WM/REUTERS FIXING RATE PREDICTION FOR ALGORITHMIC FX TRADING USING MACHINE LEARNING

Master Thesis

Author: L.D. Timmerman

UNIVERSITY OF TWENTE.

Master Thesis Industrial Engineering and Management, University of Twente

April 2022

WM/REUTERS FIXING RATE PREDICTION FOR ALGORITHMIC FX TRADING USING MACHINE LEARNING

April 2022

Author
L.D.Timmerman (Luuk)
s1864610
University of Twente, Enschede, the Netherlands
Faculty of Behavioral Mangement and Social Sciences
Master program in Industrial Engineering and Management
Specialisation track in Financial Engineering

Graduation Organisation
MN
Prinses Beatrixlaan 15, Den Haag, the Netherlands
Department of Treasury, Asset Management

University Supervisors
Prof. Dr. L. Spierdijk (Laura)
Faculty of Behavior, Management and Social Sciences
Department of Hightech Business and Entrepreneurship
Head of the Financial Engineering Section

Dr. B. Roorda (Berend)
Faculty of Behavior, Management and Social Sciences
Department of Hightech Business and Entrepreneurship
Associate Professor of Financial Engineering

External Supervisors
L. Papapoulos, CFA (Liakos)
Asset Management
Senior Investment Manager Treasury
MN

Executive Summary

When investing in assets that are traded in a foreign currency, pension funds are exposed to currency risks. Especially because monthly payments are to be paid in the home currency. The asset managers of the pension funds can use different techniques to hedge themselves against this risk. As a part of this protection, the managers execute a significant amount of foreign exchange (FX) spot transactions.

Within the FX spot market, the time frame, 15:57:30 to 16:02:30 UTC, is very significant, as it is used to determine the fixing rate (Fix). This time window is called the WMR 4 pm fixing window and is used for fixing orders and as a benchmark to measure the performance of foreign investments, as a reference rate to settle derivative contracts, and for numerous other reference purposes. Because the FX market runs 24 hours per day, Monday to Friday, the fix is used as the alternative benchmark instead of the close, which is generally the benchmark in markets for other securities e.g. equities and bonds.

This thesis aims to develop a deep learning model that can be used to predict the WMR of a currency pair, one day in advance, by finding patterns in FX datasets and market data analysis. This model can provide traders with a prediction of future movements in the market, which can then be used to optimise the execution strategy or execution moment, given the FX orders that need to be traded. While a model can provide investors with information to help increase execution efficiency and to reduce risk, the FX market's complexity makes FX prediction and forecasting a challenging research topic.

In terms of data, especially the end-of-month hedge rebalancing data is of interest. Therefore, we built a model that estimates these hedge rebalancing flows that have to occur at the end of each month. The results of this model are used as input for the neural network. Finally, the aim is to compare different models and test the best-performing model on robustness and parameter sensitivity.

The study objective is formulated as following:

Apply data wrangling and engineering techniques on relevant datasets, to prepare the data to be fed into a theoretically backed deep learning architecture, and evaluate the performance to forecast the WMR fix one day in advance

---

As input for the model, we used the following datasets, consisting of data since 01-01-2000:

1. A FX data set from New Change, consisting of the EUR/USD open, close, high, and low price, of the pair in minute granularity.

2. EuroStoxx50 and SP500 datasets from Bloomberg, consisting of the close prices in daily granularity.

3. Bond yield dataset of the US and EU 10 year generic bonds, in daily granularity. This data is used as an indication of interest rates.

4. Bond return dataset (LUATTRUU Index) from Bloomberg, consisting of the US Treasury Index, which measures US dollar-denominated, fixed-rate, nominal debt issued by the US Treasury and the Pan-European unhedged government total return index (I02513EU Index).

5. Dataset containing the daily WM/Reuters fix benchmark. Obtained from Bloomberg.

From the results, we chose a CNN-LSTM neural network as best performing model. This model consists of a convolutional layer for feature extraction and an LSTM layer for time series forecasting. Using this model, we can confirm that the end-of-month hedge rebalancing data increases the deep learning model performance. This proposed model outperforms the naive benchmark and has a Theil's U score below 1, for both direction and prediction, indicating superiority over random selection. When we assume no transaction costs, this translates to a return of 8.15% over two years when we would buy when the model predicts the WMR will go up and sell when the model predicts that the WMR will go down. The returns consist of -4.73% over 2020 and a 13.5% return over 2021. But, the black box of neural networks remains a problem, and, essentially, no causality between the hyperparameters and the model performance on validation loss can be found.

Although the results of the CNN-LSTM are significant, there is much to test and improve upon this model. Especially, because the proposed model lacks convergence. This could be the result of the neural networks having trouble learning from the data, as the data is extremely noisy. Indicating that there might be a limited amount to learn from this data to base a prediction upon.

A first step to improve the models proposed in this research is to formulate a more extensive estimation model on the end-of-month hedge re-balancing flows. A second step is to perform more model tuning and test a wide range of neural network architectures. In this study, only three possible architectures are tested, and many more can be tested. We also focused on the main parameters during tuning, yet, there are many more parameters that can be tuned to increase model performance. This thesis, however, provides a framework and a new benchmark to beat in terms of performance.

To conclude, the model can easily be applied to practice as support for the traders of MN and PGGM. They have to trade significant flows of EUR/USD, and this model can help the traders in their decision making process regarding the most efficient trading strategy for executing the required flow on a given day.

Preface

This thesis marks the end of my time as a student and of my Master in Financial Engineering and Management at the University of Twente. I have worked on this thesis from October 2021 till April 2022, 5 volatile months that were disrupted by the coronavirus, but this is also the period that marked the beginning of the reopening after the virus. I immensely enjoyed the Master, and I was lucky enough to at least complete part of it with in-person lectures and discussions with fellow students and lecturers. During the research, coding, and writing of this thesis, I got support from many people, which I would like to thank.

First of all, I want to thank Laura Spierdijk, my supervisor from the University of Twente, for her guidance, tips, and thorough feedback. I also want to thank Berend Roorda, my second supervisor, for his great and insightful lectures during this Master programme.

Furthermore, I want to thank MN for giving me the opportunity to conduct my Master thesis at the Treasury and Trading department. Especially I would like to thank Liakos Papapoulos, my supervisor from MN, and Tjerk Methorst from PGGM, for their invaluable insights and guidance, and for setting up the AXP programme at MN and PGGM, providing the students with insights from departments within both pension funds. I also want to thank the Treasury department, the Business Analytics team, Anatoliy Babic, and my fellow graduate students from AXP, who did not only help me during my thesis but also made working at MN incredibly fun.

A special thanks to the friends I already had, the new friends I made, my teammates at EHV, and the members of the Master Corner, for making my time at the University of Twente and Technische Universität München great.

Last, and definitely not least, I want to thank my family for their never-ending support and advice throughout my years at the University of Twente.

Luuk Timmerman
Utrecht, April 2022

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

Abbreviation	***Definition***
AUM	Assets Under Management
CNN	Convolutional Neural Network
EA	Execution Algorithm
FE	Feature Engineering
Fix	Fixing Rate
FX	Foreign Exchange
LSTM	Long Short-Term Memory
NN	Neural Network
ReLU	Rectified Linear Units
RNN	Recurrent Neural Network
STL	Seasonal-Trend decomposition using LOESS
TCA	Transaction Cost Analysis
TWAP	Time Weighted Average Price
UIP	uncovered interest parity
URT	Universal Representation Theorem
WMR	WM/Reuters fixing rate

1. Introduction

This first chapter describes the context of the research and provides background information on important topics to better understand this thesis. The WM/R Fix, order books, and an execution strategy are discussed. The chapter concludes with the objective of this thesis.

1.1 Background Information

For pension funds, it is important to achieve the most optimal financial returns possible for their clients, given a predefined risk profile. The aimed result and achieved result can depend on multiple factors, for example, the risk appetite and age distribution of the clients. But, by investing in assets that are traded in a foreign currency, pension funds are exposed to currency risks. Especially because monthly payments are to be paid in the home currency. The asset managers of the pension funds can use different techniques to hedge themselves against this risk. As a part of this protection, and due to the trading of foreign denominated assets, the managers execute a significant amount of foreign exchange (FX) spot transactions.

1.1.1 WMR Fix

Within the FX spot market, the time frame, 15:57:30 to 16:02:30 UTC, is very significant, as it is used to determine the fixing rate (Fix) (Michelberger & Witte, 2016). This time window is called the WMR 4 pm fixing window and is used for fixing orders and as a benchmark to measure the performance of foreign investments, as a reference rate to settle derivative contracts, and for numerous other reference purposes (Evans, O'Neill, Rime, Saakvitne, et al., 2018). Because the FX market runs 24 hours per day, Monday to Friday, the fix is used as the alternative benchmark instead of the close, which is generally the benchmark in markets for other securities e.g. equities and bonds.

Generally, the methodology of determining the fix is the following: (Reuters, 2017) Each second in the window, a sample is recorded. This sample contains a single trade, with the trade price, whether it is a bid/ask trade and the opposing side's best bid or ask price. At the end of the window, the bid (ask)trade prices and opposing bid (ask)prices are pooled together for all 300 samples. Then a median is calculated for each bid and ask pool. Finally, the midpoint between the two medians is calculated and used as the WMR 4 pm fix.

To avoid tracking errors in comparison to their benchmarks, many market participants want to execute during the window. This results in high liquidity, which generally, results in lower volatility and transaction costs. However, during the window, large orders can create an unequal liquidity supply. These aggressive, or 'liquidity taking' orders can deplete the order book on one side, with large spot rate movements as a result (Michelberger & Witte, 2016). During the window, the probability of these local and global extrema in spot price is much higher than compared to other market hours. These movements are also larger in size on average, which makes it difficult to realise the fixing rate in practice (Michelberger & Witte, 2016).

1.1.2 Order Book

When an order is submitted, it is sent to a venue and placed in an order book. Every millisecond multiple orders are sent to venues. The order book is a list of orders, which is used to record the interest of buyers and sellers in a particular currency pair. A matching engine uses this order book to determine which orders can be executed. Each venue has its own order book, which results in the possibility of discrepancies in spot prices between different venues.

An order book has two sides, the bid side, containing the buy orders and volumes, and the ask side containing the sell orders and volumes. The buy or sell is called the direction. The top-of-book bid ($B_v(t)$) is the highest available buy order for volume v at time t, and the top-of-book ask ($A_v(t)$), the lowest available sell order for volume v at time t. The difference between the bid and ask is called the bid-ask spread, or just spread, which is given by: $S(t) = A(t) - B(t)$. The price point in the middle between the best ask and best bid is the mid-price, given by: $M(t) = (A(t) - B(t))/2$.

The book is dynamic and thus constantly updated in real-time. When submitting an order, it can be placed more passively or aggressively in the order book. Where passive means lower in the order book, where the chances of fast execution are lower, or aggressive means higher in the order book, closer to the bid when buying, or ask when selling. When fast execution is important, or the price is less important, then an order can also be placed very aggressively by 'crossing the spread'. This means that the trader buys at the lowest ask, or sells at the highest bid, and thus, crosses the spread.

1.1.3 WMR Execution Strategy

An effective strategy to approach the benchmark is to spread the order over the whole window, thus making the execution price a weighted function $\hat{p} = \sum_{i=1}^{n} w_i p_i$, where w_i is the weight of transaction i, p_i the price of transaction i. The Time Weighted Average Price (TWAP) Execution Algorithm (EA) cuts the (parent)order up into smaller (child)orders and equally spreads the volume of the order over the whole window. The parent order is the complete order that we want to execute, which can be split up into multiple smaller orders called child orders. The sum of the child orders is equal to the parent order. Besides decreasing market impact, another objective of the TWAP is to execute smaller orders to achieve an average

price over a certain time window, which can be, for example, the fix.

This EA is effective, but achieving the WMR fix remains difficult. However, previous research (Evans, 2018; Husselmann & Kasikov, 2020) showed that the market dynamics during the WMR window can display predictable behaviour. Evans (2018) shows that the momentum of the price of a certain currency pair in the 15 minutes before the WMR window tends to keep going for the rest of the window, and reverses after the closing of the window. It is possible, although very unlikely, that market participants are unaware of the trading opportunity this represents. Other, more compelling, arguments why this anomaly still existed at the time of the research, is that (some) participants are exploiting it, but the effect of their trades on the rates is offset by another countervailing factor. Or it could be possible that exploiting this anomaly is not profitable enough because of transaction costs or risk exposure. In conclusion, there are reasons to believe that data-driven research can improve the FX execution strategies with respect to the Fix benchmark.

1.2 Research Objective

As previously stated, this EA is quite effective in intraday trading. Thus, our focus will go to improve interday trading. While prediction of rates can provide investors with information to help increase execution efficiency and to reduce risk, the FX market's complexity makes FX prediction and forecasting a challenging research topic (Hu, Zhao, & Khushi, 2021). However, older research (Evans, 2018) has shown some predictive possibilities.

Artificial intelligence is being used more in many research fields and practical applications due to continuous development (Hu et al., 2021). These developments have also led to an increasing number of investors applying deep learning (DL) models to forecast FX and stock markets recently (Hu et al., 2021). Previous studies have demonstrated that the non-linear, highly complex relationship of deep learning could predict fluctuations in stock and FX prices (Rundo, 2019; Sirignano & Cont, 2019). The focus of this paper is thus on deep learning models, as they have proven that they can yield better return and accuracy in the field of financial forecasting and prediction than more traditional linear models (Hu et al., 2021; Yang, Zhai, & Tao, 2020). This is in line with the Universal Representation Theorem (URT).

