for high frequency traders (HFT). For example, in [24] an explanation of the significance for HFTs is given, and in which way these models are used to reduce their risk.

One exception on the short time horizons is [27], where neural networks are used to predict the EURUSD on time horizons ranging from daily to yearly (although no LOB data is used). They conclude that the performance is better on shorter time horizons, which is probably due to the fact that FX is one of the most volatile markets [21].

In this thesis, both a classification and a regression approach is taken. Most literature formulate their problems as classification, however regression avoids certain issues associated with training classifiers [15]. Classification usually predicts whether the return over a certain horizon is positive, negative, or zero [15]. This is mostly done by creating labels from smoothed forward prices using moving averages, and then assigning a label according to thresholds. However, this introduces undesirable ad hoc modelling parameters, and there is no canonical way to perform this labeling [15].

In [15] various machine learning models (CNN, LSTM, CNN-LSTM) are considered on equities from the Nasdaq. The price prediction problem is formulated as a regression problem, where multiple time horizons are considered. They show that the effective horizon of stock specific forecasts is approximately two average price changes. Furthermore, they show that "off-the-shelf" ANNs can achieve state-of-the-art predictive performance when they are trained on stationary input derived from the order book, called order flow.

In [18] a composite ANN model is constructed based on LSTMs and CNN, called DeepLOB, and is applied to equity limit order book data from the London Stock Exchange. The model is posed as a classification problem, where the labels up, down and stationary are created using a smoothed forward price and a threshold. They use a mean and standard deviation of the previous 5 days to normalise the data. They argue that normalisation is a necessity for machine learning models as they are sensitive to input scaling, and that financial time-series usually experience regime shift, so that static normalisation is not appropriate for dataset spanning long time periods (e.g. one year). In contrast to [15], DeepLOB is not an "off-the-shelf" ANN, but is trained on the state of the order book instead of on the order flow.

As the goal of this thesis is to forecast the WMR, which is determined from a 5 minute window, these time scales are considered too short. Therefore, we aim to take the ideas of using machine learning methods on limit order books and apply them on larger time scale to forecast the value of the WMR. In light of this, the research question can slightly be refined as: Can these models be applied on longer time scales using limit order book data around the WMR? The application of machine learning in order to forecast the WMR has not been found in academic literature.

3 FX and limit order books

In this section some preliminaries will be considered. First the basics of the FX market is briefly discussed After this a theoretical aspects of limit order books will be discussed to provide details on how the data is generated.

3.1 FX

The price of a currency can be expressed with another price as numeaire, which is called exchange rate. As an example, in case a trader wishes to buy 5 euro (EUR) using US dollars (USD) and the exchange rate is 1.2 EURUSD, then he needs to pay 6 USD. EURUSD is to be interpreted as amount of US dollars per euro. The same holds for other currencies, e.g. CHFJPY (Japanese yen, JPY, per Swiss franc, CHF). This means that an exchange rate is equivalent to the price of a currency expressed in another currency. For this reason, in the remainder the terms price and (exchange) rate are used interchangeably.

One can also trade USDEUR, but this is against convention [**donnely**]. There are two main reasons. First buying USDEUR is the same as selling EURUSD. Secondly, and most importantly, USDEUR is traded far less than EURUSD. This means that if a trader wished to trade EUR against the USD, this is easiest via EURUSD as the liquidity is the highest.

In contrast to some other asset classes, the FX market is decentralized so that trading can be performed on multiple venues [28]. Currencies can be traded on multiple venues or even over-the-counter (OTC). It is not to be expected that there are large price differences across multiple venues, as this would otherwise lead to arbitrage opportunities[3]. In this thesis data from a single venue is used, so that we assume that those data represent the market. To support this assumption, due to the increase of volume during the window, interbank spreads reduce significantly [1].

3.2 Electronic trading and limit order book

Modern-day trading is performed on electronic market places, which keep track of all orders for a particular instrument placed on that market. Those orders are placed in the limit order book (LOB) which matches the orders of buyers and sellers of a particular financial instrument. In the following a general description of LOBs is given, which is not restricted to FX. This is a simplified view of order books as in reality LOBs are more complex (which will be elaborated in Section 5.2). The description of LOBs here is brief and only relevant for this work. A more detailed overview is given in [29].

The order book represents a collection of buyers and sellers, ordered by price and time. In Figure 1 a visual representation is given of the order book. An order is defined as a four-tuple (side, volume, price, time), and represents the intention of a trader to buy or sell (which is the side of the order) a certain volume (or quantity) at a certain price. The minimum price difference between two orders is the tick size, and the minimum increment in volume is the lot size.

The lowest ask and the highest bid price (also known as top of the book) are called the best ask (denoted by $p_a^1(t)$) and best bid ($p_b^1(t)$) respectively. The difference between best ask and best bid is the bid-ask spread (or spread), $p_a^1(t) - p_b^1(t)$, and the mean of best bid and best ask is the mid-price $p(t) = (p_b^1(t) + p_a^1(t))/2$. No trade can take place at mid-price, but it is often used to represent the general market value [12]. Away from the best ask and best bid are other levels in the order book, denoted by $p_b^i(t)$ and $p_a^i(t)$ for the i^{th} bid/ask level (or depth) respectively. It holds that $p_a^i(t) > p_a^j(t)$ for $i > j$ and $p_b^i(t) > p_b^j(t)$ for $i < j$. Furthermore, it generally holds that $p_b^1(t) < p_a^1(t)$; i.e. the best bid price is lower than the best ask price and that the spread is positive. The number of levels is not constant in time and can be different for bid and ask.

[3]Without further definition of arbitrage, a difference in price between two venues could lead to a strategy where currency is bought on the cheapest venue and sold on the most expensive at the same time.

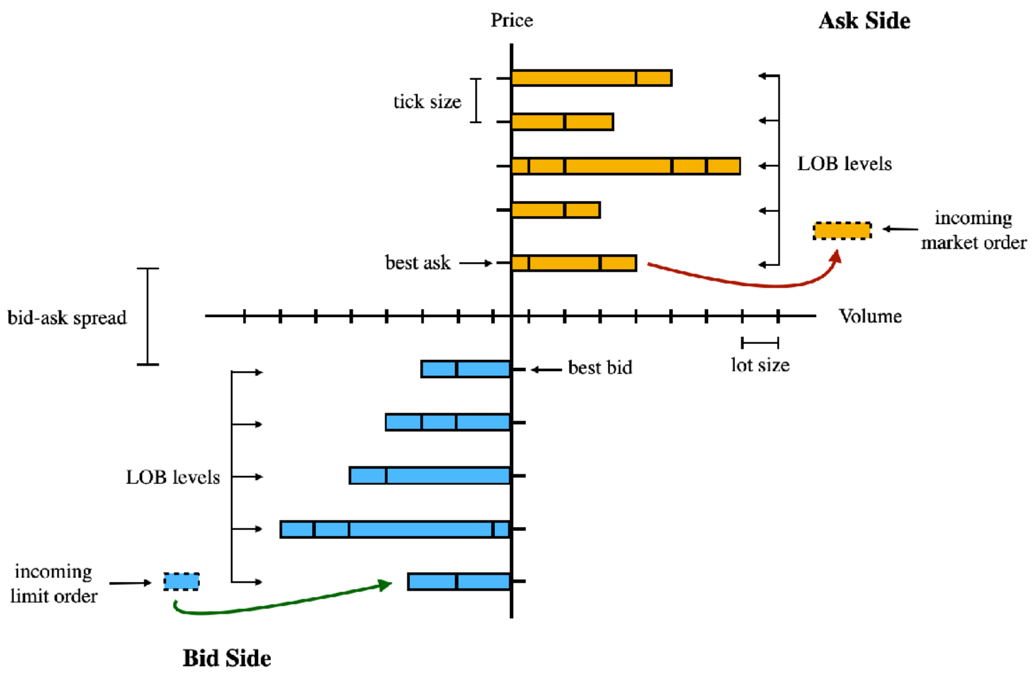


Figure 1: Visualisation of the limit order book. Retrieved from [15].

Orders come in two types: market orders and limit orders. Limit orders are not directly matched to an order on the opposite side, and are therefore placed in the LOB until they are either matched or cancelled. A market order directly matches with an order in the LOB on the opposite site. A limit order which offers a better price than the best price in the order book, is automatically changed to a market order. If two orders (partially) match, then a trade occurs.

Matching of two orders usually follows the price-time priority rule. That is, an order is first matched to the limit order which has the best price (best bid or best ask). If multiple limit orders are open on a price level, then the order which has been placed first in time is matched (the first-in-first-out, FIFO, principle). It could occur that the quantities of two matched orders are different. In case the market order has a larger quantity than the matched limit order, then the remaining quantity is matched with the next available order, again according to price-time priority.

Each trade consists of a market order and a limit order. The side of the market order is called the aggressor side of the trade. In essence, the aggressor pays the price of crossing the spread.

3.2.1 Order book states

At each point in time, the LOB can be represented by a vector representation called the LOB snapshot. As stated earlier, the number of levels is not constant in time nor equal for bid and ask. However, higher levels on either side does not contain as much information on the price direction as lower levels do [30]. For this reason, the level is kept constant and equal for bid and ask, and is denoted by L, and higher levels in the LOB are truncated. The LOB snapshot is then a vector $\mathbf{s}_t \in \mathbb{R}^{4L}$ given by,

$$\mathbf{s}_t = (p_a^1, v_a^1, p_b^1, v_b^1, \ldots, p_a^L, v_a^L, p_b^L, v_b^L)^T, \tag{1}$$

where L is the number of levels in the LOB, $p_a^i = p_a^i(t)$ and $v_a^i = v_a^i(t)$ are the ask price and volume at level i and time t, and $p_b^i = p_b^i(t)$ and $v_b^i = v_b^i(t)$ are the bid price and volume. This representation of the state of the LOB is commonly found in the literature, e.g. [15, 31].

As orders are placed, changes in the order book occur in a discrete sense. This implies that temporally the LOB can be represented in a discrete sense. I.e. s_{t_i} for $i = 1, \ldots, N$, so that $S = s_{i}{}_{i=1}^{N} \in \mathbb{R}^{N\times 4L}$.

What number of levels to use is hard to answer a priori. However, in [30] it is suggested that higher levels in the order book contain less information, and that price discovery can be mostly contributed to the best bid and best ask price. This might be attributed to electronic trading algorithms which submit and cancel vast numbers of limit orders in short periods of time [32]. In [29] it is observed that these order are placed deep in the order book, contributing to noise [12]. These conclusions are, however, for other asset classes than FX and whether they are valid for FX LOBs is uncertain.

3.2.2 Order Flow

LOB snapshots as in Table 4 are generally not stationary [15] (for the definition see for example [33]). A method for compiling input data from the LOB is using order flows, which is a common method to create stationary time series from the non-stationary limit order book states. The bid order flows (bOF) and ask order flows (aOF) are vectors $\mathbf{bOF}_t, \mathbf{aOF}_t \in \mathbb{R}^L$, where each component is given by

$$\mathrm{bOF}_{t,i} = \begin{cases} v_{t,i}^b, & \text{if } p_{t,i}^b > p_{t-1,i}^b, \\ v_{t,i}^b - v_{t-1,i}^b, & \text{if } p_{t,i}^b = p_{t-1,i}^b, \\ -v_{t,i}^b, & \text{if } p_{t,i}^b < p_{t-1,i}^b, \end{cases}$$

$$\mathrm{aOF}_{t,i} = \begin{cases} -v_{t,i}^a, & \text{if } p_{t,i}^a > p_{t-1,i}^a, \\ v_{t,i}^a - v_{t-1,i}^a, & \text{if } p_{t,i}^a = p_{t-1,i}^a, \\ v_{t,i}^a, & \text{if } p_{t,i}^a < p_{t-1,i}^a. \end{cases}$$

From the bid and ask order flow, the order flow (OF) is defined as,

$$\mathbf{OF}_t = \begin{pmatrix} \mathbf{bOF}_t \\ \mathbf{aOF}_t \end{pmatrix} \in \mathbb{R}^{2L}.$$

In [15] it is shown that the performance of "off-the-shelf" artificial neural networks trained on order flows is significantly improved compared to when they are trained on the state of the order book from (1). The transformation to the order flow is a common approach to map the non-stationary time-series of LOB states to stationary time-series [34]. Note that the dimension of the order flow is half of the dimension of the order book state, so that not only the performance is increased, but the training time can also be reduced.

4 WMR Fix

The WM/R 4pm Closing[4] Spot Rate (commonly known as the WMR Fix, the WMR, the 4pm Fix, the close, or the Fix) is a benchmark used by many major financial companies, including institutional investors such as pension funds, to valuate assets in foreign currencies. The Fix is published daily by Refinitiv, which was previously known as the WM/Reuters. It is easily identified as the most important benchmark in the foreign exchange (FX) market [1, 35], which is the most heavily traded market in the world [36]. In this section background information regarding the Fix is given.

Financial benchmarks are important for reference purposes and are widely used as reference rates to settle derivatives contracts, as a fair and transparent price to execute against, and for valuation of portfolios [35]. The WMR Fix was established as key financial benchmark in 1993, when Morgan Stanley Capital International (MSCI) announced they would use it to value foreign assets in the MSCI equity indices [37]. Since then it has been the de-facto standard in the construction of indices consisting of international securities, even though other benchmark rates exist[5], and is used in important indices such as MSCI [40] and FTSE [41].

In 2013, news came out that major banks in the FX market colluded by sharing order information ahead of the Fix. This allowed them to manipulate the fixing price away from the prevailing market price [42]. After this scandal changes were made to the fix methodology in order to improve robustness and decrease the susceptibility to manipulation[6]. These changes included the expansion of the time window during which the Fix is calculated from 1 minute to 5 minutes, and the sourcing of currency data from multiple sources. An account of this and a research into the effectiveness is given in [35] and [42].

4.1 WMR Methodology

On a daily basis Refinitiv publishes fixing rates for three FX products: spot, forwards and non-deliverable forwards. Here the focus is solely on spot rates[7]. In total WMR provides benchmarks for 157 currency pairs. The currencies are divided in trade and non-trade currencies. Markets between trade currencies are more liquid, and focus here is on trade currencies, which is relevant for the EURUSD.

The methodology is shortly described here. However it contains a number of caveats (e.g. insufficient amount of trades) which are not elaborated on. The full description of the methodology can be found in [39] and a short description can be found in [43].

Every second between 15:57:30 LN[8] to 16:02:30 LN, executed trades and bid and offer order rates are sampled from multiple venues[9]. The frequencies of incoming trades and orders are generally higher than the rate of sampling, so that not every trade or offer is captured.

Every captured trade is identified as either a bid or offer depending on the aggressor side of the trade (buy or sell). The opposing side is then determined from the sampled order at that time, given an 'opposing trade'. This gives a set of bid trades and a set of offer trades at each second (containing data from all venues). For both sets, the median is taken resulting in a median bid and offer trade. The mean of these two is the WMR Fix. In figure 2, a schematic representation of the methodology is shown.

[4]The name can be somewhat misleading as currency markets have no close and are open from 17:00 NY (New York time) on Sunday to 17:00 NY on Friday.

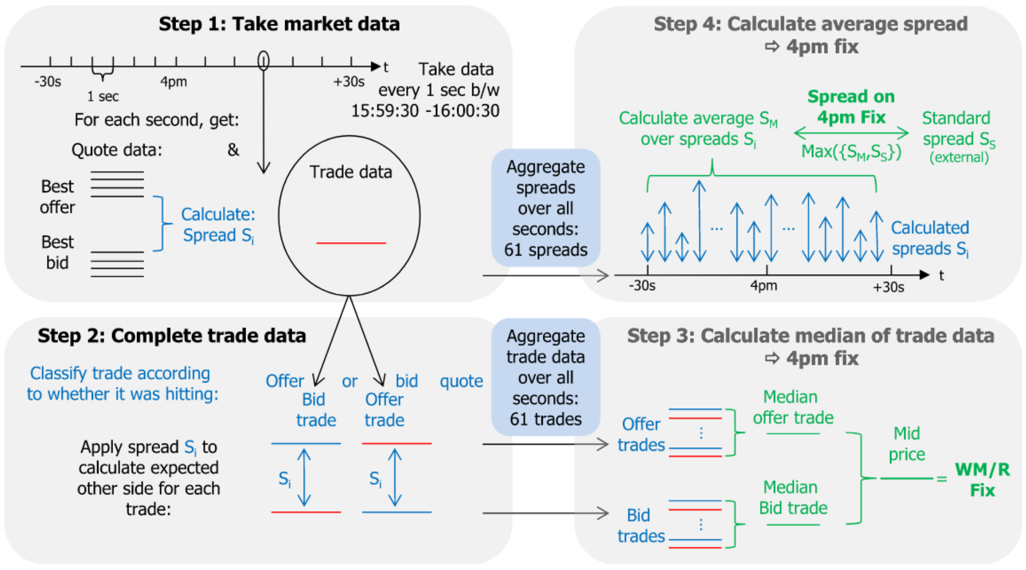
[5]For example the Bloomberg currency indices (BFIX) [38], the SIREN benchmark [], and benchmarks published by Refinitiv at other times than 16:00 [39]

[6][35] gives an account of desirable properties of benchmarks.

[7]Spot transactions have "immediate" settlement, i.e. one or two days [**donnely**]

[8]LN is used to indicate London time.

[9]For EURUSD these are EBS, Currenex and Refinitiv Matching.



Step 1: Take market data
-30s
1 sec
4pm
+30s
t
Take data every 1 sec b/w 15:59:30 -16:00:30
For each second, get:
Quote data:
&
Best offer
Best bid
Calculate: Spread S_i
Trade data
Aggregate spreads over all seconds: 61 spreads
Step 4: Calculate average spread ⇨ 4pm fix
Calculate average S_M over spreads S_i
Spread on 4pm Fix
Max({S_M,S_S})
Standard spread S_S (external)
Calculated spreads S_i
-30s
4pm
+30s
t
Step 2: Complete trade data
Classify trade according to whether it was hitting:
Offer or bid quote
Bid trade
Offer trade
Apply spread S_i to calculate expected other side for each trade:
S_i
S_i
Aggregate trade data over all seconds: 61 trades
Step 3: Calculate median of trade data ⇨ 4pm fix
Offer trades
Median offer trade
Mid price
WM/R Fix
Bid trades
Median Bid trade

Figure 2: Overview of the methodology of the Fix, retrieved from [43]. Note that the window is 1 minute, which was before the reforms, and is now 5 minutes.

There are a number of caveats in the calculation of the WMR, for example if no trade has occurred during the second interval. In certain cases expert judgement is applied to establish the WMR. For the interested reader in the caveats we refer to the methodology guide [39].

4.2 Market dynamics around the WMR Fix

Because many market participants wish to execute at the fix, for example to minimize tracking errors, volume and liquidity during the window is generally high. The large liquidity also attracts other market participants, such as high-frequency traders, increasing the complexity of the market dynamics [35]. As volume flows during the window could be described as risk minimizing and less driven by price-level these might result in inefficiencies. In this section, the market dynamics around the WMR Fix is elaborated on. Even though it might be hard to exploit these insights using machine learning models, insights in the market dynamics is useful in understanding the background of the thesis problem.

Previous analysis has shown that market dynamics around the fix might exhibit some predictability [9, 10, 11]. Reasons for this might be found in the currency hedging of foreign assets by institutional investors, an effect predominantly visible at month-end. For example, European investors with US equities in their portfolios tend to hedge most of their currency risk [3]. The hedges of their American counterparts tend to be lower, and European investors generally have larger investments in the US than vice-versa. Therefore, a change in US equities can generate a flow in a certain direction, caused by the hedge rebalances. The implicit assumption in this thesis is that these effects manifest themselves in the FX data. We aim to use these effects, and to find the auto-correlations in the data for price discovery.

To understand the market dynamics during the fix and to see if they correspond to the literature, some statistics of the order book over time are shown for the EURUSD. The market dynamics are extracted from the data which is described in Section 5. Since the Fix is calculated for every trading day, essentially a realization of a time-series is available for multiple days. From this statistics can be aggregated by time of day. To this end, suppose at day $d = 1, \ldots, D$ and time t we have some realized value $x_{d,t}$. From this, the mean and variance over multiple days can be calculated by time of day,

$$\mu_t^x = \frac{1}{D}\sum_{d=1}^{D} x_{d,t},$$
$$\sigma_t^x = \frac{1}{D-1}\sum_{d=1}^{D} \left(x_{d,t} - \mu_t^x\right)^2 .$$

This is calculated for various statistics extracted from the order book, and shown in the figures below. In all cases the data is taken from all trading days from November 2nd 2020 to October 29th 2021.

In figures 3 and 4, $x_{d,t}$ is taken to be the number of events (new orders and cancellations) in the LOB during a ten second window. We can see that during the WMR window (indicated by the vertical lines), both the mean and variance increase. Before the window, there is already a gradual increase, but for both the mean and variance it decreases sharply after the window.

In Figures 5 and 6 the mean spread during each ten second interval is considered. We can observe that the average spread decreases during the window, and that the variance slightly decreases. In contrast to the event count, the mean spread does not increase significantly before the window.

In Figures 7 and 8 the volatility (which is calculated as the standard deviation of the returns) is shown. An increase before the window in both the mean and variance can be observed with their peaks around the beginning of the window.

In Figures 9 and 10 the mean and variance of the numbers of trades by time of day is shown. Similarly to the number of event, there is a sharp increase in trading during the window. However, unlike the event count, the mean number of trades hardly increases in the time just before the window. This could be an indication of orders placed at higher levels in the order book in anticipation of the direction of the price movement [29]. Similarly, an increase in the variance can be observed during the window.

In Figures 11 and 12 the mean and the variance of the total quantity is shown. It can be seen that both the mean and variance of the total quantity increase similarly to the number of trades.

Lastly, in Figures 13 and 14 the mean and variance of the mean trade quantity during the ten second window is shown. The mean does not show a clear change during the window, except for a noticeable decrease at the beginning and end of the window. The variance shows a clear decrease during the window, which could indicate an increase in the use of execution algorithms.

Summarizing Figures 3-14, a clear effect in market dynamics during and around the WMR window can be observed. Even though the increase in volatility can be unappealing to risk-averse traders, the market during the window can adsorb large amounts of volume so that the effects of market impact is reduced. The observations in this section are in line with [9].

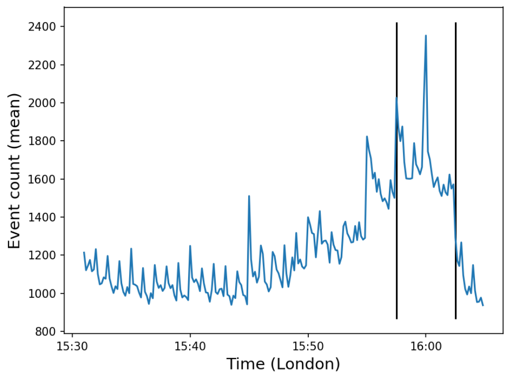


Figure 3: Event count (number of events in the order book) in ten seconds intervals, with the mean taken over twelve months of trading days.

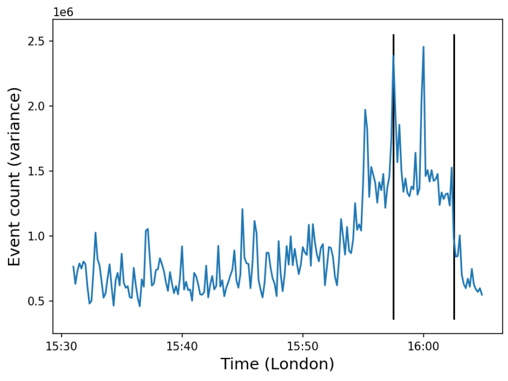


Figure 4: Event count (number of events in the order book) in ten seconds intervals, with the variance taken over twelve months of trading days.

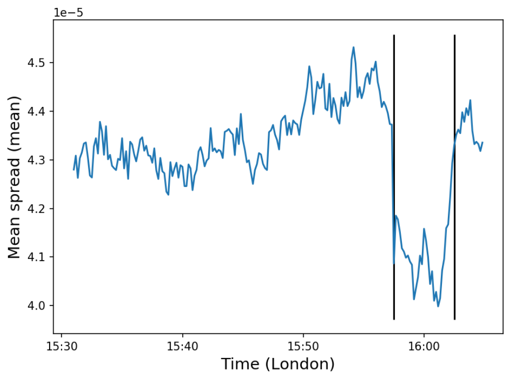


Figure 5: Mean spread in ten seconds intervals, with the mean taken over twelve months of trading days.

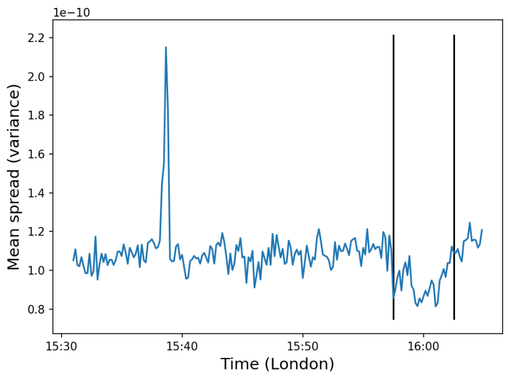


Figure 6: Mean spread in ten seconds intervals, with the variance taken over twelve months of trading days.

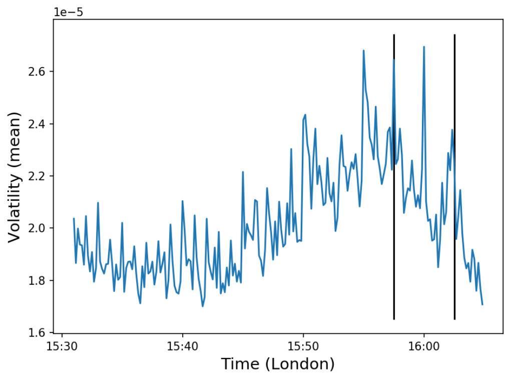


Figure 7: Volatility in ten seconds intervals, with the mean taken over twelve months of trading days.

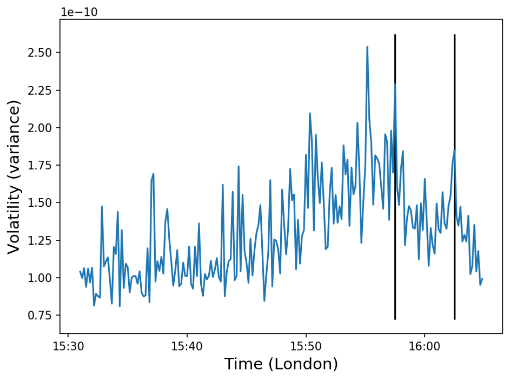


Figure 8: Volatility in ten seconds intervals, with the variance taken over twelve months of trading days.

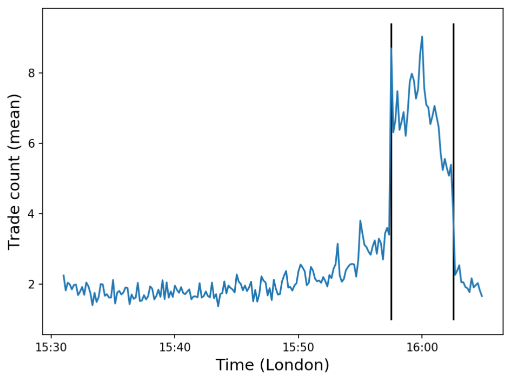


Figure 9: Trade count in ten seconds intervals, with the mean taken over twelve months of trading days.

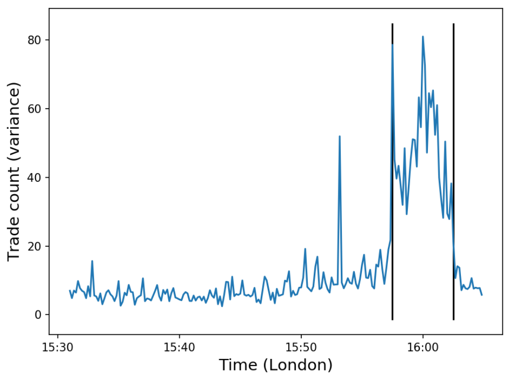


Figure 10: Trade count in ten seconds intervals, with the variance taken over twelve months of trading days.

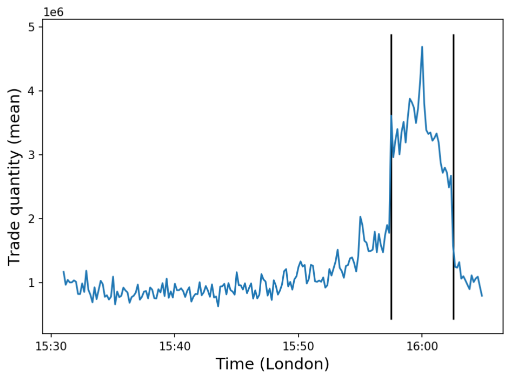


Figure 11: Mean of the total trade quantity in ten seconds intervals, with the mean taken over twelve months of trading days.

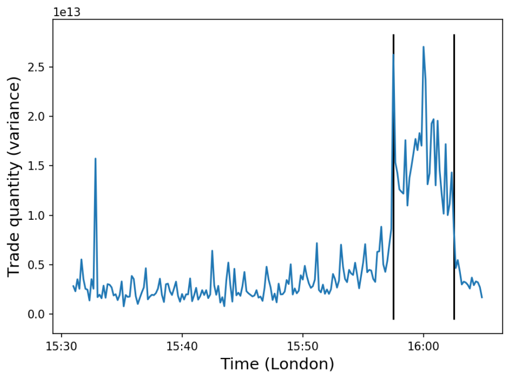


Figure 12: Mean of the total trade quantity in ten seconds intervals, with the variance taken over twelve months of trading days.

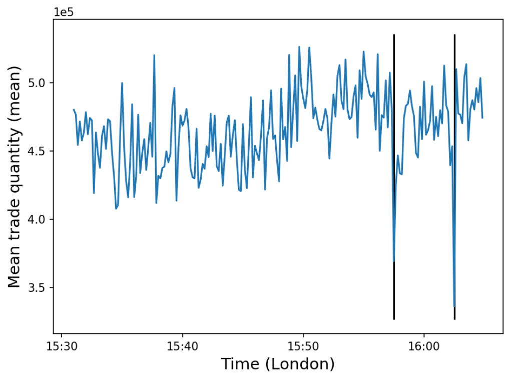


Figure 13: Mean trade quantity in ten seconds intervals, with the mean taken over twelve months of trading days.

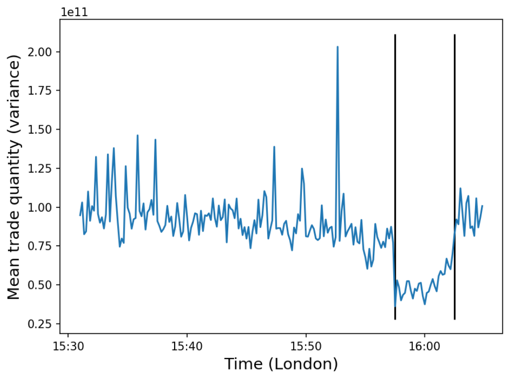


Figure 14: Mean trade quantity in ten seconds intervals, with the variance taken over twelve months of trading days.

4.3 WMR fix under price dynamics

Since the fix methodology is somewhat complicated we aim to provide a simplified model to estimate the fix given all the information available at that point in time. Besides, the fact that the WMR is calculated from data sources other than the data sources used here (and will be elaborated on in Section 5) and that expert judgement is applied in certain cases, make it hard to exactly replicate the WMR. Lastly, the value of the fix is published several minutes after the calculation has ended. A simplification can be sufficient for practical use while only making a small error.