This theory states that when the hidden layers, k, in a standard feed-forward neural network, is large enough, every continuous function that is continuously differentiable can be approximated arbitrarily closely, uniformly over any bounded set by functions realised by neural networks with one hidden layer (Dixon, Bilokon, & Halperin, 2020). What hidden layers and feed-forward networks are, is explained in Chapter 4. The URT is important because essentially a Neural Network (NN) can approximate any continuous function, however, there are some important limitations, mainly being the concern for over-fitting and while the NN can approximate any function, it does not imply that the performance can be generalised on out-of-sample datasets (Hornik, Stinchcombe, & White, 1989).

To improve the trading execution, a DL model that has predictive capabilities regarding market movements of the 4 pm WMR fixing window is constructed. This model uses both currency and 'external' data. Although the FX data is in a high frequency, the other data sets have less frequent data points. The expectation is that this exogenous data can be used to predict market behaviour of the specific currency pair in and around the fixing window. Aspects that will be looked at, for the scope of this thesis:

- Relations with the behaviour of historical and contemporaneous data of the currency pair.
- Relations with the market dynamics of other asset classes (stocks, bonds, etc.). (Cho, Choi, Kim, & Kim, 2016; Turkington & Yazdani, 2020)
- Information embedded in the month-end effects of hedge rebalancing (Noual and Kasikov, 2009).

The aim of this thesis is to develop a DL model that can be used to predict the WM/Reuters fixing rate of a currency pair, using historical and contemporaneous data of the currency pair, as well as other endogenous data. These models can provide traders a prediction of future movements in the market, which can then be used to optimise the execution strategy or execution moment, given the FX orders that need to be traded. A prerequisite is that the output of the model should be actionable. The predictive capabilities of the DL model could be applied in multiple ways:

- To optimise the trading execution, the model can dynamically adjust the EA, so that orders can be timed based on expected future market movements, instead of spreading the child orders equally over time.
- The DL model can be applied to predict the fix before the window starts, so trades can be executed before or after the window to achieve a better rate than what would have been predicted during the window.
- Finally, the model can be used to predict the fix a day into the future, such that non-time-constrained orders can be executed on the day that has the expected most favourable fix.

This thesis will focus on the last application as the implementation within an execution algorithm is out of the scope of this thesis. Thus, the goal of the DL model is to predict the price of the fix a day in advance. As results, due to the special market dynamics of the fix, we suggest that the efficient market hypothesis, during this window, does not hold (Michelberger & Witte, 2016).

1.3 Novelty

Many of the published papers focus on the creation of a novel algorithm (Hu et al., 2021). The data mostly consists of the price and order flows of the asset aimed to

predict. The novelty in this thesis is that a large range of other variables is used as input for a DL model, to predict a specific currency pair.

Although the WMR 4 pm fixing window is one of, if not the most important benchmark rate, not much academic research has been done on the observable market structure around this window (Michelberger & Witte, 2016). This thesis will also contribute to the literature on this topic.

Finally, we will do extensive hyperparameter analysis and tuning, on a DL model that is fitted on (highly noisy) time series data, to get an insight into how different hyperparameters influence the outcome of the model. This, as far as we are aware, has not been done extensively.

2. Study Objective

The aim of this thesis is to investigate if a prediction can be made of the WMR one day in advance, by applying deep learning techniques. Additionally, data wrangling and engineering techniques are applied to a collection of datasets, where especially, the end of month rebalancing data is of interest, to create a data pipeline that transforms raw data into data primed for machine learning. Finally, the aim is to compare different models and test the best performing model on robustness and parameter sensitivity.

The study objective is formulated as following:

Apply data wrangling and engineering techniques on relevant datasets, to prepare the data to be fed into a theoretically backed deep learning architecture, and evaluate the performance, to forecast the WMR fix one day in advance

---

This research is intended as a first step in applying deep learning techniques on (highly noisy) financial time-series data in this organisation. Where we define highly noisy as the phenomena where the exact same data input can result in different outputs of the target variable. Deep learning models are complex and require a lot of training and tuning before they are optimised. The study objective to find an optimised deep learning architecture is limited by time and computing power, which means that, realistically, the outcome will not be the absolute best model for the situation. The study, however, will provide a framework on which more tuning can be applied.

2.1 Research Questions

To achieve the study objective, a main question is formulated. This main question will be answered with the help of several research questions.

Main Question: "What is the forecasting performance of the proposed machine learning models on the WMR Fix one day in advance, when using historical FX, bonds, and equity market data?"

Hypothesis: We believe that the machine learning model, using FX, bonds, and equity market data, will be able to predict price movements, especially when using

end-of-month FX hedge re-balancing data of asset managers and pension funds

To indicate how accurate the machine learning model is, we define performance measurement in section 4.4.

Sub-question 1: "How can the selected models be applied to FX spot market prediction?"

An extensive literature research is conducted in order to get a broad overview of which models have the best potential to perform well on the forecasting of the WMR Fix.

Sub-question 2: "What is the performance and robustness of the models on predicting the FX spot market movements?"

The robustness of the models is tested by measuring convergence variability. The model is trained multiple times on the same data, using different seeds. This randomises the start of the optimisation and could result in divergent outcomes when the model is not stable.

Sub-question 3: "How do the different hyperparameters affect the model performance?"

The best performing model is analysed more deeply and important hyper-parameters are discussed. This results in a clearer view of the mechanics of the model.

2.2 Scope and Limitations

- The research is focused on predicting the EUR/USD currency pair. The method is applicable for other currency pairs, but that is out of the scope of this thesis.
- The DL model can be implemented for the timing of an execution algorithm, however, this implementation is out of the scope for this thesis.
- The ability to accurately predict does not guarantee a profitable strategy. Complex issues regarding exchange matching rules, position constraints, price impact, queue position, latency, and investment mandate constraints all influence the profitability of a strategy. Designing a profitable application for the developed algorithm is out of scope for this thesis.

3. Data, Algorithms, and Software

In this chapter, academic research is summarised about what data could provide insights into the WMR 4 pm fixing window. As well as research about which algorithms have the best predictive capability when using financial time series data.

Not much academic research has been conducted into what data could have predictive capabilities of the WMR 4 pm Fix specifically. So, most cited research will concentrate on a general relationship between certain data and a currency pair, not specifically during the window. However, due to the compression of a large order flow into the window, a special market dynamic is created (Michelberger & Witte, 2016). This could result in a different outcome than described in the literature.

3.1 Data Literature Review

The purpose of this section is to explore the literature for external factors that could have predictive value for currencies.

3.1.1 Stocks and Bonds

Bonds

The International debt securities market makes it possible for borrowers and lenders to get involved in the buying, selling, and issuance of bonds that are denominated in different currencies. In theory, often, the uncovered interest parity (UIP) is assumed. This theory defines that the difference in interest rates between two countries is equal to the relative change in FX rates over the same period. When this theory does not hold, there is opportunity to make a risk-free profit using FX arbitrage.

Although the theory is clear on the UIP, Cohen (2005) state that; "the empirical evidence for this relationship is weak". They find that when a given currency is strong relative to historical averages then generally more debt is issued in that currency, also when a weak currency has the same expected return. The same source finds that more debt is also issued in a certain currency when the long-term interest rate is high relative to other major currencies. Although these conclusions are based on data from 1993 Q3 till 2004 Q4, and shows the reverse causality compared to what we are looking for, the paper does indicate that historical data on debt issuance and bond yield could provide predictive value for a currency pair.

Ang and Chen (2010) find that; "in a no-arbitrage framework, variables that affect the pricing of domestic yield curve have the potential to predict foreign exchange risk premiums." In particular currencies with large changes in interest rate levels tend to appreciate and currencies where the term spread is steep tend to depreciate.

Stocks

Links between the equity and FX markets have been studied thoroughly, but evidence of a strong link is relatively rare (Turkington & Yazdani, 2020). However, Turkington and Yazdani (2020) developed a strategy where currency positions are taken based on the trailing 12-month equity index returns for the particular countries. They found that the differential in recent equity market returns between countries offers a consistent prediction of next month's currency returns. This result is remarkably robust over time to changes in construction and parameters. The proposed motivation behind this is that investors' demand increases for outperforming equity markets. This results in an appreciating exchange rate for countries with the strongest equity return in the previous year.

Djeutem and Dunbar (2018) also found empirical evidence in favour of equity returns as exchange rate predictor. The paper estimates that the US expected excess equity returns are responsible for 70-90% of the expected exchange rate changes for Canada, Japan, and the UK. Smales and Kininmonth (2016) also summarises multiple papers that found relations between equities and FX price movements. This result is promising for the objective of this thesis, as equity returns could provide predictive value for a currency pair.

3.1.2 Hedge Rebalancing

When investment managers invest in international investments, they can hedge the currency exposures associated with the exchange rate risk, e.g. in order to match the benchmark for their fund. Typically this hedge is rebalanced at the last business day of each month and during the WMR 4 pm window (Melvin & Prins, 2015). Melvin, Pan, and Wikstrom (2020) show that a mid-month rebalancing, which is 11 days before month end, realises 0.97 times the average costs. The reduction in costs is the result of seasonalities in the spreads. The same paper finds that Fridays and quarter-end, especially, have higher rebalancing costs.

In contrast to the previous section, where higher performance in the equity market indicated higher performance for that particular currency. Hau and Rey (2006) finds that higher returns in the home equity markets compared to foreign equity markets are associated with the depreciation of the home currency. Melvin and Prins (2015) find similar results, they find that around the end-of-month Fix, the equity market out-performance of a certain market is highly significantly correlated with a currency depreciation in the hours leading up to the Fix. This depreciation partially reverts after the Fix. This hedging of the exchange rate adjustment is most notable for the currencies of the largest equity markets by capitalisation; the Eurozone, Japan, and the U.S. The same paper also indicates that despite the fact that the behaviour of hedgers is known to the rest of the market, and predictable in advance, the short-run

effect remains significant.

Melvin and Prins (2015) quantify the result further, they found that a 10% appreciation in equity prices in a month leads to 14 basis points of home currency depreciation during the WMR end-of-month Fix. This result is statistically significant and although the effect seems quite small, the currency depreciation is concentrated in the hour leading up to the WMR Fix. The same paper finds evidence for price reversion after the Fix on all days, because of intra-month hedge rebalancing. To quantify this further, the paper found that 72% of the price movement in the hour leading up to the Fix is reversed before noon of the next day. Making the reversion effect on end-of-month days, two times larger than other days.

Hau and Rey (2004) also finds that exchange rate returns, equity portfolio flows and equity market returns are consistent with a dynamic rebalancing of foreign equity positions by global investors.

A seemingly opposing view is publish by Ritchie (2021) on *Bloomberg*. They report that the month-end models that look to predict the re-balancing flows of the FX exposure have deteriorated in performance since 2021. Therefore, some banks discontinued their models, while others chose to update them. These 'updated' models are less focused on the final day of the month and now have an extended period to three days before the end-of-month, where it aims to predict re-balancing flows. This seems to significantly increase returns, as the updated model is reported to have an annualised return of 5.74% when back-dated, compared to a 0.38% return of the former algorithm.

Summarising, this means that during the end-of-month period, data of the stock returns could provide serious predictive benefit for the model. Especially because, during the window, the whole month's worth of hedging is concentrated in a small time frame.

3.2 Algorithms Literature Review

This section aims to provide an overview of best-performing DL models that were applied to the relevant context in earlier published papers.

Many different models have been suggested for Forex and financial time series prediction in general, but Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) are commonly used (deep) neural network models (Yang et al., 2020). Each neural network has its strengths and weaknesses, it is important to choose the right algorithm for the right task (Islam, Hossain, Rahman, Hossain, & Andersson, 2020). To take advantage of the different strengths, it is popular to construct a combination model from these neural networks to enhance the performance, such as a hybrid CNN and LSTM model (Yang et al., 2020). Jain, Gupta, and Moghe (2018) found that the CNN-LSTM architecture is superior to an LSTM and CNN in stock price movement prediction. An explanation is that such a hybrid model can take advantage of the superior feature extraction of CNN and the preferable time series prediction capabilities of LSTM.

Li et al. (2019) improved upon this architecture by adding an attention mechanism, further increasing prediction accuracy. Interestingly, more recently, van der Meulen, Vermeulen, and Tetereva (2021) found that, in their context, the LSTM standalone outperforms a.o. an Attention-LSTM hybrid. Suggesting that a hybrid, or more sophisticated model, does not necessarily outperform the more simple architectures.

3.2.1 Feature Extraction

There is a large variety of variables that can be used for making market predictions. Raw price data, data from other markets that are connected to the target market, and technical indicators extracted from historical data are some examples. Because of the diversity of the information, it is not straightforward to accumulate the data in a way that the prediction algorithm can use it. Therefore, ideally, an automatic approach is used to dissect the useful features from the different data sets that could be beneficial for Forex prediction.

Hoseinzade and Haratizadeh (2019) designed a feature extraction algorithm based on a CNN and Yang et al. (2020) expanded on that model. The CNN is designed in the computer vision and image processing field and is more recently being applied in extracting market features. Experimental results show high effectiveness and this research will thus deploy the CNN for feature extraction.

3.2.2 Time Series Forecasting

RNN is a deep learning methodology that is mainly used for time series analysis because it has feedback connections inside the network that allow past information to remain. This results in superior non-linear and time series prediction capabilities, compared to conventional artificial neural networks (Islam et al., 2020). An LSTM network, or just LSTM, is a deep RNN model that is built from LSTM units. The advantage of an LSTM is that it assigns weights to data, which extends the RNN's memory. In this paper, LSTM is used for the time series forecasting segment of the algorithm. An LSTM uses hidden neurons, which makes it very efficient in time series prediction (Islam et al., 2020). The same reason also makes it a popular choice as a Forex prediction method in the literature (Hu et al., 2021). We develop an LSTM because it has been shown to outperform other RNNs on tasks involving long time lags (Gers, Schraudolph, & Schmidhuber, 2002).

3.2.3 Attention Mechanism

Attention mechanisms were initially developed to improve neural machine translation (Luong et al., 2015). When applied to time series forecasting, the attention mechanism can adaptively select relevant driving series and capture long-term temporal dependencies (Qin et al., 2017). NNs using this mechanism can outperform state-of-the-art methods for time series prediction (Li et al., 2019; Qin et al., 2017) However, Qin et al. (2017) also warn that this mechanism might not be suitable for time series forecasting. When there are multiple driving exogenous series are available, as can be the case with time series, it can result in difficulties for the network to explicitly select driving exogenous series.

3.3 Data Sets

The following data sets are used, consisting of data since 01-01-2000:

1. A FX data set from New Change, consisting of the EUR/USD open, close, high, and low price, of the pair in minute granularity.

2. EuroStoxx50 and SP500 datasets from Bloomberg, consisting of the close prices in daily granularity.

3. Bond yield dataset of the US and EU 10 year generic bonds, in daily granularity. This data is used as indication of interest rates.

4. Bond return dataset (LUATTRUU Index) from Bloomberg, consisting of the US Treasury Index, which measures US dollar-denominated, fixed-rate, nominal debt issued by the US Treasury and the Pan-European unhedged government total return index (I02513EU Index).

5. Dataset containing the daily WM/Reuters fix benchmark. Obtained from Bloomberg.

3.4 Software

Preprocessing, data handling, data visualisation, and model creation are conducted in Python (version 3.8). The scientific libraries, Pandas, and Numpy are extensively used. For model creation the Scikit-learn, Tensorflow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019) libraries are used. For model tuning the AWS cloud is used.

4. Methodology

In this chapter relevant research is dissected and used as a solid methodology, this will provide a theory-ingrained backbone to the thesis and algorithm. This backbone is used at the end of the chapter to answer sub-question one:

Sub-question 1: "How can the selected models be applied to FX spot market prediction?"

We have the following dilemma for the Methodology section. This section goes over the functioning of neural networks, and how the different layers and gates work. This, however, is not the central matter of this thesis, and this thesis does not provide new insights into this knowledge field. This also is, by no means, the expertise of the author.
Nevertheless, we do think that it is crucial to understand the complexities and the general fundamentals of neural networks for this thesis. For this reason, the chapter is not placed in the Appendix.

4.1 Neural Networks

Before going into the technical side of the neural network that is built during this thesis, we will go over how a basic neural network functions and over some basic definitions.

The input of an NN is features. A feature is a column of data values, that the engineer of the NN thinks will provide information that the NN needs to make a prediction. For example, the close prices of a stock index when predicting the stock price of that index. The NN itself consists of multiple layers of neurons. The network always starts with the input layer, where every neuron represents a feature. So when the input data consist of four features, the input layer needs four neurons to process this data.

The last layer of the NN is the output layer. The number of neurons in this layer depends on the number of outputs the engineer requires from the network. When only one variable needs to be predicted, the final layer requires only one neuron. But when the network has to identify a picture and decide if it contains a cat, a dog, or neither, then the network requires three neurons in the output layer.

The layers between the input and the output layer are called the hidden layers. These hidden layers can theoretically have any number of neurons, but, more neurons require more computing power and do not always increase performance. The neurons of each sequential layer are connected, but the neurons within a layer are not, as can be seen in 4.1. These connections are the weights, which dictate how much information of a neuron gets passed on to the next neuron.

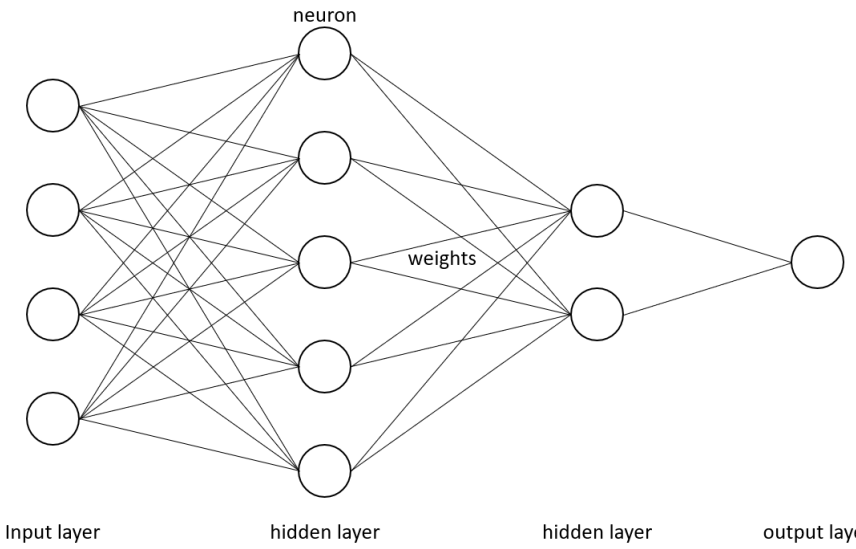

Figure 4.1: The architecture of a basic neural network with two hidden layers

To optimise the NN, the weights get changed between each training round, where the loss function of the output dictates if the change in weights is beneficial or not. By training and sub-sequentially changing weights, the NN can 'learn' which information is important to predict the target variable.

4.2 The Models

This section provides the framework of the different models. During this thesis, three different Neural Networks are built based on the literature from 3.2. The models are an LSTM, a CNN-LSTM, and an Attention-CNN-LSTM, making it possible to compare more sophisticated and hybrid models with a simpler model.

4.2.1 Input Tensor

A tensor is the way the input data of the NN is stored. For example, when a scalar is used as input it is a 0-dimension tensor, and an array and a matrix are respectively a 1-dimensional and 2-dimensional tensor. The difference is that a tensor can take advantage of hardware acceleration, making the computations more quickly. The tensor used for the models in this thesis is a 2-dimensional tensor, where the dimensions represent the input dimension (features), the time steps, and the batch size. An example of such a tensor is visualised below in figure 4.2.

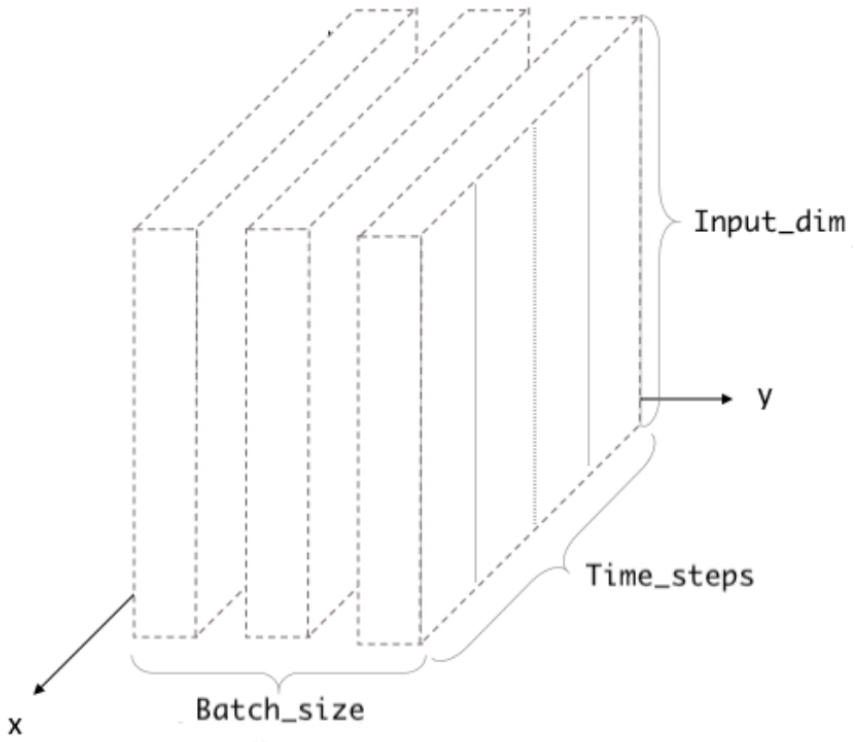

Figure 4.2: An example of a three-dimensional input tensor (Govani, 2020)

Time steps is defined as the number of historical observations that are used to make the current forecast. In this study the time steps is set at 260, or a little more than a year of data. This is discussed further in subsection 5.2.2. This means that for a prediction, the 260 previous observations are used as information.
The batch size is the number of training examples that are utilised in one iteration, which is discussed further in subsection 5.6.2. The batch size is set by an tuning algorithm. The input dimension or number of features used as input is 33, which is discussed in Chapter 5.

Input dimension

Gunduz, Yaslan, and Cataltepe (2017) find that a CNN presented with specifically ordered features outperforms a CNN that is presented with randomly ordered features. For this reason, the Pearson Product-Moment Correlation Coefficient (PPMCC) is used to order the features. PPMCC is used to determine the degree of linear correlation between two variables, the formula of PPMCC is found in equation 4.1.

$$PPMCC_{p,q} = \frac{\sum_{t=1}^{n}(x_{p,t} - \bar{x_p})(x_{q,t} - \bar{x_q})}{\sqrt{\sum_{t=1}^{n}(x_{p,t} - \bar{x_p})^2 \sum_{t=1}^{n}(x_{q,t} - \bar{x_q})^2}} \tag{4.1}$$

Where $\bar{x_p}$ and $\bar{x_q}$ are the average values of the p-th and q-th feature and $x_{p,t}$ and $x_{q,t}$ are the values of the p-th and q-th feature on the t-th day index. n is the number of data points, like the number of trading days in a sample. When PPMCC < 0, then the correlation is negative, consequently when PPMCC > 0 then the correlation is positive.

The batch size and time steps of the tensor are discussed in subsection 5.6.2 and 5.2.2 respectively.

4.2.2 CNN

A CNN consists of multiple layers, such as the input layer, output layer, the convolutional layer, the fully connected layer, and the pooling layer.

Convolution Layer

The convolution layer is designed for convolution operations on the input data, which can be considered as a filter for the input data. The filter works the following way: there is an input layer, $L-1$, which is an NxN matrix, followed by a convolutional filter, which is an FxF matrix. Layer L is the output layer, and the input of this layer is calculated by applying the filter to the input data, which is visualised below in Figure 4.3.

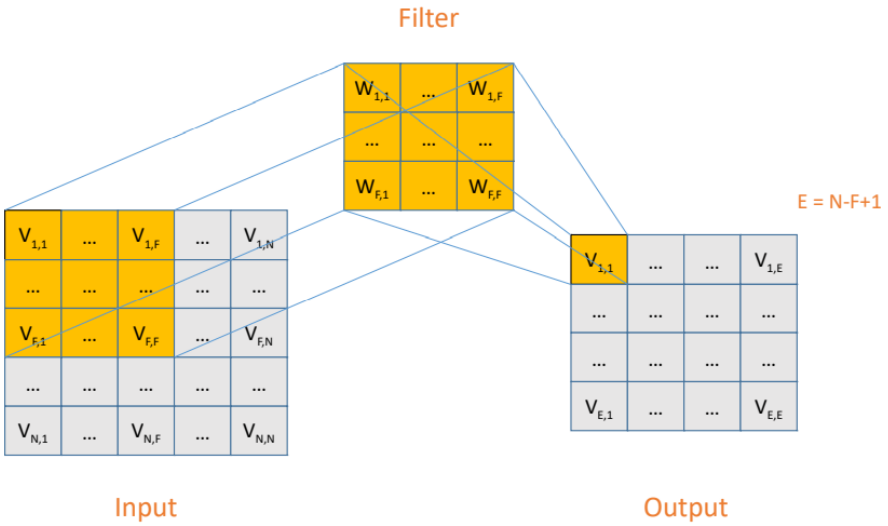

Figure 4.3: Applying a FxF filter to the NxN input layer, to get outcome $v_{1,1}$ in the output layer (Hoseinzade & Haratizadeh, 2019)

The values of the output layer are calculated by the filter, this is displayed in Formula 4.2:

$$v_{i,j}^{l} = \delta(\sum_{k=0}^{F-1}\sum_{m=0}^{F-1} w_{k,m} V_{i+k,j+m}^{l-1}) \tag{4.2}$$

Where $v_{i,j}^{l}$ is the output value at row i, column j of layer l, $V_{i+k,j+m}^{l-1}$ is the input of the filter, $w_{k,m}$ is the weight at row k, column m of the filter and $\delta$is the activation function. Equation 4.3 is a commonly used non-linear activation function called ReLU, which we will go over more in-depth in subsection 5.5.2.

$$f(x) = max(0, x) \tag{4.3}$$

Pooling Layer

Deep models, like CNN, generally, have many parameters. This makes the models susceptible for overfitting (Hoseinzade & Haratizadeh, 2019). The pooling layer is responsible for subsampling the data, which reduces the computational costs of the learning process but also helps manage the overfitting problem. In a pooling layer, the input values are transformed into only one value. This conversion reduces the input size, and so the number of parameters that the model has to learn, which reduces the risk of overfitting. The most common type of pooling is Max Pooling, where simply the maximum value in a certain window is chosen.

Fully Connected Layer

The fully connected layer has the purpose to convert the, in previous layers extracted, features to the final output. The relation is defined in the formula below.

$$v_i^j = \delta(\sum_k^v v_k^{j-1} w_{k,i}^{j-1}) \tag{4.4}$$

Where, $\delta$is again the activation function, $w_{k,i}^{j-1}$ is the weight of the connection between neuron k from layer $j-1$ and neuron i from layer j. The value of neuron i at the layer j is represented by v_i^j (Hoseinzade & Haratizadeh, 2019).