Throughout this section an underlying probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is assumed. Suppose that the mid-price S_k at time $t_k,\ k = 0, \dots, N$ follows a discrete stochastic process,

$$S_k = S_{k-1} + \sigma\sqrt{\tau_k}\xi_k, \tag{2}$$

where σ is the volatility, $\tau = t_k - t_{k-1}$ and $\xi_k \sim \mathcal{N}(0,1)$ i.i.d., and we denote its natural filtration by $\{\mathcal{F}_n, n \in \mathbb{N}\}$. As seen before, the volatility is not constant during the window, but the approximation later on will be independent of the volatility. Suppose that the benchmark $\bar{S}_{m,l}$ (which is calculated between t_m and t_l) is a function of S_k, $k = m, \dots, l$,

$$\bar{S}_{m,l} = f(S_k, k = m, \dots, l).$$

We take $\bar{S}_{m,l}$ to be the time-weighted average from time $t_m, \dots, t_l$,

$$\bar{S}_{m,l} = \frac{1}{l - m + 1} \sum_{k=m}^{l} S_k. \tag{3}$$

Then the conditional expectation of $\bar{S}_{m,l}$ given $\mathcal{F}_n$ at time t_n is given by,

$$\mu_n = \mathbb{E}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) = \begin{cases} S_n & n < m, \\ \frac{1}{l-m+1}\sum_{k=m}^{n} S_k + \frac{l-n}{l-m+1} S_n & m \le n \le l, \\ \frac{1}{l-m+1}\sum_{k=m}^{l} S_k & l < n. \end{cases} \tag{4}$$

For the variance we have

$$\sigma_n^2 = \mathbb{V}\text{ar}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) = \begin{cases} -\sigma^2\tau n + K_1 & n < m, \\ -\sigma^2\tau\left(\beta_1 n^3 + \beta_2 n^2 + \beta_3 n\right) + K_2 & m \le n \le l, \\ 0 & n > l. \end{cases}$$

A derivation of the conditional expectation and the variance is given in Appendix A. Note that the conditional distribution $\bar{S}_{m,l}|\mathcal{F}_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$, and is dependent on n. Under these simple price dynamics, the process $X_n = \bar{S}_{m,l}|\mathcal{F}_n$ is not a stationary process (for the definition see [33]).

The volatility can be estimated (e.g. by means of maximum likelihood), but this is not the aim of this derivation. The main takeaway is that the variance decreases linearly before the start of the window, and cubic in the window. This insight in the uncertainty of the estimation will be used later on when analysing the results of the machine learning predictions. Note that the conditional expectation holds if the stochastic process of S_k is a martingale[10]. In case the volatility is non-constant, the expected value still holds but the variance will be different.

[10]Exchange rates are often described using mean reverting processes [44]. On short time scales the drift can thus be considered small, justifying the use of martingales.

This estimate can also be done in a continuous-time setting,

$$\mathrm{d}S_t = \bar{\mu}(S_t, t)\mathrm{d}t + \bar{\sigma}(S_t, t)\mathrm{d}W_t,$$

where S_t follows an Itô process [45] with drift $\bar{\mu}$ and volatility $\bar{\sigma}$, and S_0 given, and we take the estimate to be,

$$\bar{S}_{a,b} = \frac{1}{b-a}\int_a^b S_t \ \mathrm{d}t.$$

In case the drift term $\bar{\mu}$ is zero, then the process is a martingale with respect to $\mathcal{F}_t$. Then the continuous case yields a comparable result for the estimate,

$$\mathbb{E}\left(\bar{S}_{a,b}|\mathcal{F}_t\right) = \frac{1}{b-a}\left(\int_a^t S_\tau \ \mathrm{d}\tau + (b-t)S_t\right).$$

However, given the discrete nature of forecasting time-series using machine learning models, the discrete setting is chosen.

The expected value $\mathbb{E}\left(\bar{S}_{m,l}|\mathcal{F}_n\right)$ is called the naive estimate of the WMR at time t_n, and is valid as long as the underlying process is a martingale. Note from the discussion above that the naive estimate is not a stationary process. Figure 15 shows an example of the naive estimate, taken from the EURUSD on October 29th, 2021.

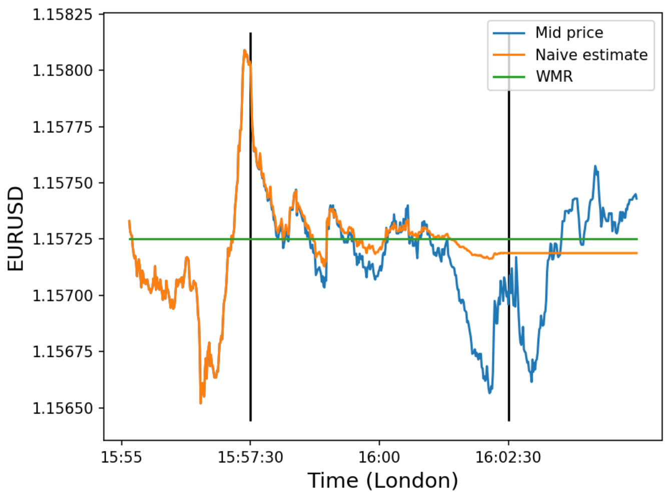


Figure 15: Plot of the naive estimate of the WMR, along with the mid-price of EURUSD. Before 15:57:30 the naive estimate and the mid-price coincide.

In the use of the approximation, it is necessary to get an idea on the error this gives. In Figure 16 the histogram is shown of the error between the WMR approximation and the value of the WMR. The error in this case is the difference between the approximation and the WMR, and is expressed in pips (for EURUSD one pip equals \$0.0001 [**donnely**]). The approximation in this case is done by taking the average of the mid-price every second from 15:57:30 LN to 16:02:30 LN. This is done for each day data is available between November 2nd, 2020 and October 29th, 2021. The error is generally small and in nearly all observations less than 1 pip. Given that this is expected to be less than the uncertainty in the prediction of the WMR, we believe that the use of the approximation is justified.

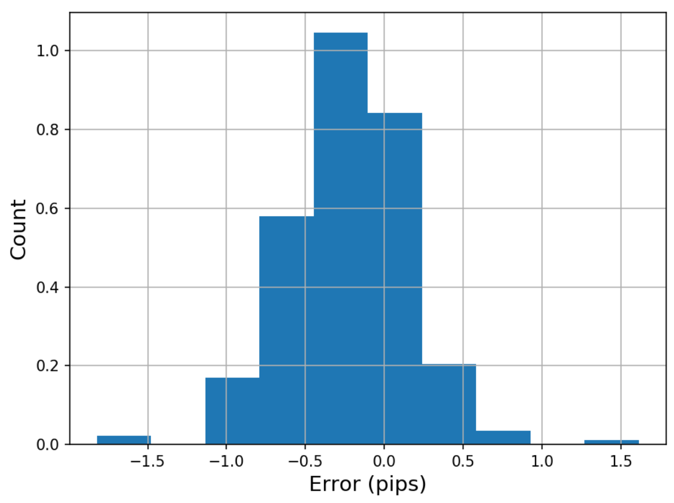


Figure 16: Histogram of the error of the WMR approximation in pips.

5 Raw Data

This section describes the raw data used in the predictions by the machine learning models. Furthermore, it describes the pre-processing of the data which is necessary to be used as input.

5.1 WMR Fix data

The value of the WMR Fix for each day is available for the same period as the LOB data. This data is obtained through Refinitiv. Table 1 shows a sample of the value of the EURUSD Fix for several days.

Table 1: Sample of the EURUSD Fix for several days.

Date	EURUSD
13-09-2021	1.18100
14-09-2021	1.18215
15-09-2021	1.18190
16-09-2021	1.17635
17-09-2021	1.17370
20-09-2021	1.17250
21-09-2021	1.17245
22-09-2021	1.17335
23-09-2021	1.17495
24-09-2021	1.17135
27-09-2021	1.17005

5.2 Limit order book data

Generally, LOB data comes in three degrees of granularity [31] with L1 data providing the best bid/ask price and volumes, L2 the prices and volumes at all price levels, and L3 containing the non-aggregated orders placed by market participants.

L3 LOB data from Cboe is available for the Euro-US Dollar exchange rate. A description of the data can be found in [46]. EURUSD is chosen as this currency pair has the largest share in FX spot trading at 24% in 2019 [2]. The data is available for a time period from November 2nd 2020 to October 29th 2021. The total size of the data is 142.5 GB, however not all data is used as the focus is specifically on times around the Fix.

Two types of data is available, the limit order book event history, and trade data. From the former the order book can be reconstructed at any point in time. The time stamps of the raw data is in New York time, and is converted to London time[11]. The tick size of EURUSD Cboe is equal to \$0.00001 (0.1 pip).

Of the event history a sample is shown in Table 2. Each row represents an update in the orderbook, which can be any of four types:

- N: new order;
- C: cancellation of an order;
- S: snapshot;
- M: modification of an order.

[11]London and New York do not change between summer and winter time on the same date, so that the conversion between the time zones has to be done carefully.

A new order consists of the side of the order (1 is bid, 2 is ask), the price, and quantity. Every order is identified by a unique identifier. A cancellation only consists of the identifier, which matches with either an S or N. The side, price, and quantity of the cancellation can be found by matching it with the corresponding S or N. It is important to note that a cancellation in the data is not necessarily the cancellation of the corresponding order, as it could also indicate that that order has been matched. A snapshot S comes in groups, and that group gives the state of the order book. A snapshot is similar to a new order, but when a group of snapshots is encountered, the current state of the order book is discarded. Lastly, a modification indicates a change in quantity for a given order. This type however has not been observed in the data. The number of new events as plotted in Figure 3 is calculated form the event history (in this snapshots are not considered new events). Furthermore, Figures 4-8 are calculated from the event history.

Table 2: Sample of the event history of the EURUSD on 27-07-2021.

Time (NY)	EventIdentifier	Type	Bid/Ask	Price	Quantity
15:00:14.301	49773689	N	2	1.18219	1000000
15:00:14.312	49773688	C			
15:00:14.312	49773699	N	1	1.18216	1000000
15:00:14.312	49773689	C			
15:00:14.312	49773700	N	2	1.18221	1000000
15:00:14.368	49773304	C			
15:00:14.368	49773726	N	2	1.18225	1000000
15:00:14.368	49773401	C			
15:00:14.368	49773480	C			

Next to the event history, the trade data is available, of which a sample is shown in Table 3. This data consists of actual trades which have been executed. Each row contains the price and quantity of a trade together with the aggressor side. Trade data corresponds to changes in the limit order book, but could differ from the mid-price at any given point in time as no trading occurs. Note that the frequency of updates in the order book event history is higher than the trade history. Besides the traded quantities are different from the quantities in the LOB, which are usually multiples of 1000000. Figures 9-14 are calculated from the trade data.

Table 3: Sample of the trade history of the EURUSD on 27-07-2021.

Time (NY)	Side	Price	Quantity
15:00:12.789	B	1.18220	437000
15:00:16.176	B	1.18222	100000
15:00:22.622	B	1.18222	100000
15:00:22.709	B	1.18223	100000
15:00:24.656	B	1.18223	5000
15:00:40.225	S	1.18220	225955
15:00:43.526	B	1.18222	500000
15:00:47.002	S	1.18219	232598

As mentioned in Section 3.2, in practice LOBs are more complex, which is mostly due to rules instituted by venues. For instance, Cboe offers customized liquidity to market participants so that certain participants are excluded from trading with each other (e.g. due to credit limits). This means that the top of the book (the best bid and ask prices) can become inaccessible to other market participants. Among other things, this implies that spreads can become negative, and trades can occur at higher levels in the order book.

Next to the customized liquidity, it can happen that trades occur which were not listed in the order book at all. This is due to the fact that Cboe offers hidden resting orders. Furthermore, if a market order matches a limit order in the LOB, it is removed from the LOB but does not appear directly in the trade history. This is caused by last look, which is the option for a market participant to accept the trade within 70 milliseconds. Only after acceptance the trade is placed in the list of trades.

5.3 Data processing

Before the machine learning models are trained, prepossessing of the raw data needs to be performed. The reason for this is twofold. Firstly, the machine learning models used are sensitive to the scaling of the input features. Secondly, data needs to be resampled as long input sequences make the machine learning models computationally heavy.

Firstly, event history data needs to be transformed to the LOB snapshot format from 1. A sample of this is shown in Table 4, and in Figure 17 a visualisation at a single point in time is shown for 15 levels, which is similar to the visualisation of the order book from Figure 1. As can be seen from Table 2, the time stamp for an event is not unique. All events with the same time stamp are aggregated in a single snapshot. This means that in each state of the order book as in Table 4 multiple changes in the order book can occur.

Table 4: Sample of the state of the LOB with three levels on July 27, 2021. All volumes are multiples of 10^6.

	p_b^1	v_b^1	p_b^2	v_b^2	p_b^3	v_b^3	p_a^1	v_a^1	p_a^2	v_a^2	p_a^3	v_a^3
15:31:08.400	1.18091	1	1.18089	2	1.18088	4	1.18095	0.5	1.18096	2	1.18097	5
15:31:08.408	1.18091	1	1.18089	2	1.18088	4	1.18095	0.5	1.18096	2	1.18097	4
15:31:08.451	1.18091	1	1.18089	2	1.18088	4	1.18095	0.5	1.18096	2	1.18097	3
15:31:08.469	1.18091	1	1.18089	1	1.18088	5	1.18095	0.5	1.18096	2	1.18097	3
15:31:08.472	1.18091	1	1.18089	1	1.18088	5	1.18095	0.5	1.18096	2	1.18097	4
15:31:08.480	1.18089	2	1.18088	5	1.18087	2	1.18095	0.5	1.18096	3	1.18097	4
15:31:08.488	1.18089	2	1.18088	5	1.18087	2	1.18095	0.5	1.18096	3	1.18097	4.5
15:31:08.493	1.18089	2	1.18088	5	1.18087	2	1.18095	0.5	1.18096	3	1.18097	5
15:31:08.500	1.18089	1	1.18088	4	1.18087	4	1.18095	0.5	1.18096	3	1.18097	5
15:31:08.570	1.18089	2	1.18088	4	1.18087	3	1.18095	0.5	1.18096	3	1.18097	5

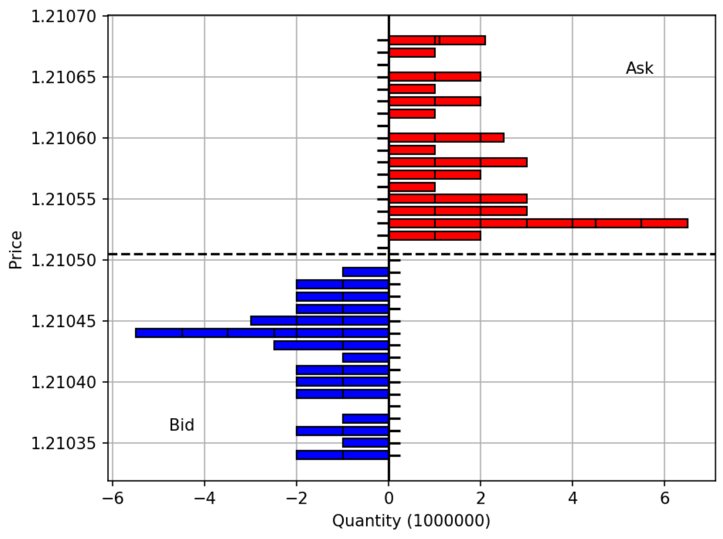


Figure 17: Visualisation of the limit order book (with 15 levels on either side) from the Cboe data on January 20, 2021, 15:58:30 LN.

The LOB representation of Table 4 is the basic level of input to the models. However, in certain cases we might want to add additional features to the data in order to improve predictive performance. These features can be a direct function of the raw data (e.g. volatility of the mid-price for the past 10 seconds or essentially any technical indicator), or a combination of multiple data sources (e.g. combination of the event and trade data). This is commonly referred to as feature engineering.