4.2.3 RNN

The advantage an RNN provides over other Artificial Neural Networks is its memory feature, which makes it better suited to extract temporal patterns in data (Chung & Shin, 2018). In an RNN the specific timestamp or position in the sequence is preserved. Each layer has a single input corresponding to the specific position in the sequence, instead of having a sequence in a single input layer. Depending on the position in the sequence, the input of a timestamp is allowed to interact with the hidden layer h_t. This is repeated for all timestamps in the sequence, which gives the neural network the name Recurrent. Each layer takes as an additional argument the previous timestamp's hidden state h_{t-1}. This results in Formula 4.5.

$$h_t = \delta(W_h[h_{t-1}, x_t] + b_h) \tag{4.5}$$

Where, W_h is the hidden weight matrix, x_t is an input vector and b_h is a bias. This produces hidden activation h_t, and the prediction $\hat{y}_t$ is given by Formula 4.6. The flow of an RNN process is shown in figure 4.4. When the RNN cannot extract consequential temporal patterns, then it can happen that the bias b_h becomes the main output of the hidden layer, this is called saturation.

$$\hat{y}_t = \delta(W_y^T h_t b_h) \tag{4.6}$$

The weights and biases determine the mapping from x_t to $\hat{y}_t$. During the training of the network, the goal is to optimise the value of the loss function, by changing the weights and biases of the network. This is achieved by the gradient descent-based training which starts at the gradient with respect to the output and propagates the derivatives backwards through the network using the chain rule, to find the local minima (Rumelhart, Hinton, & Williams, 1986). In RNN the gradient descent

training of the network is called the backpropagation through time because the error derivatives are backpropagated through the network and through time via the recurrent connections (Werbos, 1990).

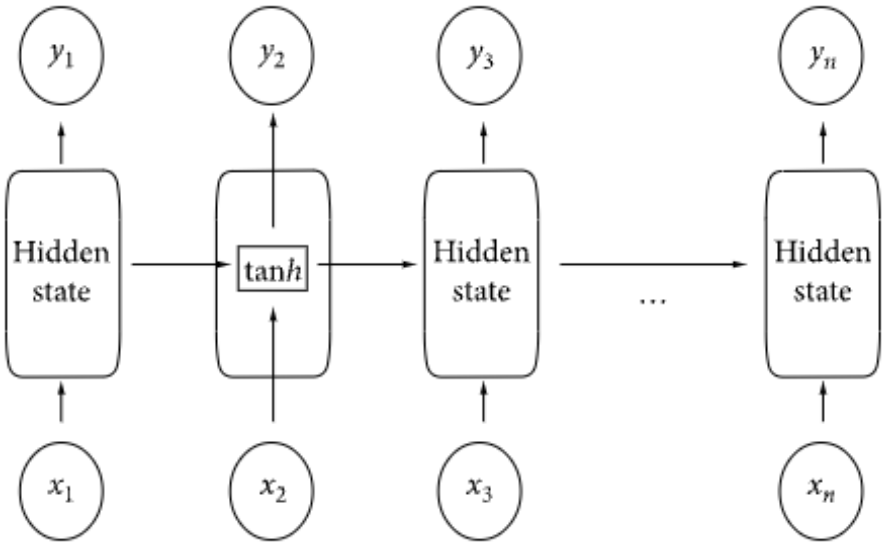

Figure 4.4: Flow of an RNN process (Jung & Choi, 2021)

Typically, RNNs use activation functions such as the logistic sigmoid (σ) or the hyperbolic tangent (tanh). The derivative of both these functions lies in the interval [0,1], this results in a smaller squeezed gradient with respect to a weight (Hochreiter, 1998). Because the propagating backwards uses the chain rule, the gradient gets smaller and smaller the further the network goes back in time. This results in a vanishing gradient for earlier layers, or that new layers become unproportionally large. This vanishing and exploding gradient is a known problem for RNNs. A way to overcome this problem is by using an LSTM architecture which has an additional pathway called the cell state. The cell state allows the LSTM to remember gradients through long time sequences (Hochreiter & Schmidhuber, 1997).

4.2.4 LSTM

LSTM networks are specially designed for learning long-term dependencies (Hochreiter & Schmidhuber, 1997). The network has an additional pathway, the cell state, which is updated by three different gates. The gates are described later in this section. These gating units control how the input, x_t and the recurrent input h_{t-1}, change the cell state in order to produce an update cell state C_t. The logistic sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, is important in the gating operations of the LSTM cell. It controls the information flow through the gates, by taking the weighted inputs and recurrent inputs and maps them to the [0,1] interval. Where 0 means no information is passed through, and 1 means that all information is passed through the gate. The flow process of an LSTM network is visualised in figure 4.5.

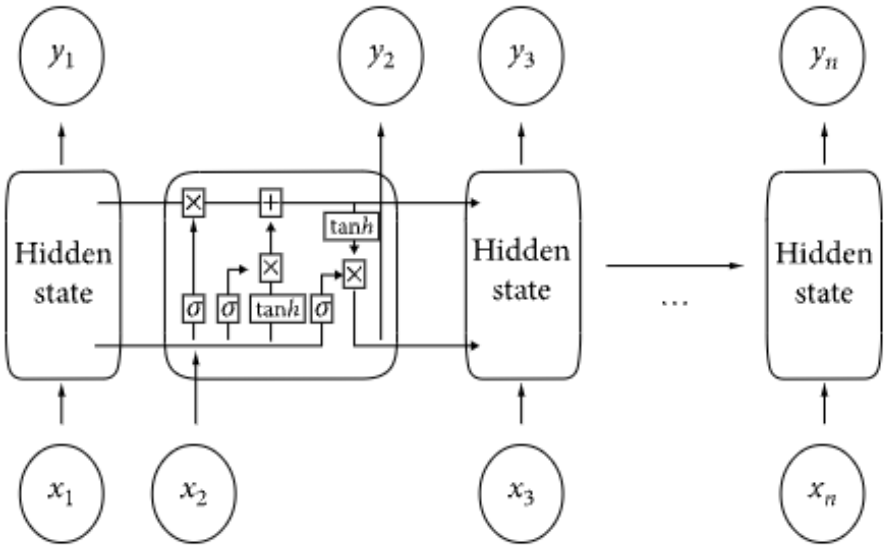

Figure 4.5: Flow of an LSTM process (Jung & Choi, 2021)

The Input Gate

The objective of the input gate is to preserve the information of the cell state, which is a representation of the information from previous time steps. It selectively updates the cell state with new information. The gate i_t employs a sigmoid to control this information flow:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{4.7}$$

And to update the cell state with new information a new set of candidate value $\hat{C}_t$ is generated, generally a hyperbolic tangent is defined by equation 4.8.

$$\hat{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{4.8}$$

The Forget Gate

The purpose of the forget gate is to determine which cell state is passed on to the next time step. This is again achieved by a sigmoid gatekeeper function:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{4.9}$$

Where f_t is the forget gate. Then, f_t is multiplied by C_{t-1} to select the information that is passed onto the next step, 0 implying no information, and 1 implying that all information is kept.

The Output Gate

This gate determines the prediction of the LSTM, using the current input x_t and cell state C_t. To produce a version of the cell that is scaled to the interval [-1,1] a hyperbolic tangent is applied to the values of the current cell state:

$$C_t^* = tanh(C_t) \tag{4.10}$$

The output gate determines which information to pass to the output layer and thus to the next time steps in the hidden state h_t:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4.11}$$

To construct the output of the current timestep, o_t and C_t^* are multiplied:

$$h_t = o_t \circ C_t^* \tag{4.12}$$

h_t represents the recurrent input at time t+1 and functions as a basis for the prediction at time t. The $\circ$ operator stands for a composition of functions. If two functions $g : X \to Y$ and $f : Y \to Z$, then f $\circ$ g is a function from X to Z. For example, for x $\in X, (f \circ g)(x) = f(g(x))$ holds.

New Cell State

The new cell state, C_t, is based on the input and forget gate and is obtained by updating the old cell state, C_{t-1}, with the new candidate values from $\hat{C}_t$. The updated cell is passed on to the next time step, $t + 1$:

$$C_t = f_t \circ C_{t-1} + i_t \circ \hat{C}_t \tag{4.13}$$

4.3 Attention Layer

Luong et al. (2015) have proposed two different attention categories, global and local. In this study, we focus on the global category, also known as the Luong-style attention.

At time step t, the attention layer takes hidden state h_t as input, with the objective to derive a context vector c_t. This vector captures relevant source information to help predict the target y_t. The idea of a global attention mechanism is to consider all the previous hidden states when deriving context vector c_t. A variable-length alignment vector a_t, whose size is equal to the time steps in the input, is derived by comparing the current hidden state h_t, with each respective input hidden state $\bar{h_s}$:

$$a_t(s) = align(h_t, \bar{h_s}) = \frac{exp(score(h_t, \bar{h_s}))}{\sum_{s'} exp(score(h_t, \bar{h_s'}))} \tag{4.14}$$

Where score is defined, in this study, as:

$$score(h_t, \bar{h_s}) = h_t \cdot \bar{h_s} \tag{4.15}$$

This results in a layer that, at each time step t, infers a variable-length alignment weight vector a_t, based on the current target state h_t and all input states $\bar{h_s}$. Than a global context c_t is computed as the weighted average according to a_t, over all input states. This process is visualised in Figure 4.6.

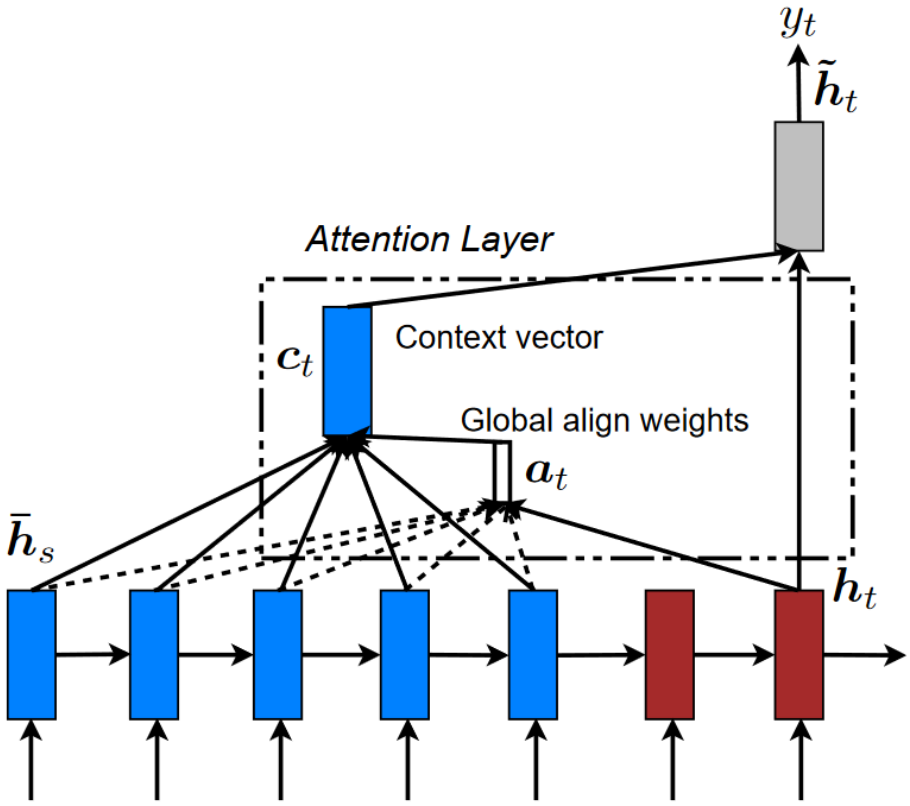

Figure 4.6: Attention Mechanism (Luong et al., 2015)

4.4 Operationalisation of Performance

The purpose of this section is to operationally define performance, we defined a cost function for this purpose. This function is optimised during the training of the model. A well-performing cost function is crucial, as the model is trained on this metric. This function is also called the loss function, or objective function.

4.4.1 Loss function

The loss function evaluates the fitness of $\hat{y}$ to $f(x)$, or of the predicted values on the actual values. The difference between $\hat{y}$ and $f(x)$ are called residuals and can be calculated in several different manners. In this paper, the loss is calculated by the Huber loss function, which is defined below in formula 4.16.

$$L_\delta(\hat{y}, f(x)) = \begin{cases} \frac{1}{2}(\hat{y} - f(x))^2, & \text{for } |\hat{y} - f(x)| \leq \delta \\ \delta(|\hat{y} - f(x)| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{4.16}$$

Where δ is the threshold where the Huber loss function changes from a quadratic to a linear loss function. In this study, the δ is set at 1.0, as this is the default in the Tensorflow package. The results of this loss function are quadratic for small residuals and linear for large residuals. This is desirable because the model gets punished harshly for wrong predictions, similar to the Mean Squared Error (MSE) loss function. however, when there are outliers in the data then the model gets

punished less harshly, similar to a Mean Absolute Error (MAE) loss function. This makes the optimisation less sensitive to outliers.

4.4.2 Validation Loss

The dataset is split into three different sets, the training-, validation-, and test-set. This will be discussed in 5.2. During training, the performance of the NN is measured, and thus optimised on the validation loss. This increases generalisation and avoids overfitting on the training set. The best-performing model on the validation set is evaluated on a held-back test set.

4.4.3 Benchmark, and Performance Metrics

The model performances are measured on the test set. As, performance reporting metrics, the MAE, MSE, and Theil's U are used and compared to a naive prediction, which is the benchmark. A naive prediction is using the WMR Fix at time t as prediction for time $t+1$. Meaning, using the WMR Fix value of today as prediction for tomorrow. The equations for each metric are formulated below.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{4.17}$$

Where y_i is the prediction of sample i, x_i is the actual value of sample i, and n is the total number of data points. This metric measures the absolute difference between the predictions and actual values, and averages it out over the whole dataset. Measuring the expected residual per prediction regardless of the error being negative or positive.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \tag{4.18}$$

The MSE is quite similar, yet, there is an important difference. This metric squares all residuals, meaning, that outlier predictions, where the predictions are far from the actual values, are weighted larger than with the MAE, due to the squaring of the errors. A disadvantage of this metric is that when the model makes one bad prediction, or if the data has outliers that the model does not generalise well on, it results in a large MSE.

$$Theil'sU = \sqrt{\frac{\sum_{t=1}^{n-1}(\frac{y_{t+1}-x_{t+1}}{x_t})^2}{\sum_{t=1}^{n-1}(\frac{y_{t+1}-x_t}{x_t})^2}} \tag{4.19}$$

The Theil's U is a relative accuracy measure that compares the forecasted result with naive forecasting. When the outcome $U < 1$, then the forecasting technique is better than guessing, when $U = 0$, it is just as good as guessing, and when $U > 1$, the forecasting technique is worse than guessing.

4.5 Data Wrangling

One of the most crucial, and most time-consuming, aspects of data analysis is data wrangling (Kandel et al., 2011). Data wrangling is putting the data into a format

usable for downstream analysis tools, which can be anything from data visualisation to model fitting. During data wrangling it is important to look at data quality issues, missing data, inconsistent values and unresolved duplicates. But also, deciding if the data has enough quality, and how the data has to be transformed and cleaned to bring it to a usable state for the model.

This section goes over the data model of the thesis to set a framework of what data is necessary and in which form it is usable. This section also aims to provide an overview of the data and data structures and how this data is transformed to be usable.

4.5.1 Data Model

A data model, for this thesis, is a conceptualisation of the data that is necessary for the, in this case, algorithm. It visualises how the data is related to each other and what elements of the data sets are used by the model. This framework is helpful during the data wrangling phase, as it defines what the data should look like after the wrangling. The main data model is displayed in table: 4.1, where a column can represent multiple variables. These variables are further dissected in subsequent subsections and presented in tables 4.2, 4.3 and 4.6.

Table 4.1: Data Model: Conceptual Overview

Date	FX variables	Index and Bond variables	End-of-Period variables	Hedge re-balance variables	Target variable values
2000/01/01	...	...	...	...	...
...	...	...	...	...	...
2021/12/31	...	...	...	...	...

All data sets are linked through the date variable. The FX variables, Index, and Bond variables in the table are comprised of their respective daily returns among other variables. The end-of-period variables are represented by one column, however in the model each month, quarter and year have their own feature to indicate when the end of the period occurs. This variable is explained more in-depth in the feature engineering chapter 5.3. The hedge re-balance column represents the estimated data of the hedge re-balance flows that have to occur in that particular month. This variable is also discussed more in-depth in section 5.3. The final column represents the target variables.

FX Features

The FX column in the conceptual overview consists of the daily "EUR/USD" FX performance. To summarise the properties of each day, variables are added like the standard deviation of the rates within this day.

Table 4.2: Data Model: FX Features

Date	FX performance "EUR/USD" daily	daily FX data std.dev	daily FX data high-low spread
2000/1/1	...	...	...
...	...	...	...
2021/12/31	...	...	...

Index and Bond Features

The Index and Bond column in the main overview consists of the daily returns of the US and EURO indexes, as well as the daily returns of the US and EURO bonds. Then a variable is added that represents the compounded return of each index and bond. The aim of the added variable is to better capture the possible re-balancing flows at the end of the week, month, quarter, and year.

Table 4.3: Data Model: Index and Bond Features

Date	US index and bond daily return	EURO index and bond daily return	compounded return per month/year for each index/bond
2000/1/1	...	...	...
...	...	...	...
2021/12/31	...	...	...

End-of-Period Features

The end-of-period variables are required to give an indication to the NN of when the end of, for example, a month is approaching. It is important that the NN has an indication of when certain periods end, as the hedge re-balancing flows have to be re-balanced before the start of the next period. This feature does not occur in the data, and as such has to be engineered.

Table 4.4: Data Model: End-of-Period Features

Date	End-of-Month	End-of-Quarter	End-of-Year
2000/1/1	...	...	...
...	...	...	...
2021/12/31	...	...	...

Hedge Re-balance Flows Feature

The hedge re-balancing flows variable has to quantify which net hedge flows are expected at the end of each period. The net hedge flow means that the hedge re-balancing flows of the EU and US are cancelled out against each other, resulting in the net hedge re-balancing flow. This flow is calculated for every day of the month cumulatively, and 'resetting' each start of the month, indicating that all hedges should have been re-balanced.

Table 4.5: Data Model: Hedge re-balance flows

Date	Net Hedge Flow
2000/1/1	...
...	...
2021/12/31	...

Target Variable

The final column in the main data model represents the target variable. The target variable is the variable that the algorithm will predict. Important is to keep in mind the stoppage time, so no information 'leaks' into the future. As the model uses data that is measured end-of-day, no prediction can be made of the WMR that is measured during that same day. When the model is trained in using information at time t, it would use information that is only available in the future, after the WMR of that day is already established. Therefore, each day, at the end of the day, the WMR fix of the next day is predicted.

Table 4.6: Data Model: Target Variable

Date	Target variable: WM/R + 1 day
2000/1/1	...
...	...
2021/12/31	...

Overview

Essentially the process looks like the following: all data is cleaned and necessary features are added, then all datasets are merged together. The next stage is feature engineering, where the 'end-of-period' and 'hedge-rebalancing' features are added, based on domain knowledge. This complete dataset is used as input for the neural network, where hyperparameters are set that fit our specific objective. Eventually, this NN will make a prediction for the target variable. This process is set out in Figure 4.7.

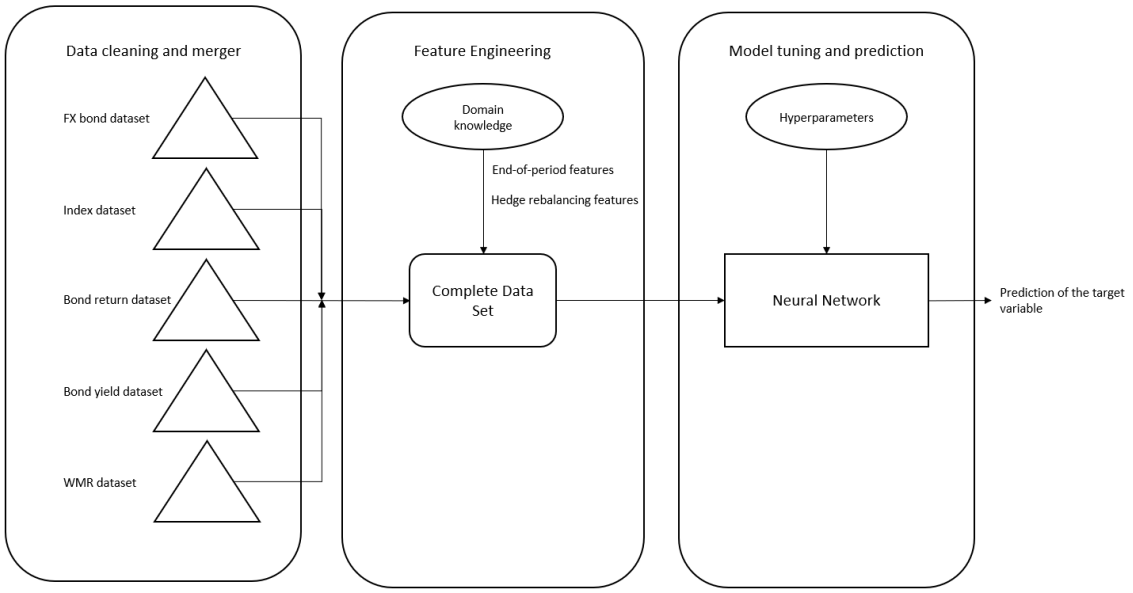

Figure 4.7: Overview data model to predicted value

4.5.2 Data Cleaning

All data is fairly clean. There are no unexplainable outliers or quality issues. Therefore, the data cleaning process is relatively straightforward. The five data sets have most cleaning steps in common, these are displayed in Figure 4.8.

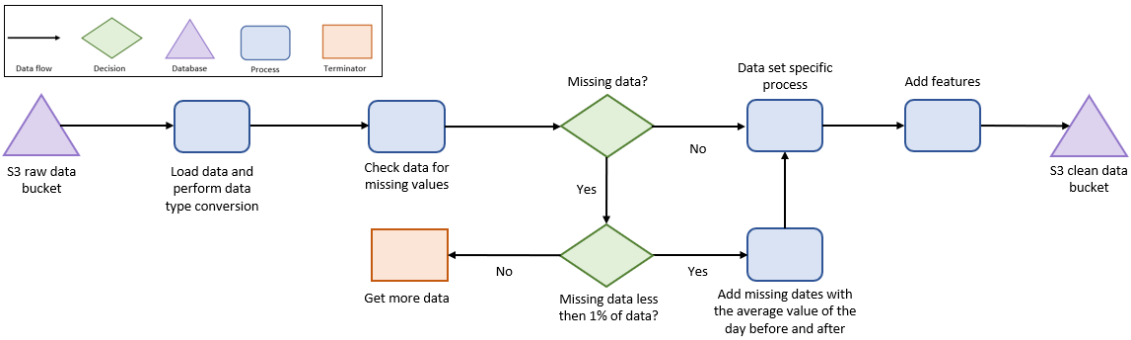

Figure 4.8: Data flow chart of the cleaning process

Additionally, the equity and bond return datasets have in common that we add the daily, monthly, and yearly cumulative return features.
Because the New Change FX dataset is in a minute granularity it requires data processing as well. We take the mean price of each day to set the EUR/USD value of each day. To capture the characteristics of the EUR/USD price development during a day, the high-low spread of the day, the EUR/USD price standard deviation, and the high-low spread standard deviation are added as variables. Where the high-low spread is the difference between the highest price and lowest price during a certain time period.

4.6 Methodology Conclusion

This section brings together the outcomes of previous literature review subsections 3.1 and 3.2, and the current chapter, in order to answer sub-question one.

Sub-question 1: "How can the selected models be applied to FX spot market prediction?"

In terms of data, both equity and bond data have the potential to be valuable in the prediction of FX spot market behaviour. Most promising, however, is data on the end-of-month hedge rebalancing flows.
The most promising models to be applied to this data and forecast the WMR fix are neural networks. Although the literature is divided on which neural networks have the best potential in this situation, the LSTM architecture, the CNN-LSTM and Attention CNN-LSTM hybrid architectures seem to be the best choices.

5. Solution Design

In this chapter, the data is visualised, analysed, and prepared for the model. Stationarity is discussed and several statistical tests are performed. This chapter discusses the design of the DL algorithm and the different hyper-parameters that need to be set to optimise the DL model.

5.1 Data Analysis

To get more insight into the data, we decompose the WMR variable utilising the STL (Seasonal-Trend decomposition using LOESS) filtering procedure, proposed by Cleveland, Cleveland, McRae, and Terpenning (1990). The STL consists of a sequence of LOESS smoothers, where LOESS stands for LOcal regrESSion, and it decomposes a seasonal time series into three components; trend, seasonality, and remainder (Resid). Where the trend is the low-frequency variation in the data together with long-term nonstationary changes in level. The seasonal component is the variation in the data at or near the seasonal frequency. Lastly, the remainder is the variation in the data that is not yet captured in the previous two components.

When we decompose the WMR fix time series, it is clear that this data is non-seasonal, as can be seen in figure 5.1.
There is no clear long-term trend, no seasonal short-term cycle, and large residuals. The decomposed data is not used as input for the neural network. As, Ouyang, Ravier, and Jabloun (2021) show that although the STL decomposition as a preprocessing step of the data can benefit forecasting using statistical methods, it harms the machine learning ones.

5.1.1 Delta WMR Fix

In order to predict the WMR fix at $t+1$, we aim to predict the difference, or delta (Δ), between the WMR fix of t and $t+1$, making the naive prediction of the WMR delta equal to 0. After which we add this delta to the WMR fix of t to make the prediction of the WMR fix of $t+1$. This method is called differencing and results in the target variable being the delta WMR fix. Besides being a more accurate target variable, this could also result in the target variable being normally distributed. Yet, based on the D'Agostino and Pearson's test (D'Agostino & Pearson, 1973), that resulted in a p-value of 3.17758e-109, we reject the null hypothesis that the WMR delta observations come from a normally distributed population. The distribution

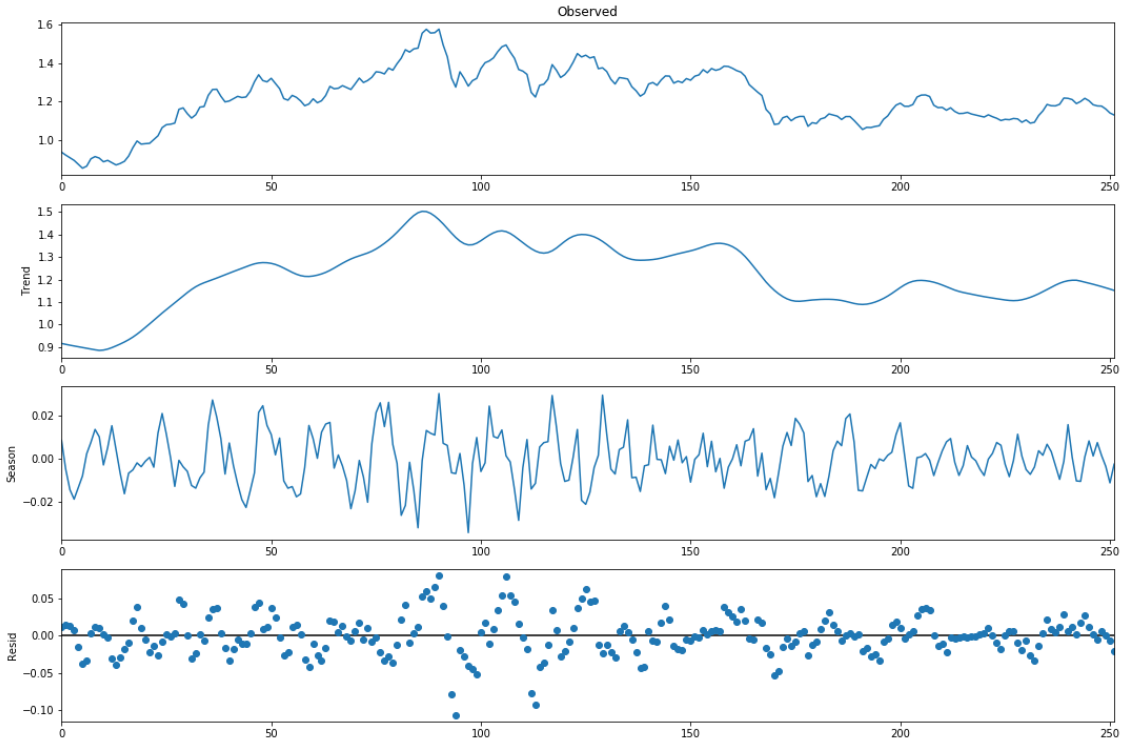

Figure 5.1: WMR decomposition

of the WMR delta is shown in 5.2.