However, one of the big strengths of machine learning is the fact that it obviates the need of domain knowledge in the construction of the models. It aims to learn the patterns in the raw data, without general information on the data. For this reason, this is not done (with a single exception).

Added features to the state of the LOB used in the results of this thesis are:

1. Naive estimate (Section 4.3);
2. Time until 16:02:30;
3. last traded price;
4. last traded quantity;
5. aggressor side of last trade.

The naive estimate is the only input feature which is a function of the LOB data. Although a lot of features can be added, it is limited in this thesis to just these. In the results the input features is varied in order to determine which hold most predictive performance.

After all features are present the data is resampled. Resampling is necessary as long input sequences can make the machine learning models too computationally heavy. Moreover, update in the LOB on millisecond time scales might not provide enough additional information to improve the predictive power. Even though resampling is performed, it essentially is an ad hoc step in the preparation of the data, and there is no canonical way to do so. It is hard to say what the effect is of resampling on the predictions, and what the most optimal approach is to resampling.

After resampling the data is normalized using the rolling mean and standard deviation, similarly to [12]. Data normalisation is needed as the machine learning models used are sensitive to the scaling of the different input features. The choice of rolling mean and standard deviation comes from the fact that using static normalisation would imply that the data from multiple days is used for normalisation and the intraday mean and standard deviation can vary significantly from that of multiple days. Arguably, normalisation is the weakest spot in the preparation of the data, and the choice of window lack is hard to provide a solid foundation for.

6 Models

In this section the artificial neural networks (ANN) used in this thesis are described. In particular the focus will be on recurrent neural networks (RNN). First a general introduction to RNNs will be given, including their main assumptions and the most important paradigms.

Broadly speaking, in order to reach conclusions from data two groups can be distinguished [47]. One assumes the data is generated by an underlying stochastic model, the other treats the data-generating mechanism as unknown. The latter is algorithmic in nature and forms the basis of machine learning.

The strength in using machine learning models is that they are agnostic on the structure of the data they are trained on, i.e. no domain knowledge is assumed. This allows the models to be general and complex, and avoids the need of assuming any stochastic process which generates the data. This is considered one of the great strengths of machine learning.

The essence of this section can briefly be summarized. The data is represented by $D = \{(\mathbf{x}_t, \mathbf{y}_t),\ t = 1, \ldots, T\}$, with $\mathbf{x}_t \in \mathbb{R}^P$, $\mathbf{y}_t \in \mathbb{R}^M$, and is assumed to be observations of an autocorrelated process. The goal is to construct a predictor $\hat{\mathbf{y}}_{t+h}$ at a horizon of h steps, of a response $\mathbf{y}_{t+h}$ given N past observations,

$$\hat{y}_{t+h} = f(\mathcal{X}_t; \boldsymbol{\theta}),\ \mathcal{X}_t = (\mathbf{x}_{t-N+1}, \ldots, \mathbf{x}_t),$$

with $f : \mathbb{R}^{T \times P} \to \mathbb{R}^M$ some (non-linear) function parameterised by $\boldsymbol{\theta}$. In this way, implicitly the joint distribution of the in- and output is modelled, as well as the autocovariance of the data. This is very flexible as no assumptions are made regarding the distribution of the input or output.

6.1 Machine learning paradigms

Machine learning is a broad term, and covers various classes of algorithms. The focus in this work will be on supervised learning. In this we are given pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$, with $\mathbf{x}_i \in X$ and $\mathbf{y}_i \in Y$, and the goal is to find the relationship between X and Y. Each element $\mathbf{x}_i \in X$ is referred to as a feature (vector), with $\mathbf{y}_i$ the corresponding label or response.

Supervised learning addresses a fundamental prediction problem, namely the construction of a non-linear predictor $\hat{f}(\cdot)$ of an output $\mathbf{y}_i \in Y$ given an input $\mathbf{x}_i \in X$. In this it is assumed that X and Y are related through a "true" function f with an added noise term $\boldsymbol{\varepsilon}$,

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\varepsilon}. \tag{5}$$

Supervised learning can be split in two sorts, regression and classification. In regression the label is a continuous variable, $Y \subset \mathbb{R}^M$, whereas in classification it is discrete. In the latter case we denote the output space by $G \subset \mathbb{N}^M$, and the predictor by $\hat{g}$. As an example for classification, suppose that a response can be either of K categories so that $G = \{0, \ldots, K-1\}$.

In the case of classification it is usual to represent a label as a vector of zeros and a one in the k^{th} place, called one-hot encoding [33]. Then if we have a predictor $\hat{g}(x; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ with $\hat{g} : X \to [0, 1]^K$, then we can interpret the output at the k^{th} index as the probability of the corresponding category given x,

$$\hat{g}_k(x; \boldsymbol{\theta}) = \mathbb{P}(g = k | X, \boldsymbol{\theta}).$$

In general it is hard to determine whether a predictor performs well. As there is no explicit assumption of the distribution of the data, traditional performance measures, such as likelihood, p-values, or AIC, cannot be used [33]. However, for regression some insight can be gained by assuming that the noise ε from Equation (5) has zero mean and variance σ^2. The expected value[12] of the squared error, $(y - \hat{f})^2$, made by the predictor can be decomposed in three terms,

$$\mathbb{E}\left((y - \hat{f})^2\right) = \underbrace{\left(f - \mathbb{E}\left(\hat{f}\right)\right)^2}_{\text{Bias}(\hat{f})^2} + \sigma^2 + \mathbb{V}\text{ar}\left(\hat{f}\right) \tag{6}$$

where the first term is the square of the predictor bias, the second is the variance of the noise, and the last term is the variance of the predictor. Note that the lower bound of the expected value is given by the variance of the noise. Furthermore, the other two terms constitute what is called the bias-variance trade-off. Bias can occur if a predictor misses relevant structures in the data and is commonly associated with underfitting. On the other hand high variance of the predictor might be the results of overfitting, so that the predictor essentially models the noise. In case of overfitting the model performs well the data is trained on, but lacks performance on out-of-sample data.

In supervised learning, the dataset is split in three groups, a training, validation, and test set. The model is fitted to the training set by minimizing some loss function. The validation set is used to valuate performance of the model during training, i.e. to check whether the loss evaluated on the validation set decreases similarly to the loss of the training set. Validation is important in order to avoid overfitting, since no improvement of the validation loss, but a decrease of the training loss signals overfitting. Lastly the test set is used to validate the best model's performance on unseen data.

6.2 Recurrent neural networks

Recurrent neural networks (RNNs) are part of the artificial neural network (ANN) family. RNNs are designed to explicitly handle temporal dependencies in sequential data [15]. Although only derived methods from RNNs are used in this thesis, it is useful to consider their construction as it provides some insight.

General recurrent neural networks originate in feedforward neural networks (FFNN). Feedforward neural networks are universal approximators [48], meaning they can essentially approximate any continuous function over the input space arbitrarily closely given that the number of hidden units is large enough (there are some technicalities, for this see [33]). Since FFNNs are can approximate any function, this raises the question why other types of networks exist. The answer can be found in efficiency and an improvement on fit. In case structures in data can be exploited, the number of weights can be decreased resulting in less susceptibility to over-fitting, and a reduction in training time. Potentially, the most important class of neural networks for time series is the RNN.

General RNNs construct f by introducing a hidden state $\mathbf{z}_t$ at each time step as a function of the previous hidden state $\mathbf{z}_{t-1}$ and the input at t, $\mathbf{x}_t$,

$$\mathbf{z}_t = g(\mathbf{z}_{t-1}, \mathbf{x}_t), \tag{7}$$

where g is a non-linear function, and $\mathbf{z}_t \in \mathbb{R}^H$. The initial hidden state $\mathbf{z}_{t-N} = 0$. The response is a function of the last hidden state,

$$\hat{\mathbf{y}}_{t+h} = f(\mathbf{z}_t).$$

In Figure 18 a graphical representation of RNNs is shown.

[12]Implicitly we assume an underlying probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

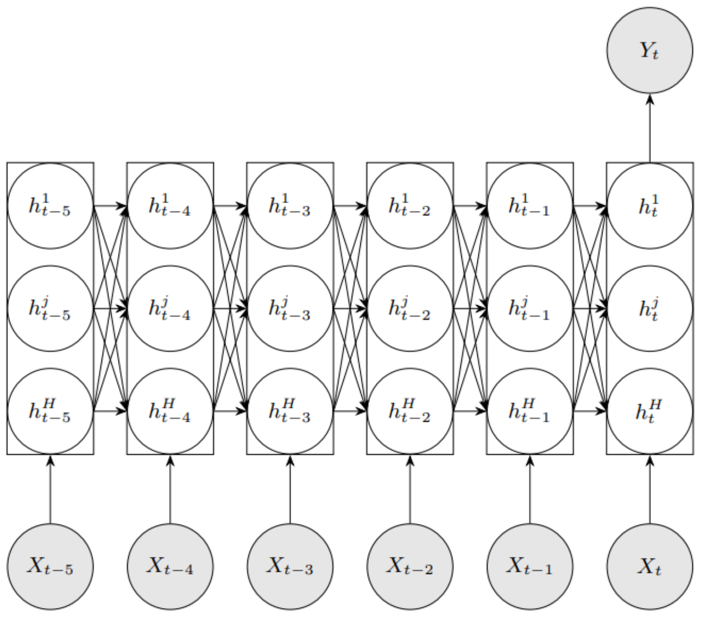


Figure 18: Graphical representation of Recurrent Neural Networks. The notation is slightly different, and the hidden units are denoted by h. Retrieved from [24]

In the most simple form of a RNN, at each time step the hidden state $\mathbf{z}_t$ is a semi-affine transformation of the previous affine transformation $\mathbf{z}_{t-1}$ and the input $\mathbf{x}_t$. The output is given by a semi-affine transformation of the last state. This yields the following set of equations,

$$\begin{aligned}\hat{\mathbf{y}}_{t+h} &= \sigma^{(2)}\left(W^{(2)}\mathbf{z}_t + \mathbf{b}^{(2)}\right)\\ \mathbf{z}_t &= \sigma^{(1)}\left(U^{(1)}\mathbf{x}_t + W^{(1)}\mathbf{z}_{t-1} + \mathbf{b}^{(1)}\right),\end{aligned}$$

where $\sigma^{(1)}$ and $\sigma^{(2)}$ are the non-linear activation functions, $W_x^{(1)}$, $W_z^{(1)}$, and $W^{(2)}$ are the weight matrices, and $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ are the bias vectors.

In case of regression, the activation function of the last state $\sigma^{(2)}$ is the identity function. For classification, $\sigma^{(2)}$ can be any monotone increasing function with $\lim_{x\to-\infty}\sigma^{(2)}(x) = 0$, and $\lim_{x\to\infty}\sigma^{(2)}(x) = 1$. Typical choices for the activation functions are the hyperbolic tangent, the sigmoid function ($(1+e^{-x})^{-1}$), the ReLu function ($\max(0, x)$), and the softmax function $\sigma_s : K \to [0, 1]^K$ defined as,

$$\sigma_s(\mathbf{x})_k = \frac{\exp(x_k)}{\|\exp(\mathbf{x})\|_1},\ k \in \{1, \dots, K\},$$

where $\|\cdot\|_1$ is the 1-norm. The softmax function is used in the classification problem for $\sigma^{(2)}$. For the recurrent activation function $\sigma^{(1)}$ the sigmoid function is used.

In the RNN the level of non-linearity is determined by the number of hidden layers H, and should be at least the dimensionality of $\mathbf{x}_t$ [33].

The RNN can be considered in certain cases to be a non-linear generalization to autoregressive (AR) processes. This is apparent if the activation functions are linear, in case the output becomes a linear combination of the input.

6.2.1 LSTM

Even though the main aim of RNNs is to take into account long-term dependencies, training those dependencies in RNNs is quite hard due to the vanishing gradient problem [49, 50]. This deficiency is overcome by a generalization of RNNs called LSTM. Long short-term memory (LSTM) is an extension to the standard RNN introduced by Hochreiter et al. [51]. This is done by introducing a cell memory $\mathbf{s}_t$ next to the hidden states, which represents the state of the LSTM over time. The memory is input to non-linear gates which determine the information flow. LSTMss have been found to be able to forecast high-frequency returns using order book data [22, 15].

In LSTMs each layer (see Equation (7)) is given by,

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(U^f \mathbf{x}_t + W^f \mathbf{z}_{t-1} + \mathbf{b}^f\right) \\
\mathbf{i}_t &= \sigma\left(U^i \mathbf{x}_t + W^i \mathbf{z}_{t-1} + \mathbf{b}^i\right) \\
\mathbf{o}_t &= \sigma\left(U^o \mathbf{x}_t + W^o \mathbf{z}_{t-1} + \mathbf{b}^o\right) \\
\mathbf{s}_t &= \mathbf{f}_t \circ \mathbf{s}_{t-1} + \mathbf{i}_i \circ \tanh\left(U^s \mathbf{x}_t + W^s \mathbf{z}_{t-1} + \mathbf{b}^s\right) \\
\mathbf{z}_t &= \mathbf{o}_t \circ \tanh\left(\mathbf{s}_t\right)
\end{aligned}$$

where $\circ$ is the Hadamard (element-wise) product. σ is the activation function, which is chosen to be the sigmoid function $\left((1+e^{-x})^{-1}\right)$, similarly to [15]. Furthermore, $\mathbf{f}_t \in \mathbb{R}^H$ is the forget gate's activation vector, $\mathbf{i}_t \in \mathbb{R}^H$ is the input gate's activation vector, $\mathbf{o}_t \in \mathbb{R}^H$ is the output gate's activation vector, $\mathbf{z}_t \in \mathbb{R}^H$ is the hidden state, $\mathbf{s}_t \in \mathbb{R}^H$ is the cell's memory, and U, W, and b are weight matrices and vector respectively.

The final state $\mathbf{z}_t$ is fed to a semi-affine transformation in the same way for general RNNs,

$$\hat{y}_{t+h} = \sigma^{(2)}\left(W^{(2)} \mathbf{z}_t + \mathbf{b}^{(2)}\right).$$

In case of classification $\sigma^{(2)}$ is the softmax function, and the identity in case of regression.

As LSTMs produce hidden state at each time step, they create a new sequence $\mathbf{z}_t$, $t = T-1, \ldots, 0$. This sequence can be passed on to another LSTM, creating a stacked LSTM architecture [52]. A stacked LSTM containing multiple LSTM layers is better able to approximate non-linear functions than single-layer LSTMs.

6.2.2 CNN-LSTM

Another method to model time-series is the CNN-LSTM, which is a combination of a Convolutional Neural Network (CNN), and an LSTM. CNNs are artificial neural networks that can exploit local spatial structures in the input [12]. Combinations of both methods have shown to increase performance as both methods combine strengths, CNN in the spatial part (i.e. the state of the order book) and LSTM in temporal structures.

Convolutional neural networks consist of layers of discrete convolutional filters. For two general time-series b_t and x_t, the convolution is given by,

$$y_t = \sum_{k=-\infty}^{\infty} b_k x_{t-k}.$$

Convolution is commonly used in stochastic signal analysis (e.g. infinite impulse response filters). Unlike in traditional signal filtration theory, the weights are taken as free parameters [12].

CNNs provide ways to extract features in the spatial dimension. Here, the convolution is performed in the spatial dimension, yielding a new time-series,

$$y_{i,t} = \sum_{k=-\infty}^{\infty} K_k x_{i-k,t}.$$

K is called the kernel, and its dimension determines the spatial dimension of the resulting time-series $\mathbf{y}_t$. Similarly to LSTMs, the resulting time series can be used as input for another CNN layer.