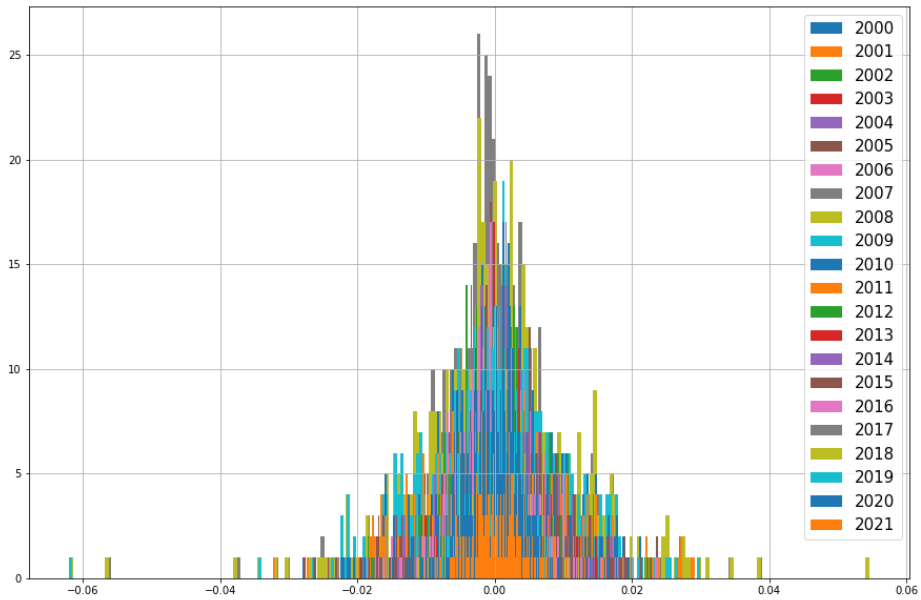

Figure 5.2: WMR delta distribution

Stationarity

Many engineering studies overlook verifying essential diagnostics such as the Dickey-Fuller test for testing stationarity of the time series (Shivarova, 2021). Informally, stationarity is when the auto-covariance is independent of time. It is decisive in characterising the prediction problem and the choice whether to use a more advanced architecture. We tested the target variable on stationarity and this resulted in a p-value < 0.05, rejecting non-stationarity based on this test, as shown in Table 5.1.

Table 5.1: Dickey-Fuller test for stationarity

ADF Statistic: -71.072384	
p-value: 0.000000	
Critical Test Statistics Values:	
1%:	-3.432
5%:	-2.862
10%:	-2.567

Other input variables are not stationary, this, however, is not an issue. We use LSTMs which have the ability to take into account temporal dynamics, since these models exhibit dynamic autocorrelation structure (Ryll & Seidens, 2019; Shivarova, 2021). It is important to not use a plain RNN, as they are not suited for non-stationary time series modeling (Shivarova, 2021).

5.2 Data Preparation

5.2.1 Data split

The data split is according to a straightforward 0.8/0.1/0.1 split. Meaning that 80% of the data set is used for training, 10% for validation during training, and 10% as a test set for the model. We used a larger portion of the data for the training set than what might be usual, the reason being that the total data set is relatively small for the training a neural network. The data set is not shuffled and is split according to time order, meaning the 80% that is the training data is the first 80% of the data set, ordered by time. Random sampling is not a good idea as this causes look-ahead bias. Each observation in the time series is dependent on previous observations. The ordering of the observations therefore matters and the data is not i.i.d. Thus, the split has to be in the training, validation, test order. The look-ahead bias is touched upon further in subsection 5.2.4. This results in a training set of 3801 observations, a validation set of 507 observations, and a test set of 501 observations.

5.2.2 Time Steps

By defining the time steps variable, we define how many past observations are used to make the current forecast. We chose 260 observations, which is a little more than the number of working days per year. This makes sure the all information of the past year, including the same date in the previous year, is used for the forecast. This results in the finished input tensor for the training-, validation- and

test-set of respectively (3801, 260, 33), (507, 260, 33), and (501, 260, 33). Where the first dimension is the number of observations in each set, the second dimension is the number of time steps, and the third dimension is the number of input features.

The 260 time steps together is called a window. This window is used to make a forecast of the next day, after which this window slides forward one observation. Meaning that the 260th observation is dropped, and the current WMR Fix is added as the first time step. This is called a rolling window, which is shown in Figure 5.3.

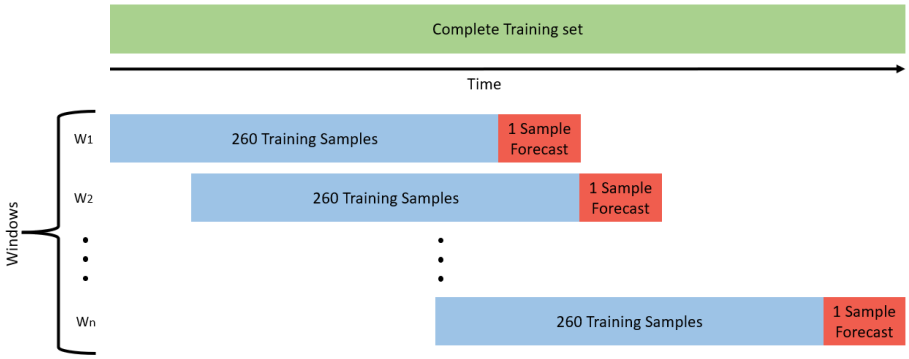

Figure 5.3: Rolling window dataset

5.2.3 Data scaling

When using machine learning for prediction, it is important to scale the data to avoid one feature dominating over another, due to disparate scales. For example, when hypothetical variables $X_1 >> X_2$, X_1 gets assigned a larger weight, with as a result unequal importance for X_1 and X_2. Additionally, unequal weights of variables result in slower model convergence during training, as is displayed in figure 5.4. The convergence is slower as the optimiser has more trouble when optimising the gradient descent. This hypothetical gradient decent is shown in red in the figure, where in blue a function $f(w)$ is visualised.

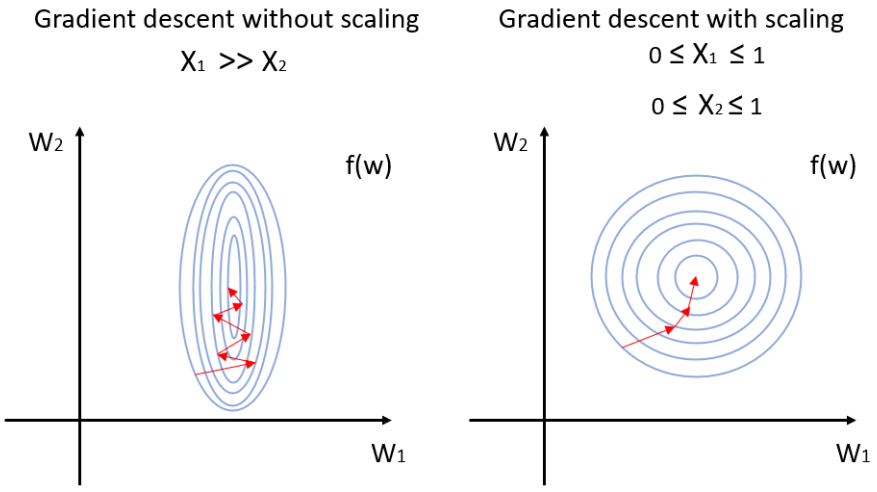

Figure 5.4: Example of gradient descent, with and without scaling

To scale the data a quantile transformer is used. This method transforms the features into a uniform or normal distribution. This results in the most frequent values being spread out and reduces the impact of outliers. This scaler distorts the linear correlations between features measured at the same scale, however, it makes variables measured at different scales more directly comparable. data leakge can occur when the scalar it not fitted correctly, this is discussed in subsection 5.2.4.

Below, in table 5.2, the comparison between different scalers is shown. This comparison is based on a CNN-LSTM model, which architecture is shown in table 5.3. The model performance is measured on validation-loss, while training is done for 20 epochs. 20 epochs is chosen based on trial and error, more epochs were not necessary, as can be seen in table 5.2. Model training is stopped early when the model didn't improve for 5 consecutive epochs, to save time and computing power. The number of epochs is the number of passes that the fine-tuning algorithm takes through the training dataset. Every epoch the weights of the neurons are changed based on the outcome of the previous epoch. While the MAE between the different scalers differs greatly, it is important to realise that also the naive prediction is scaled differently. Thus, the MAE results cannot be directly compared to each other. Another noticeable statistic is that different scalers do not influence the direction that the model predicts the WMR moves towards (up or down), only how the same data and outliers get represented.

Table 5.2: Comparison of Scalers

Scaler	Early Stopping(Y/N)	MAE	MAE Naive Prediction	% Improvement Upon Naive MAE	MSE ($*10^{-5}$)	% Correct: up-down
MinMax Scaler	N(Epoch 20)	0.0330	0.0383	%15.95	1.96 211440	%53.04
Robust Scaler	Y(Epoch 7)	0.4300	0.5513	%28.19	1.96 210603	%53.04
Standard Scaler	Y(Epoch 10)	0.4645	0.5899	%27.00	1.96 210585	%53.04
Quantile Transformer	Y(Epoch 11)	0.1852	0.2342	%26.44	1.96 211022	%53.04

%Improvement Upon Naive MAE is calculated according to the following:
$\%\text{Improvement} = \frac{NaiveMAE - MAE}{MAE} * 100$.

The quantile transformer does not realise the largest improvement in MAE upon the naive forecast. However, we still use this transformer for the data, as it makes variables measured at different scales more directly comparable.

Table 5.3: Scalar comparison CNN-LSTM

Layer (type)	Neurons	Param #
Conv1D	256	42496
MaxPooling1D	None (Pool size 2)	0
Dropout	None (10% dropout)	0
LSTM	32	36992
Dropout	None (10% dropout)	0
Flatten	None	0
Dense	1	3809
Total params:	83297	
Trainable params	83297	
Non-trainable params:	0	

Batch Size = 64, Optimiser = Adam(Learning rate = 0.0001),
Loss Function = Huber Loss, Validation Split = 0.1

5.2.4 Look-Ahead Bias

In time series analysis, data 'leakage' happens easily. Data leakage is when data is used by the model, that should not have been available to it at that point in time. For example, when a model needs to predict $\hat{y}$ at time t, using data from time $t-1$, but also has access to some information from t or $t+1$. Less abstractly this means that a model that needs to predict the WMR fix today, using information from the past, accidentally can access information from today or tomorrow on which it can base its prediction.

Another form of data leakage can occur during the data split. For example, in the subsection 5.2.1 on the data split, we did not mention a k-fold split. Although this can be a useful tool to make the most of small datasets, it also is the cause of data leakage during time series analysis. During a k-fold split, the data is sliced into smaller pieces, and each of these pieces is used $k-1$ times in the training set, and one time as test set. This results in data of $t+1, t+2...t+n$, being used to predict t for example, as shown in figure 5.5. Using a k-fold split would eliminate the auto-correlation structure of the data.

The look-ahead bias is also a topic during data scaling. To avoid this bias, we must scale the training data without knowledge of the validation and the test sets. Thus, we simply fit the data scaler on the training set, and to avoid systematic bias into the other sets we use the identical scaler on them as well.