A CNN can be considered to be a generalization of the micro-price. The mirco-price is similar to the mid-price, but it is weighted by the volume of the opposite side [53],

$$p^{\text{micro}} = \frac{v_a^1 p_b^1 + v_b^1 p_a^1}{v_b^1 + v_a^1}.$$

If $\mathbf{x}_t = (p_a^1, p_b^1, v_a^1, v_b^1, \ldots)$ is the state of the order book similar to Equation (1), then applying a CNN with kernel of size 2 results in the linear combination of the two prices in the first output of the CNN, similar to the micro-price. This idea is further elaborated on in [12].

The time-series output of the CNN is pasted onto the LSTM creating the CNN-LSTM. The CNN serves as a way to automate feature extraction [12], and is used to learn the spatial structures in the data. The derived features are then passed on to the LSTM which can extract temporal structures. In [12] the idea is used of combining CNNs and LSTMs, but use a more elaborate method.

6.3 Training

Training is done on the input and output pairs $D = \{(x_t, y_t),\ t = 1, \ldots, T\}$. The goal is to find the approximator $\hat{f}(x)$ of $f(x)$. We define a loss function $\mathcal{L}(y, \hat{f})$ for a predictor $\hat{f}$. The goal of training the models comes down to minimizing the loss function over all observations,

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda\phi(\boldsymbol{\theta}),$$
$$L(\boldsymbol{\theta}) = \frac{1}{T}\sum_{i=1}^{T} \mathcal{L}(y_t, \hat{f}(x_t)),$$

where ϕ is a regularisation penalty weighted by λ. Regularization is often used to make the model less susceptible to over-fitting by penalising large weights [33]. However, in this thesis no regularisation is used (as the results show no indication of over-fitting).

In case of regression, the loss function is taken to be the square error,

$$\mathcal{L} = (y - \hat{f})^2,$$

and L is called the mean-squared error (MSE). In case of classification, the loss function is taken to be the negative cross-entropy,

$$\mathcal{L}(g, \hat{g}) = -\sum_{i=1}^{K} g_k \ln \hat{g}_k.$$

The use of negative cross-entropy finds is justification in the fact that the negative cross-entropy is minimal if the distribution of the estimator equals the “true” distribution [33].

In case there is no regularisation penalty, then the training is equivalent to minimising some function of the error $\varepsilon_t = y_t - \hat{f}_t$ over all observations. Note that we do not make any assumptions regarding the distribution of the error. However, if there is an underlying probability model, $p(y|\hat{f})$, then the loss function is the negative log probability $\mathcal{L}(y, \hat{f}) = -\log p(y|\hat{f})$.

For example, in the case of MSE it is implicitly assumed that the error is Gaussian and independent and identically distributed (i.i.d.). To see this, suppose $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$, then we can fit the model by maximizing the log-likelihood L,

$$\begin{aligned} L &= \sum_{i=1}^{N} \log \left(\frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(\hat{f} - y_t)^2}{2\sigma^2} \right) \\ &= \sum_{i=1}^{N} - \log \left(\sigma\sqrt{2\pi} \right) - \frac{(\hat{f} - y_t)^2}{2\sigma^2}. \end{aligned}$$

We can see that under this assumption maximizing the log-likelihood with respect to the parameters of $\hat{f}$ is equivalent to minimizing the MSE (regardless of σ), so that we indeed implicitly assume normality and i.i.d.

Generalizations where the error is heteroscedastic do exist, much in the way that GARCH extends AR. However, heteroscedasticy in neural networks is nascent in academic literature [33]. Furthermore, in the current application of RNNs in this thesis, the predicted value is not known in the next time step. The error for each forecast is only known at the end of the Fix window each day, making heteroscedastic generalizations infeasible.

6.3.1 Optimizers

Optimizers are algorithms to minimize loss functions. There are many forms, but most of them employ a combination of (stochastic) gradient descent and backpropagation.

Stochastic gradient descent (SGD) is used to solve the minimization problem and train the model. SGD uses an estimate of the gradient of the loss function, which at the k^{th} iteration is given by,

$$g^k = \frac{1}{b_k} \sum_{i \in E_k} \nabla \mathcal{L}_{\boldsymbol{\theta}}(y^i, \hat{f}^k)$$

with $E_k \subset \{1, \ldots, N\}$ and $b_k = |E_k|$. b_k is called the batch size. The estimates of $\boldsymbol{\theta}$ are then updated at each iteration by

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - t_k g^k,$$

where t_k is called the learning rate which controls the speed of convergence. If a local minimum of the loss function exists, then the stochastic gradient descent converges to a minimum as $k \to \infty$. The gradient is evaluated over all points in the dataset numerous times. A single iteration over all datapoints is called an epoch. The evaluation of the gradient of the loss function is performed by backpropagation [33]. In this thesis the Adam optimizer [54] is used, which implements SGD in an efficient way.

Research indicates that small batch sizes lead to local minima of the loss function which generalize better [55]. Furthermore, due to the high non-linearity of these models, using low learning rates the loss converges to local minima which generalize better [15].

6.3.2 Hyperparameter optimization

All of the described models include hyperparameters, which characterize their behaviour. In case of RNNs examples of hyperparameters are the number of hidden units and the number of layers in the model.

Hyperparameter optimization (HPO) can be used to find the hyperparameters which result in the model with the smallest loss. The most simple form of hyperparameter optimization is using random grid search, where hyperparameters are chosen uniformly over a domain. More complex variants of HPO include Bayesian hyperparameter optimizations, where the selection of new hyperparameters is determined using a posterior distribution. However, due to the training times HPO is not used in this thesis.

6.4 Code implementation

The models described above are implemented in Python 3.7 using the TensorFlow package (framework version 2.5). The execution of the code is performed on Amazon Sagemaker, which is a cloud computing environment. Sagemaker allows for training on large data sets by providing additional TensorFlow containers. Using these containers, various instances can be initiated on which the training is performed. These instances act as virtual machines, which makes it easy to request more computational power for training the machine learning models. The instances used in the training is the *ml.p3.2xlarge* and includes a GPU to significantly decrease the training time.

6.5 Optimal execution strategies

Before continuing on to the results of the models, first the potential use of machine learning predictions is elaborated. Even though the predictions can be given to traders in real-time, that is not useful as traders get ample trading signals from the market. Machine learning models are particularly useful for high-frequency traders on very short time horizons [24], but here a different idea is presented. This is a rough outline on how to incorporate the predictions in an execution algorithm (EA), which are algorithms to efficiently execute trades. Execution algorithms can be designed for trading against the Fix.

The use of EAs in the FX market is surveyed by the Bank of International settlements [56]. They also provide a basic overview of EA archetypes, why market participants use certain EAs, and the key trade-off between execution objectives (the execution algorithm trilemma). This trilemma entails that an EA balances between market impact, market risk (risk arising from changing market price during execution), and opportunity cost (not being able to execute the entire order within the time period).

The work in this section is based on the work of Almgren and Chriss [57]. This is based on the thought that an exchange rate follows a stochastic process, but that in addition we can make a small prediction on the future exchange rate.

In line with [57], suppose we hold a position of X units of currency, that need to be liquidated before time T. The interval is divided in N intervals of length $\tau = T/N$, and from this we define $t_k = k\tau$, $k = 0, \dots, N$. A trading trajectory is defined as a list $x_0, \dots, x_n$, where x_k is the number of units at time t_k, with $x_0 = X$ and $x_N = 0$. From this a trade list is defined as $n_k = x_{k-1} - x_k$, which is the amount of units sold between t_{k-1} and t_k.

It is assumed in [57] that the price dynamics follows,

$$S_k = S_{k-1} + \sigma\sqrt{\tau}\xi_k - \tau g\left(\frac{n_k}{\tau}\right),$$

where S_k is the exchange rate at time t_k, σ is the volatility, $\xi_k \sim \mathcal{N}(0,1)$ i.i.d., g is a function wihich determines the permanent price impact of trading n_k units. Next to that, they suppose that there is a temporary price impact, so that the actual price for each trade is

$$\tilde{S}_k = S_{k-1} - h\left(\frac{n_k}{\tau}\right),$$

where h determines the temporary market impact. h can also take into account the cost of crossing the spread, and other trading costs.

In [57] the total cost of trading is defined as $XS_0 - \sum n_k\tilde{S}_k$, so that the liquidation is optimized against the initial exchange rate. Here, a slight change is made, since we wish to optimally execute against the Fix. With this in mind, we define the benchmark as $\bar{S}_{m,l} = f(S_k, k = m, \dots, l)$, and $0 \leq m < l \leq N$. Then we wish to optimize the execution against $\bar{S}_{m,l}$, to which end we define the cost as

$$C = X\bar{S}_{m,l} - \sum_{k=1}^{N} n_k\tilde{S}_k.$$

We assume that S_k is adapted to the filtration $\mathcal{F}_n$. Then the expected value of C given $\mathcal{F}_n$ is,

$$\mathbb{E}\left(C|\mathcal{F}_n\right) = X\,\mathbb{E}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) - \sum_{k=1}^{N} n_k\,\mathbb{E}\left(\tilde{S}_k|\mathcal{F}_n\right).$$

Similarly the conditional variance of C can be calculated, which is slightly more complicated as $\bar{S}_{m,l}$ and $\tilde{S}_k$ are not independent.

One way to find the optimal liquidation strategy is to minimize the conditional expected value of the cost subject to a constraint on the conditional variance. The minimization problem is then given by,

$$\min_{C:\mathbb{V}(C|\mathcal{F}_n)\leq\mathbb{V}_*} \mathbb{E}(C|\mathcal{F}_n), \tag{8}$$

for a given maximum variance $\mathbb{V}_*$. This yields the (dynamic) optimal strategy n_k, $k = n+1,\ldots,N$. This is similar to [57], but they have a static optimization since they use the unconditional expectation and variance. Note that we expect the conditional variance to decrease over time. A case is worked out in Appendix B where the optimisation is static, and it is assumed that $\bar{S}_{m,l} = \frac{1}{l-m}\sum_{k=m}^{l} S_k$.

The machine learning essentially aims to predict $\mathbb{E}\left(\bar{S}_{m,l}|\mathcal{F}_n\right)$. Since at each time step a new prediction is given, a dynamic strategy can be designed which solves Equation (8) at every time step. Similarly, the machine learning can include predictions on the price ΔS_k^{pred} on short time scales,

$$S_k = S_{k-1} + \sigma\sqrt{\tau}\xi_k - \tau g\left(\frac{n_k}{\tau}\right) + \Delta S_k^{\text{pred}}.$$

7 Results

In this section the results are presented. First the results from the regression are shown, after which the classification is given.

7.1 Regression

First the number of levels to use in the state of the LOB needs to be determined. In Figures 19 and 20 scatter plots are shown of the quantity of an order versus the initial level at which that order is placed in the LOB, using data of January 20th, 2021 between 15:52:30 - 16:02:30 LN. As many orders are placed at the same quantity and initial level, in Figure 20 only points are plotted with more than 10 observations.

From the figures it can be seen that the quantities of orders placed at higher levels in the LOB are mostly 10^6, $3 \cdot 10^6$, and $5 \cdot 10^6$. Note in Figure 20 that the number of trades at level 4 and quantity 10^6 is much higher than other levels/quantities. This could be an indication of orders placed in anticipation of price movements in line with [29]. Furthermore orders with quantities lower than 10^6 are often placed at lower levels in the LOB. Based on these figures, 5 levels are used in the state of the LOB for use the models.

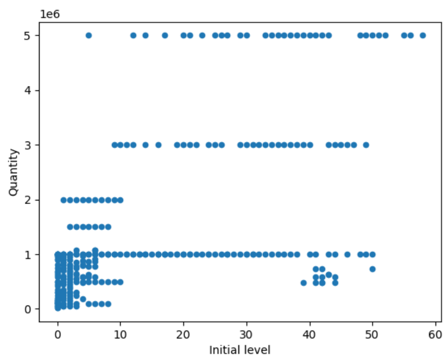


Figure 19: Scatter plot of the initial level versus the quantity.

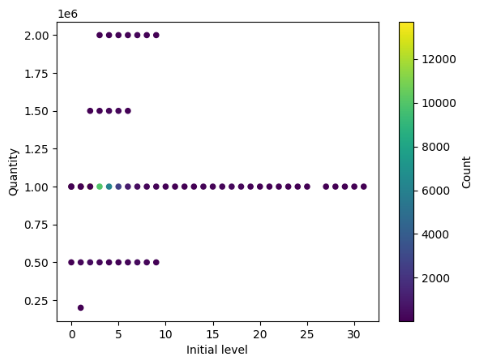


Figure 20: Scatter plot of the initial level versus the quantity.

In the first regression problem, we fit a single layer LSTM on EURUSD LOB data from January 1, 2021 until October 29, 2021. For each day, data is chosen from 15:50-16:02:30 London time. The state of the LOB is sampled every 0.5 seconds for 100 time steps, which gives an input size of $100 \times 4 \cdot 5$. After resampling the input data is normalised using running mean and standard deviation of length 15. The target variable is the difference between the WMR and the 15 time steps running mean of the naive estimate. The data set is split into a training (70% of the data), validation (20%), and test (10%) set. For the data set used this means there are 214 training days, 43 validation days, and 21 test days[13][14]. The batch size used is 64 and the learning rate is set to 10^{-5}. The number of epochs is taken to be 200.

With this data and parameters, in Figure 21 an example of a daily prediction of the WMR (with data from October 15th, 2021) is shown, which is the output of the LSTM after de-normalisation. The fix window is indicated using vertical lines. As can be seen, the predictions are very noisy, which is an undesirable property for predictions used in trading algorithms. The error (difference between the prediciton and the WMR) over time is shown is Figure 23, and a histogram of the error in the validation set is displayed in Figure 25.

[13]Rounding occurs in the splitting of the dataset, so that the proportions of the data set is not precisely 70-20-10%. This is to avoid that data from a single day is both in the validation and training set.

[14]The size of this data stored in binary format was 5.9 GB and consist is roughly 400 million data points.

To this end, the predictions are smoothed using a 30 seconds rolling mean. The use of a rolling mean is quite undesirable as the length of the window is chosen somewhat arbitrarily. Furthermore, smoothing diminishes the model's ability to predict sudden price movements.

The smoothed prediction is shown in Figure 22. Since the convergence of the prediction by the LSTM is not enforced, a weighted estimate is added as prediction. The weighted estimate before the window is equal to the prediction by the LSTM, but during the window is weighted with the naive estimate, where the weights change linearly over time. In Figure 24 the smoothed prediction is shown around the window, so that the weighted estimate is more clear.

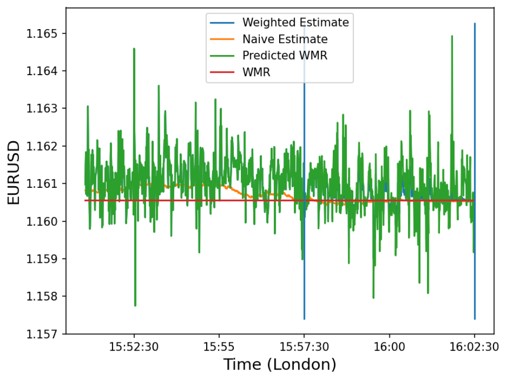


Figure 21: Predictions from LSTM without smoothing.

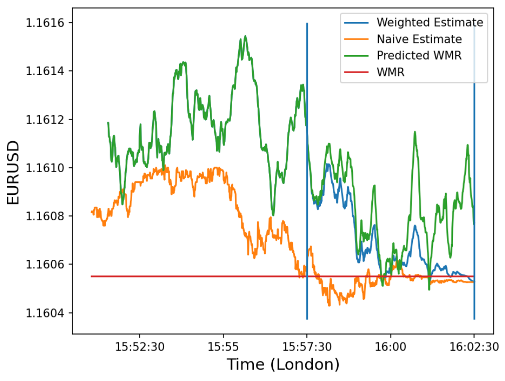


Figure 22: Predictions from the LSTM smoothed with a 30 seconds rolling mean.

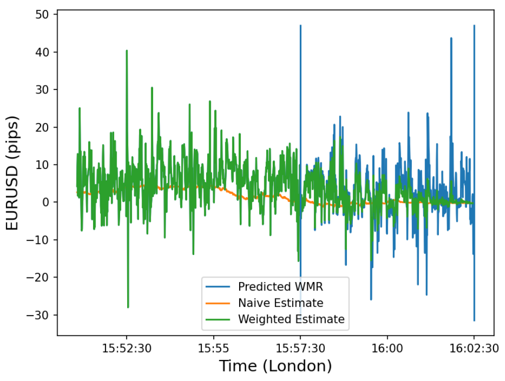


Figure 23: Error of the predictions from LSTM without smoothing.

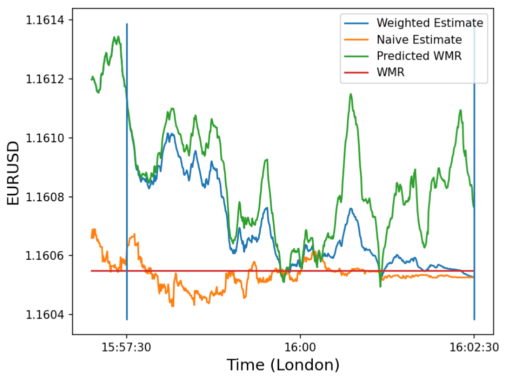


Figure 24: Predictions during the window from the LSTM smoothed with a 30 seconds rolling mean.

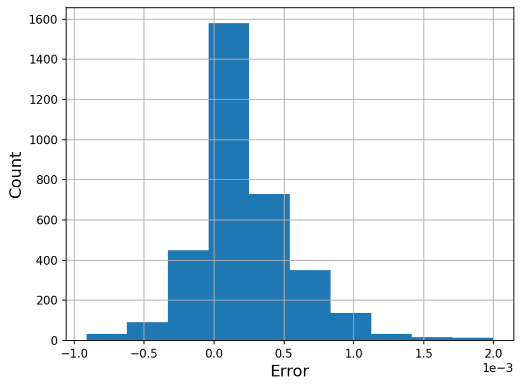


Figure 25: Histogram of the error in the validation set.

In figure 26 a plot of the mean-square error loss is plotted on a log-scale against the epoch number during training. We can see that the validation loss decreases together with the training loss, indicating that the trained LSTM does not overfit the data.

In line with [15], the performance of a trained model is compared to a benchmark estimate. In case of the fix, the benchmark used is the naive estimate from equation (4). In figures 27, 28, and 29 the performance of the model and the naive estimate can be seen on the training, validation, and test set, respectively. The performance in this case is the root of the cumulative root-mean-squared error (RMSE), calculated from time t to the end of the window,

$$RMSE_{\text{cumulative}} = \sqrt{\frac{1}{TN}\sum_{i=t}^{T}\sum_{j=1}^{N}(\hat{y}_{i,j} - y_{i,j})^2}.$$

where N is the number of days, T is the number of time steps per day, and $\hat{y}_{i,j}$ is the prediction of $y_{i,j}$ on day j and time i. The cumulative RMSE is done in order to get an idea of the performance of the model over time, and since the RMSE calculated at each point is time is more noisy.

As can be seen in Figures 28 and 29 the out-of-sample (validation and test) performance of the LSTM is worse than the RMSE of the naive estimate. Note that in all three figures the RMSE of the predicted WMR is is worse than for the naive estimate. Furthermore, the RMSE of the prediction by the LSTM does not clearly decrease during the window, which is due to the fact that the convergence is not enforced.

The RMSE of the validation set by time of day (i.e. not cumulative) is also shown in Figure 30. The RMSE can increase in time and is not to be expected. This could be caused by the low number of observations of 43 days.

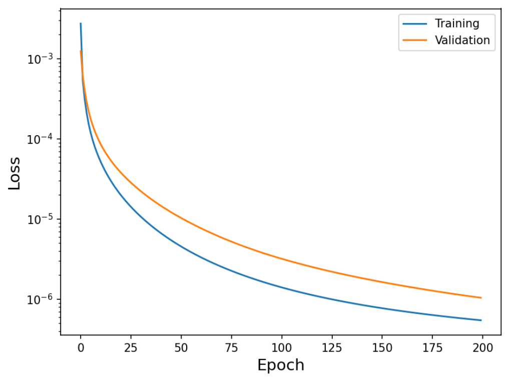


Figure 26: Plot of the loss of the training and validation set on a log scale.

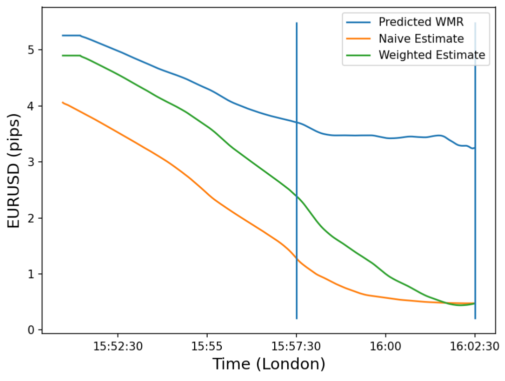


Figure 27: Root mean squared error from time t until the end of the fix window, calculated on the train set.

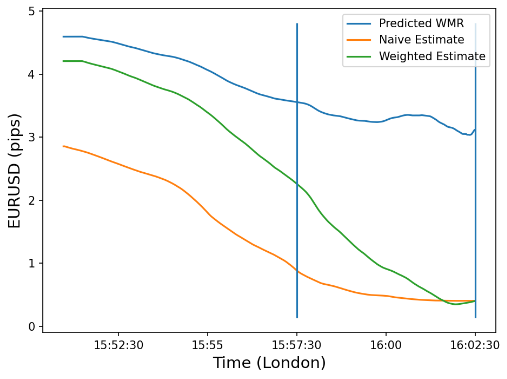


Figure 28: Root mean squared error from time t until the end of the fix window, calculated on the validation set

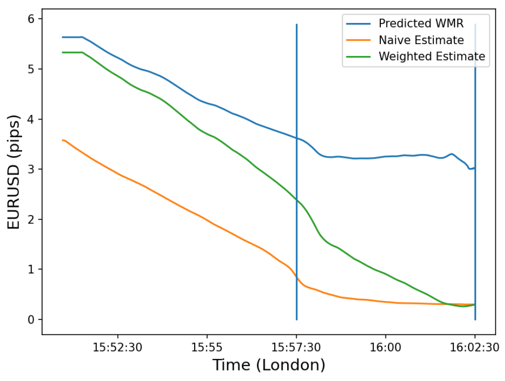


Figure 29: Root mean squared error from time t until the end of the fix window, calculated on the test set.

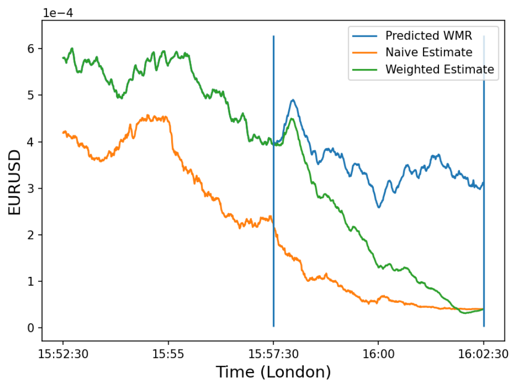


Figure 30: Root mean squared error at time t, calculated on the validation set.

7.1.1 Comparison of various models and input features

The previous results are repeated for various inputs and models to get an idea which combination provides the best performance[15]. A sample of the inputs is shown in Appendix C (without the order flow OF). The input data is split in several groups:

1. LOB is the state of the order book for 5 levels;
2. OF is the order flow;
3. naive is the naive estimate as input;
4. time is the time until the end of the window;
5. trade is the last trade price, quantity and side.

Time is included as the uncertainty of the estimate theoretically decreases as the time approach the end of the window (see Section 4.3). This is not enforced by the models, however. Therefore, by including time explicitly, additional information is provided to the models with the aim of improving the fit (i.e. "learning" the convergence). The same goes for including the naive estimate as input.

The mean-squared error for the various models and inputs is shown in Table 5, where the MSE is calculated over the entire training, validation, and test sets. LSTM(3) indicates a stacked LSTM with three layers. Note that the MSE is calculated of the weighted estimate. Using the prediction directly would result be a worse comparison as these prediction do not converge.

Firstly, none of the models and input features perform better than the naive estimate. This is an indication that the time horizon is generally too far in the future, and that there are big effects which do not manifest themselves in the input data but have an impact on the price formation.

In general, LSTM(3) and CNN-LSTM outperform the single layer LSTM. This is because they allow for more non-linearity, improving the fit. However, they do not sufficiently improve the performance of the models. Adding additional input features does improve the performance, but only marginally. Using order flow shows a relatively large improvement from using the LOB as input, and shows the best results across the train, validation, and test set.

Furthermore, from Table 5 it can be seen that the performance of every model on the test set is worse than on the validation set. As the test set contains data further in time from the validation set, this is an indication that the performance of the models decreases over time. The parameters of the models capture the market dynamics, which can change over time as the dynamics undergo a regime change.

Figure 31 shows the some predictions of one of the best models from Table 7, which is the CNN-LSTM trained on OF. The RMSE by time of this model is plotted in Figure 32 and shows better performance than the model from Figure 30.

Even though the performance of these models is worse than the naive estimate, the convergence of the validation loss show that the training is performed well, and that it does not cause the worse performance. Figure 33 shows the convergence of an LSTM with three layers and trained on LOB + naive + time + trade. Even though Figure 26 suggests that the loss can be even smaller for a larger amount of epochs, the decrease is very slow. In Figure 33 the loss decreases very slowly as the number of epochs increases. Observe that the rate of convergence of the LSTM with three layers is faster for a low number of epochs than for a single layer LSTM. All trained models in Table 5 show similar convergence plots, and the LSTM(3) and CNN-LSTM showed faster convergence than the single layer LSTM. The value the loss converges to can be interpreted as the irreducible term in the bias-variance trade-off form Equation (6).

[15]Note that the size of the dataset for the various input features is larger than the numbers mentioned before as the dimension of the input is increased.

Table 5: Mean-squared error of the train, validation and test sets for various inputs. All mean-squared errors are in units of 10^{-7}.

Input data	Model Model	Train MSE	Validation MSE	Test MSE	Training Time (hours)
Naive estimate		1.6	0.8	1.2	
LOB	LSTM	2.4	1.8	2.8	1.6
LOB	LSTM(3)	2.0	1.5	2.3	5.7
LOB	CNN-LSTM	1.9	1.6	2.3	8.3
LOB + naive	LSTM	2.2	1.7	2.7	1.4
LOB + naive	LSTM(3)	2.0	1.5	2.2	5.5
LOB + naive	CNN-LSTM	1.9	1.5	2.3	8.5
LOB + naive + time	LSTM	2.0	1.4	2.3	1.6
LOB + naive + time	LSTM(3)	2.0	1.6	2.3	5.6
LOB + naive + time	CNN-LSTM	1.9	1.5	2.2	8.4
LOB + naive + time + trade	LSTM	2.0	1.2	1.8	1.7
LOB + naive + time + trade	LSTM(3)	1.8	1.4	2.1	5.5
LOB + naive + time + trade	CNN-LSTM	2.0	1.6	2.4	8.7
OF	LSTM	2.0	1.4	2.1	1.7
OF	LSTM(3)	1.9	1.3	2.1	5.7
OF	CNN-LSTM	1.8	1.2	1.9	8.6
OF + naive (differenced)	LSTM	2.0	1.5	2.2	1.8
OF + naive (differenced)	LSTM(3)	1.9	1.4	2.1	5.9
OF + naive (differenced)	CNN-LSTM	1.8	1.2	1.9	8.6

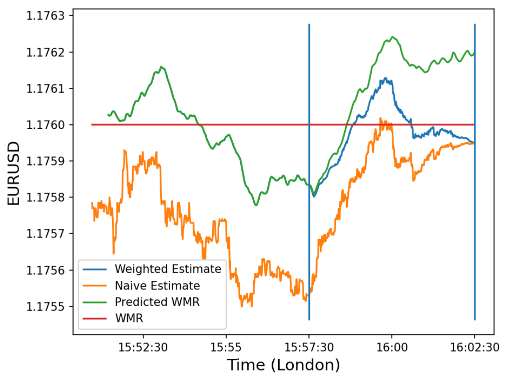


Figure 31: Predictions of CNN-LSTM trained on order flow using data from August 6th, 2021.

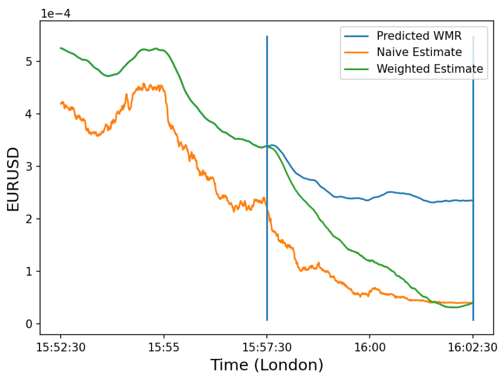


Figure 32: RMSE by time of CNN-LSTM trained on order flow using data from August 6th, 2021.

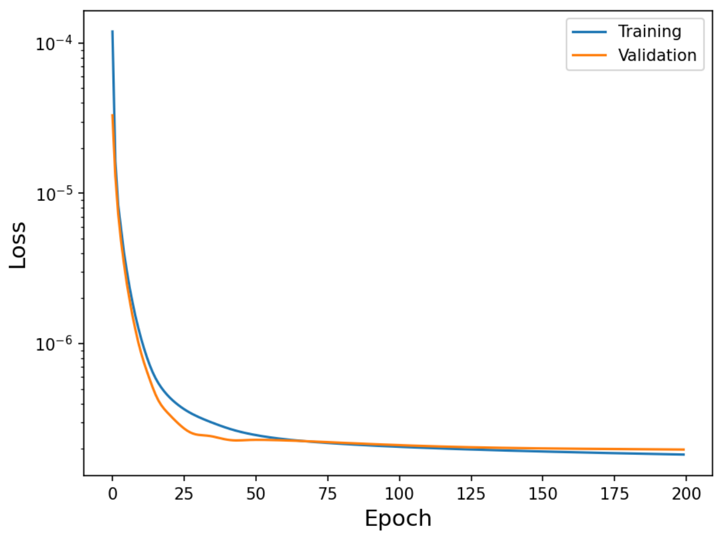


Figure 33: Convergence of the training and validation loss of an LSTM(3) trained on LOB + naive + time + trade.

7.2 Classification

Classification has been been performed on the same data as in the regression. However, in the case of classification the response has changed from the value of the WMR that day, to a category. In case the WMR is more than 2.5 pips over the mid-price, then the label is “up”, if it less than 2.5 pips under the mid-price the label is “down”, and “static” otherwise. The choice of 2.5 pips is loosely based on the root mean-squared error of the naive estimate (see for example Figure 29).

The output of the classifier is a probability for each category. A sample of this is shown in Table 6, where an LSTM is trained on 5 levels of LOB and the naive estimate (LOB + naive from Table 7). The prediction of the classifier is the category with has the highest probability. Figure 34 shows the confusion matrix of this classifier, which is used to visualise

Table 6: Sample of the output of an LSTM classifier from October 1st, 2021.