5.3 Feature Engineering

Feature Engineering (FE) is well defined by Khurana, Samulowitz, and Turaga (2018): "The task or process of altering the feature representation of a predictive modeling problem to better fit a training algorithm is called feature engineering". FE generally involves using mathematical functions to transform a certain feature

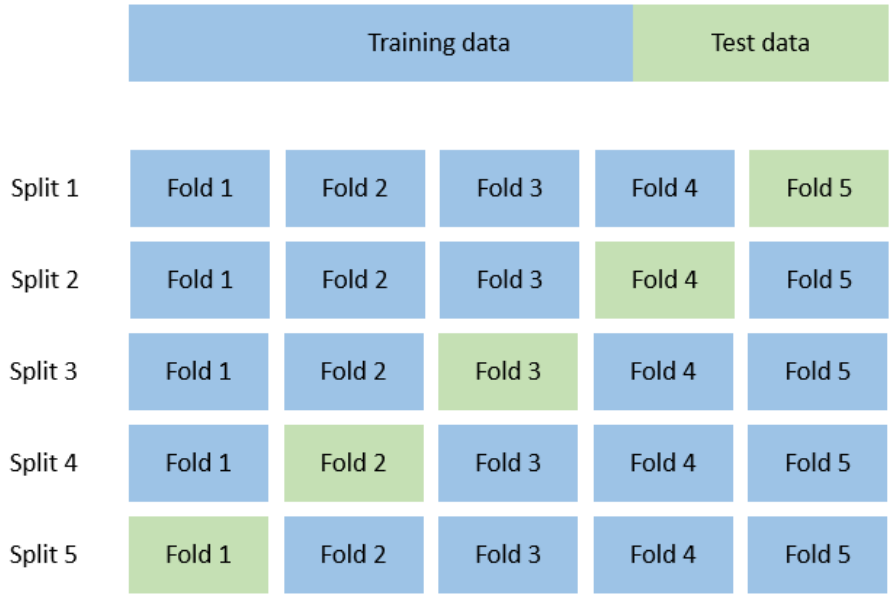

Figure 5.5: K-fold split example, with five folds

space, with the goal to reduce the prediction error. Which features need to be changed or created is often the result of domain knowledge or intuition.

5.3.1 End-of-period Features

In subsection 3.1.2 we found that the hedge re-balancing flows at the end of certain periods can provide significant value for the NN in terms of prediction. As such, it is decisive that we provide a feature to the NN that indicates the end of these periods. This feature is constructed so the last five data points before the next month get a sequentially bigger value. Resulting in month-end getting assigned a value of one, the day before 0.8, two days before 0.6, etc. This example is displayed in table 5.7.

Table 5.4: Example: End-of-Period variables where T = end-of-quarter

Time	End-of-Month	End-of-Quarter	End-of-Year
T-5	0.0	0.0	0.0
T-4	0.2	0.2	0.0
T-3	0.4	0.4	0.0
T-2	0.6	0.6	0.0
T-1	0.8	0.8	0.0
T	1.0	1.0	0.0
T+1	0.0	0.0	0.0

For each end-of-month, quarter and year variable a separate feature is constructed that will keep track of its respective cycle. It is chosen to start each month with zero. Another option would be to decrease the value from one to zero in a similar way after month-end. So that the first day of the month gets 0.8, the second day

0.6 etc. However, we believe that hedge re-balancing flows are pulled earlier in the month, and not pushed after the month-end. This is supported by the decisions the banks made, mentioned in the Bloomberg article in subsection 3.1.2. Where they extended the end-of-month hedge rebalancing model to include the few days before the end-of-month, but not the few days after month-end

5.3.2 Hedge Re-balancing Flows Feature

Estimating the hedge re-balancing flows that will have to occur each month could be a thesis by itself. In this research, however, we formulated a relatively simple model, using some assumptions. The goal of this feature is to make an approximation of the actual flows.

The Model

Previously, in subsection 3.1.2, we mentioned that an increase in the home equity markets compared to foreign markets leads to a depreciation of the home currency. The logic behind the depreciation is that foreign asset managers need to re-balance their hedges when the home markets have increased more in value than the foreign markets. Thus, to estimate the net hedge re-balancing flows at the end of a month, it is required to estimate how much re-balancing the home asset managers and foreign asset managers have to do, due to the performance of their respective asset markets. In this case, we have decided to split the returns, in returns of risky assets, like equity, and non-risky assets, like government bonds.
This resulted in the following model:

$$\Delta Hedge_{B,t}^{A,j} = AUM_B * W_B^{A,j} * H_B^{A,j} * r_t^{A,j} + \varepsilon \tag{5.1}$$

The delta hedge represents the hedge that needs to be re-balanced of country B (B) that invests in the stock market of country A during time t. Where A is country A, with j as the asset market denominated in its currency. This is calculated by multiplying the Assets Under Management (AUM) of country B with the Weight (W) of the portfolio that is invested in the asset markets of A, with the hedge ratio (H) that investors from B use for assets from A and eventually it is multiplied with the return (r) of the investment in that particular period. At the end, we add an error term, denoted by ε, as the equation does not result in the exact $\Delta Hedge$ flows.

For example:

$$\Delta Hedge_{EUfunds,t}^{USEquities} = AUM_{EUfunds} * W_{EUfunds}^{USEquities} * H_{EUfunds}^{USEquities} * r_t^{USEquities} \tag{5.2}$$

The result is the amount of money that asset managers from B have to re-balance as a consequence of their hedges and the investment performance of the assets from country A. The data required to make this basic estimation of the hedge re-balancing flows consist of, at least, yearly AUM data of EU and US funds, average portfolio weights of equity and fixed income assets, and the hedge ratio on these respective asset categories. Unfortunately, not much academic research is published on this data.

Data: Assets Under Management

Boston Consultancy Group (BCG) publishes every year a Global Asset Management Report, which includes specifically the AUM of European and North-American asset managers. We use these reports as estimation for the AUM for each year from 2007 till 2020 and 2002[1]. Although this data might not be exactly correct, due to the nature that the exact numbers of AUM are practically impossible to measure, the methodology stays consistent over the years, which is crucial. For the years where no AUM amounts can be found, using the same methodology, the AUM is linearly interpolated between known years. For 2000, 2001, and 2021, the same growth rate as reported in 2002 and 2020 are used respectively for extrapolation. This data can be found in table 5.5, where the years that are interpolated or extrapolated are in bold.

Table 5.5: Assumed AUM of EU and US asset managers from 2000 till 2021 in Trillion Dollars

Year	EU AUM	US AUM	EU growth rate	US growth rate
2000	9.3	13.5	1.1	1.1
2001	10.2	15.1	1.1	1.1
2002 (Kramer et al., 2011)	11.1	16.8	1.1	1.1
2003	12.1	18.7	1.1	1.1
2004	13.3	20.8	1.1	1.1
2005	14.5	23.2	1.1	1.1
2006	15.8	25.9	1.1	1.1
2007 (Shub et al., 2012)	17.3	28.8	1.1	1.1
2008 (Heredia et al., 2020)	11.6	18.7	0.7	0.6
2009 (Heredia et al., 2021)	13.5	22.1	1.2	1.2
2010 (Shub et al., 2012)	17.5	27.6	1.3	1.2
2011 (Shub et al., 2012)	17.4	27.7	1.0	1.0
2012 (Shub et al., 2014)	18.0	29.4	1.0	1.1
2013 (Shub et al., 2014)	19.3	34.0	1.1	1.2
2014 (Shub et al., 2016)	18.9	36.4	1.0	1.1
2015 (Beardsley et al., 2017)	17.2	31.2	0.9	0.9
2016 (Fages et al., 2018)	20.8	33.0	1.2	1.1
2017 (Fages et al., 2018)	21.2	37.2	1.0	1.1
2018 (Heredia et al., 2020)	20.2	35.4	1.0	1.0
2019 (Heredia et al., 2021)	23.5	42.2	1.2	1.2
2020 (Heredia et al., 2021)	25.7	48.6	1.1	1.2
2021	28.1	56.0	1.1	1.2

[1]The reason that some data is missing is because these reports are not stored centrally, but are distributed on different web pages on the internet. Therefore, not all reports since 2000 were found.

Data: Assets Under Management Invested in Foreign Assets

With the data on AUM of the US and EU obtained, we need to know what percentage of those AUM of European investors are invested in the US and vice versa. For the European asset managers, we assume that 19.86% of the AUM are invested in dollar-denominated assets (Pojarliev, 2018). For asset managers from the US, we assume that 7.92% of the AUM have Euro exposure (Pojarliev, 2018).

Data: Risky/Non-risky Asset Allocation

The next piece of the required information is the ratio of which EU and US investors invest in risky and non-risky assets. This information is not widespread either, however, Andonov et al. (2012) published a paper in 2012 from which the risky/non-risky asset ratios of EU and US investors can be calculated. The numbers in table 5.6 are extracted from that paper, with as result, a 0.635 and 0.702 ratio of risky/non-risky assets for the EU and US investors respectively.

Table 5.6: Statistics on pensionfunds extacted from Andonov et al. (2012)

Summary statistics in 2012	U.S.		Europe	
	Public	Private	Public	Private
Funds	63	127	4	34
%Risky	0.745	0.617	0.749	0.560
Averege Fund Size in $Billion	32.543	8.085	100.131	17.896
Funds * Average Size	2050.209	1026.795	400.524	608.464
%Risky Total	0.702		0.635	

Data: Asset Hedge Ratio's

Now we know the AUM that are invested in the EU and the US by, respectively, the US and the EU, and how those funds are divided over risky and non-risky assets, the next data that has to be acquired is which share of the risky and non-risky asset exposures are hedged. We could not find any published data on the specific hedge ratios of the risky and non-risky assets by European and American funds. Hence, we assume that the hedge ratio of the risky and non-risky assets is the same. This results in the assumption that European investors hedge 80% of the Dollar exposure, while investors from the US hedge 50% of the Euro exposure (Melvin & Prins, 2015).

Assumptions and Limitations

To start with the assumptions on AUM data, we only have data from the end of each year. It is assumed that the AUM grows in the growth rate mentioned in table 5.5. To estimate the AUM at each day between the known periods, which are each end of the year, the AUM is calculated according to formula 5.3.

$$AUM^{A}_{rf,r}(t) = AUM^{A}(t-1)(W^{A_a}_{rf} * rf^{B}(t) + W^{A_a}_{r} * r^{A}(t) + W^{A_b}_{rf} * rf^{B}(t) + W^{A_b}_{r} * r^{B}(t)) + F_t \tag{5.3}$$

Where $AUM^{A}_{rf,r}(t)$ is the AUM of the asset managers of country A. The AUM on t depens on the AUM of the managers on $t-1$, the weight of the portfolio that is invested respectively in risk free and risky assets assets from country A ($W^{A_a}_{rf}$, $W^{A_a}_{r}$) multiplied with their respective returns on time t ($rf^{A}(t), r^{A}(t)$). And it depends on the weight the managers invested in risk free and risky assets from country B ($W^{A_b}_{rf}$, $W^{A_b}_{r}$) multiplied with their respective returns on time t ($rf^{B}(t), r^{B}(t)$). Finally, F_t denotes the inflow or outflow of AUM.

This daily change is calculated with the risky/non-risky asset allocation together with the amount that is invested in the EU for US investors, and vice versa. The discrepancy between the AUM that is measured by BCG and the AUM that is calculated, is assumed as inflow/outflow of money in/out of the market. This flow is uniformly distributed over the whole year and represented as F_t in formula 5.3. An example of this flow from the period 2001 to 2002 for European AUM is depicted below in figure 5.6. $EU\_AUM\_w\_R$ is the AUM that we expect based on returns. To make this fit with the AUM measurement of 2002, from BCG, we calculated the cumulative (in)flow over the year, which is denoted by $CUMU\_EU\_inflow$ in the previously mentioned figure 5.6 and F_t in the previously mentioned formula 5.3

Figure 5.6: European AUM 2001-2002 with flow

Other assumptions we have to make as a consequence of the lack of data are that the hedge ratios, the risky/non-risky asset ratios, and the portfolio weights of assets invested in the EU and US stay constant over the whole period from 2001 till 2021.

Finally, the current model is restricted to the re-balancing flows as a result of EU and US investors. However, re-balancing flows also occur because of other countries investing in the EU and US, these flows are not taken into account. The flows that occur because of corporate actions and hedged corporate revenue are also not taken into account. The consequence being that there might be a larger discrepancy between the estimated hedge flows and the actual hedge flows that occur, then what would be the case if we would take these other flows into account as well.

Implementation

When fully implemented, the estimation of formula 5.3 results in the 'weight' of which part of the returns needs to be re-balanced as a consequence of the hedges. For example, these 'weights' mean that all things being equal, and US equities increase by 1%, the expected hedge re-balancing flow x is estimated by multiplying the weight with the returns, so $x = W * r$. These weights over time are depicted in figure 5.7, where the y-axis is in Billion US Dollars.

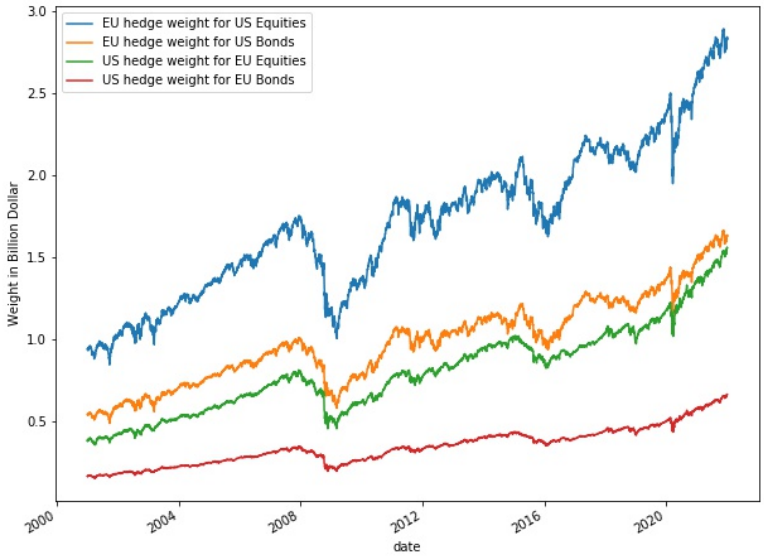

Figure 5.7: Evolution of hedge rebalancing weights over time (2001-2021)

With these weights and the returns of each respective category, the total hedge rebalancing flows are calculated. The results are shown in figure 5.8, where $EU\_total\_hedge\_flow$ is the hedge flow expected from EU investors in the US, and $US\_total\_hedge\_flow$ vice versa. Net_hedge_flow is the sum of the EU and US expected hedge flows. These net hedge flows are isolated in figure 5.9 for better visualisation.