	Down	Static	Up
15:50:58.000	0.29	0.71	0.00
15:50:58.500	0.40	0.60	0.00
15:50:59.000	0.83	0.17	0.00
15:50:59.500	0.26	0.74	0.00
15:51:00.000	0.29	0.70	0.00
15:51:00.500	0.98	0.02	0.00
15:51:01.000	0.29	0.71	0.00
15:51:01.500	0.16	0.82	0.02
15:51:02.000	0.97	0.03	0.00
15:51:02.500	0.96	0.04	0.00

the performance of the model. Each entry in the confusion matrix contains the number of occurrences where the column indicates what the true label is, and the row indicates the predicted label. Entries on the anti-diagonal are correctly predicted. As can be seen in Figure 34, most predictions where “static”, indicating that the used threshold of 2.5 pips was too large. The classification has been repeated with a threshold of 1 pip in order to

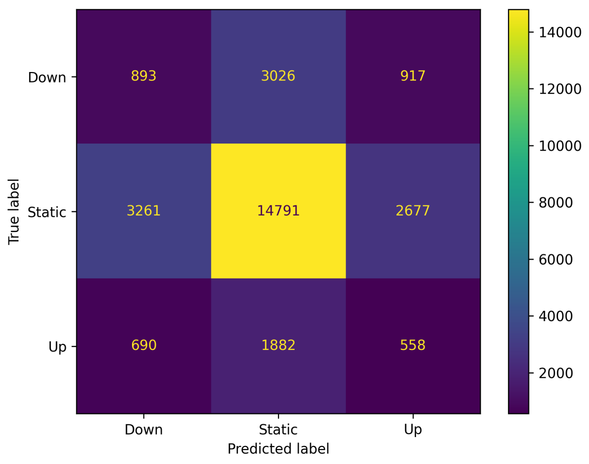


Figure 34: Confusion matrix calculated from a test set of predictions of an LSTM trained on LOB.

reduce the number of static observations. However, none of these yielded any useful results. For example, in Figure 35 the training and validation loss (negative cross-entropy) are shown of a LSTM classifier trained on order flow. The training loss converges, but the validation loss does not. This is indicative for all other attempts. The precise reason of this is unknown, but it could indicate that the time horizon is too large. Furthermore, since the uncertainty of the estimate of the Fix decrease towards the end of the window, the fact that a constant threshold is used could contribute to this bad convergence.

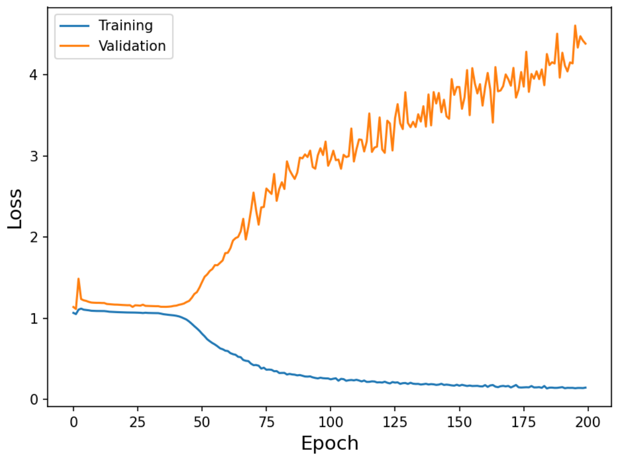


Figure 35: Training and validation loss of an LSTM classifier trained on order flow.

8 Conclusion and Discussion

In this thesis, LSTMs have been applied on FX limit order book data around 16:00 London time to forecast the WMR fix. The work is split in two approaches: regression and classification. The former constitutes the prediction of the WMR, whereas the latter concerns whether the WMR is lower or higher than the current mid-price.

In the case of regression, the performance of the models in the training, validation, and test set was worse than the naive estimate used as baseline. The naive estimate assumes that the price process is a martingale, and turns out to be a better estimator.

Adding additional features to the input does generally improve the result, but only marginally. However, transforming the non-stationary input features to the stationary order flow did improve the performance, and yielded one of the best performing models. Furthermore, the stacked LSTM and CNN-LSTM performed better than the single layer LSTM due to increased non-linearity, but only marginally. Thus the "off-the-shelf" recurrent neural networks used do not appear to be suitable to be used on these time scales. The markets appear to be more efficient than was assumed in this work.

The volatility in all predictions is very high. This could indicate that the time horizon of the predictions is too far in the future. Most literature only considers short term predictions (e.g. seconds or less), whereas the time scales used in this thesis are over 10 minutes. Furthermore, the large volatility makes the use of these predictions impractical in trading algorithms.

The convergence of the methods is an indication that the models are not overfitted, and that state of the order book does not contain sufficient information for the prediction on longer time scales. The latter can also be seen in the light of the bias-variance trade-off from Section 6.1, as the loss converges to the irreducible noise which is implicitly assumed. The fact that the state of the order book does not provide enough information means that the noise term has to model all other market dynamics, contributing to the noise.

Furthermore, the naive estimate might be a good estimator for the following reason. As market participants wish to execute against the Fix, they make use of execution algorithms. From the problem of optimal execution, it can be seen that a TWAP (time-weighted average price) algorithm is optimal given the price dynamics. If many parties at the same time execute using similar algorithms, this might create herd behavior. Additionally, in [9] it is stated that the first part of the Fix window is indicative of the second part. However, since the uncertainty in the estimate of the fix decreases cubically, the second part of the Fix window has not so much influence on the outcome of the value of the Fix.

Next to regression, the problem was also stated as a classification problem. However, all attempts at classification did not yield useful results. In all cases the training loss converged but not the validation loss. The precise reason for this is unknown, but the constant threshold used could contribute to this, as the uncertainty of the Fix estimate decreases towards the end of the window.

The used machine learning models suffer from three large weaknesses: normalisation, resampling, and smoothing. All three introduce ad hoc modelling parameters, such as the window of normalisation, the resampling frequency, the smoothing window, and the weighted estimate. Moreover, smoothing is undesirable as the predictions become less sensitive to quick changes in the underlying dynamics of the order book. The weighted estimate is introduced to enforce convergence of the LSTM predictions to a single value at the end of the window.

The special market dynamics are not captured by the LSTMs, even though they are designed to avoid the need of domain knowledge. This might be improved by investigating the assumption that the error is independent and identically distributed, and heteroscedastic extensions could be considered.

More complex models can be considered, such as DeepLOB [12]. However, given that the performance of the models is worse than the naive estimate, it can be argued whether using more complex models will lead to different results. Relatively simple models are often able to make good first predictions, which can be improved by more complex ones. However, in this case the “off-the-shelf” methods do not yield a satisfactory prediction.

The inclusion of the naive estimate as feature has not improved the results of the regression. Further research can investigate other ways how to include the naive estimate in the input, or how to enforce the convergence to a single value at the end of the window.

The training in this thesis has been performed in a static manner, i.e. the entire data set is split into a training, validation, and test set. Instead training can be performed in a rolling-window backtesting fashion, which better resembles a production setting. In this case, the model is trained on a small period of time, for example 4 weeks, where the first week is reserved for validation, the following three weeks for training, and the last week for testing. Then, if a new time period is available, for example a week of data, then the model is retrained where new week is used for testing, and the previous test data becomes part of the training data. This gives several benefits, such as the fact that nearly the entire dataset has been used for training, and the performance can also be measured on nearly the entire dataset. Furthermore, in this way regime changes in the market dynamics can be captured. Besides, the results from the regression indicate that the performance deteriorates on longer time scales, as the MSE of the test set for all models is worse than the validation MSE. This could be overcome as well.

Lastly, regarding optimal execution abundant research can be done. The analysis of the market dynamics in Section 4.2 can be used together with the optimal execution problem worked out in Appendix B to lead to an improved optimal execution during the fix window. For example, from the number of events placed in the order book an estimate can be made on the permanent market impact. Similarly the temporary market impact can be estimated as well as the transaction costs (which include the spread). Furthermore, in optimal execution it can be interesting to consider reinforcement learning, which is another branch of machine learning next to the supervised learning performed here. The reinforcement learning can be applied in a setting of optimal execution against a benchmark.

In the end forecasting financial time series is incredibly hard as market continually change dynamics and many parties are performing research in order to benefit form it. Even as a model might perform well on a certain time period, there is no guarantee that it will work on another.

A Expected value and variance of $\bar{S}$

Here the conditional expected value and variance of $\bar{S}_{m,l}$ as defined in Equation 3 are derived. The expected value follows directly from the linearity of the expected value,

$$\mu_n = \mathbb{E}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) = \begin{cases} S_n & n < m, \\ \frac{1}{l-m+1}\sum_{k=m}^{n} S_k + \frac{l-n}{l-m+1}S_n & m \le n \le l, \\ \frac{1}{l-m+1}\sum_{k=m}^{l} S_k & l < n. \end{cases} \tag{9}$$

The calculation of the variance,

$$\mathbb{V}\mathrm{ar}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) = \frac{1}{(l-m+1)^2}\mathbb{V}\mathrm{ar}\left(\sum_{k=m}^{l} S_k|\mathcal{F}_n\right),$$

is split in three cases. Firstly, if $n < m$,

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) &= \frac{1}{(l-m+1)^2}\mathbb{V}\mathrm{ar}\left(\sum_{k=m}^{l} S_k|\mathcal{F}_n\right)\\ &= \frac{1}{(l-m+1)^2}\mathbb{V}\mathrm{ar}\left(\sum_{k=m}^{l}\left(\sum_{i=n+1}^{k}\sigma\sqrt{\tau}\xi_k + S_n\right)|\mathcal{F}_n\right)\\ &= \frac{\sigma^2\tau}{(l-m+1)^2}\mathbb{V}\mathrm{ar}\left(\sum_{k=m}^{l}\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right).\end{aligned}$$

The variance can be further simplified,

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left(\sum_{k=m}^{l}\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right) &= \sum_{k=m}^{l}\mathbb{V}\mathrm{ar}\left(\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right) + 2\sum_{k=m}^{l-1}\sum_{q=k+1}^{l}\mathbb{C}\mathrm{ov}\left(\sum_{i=n+1}^{k}\xi_k,\sum_{i=n+1}^{q}\xi_k|\mathcal{F}_n\right)\\ &= \sum_{k=m}^{l}\mathbb{V}\mathrm{ar}\left(\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right) + 2\sum_{k=m}^{l-1}\sum_{q=k+1}^{l}\mathbb{V}\mathrm{ar}\left(\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right)\\ &= \sum_{k=m}^{l}\mathbb{V}\mathrm{ar}\left(\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right) + 2\sum_{k=m}^{l-1}(l-k)\mathbb{V}\mathrm{ar}\left(\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right)\\ &= \sum_{k=m}^{l}(k-n) + 2\sum_{k=m}^{l-1}(l-k)(k-n)\\ &= \sum_{k=m}^{l}\left(1+2(l-k)\right)(k-n)\\ &= \frac{l-m+1}{6}\left(2l^2+2lm-6ln+l-4m^2+6mn+5m-6n\right).\end{aligned}$$

Using this the variance for $n < m$ is,

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) &= \frac{\sigma^2\tau}{(l-m+1)^2}\sum_{k=m}^{l}\left(1+2(l-k)\right)(k-n)\\ &= \frac{\sigma^2\tau}{6(l-m+1)}\left(2l^2+2lm-6ln+l-4m^2+6mn+5m-6n\right)\\ &= -\sigma^2\tau n + K_1,\end{aligned}$$

where K_1 is a constant independent of n. The derivation for $m \leq n < l$ is similar,

$$\begin{aligned}
\mathbb{V}\text{ar}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) &= \frac{1}{(l-m+1)^2}\mathbb{V}\text{ar}\left(\sum_{k=n+1}^{l} S_k|\mathcal{F}_n\right) \\
&= \frac{1}{(l-m+1)^2}\mathbb{V}\text{ar}\left(\sum_{k=n+1}^{l}\left(\sum_{i=n+1}^{k}\sigma\sqrt{\tau}\xi_k + S_n\right)|\mathcal{F}_n\right) \\
&= \frac{\sigma^2\tau}{(l-m+1)^2}\mathbb{V}\text{ar}\left(\sum_{k=n+1}^{l}\sum_{i=n+1}^{k}\xi_k|\mathcal{F}_n\right) \\
&= \frac{\sigma^2\tau}{(l-m+1)^2}\sum_{k=n+1}^{l}\left(1+2(l-k)\right)(k-n) \\
&= \frac{\sigma^2\tau}{6(l-m+1)^2}(2l-2n+1)(l-n)(l-n+1) \\
&= -\sigma^2\tau\left(\beta_1 n^3+\beta_2 n^2+\beta_3 n\right)+K_2,
\end{aligned}$$

where β_1, β_2, β_3, and K_2 are independent of n. Lastly, for $l \leq n$, the variance is zero. Summarizing the three cases, the conditional variance of $\bar{S}$ is,

$$\sigma_n^2 = \mathbb{V}\text{ar}\left(\bar{S}_{m,l}|\mathcal{F}_n\right) = \begin{cases} -\sigma^2\tau n + K_1 & n < m, \\ -\sigma^2\tau\left(\beta_1 n^3+\beta_2 n^2+\beta_3 n\right)+K_2 & m \leq n \leq l, \\ 0 & n > l. \end{cases}$$

B Optimal liquidation under price dynamics

In line with [57], suppose we hold a position of X units of currency, that need to be liquidated before time T. The interval is divided in N intervals of length $\tau = T/N$, and define $t_k = k\tau$, $k = 0, \ldots, N$. A trading trajectory is defined as a list $x_0, \ldots, x_n$, where x_k is the number of units at time t_k, with $x_0 = X$ and $x_N = 0$. From this the trade list is defined as $n_k = x_{k-1} - x_k$, which is the amount of units sold between t_{k-1} and t_k. Suppose the exchange rate S_k follows the following process,

$$S_k = S_{k-1} + \sigma_k \sqrt{\tau} \xi_k - \tau g_k \left(\frac{n_k}{\tau} \right),$$

where σ_k is the time-dependent volatility, $\xi_k \sim \mathcal{N}(0,1)$ i.i.d., and g_k is the time-dependent permanent market impact. Furthermore, suppose that the price at which we can trade $\tilde{S}_k$ depends on the quantity of the trade,

$$\tilde{S}_k = S_{k-1} - h_k \left(\frac{n_k}{\tau} \right),$$

where we refer to h_k as the time-dependent temporary market impact. This is similar as in [57], with the difference that the volatility, permanent, and temporary market impact are time-dependent. The reason for this is that during and around the WMR fix, these can vary quite a lot, as is shown in Section 4.2. In theory, the analysis from Section 4.2 can be used in this optimal liquidation problem to devise a strategy to trade against the Fix.

Next we define the benchmark against we wish to optimally execute as

$$\bar{S}_{m,l} = \frac{1}{l - m + 1} \sum_{k=m}^{l} S_k = \sum_{k=m}^{l} w_k S_k.$$

The cost of trading against this benchmark is defined as,

$$C = X \bar{S}_{m,l} - \sum_{k=1}^{N} n_k \tilde{S}_k.$$

The goal is the find the n_k such that the cost optimal in some sense. Note that the cost can be written as

$$\begin{aligned} C &= -XS_0 + \sum_{k=1}^{N} \left(\tau g_k \left(\frac{n_k}{\tau} \right) - \sigma_k \sqrt{\tau} \xi_k \right) x_k + \sum_{k=1}^{N} n_k h_k \left(\frac{n_k}{\tau} \right) + X \sum_{k=m}^{l} w_k S_k \\ &= -XS_0 + \sum_{k=1}^{N} \left(\tau g_k \left(\frac{n_k}{\tau} \right) - \sigma_k \sqrt{\tau} \xi_k \right) x_k + \sum_{k=1}^{N} n_k h_k \left(\frac{n_k}{\tau} \right) \\ &\qquad + X \sum_{k=m}^{l} w_k \left(S_0 + \sum_{i=1}^{k} \sigma_i \sqrt{\tau} \xi_i - \tau g_i \left(\frac{n_i}{\tau} \right) \right). \end{aligned}$$

The expected value of the cost is

$$\mathbb{E}(C) = -XS_0 + \sum_{k=1}^{N} \tau x_k g_k \left(\frac{n_k}{\tau} \right) + \sum_{k=1}^{N} n_k h_k \left(\frac{n_k}{\tau} \right) + X \sum_{k=m}^{l} w_k \left(S_0 - \sum_{i=1}^{k} \tau g_i \left(\frac{n_i}{\tau} \right) \right).$$

Similarly, the variance is given by

$$\begin{aligned}\mathbb{V}\mathrm{ar}\,(C) &= \mathbb{V}\mathrm{ar}\left(\sum_{k=1}^{N} -\sigma_k\sqrt{\tau}\xi_k x_k + X\sum_{k=m}^{l} w_k S_k\right)\\ &= \mathbb{V}\mathrm{ar}\left(\sum_{k=1}^{N} -\sigma_k\sqrt{\tau}\xi_k x_k + X\sum_{k=m}^{l} w_k \sum_{i=1}^{k}\sigma_i\sqrt{\tau}\xi_i\right)\\ &= \sum_{k=1}^{N}\sigma_k^2\tau(X\mathbb{1}_k w_k - x_k)^2 - 2\sum_{k=1}^{N-1}\sigma_k^2\tau X x_k \sum_{q=k+1}^{N}\mathbb{1}_q w_q + K,\end{aligned}$$

where K is a term independent of x_k, and

$$\mathbb{1}_k = \begin{cases}1, & m \le k \le l,\\ 0, & \text{otherwise.}\end{cases}$$

The optimal strategy might be found by minimizing the expected cost, under a maximum level of variance,

$$\min_{C:\mathbb{V}(C)\le\mathbb{V}_*} \mathbb{E}(C).$$

To solve this, we introduce a Lagrange multiplier λ, and minimize $U(C) = \mathbb{E}\,(C) + \lambda\mathbb{V}\mathrm{ar}\,(C)$, which is equivalent to minimizing,

$$\begin{aligned}\tilde{U}(C) = \sum_{k=1}^{N} \tau x_k g_k\left(\frac{n_k}{\tau}\right) + n_k h_k\left(\frac{n_k}{\tau}\right) - X\mathbb{1}_k w_k \sum_{i=1}^{k}\tau g_i\left(\frac{n_i}{\tau}\right) + \lambda\sigma_k^2\tau(-2X\mathbb{1}_k w_k x_k + x_k^2)\\ - 2\lambda\sigma_k^2\tau X x_k \sum_{q=k+1}^{N}\mathbb{1}_q w_q.\end{aligned}$$

No market impact and constant volatility

If we assume no market impact $g_k = h_k = 0$, then the minimization problem reduces to the minimization of the variance. If we in addition assume that the volatility is constant, then the minimization problem is easy to solve. Taking partial derivatives with respect to w_j, and setting these to zero, the optimal trading trajectory is given by

$$x_j = X\sum_{q=j}^{N}\mathbb{1}_q w_q = \frac{X}{l-m+1}\sum_{q=j}^{N}\mathbb{1}_q.$$

Note that this solution satisfies $x_0 = X$ and $x_N = 0$. This optimal strategy is called a time-weighted average price (TWAP), executed during the time window of the benchmark.

Linear market impact

As in [57], we can specify the market impact functions as

$$\begin{aligned}g_k\,(v) &= \gamma_k v,\\ h_k\left(\frac{n_k}{\tau}\right) &= \epsilon_k \mathrm{sgn}(n_k) + \frac{\eta_k}{\tau}n_k.\end{aligned}$$

Then $\tilde{U}(C)$ reduces to,

$$\tilde{U}(C) = \sum_{k=1}^{N} x_k \gamma_k n_k + \epsilon_k |n_k| + \frac{\eta_k}{\tau} n_k^2 - X \mathbb{1}_k w_k \sum_{i=1}^{k} \gamma_i n_i \\ - 2\lambda\sigma_k^2 \tau X \mathbb{1}_k w_k x_k + \lambda \sigma_k^2 \tau x_k^2 - 2\lambda\sigma_k^2 \tau X x_k \sum_{q=k+1}^{N} \mathbb{1}_q w_q. \quad (10)$$

Note that in case the permanent market impact is constant, this can be simplified further. First, since $n_k = x_{k-1} - x_k$,

$$\sum_{i=1}^{k} \gamma n_i = \gamma(X - x_k).$$

Secondly (see [57]),

$$\sum_{k=1}^{N} \gamma x_k n_k = \frac{1}{2}\gamma X^2 - \frac{1}{2}\gamma \sum_{k=1}^{N} n_k^2.$$

With this 10 becomes (dropping out terms independent of x_k),

$$\tilde{U}(C) = \sum_{k=1}^{N} \epsilon_k |n_k| + \left(\frac{\eta_k}{\tau} - \frac{1}{2}\gamma \right) n_k^2 + X \mathbb{1}_k w_k \gamma x_k \\ - 2\lambda\sigma_k^2 \tau X \mathbb{1}_k w_k x_k + \lambda \sigma_k^2 \tau x_k^2 - 2\lambda\sigma_k^2 \tau X x_k \sum_{q=k+1}^{N} \mathbb{1}_q w_q. \quad (11)$$

Grouping all terms,

$$\tilde{U}(C) = \sum_{k=1}^{N} \epsilon_k |n_k| + \tilde{\eta}_k n_k^2 + \alpha_k x_k + \beta_k x_k^2,$$

where

$$\begin{aligned} \tilde{\eta}_k &= \frac{\eta_k}{\tau} - \frac{1}{2}\gamma, \\ \alpha_k &= X \mathbb{1}_k w_k \gamma - 2\lambda\sigma_k^2 \tau X \sum_{q=k}^{N} \mathbb{1}_q w_q, \\ \beta_k &= \lambda \sigma_k^2 \tau. \end{aligned}$$

$\tilde{U}(C)$ can be minimized numerically.

C Input features

	p_b^1	v_b^1	p_b^2	v_b^2	p_a^1	v_a^1	p_a^2	v_a^2
15:50:08.500	1.22916	2	1.22915	4	1.22920	2	1.22921	4
15:50:09.000	1.22916	2	1.22915	4	1.22920	2	1.22921	4
15:50:09.500	1.22917	2	1.22916	2	1.22920	1	1.22921	3
15:50:10.000	1.22917	2	1.22916	2	1.22921	4	1.22922	4
15:50:10.500	1.22908	1	1.22907	1	1.22911	2	1.22912	2

	last trade price	last trade quantity	last trade side	naive. WMR estimate	time to end window
15:50:08.500	1.22919	1	1.0	1.229180	741.5
15:50:09.000	1.22919	1	1.0	1.229180	741.0
15:50:09.500	1.22919	1	1.0	1.229185	740.5
15:50:10.000	1.22919	1	1.0	1.229190	740.0
15:50:10.500	1.22914	0.2	1.0	1.229095	739.5

Table 7: Sample of the input data used in Section 7.1 taken from January 6th, 2021. Volumes in the table are in units of 1000000

References

[1] Ian W Marsh, Panagiotis Panagiotou, and Richard Payne. "The WMR fix and its impact on currency markets". In: (2017).

[2] Bank for International Settlements. *Triennial Central Bank Survey: Foreign exchange turnover in April 2019*. Sept. 2019. URL: `https://www.bis.org/statistics/rpfx19_fx.pdf`.

[3] Michael Melvin and John Prins. "Equity hedging and exchange rates at the London 4 pm fix". In: *Journal of Financial Markets* 22 (2015), pp. 50–72.

[4] Ananda Chatterjee, Hrisav Bhowmick, and Jaydip Sen. *Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models*. 2021. arXiv: `2111.01137 [q-fin.ST]`.

[5] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019". In: *Applied soft computing* 90 (2020).

[6] Alexander Dautel et al. "Forex exchange rate forecasting using deep recurrent neural networks". In: *Digital Finance* 2 (Sept. 2020). DOI: `10.1007/s42521-020-00019-x`.

[7] Philippe Bacchetta, Elmar Mertens, and Eric Van Wincoop. "Predictability in financial markets: What do survey expectations tell us?" In: *Journal of International Money and Finance* 28.3 (2009), pp. 406–426.

[8] Burton G Malkiel. "Reflections on the efficient market hypothesis: 30 years later". In: *Financial review* 40.1 (2005), pp. 1–9.

[9] Pragma. *New Trading Patterns Around the WM/R Fix*. 2015.

[10] Pragma. *WM/R Trading Patterns: Month-End and Quarter-End*. 2015.

[11] Arnav Sheth and Keisuke Teeple. "Leveraging Herd Behavior in Foreign Exchange Markets". In: *SSRN Electronic Journal* (Jan. 2017). DOI: `10.2139/ssrn.3069629`.

[12] Zihao Zhang, Stefan Zohren, and Stephen Roberts. "DeepLOB: Deep Convolutional Neural Networks for Limit Order Books". In: *IEEE Transactions on Signal Processing* 67.11 (June 2019), pp. 3001–3012. ISSN: 1941-0476. DOI: `10.1109/tsp.2019.2907260`. URL: `http://dx.doi.org/10.1109/TSP.2019.2907260`.

[13] JG Agrawal, V Chourasia, and A Mittra. "State-of-the-art in stock prediction techniques". In: *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 2.4 (2013), pp. 1360–1366.

[14] Rodolfo C Cavalcante et al. "Computational intelligence and financial markets: A survey and future directions". In: *Expert Systems with Applications* 55 (2016), pp. 194–211.

[15] Petter N. Kolm, Jeremy D. Turiel, and Nicholas Westray. "Deep Order Flow Imbalance: Extracting Alpha at Multiple Horizons from the Limit Order Book". In: *Econometric Modeling: Capital Markets - Portfolio Theory eJournal* (2021).

[16] Frédéric Abergel and Aymen Jedidi. "A mathematical approach to order book modeling". In: *International Journal of Theoretical and Applied Finance* 16.05 (2013), p. 1350025.

[17] Rama Cont, Sasha Stoikov, and Rishi Talreja. "A stochastic model for order book dynamics". In: *Operations research* 58.3 (2010), pp. 549–563.

[18] Justin Sirignano. *Deep Learning for Limit Order Books*. 2016. arXiv: `1601.01987 [q-fin.TR]`.

[19] Rama Cont and Adrien De Larrard. "Price dynamics in a Markovian limit order market". In: *SIAM Journal on Financial Mathematics* 4.1 (2013), pp. 1–25.

[20] George S Atsalakis and Kimon P Valavanis. "Surveying stock market forecasting techniques–Part II: Soft computing methods". In: *Expert systems with applications* 36.3 (2009), pp. 5932–5941.

[21] Avraam Tsantekidis et al. "Using deep learning for price prediction by exploiting stationary limit order book features". In: *Applied Soft Computing* 93 (2020), p. 106401.

[22] Justin Sirignano and Rama Cont. *Universal features of price formation in financial markets: perspectives from Deep Learning.* 2018. arXiv: `1803.06917 [q-fin.ST]`.

[23] Avraam Tsantekidis et al. "Forecasting stock prices from the limit order book using convolutional neural networks". In: *2017 IEEE 19th conference on business informatics (CBI).* Vol. 1. IEEE. 2017, pp. 7–12.

[24] Matthew Dixon. "Sequence Classification of the Limit Order Book using Recurrent Neural Networks". In: *Journal of Computational Science* (Jan. 2018). DOI: `10.1016/j.jocs.2017.08.018`.

[25] Zexin Hu, Yiqi Zhao, and Matloob Khushi. "A Survey of Forex and Stock Price Prediction Using Deep Learning". In: *Applied System Innovation* 4 (Feb. 2021), p. 9. DOI: `10.3390/asi4010009`.

[26] Antonio Briola, Jeremy Turiel, and Tomaso Aste. *Deep Learning modeling of Limit Order Book: a comparative perspective.* 2020. arXiv: `2007.07319 [q-fin.TR]`.

[27] Svitlana Galeshchuk. "Neural networks performance in exchange rate prediction". In: *Neurocomputing* 172 (2016), pp. 446–452.

[28] Antonio Gargano and Steven Riddiough. "The Value of Volume in Foreign Exchange". In: *SSRN Electronic Journal* (Jan. 2018). DOI: `10.2139/ssrn.3019870`.

[29] Martin D Gould et al. "Limit order books". In: *Quantitative Finance* 13.11 (2013), pp. 1709–1742.

[30] Charles Cao, Oliver Hansch, and Xiaoxin Wang. "The information content of an open limit-order book". In: *Journal of Futures Markets: Futures, Options, and Other Derivative Products* 29.1 (2009), pp. 16–41.

[31] Yufei Wu et al. "How Robust are Limit Order Book Representations under Data Perturbation?" In: *arXiv preprint arXiv:2110.04752* (2021).

[32] Terrence Hendershott, Charles M Jones, and Albert J Menkveld. "Does algorithmic trading improve liquidity?" In: *The Journal of finance* 66.1 (2011), pp. 1–33.

[33] Matthew F. Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance: From Theory to Practice.* Springer, 2020. DOI: `10.1007/978-3-030-41068-1`.

[34] Rama Cont, Arseniy Kukanov, and Sasha Stoikov. "The price impact of order book events". In: *Journal of financial econometrics* 12.1 (2014), pp. 47–88.

[35] Martin Evans et al. "Fixing the Fix? Assessing the Effectiveness of the 4pm Fix Benchmark". In: *SSRN Electronic Journal* (Jan. 2018). DOI: `10.2139/ssrn.3270844`.

[36] Tim Weithers. *Foreign exchange: a practical guide to the FX markets.* John Wiley & Sons, 2011.

[37] Martin Evans. "Forex Trading and the WMR Fix". In: Nov. 2014.

[38] Carl Husselmann and Kristjan Kasikov. "Trend-following market behaviour at the 4pm London time BFIX and WMR fixing windows". In: *Quantitative Finance* 20 (Aug. 2019), pp. 1–8. DOI: `10.1080/14697688.2019.1638154`.

[39] Refinitiv Benchmark Services Limited. *WM/Refinitiv ('WMR') FX Benchmarks": Spot, Forward and NDF Rates Methodology Guide.* Jan. 2022. URL: `https://www.refinitiv.com/content/dam/marketing/en_us/documents/methodology/wm-refinitiv-methodology.pdf`.

[40] MSCI. *MSCI Index Calculation Methodology*. Feb. 2022. URL: https://www.msci.com/eqb/methodology/meth_docs/MSCI_IndexCalcMethodology_Feb2022.pdf.

[41] FTSE Russel. *Guide to Calculation: FTSE Global Equity Index Series*. Aug. 2021. URL: https://research.ftserussell.com/products/downloads/FTSE_Global_Equity_Index_Series_Guide_to_Calc.pdf.

[42] Takatoshi Ito and Masahiro Yamada. *Did the Reform Fix the London Fix Problem?* Working Paper 23327. National Bureau of Economic Research, Apr. 2017. DOI: 10.3386/w23327. URL: http://www.nber.org/papers/w23327.

[43] Patrick Michelberger and Jan Witte. "Foreign Exchange Market Microstructure and the WM/Reuters 4pm Fix". In: *The Journal of Finance and Data Science* 2 (Jan. 2015). DOI: 10.1016/j.jfds.2016.01.002.

[44] Georgios Chortareas and George Kapetanios. "Getting PPP right: identifying mean-reverting real exchange rates in panels". In: *Journal of Banking & Finance* 33.2 (2009), pp. 390–404.

[45] Cornelis W Oosterlee and Lech A Grzelak. *Mathematical Modeling and Computation in Finance: With Exercises and Python and Matlab Computer Codes*. World Scientific, 2019.

[46] Cboe. *Historical Market Data*. URL: https://fx.cboe.com/products/historical_market_data.jsp.

[47] Leo Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.

[48] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[49] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[50] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[51] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[52] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2013. DOI: 10.48550/ARXIV.1308.0850. URL: https://arxiv.org/abs/1308.0850.

[53] Jim Gatheral and Roel CA Oomen. "Zero-intelligence realized variance estimation". In: *Finance and Stochastics* 14.2 (2010), pp. 249–283.

[54] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[55] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[56] Bank for International Settlements. *FX execution algorithms and market functioning*. Oct. 2020. URL: https://www.bis.org/publ/mktc13.pdf.

[57] Rober Almgren and Neil Chriss. "Optimal Execution of Portfolio Transactions". In: (Dec. 2000). URL: https://www.smallake.kr/wp-content/uploads/2016/03/optliq.pdf.