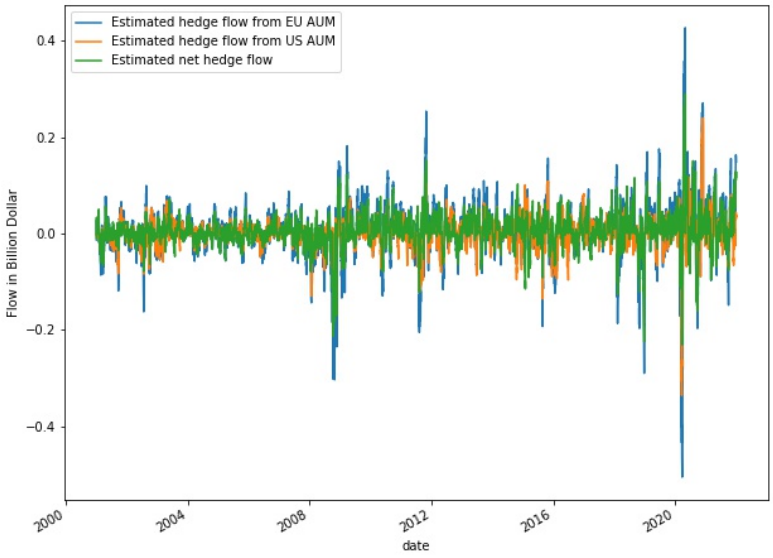

Figure 5.8: EU,US and net hedge flows (2001-2021)

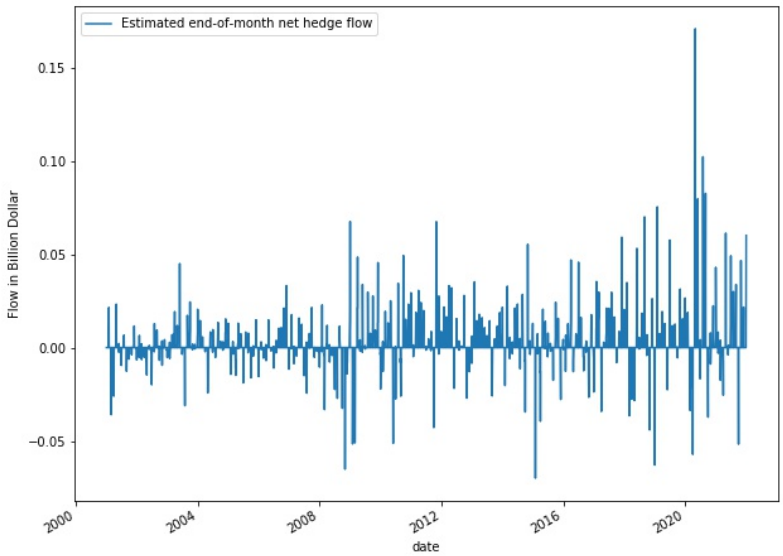

Figure 5.9: Net end-of-month hedge rebalancing flows (2001-2021)

5.4 Complete Data Set

The complete data set has 5069 samples, starting on 02-01-2001 and ending on 23-12-2021. The number of samples is the first dimension in the tensor. The second dimension is the window size, which is the 260 time steps. The third, and final,

dimension are the 33 features, which can be found in Appendix A.2.

5.5 Algorithm Design

Empirical studies have shown that, in most circumstances, a more complex ML model, with deeper (i.e., more layers) and more width (i.e., more neurons per layer) work better than the models with a simple structure (He, Zhang, Ren, & Sun, 2016; Zagoruyko & Komodakis, 2016). The downside of the more complex models is that they are more difficult to train, and require more computational resources. Zagoruyko and Komodakis (2016) show that making a network deeper quickly results in diminishing returns, where each fraction of a percentage of improved accuracy costs nearly a doubling of the number of layers. The same paper suggests a network with a much wider architecture, that achieves a far superior result over the deep counterparts.

5.5.1 Optimiser

The cost function is optimised by the optimiser. A standard optimiser is a gradient descent, which is described in Subsection 4.2.3. The optimiser's task is essentially to optimise the loss function, it depends on the function if it has to be minimised or maximised. Essentially, the loss function measures how good the predictions of the model are compared to the actual values. The optimiser determines in which direction and with what magnitude changes should be made to the parameters in order to improve the cost function.

Learning Rate

The learning rate influences the size of the steps that the optimiser takes when optimising the parameters of the model. When the learning rate is large, the optimiser makes large steps towards the seemingly right direction, sometimes overshooting the actual optimum value. When the learning rate is small, the optimiser takes small steps towards the optimum, however, as these steps are small, the optimisation process is slow and overfitting can occur (Smith, 2018).

Adam Optimser

The optimiser we use is the Adam optimiser. This is a method for efficient stochastic optimisation (Kingma & Ba, 2014). The advantage of this method is a.o. that it does not require a stationary objective function and is appropriate for very noisy gradients (Kingma & Ba, 2014). Reddi, Kale, and Kumar (2019) proposed an updated version of Adam, as situations do occur when Adam does not converge to the optimal solution. This updated version is called AMSGrad, which we implement in our algorithm.

5.5.2 Activation Functions

The most commonly used activation functions are non-linear activation functions. Due to non-linear characteristics of real world errors, non-linear activation functions

are preferred over their linear counterparts in a Neural Network (Sharma, Sharma, & Athaiya, 2017). In our case, the tanh function seems to be the most promising. The function can be defined as:

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{5.4}$$

The function is continuous and differentiable, which is important for the optimisation phase. The values of the function lie in the range -1 to 1, where large input values get mapped closer to 1, and large negative values closer to -1.

Using the Rectified Linear Units (ReLU) as activation function is generally conventional for deep NNs (Agarap, 2018). The ReLU function is defined as:

$$f(x) = ReLU(x) = max(0, x) \tag{5.5}$$

This function works by thresholding values at 0, so it outputs 0 when x < 0, and otherwise it outputs a linear function when x ≥ 0. The tanh and ReLU functions are shown in appendix A.1. Both functions are used to find the best-performing model.

5.6 Hyper-parameters and Optimisation

Hyper-parameters optimisation is crucial, as sub-optimal parameters result in unnecessarily long training times and an underperforming model (Smith, 2018). However, optimisation of the parameters is not easy, Smith (2018), and Snoek, Larochelle, and Adams (2012) even call it a 'black art' that needs years of experience to master.

5.6.1 Epochs

The epochs are the number of passes that the fine-tuning algorithm takes through the training dataset. After each epoch, the weights of the neurons are changed based on the outcome of the previous epoch. After a certain number of epochs the model improvements diminish or vanish completely. That is why generally, the model training is stopped early if the model does not improve anymore after a certain number of epochs.

5.6.2 Batch Size

The batch size is the number of training examples that are utilised in one iteration. So when batch size = 10, ten samples are passed to the network in a 'batch', after which the weights are updated. Small batch sizes have been recommended for regularisation effects, but other research has shown that larger batch sizes could achieve a higher forecasting accuracy (Smith, 2018).

5.6.3 Dropout

Dropout is a way to perform regularisation. This is crucial, as a relatively small dataset can easily overfit, and perform poorly on held-out test data. Dropout removes randomly a certain percentage of neurons from a layer during training. This

makes the network more robust as the network cannot rely on just a few specific neurons (Smith, 2018).

5.7 Hyper-parameter Optimisation

To assist with the hyper-parameter optimisation, AWS cloud is used. All three models are given a hyperparameter-range from which 50 different models are created and trained per architecture. The 50 architectures per model are not chosen randomly but optimised by a Bayesial search algorithm that estimates the possibly best model on the previously trained models. This algorithm is discussed in subsection 5.7.1 below. The hyperparameter ranges are discussed later in subsection 5.7.2.

5.7.1 Bayesian Optimisation

To optimise the hyperparameter tuning, a Bayesian search algorithm is deployed. Snoek et al. (2012) have shown that using this automatic approach to optimise the hyperparameters can reach or surpass human expert-level- optimisation.

Bayesian optimisation is used to find the minimum of a function $f(x)$, in our case the objective function, on some bounded set χ. The algorithm constructs a probabilistic model for $f(x)$ and then makes a decision about where in χ to next evaluate the function, using all information available from previous evaluations of $f(x)$.

During the model tuning, it is chosen to train five models simultaneously. The Bayesian optimiser can benefit from previously trained models, but training all models sequentially takes too much time, while the probability of better model performance is small.

5.7.2 Hyperparameter Ranges

For the Bayesian optimisation to work we need to define the bounded set χ. In the next subsubsections, this bounded set is discussed per hyperparameter.

Epochs

The maximum epochs that each model will train is set at 100. However, when the model does not improve for 15 consecutive epochs then the training is halted early, to save computing time.

Hyperparameter Range: Batch Size

For the batch size hyperparameter values of 2^4, 2^5,...,2^9 are chosen. This results in a minimum batch size of 16 and a maximum of 512 which is a relatively standard range for our small dataset. It is also standard to use exponents of two (Smith, 2018).

Hyperparameter-Range: Learning Rate

The learning rate is set as a continuous parameter between $1 * 10^{-4}$ and $1 * 10^{-1}$, meaning that any value between those boundaries can be set as learning rate. The scale on which the learning rate is chosen is logarithmic, so the smaller values are chosen more often. This results in a minimum learning rate of 0.0001 and a maximum learning rate of 0.1. This is smaller than standard (Smith, 2018), however, a lower learning rate is appropriate due to the extreme noise in our data.

Hyperparameter Range: Neurons

The three models all have LSTM layers, the neurons for this layer are set as any integer between 8 and 256. For the models that have CNN layers, the neurons are separately set, as any integer between 8 and 256 as well. A useful reference point is that the number of hidden units should be, at least, equal to the number of input neurons for enough expressibility (Shivarova, 2021). We use 33 input neurons, so we expect that the optimised architectures have at least 33 hidden neurons.

Hyperparameter Range: Dropout

The allowed dropout values for the bounded set are set at 0.2, 0.3, and 0.4, meaning that at each layer a random selection of neurons between 20% and 40% are dropped.

Hyperparameter Range: Activation Functions

As mentioned earlier in subsection 5.5.2, the tanh activation function seems promising in our particular case, and the ReLU activation function is conventional, therefor both functions are tested during the tuning.

Table 5.7: Hyperparameter Ranges for Tuning

Hyperparameter	Range	Type
Batch Size	2^4, 2^5, ..., 2^9	Categorical
Learning Rate	$1 * 10^{-4} - > 1 * 10^{-1}$	Continuous
CNN Neurons	$8 - > 256$	Integer
LSTM Neurons	$8 - > 256$	Integer
Dropout	0.2, 0.3, 0.4	Categorical
Activation Function	Tanh, ReLU	Categorical

6. Results

This chapter describes the architecture and performance metrics of the best models of each type. As well as how different parameters affect the objective function during tuning. Eventually, the best-performing model is tested on robustness. At the end of this chapter sub-questions two and three are answered:

Sub-question 2: "What is the performance and robustness of the models on predicting the FX spot market movements?"

Sub-question 3: "How do the different hyperparameters affect the model performance?"

6.1 Model Architectures

In this section, all model architectures are quickly discussed. The architecture of the Attention CNN-LSTSM has a lot more parameters than the other models. Initially, the number of parameters was more close, however, the architecture of the LSTM and CNN-LSTM seem to cause saturation, which resulted in the choice for fewer layers.

6.1.1 LSTM

The architecture of the LSTM NN consists of one LSTM layer and one dropout layer, in total with 27,813 trainable parameters, as shown in table 6.1. The initial architecture consisted of three of both layers, as shown in appendix B.1, however, this possibly resulted in neuron saturation, as the prediction of this model is constant as well.

6.1.2 CNN-LSTM

For this model, the initial architecture was more sophisticated as well. It consisted of two convolution layers, followed by the combination of three LSTM and three dropout layers, as shown in appendix B.2, however, this possibly resulted in neuron saturation, as the prediction of this model is almost constant.

Table 6.1: LSTM Architecture Results

Layer (type)	Neurons	Param #
LSTM	64	27744
Dropout	None (40% dropout)	0
Dense	1	69
Total params:	27.813	
Trainable params	27.813	
Non-trainable params:	0	

Batch Size = 260, Optimiser = Adam(Learning rate = 0.04137), Loss Function = Huber Loss, Validation Split = 0.1, Activation = Tanh

The current model consists of a one-dimensional convolutional layer, followed by a pooling layer and a dropout layer. The final layers of the model are an LSTM and a dropout layer, followed by the output layer, as shown in table B.2. This results in a model with 68,133 parameters.

Table 6.2: CNNLSTM Architecture Results

Layer (type)	Neurons	Param #
Conv1D	218	21800
MaxPooling1D	0 (pooling size = 2)	0
Dropout	None (20% dropout)	0
LSTM	44	46288
Dropout	None (20% dropout)	0
Dense	1	45
Total params:	68.133	
Trainable params	68.133	
Non-trainable params:	0	

Batch Size = 512, Optimiser = Adam(Learning rate = 0.00012), Loss Function = Huber Loss, Validation Split = 0.1, Activation = Tanh

6.1.3 Attention CNN-LSTM

The Attention CNN-LSTM has significantly more parameters than the other two models, partly because it has more layers, and also because the tuning resulted in more neurons for the CNN and LSTM layers. It starts with an LSTM layer, followed by an Attention layer, one-dimensional convolutional layer, and dropout layer. The final layers consist of an LSTM layer, dropout layer, and output layer, as shown in table 6.3. This results in a model that has 682,332 parameters.

Table 6.3: Attention-CNNLSTM Architecture Results

Layer (type)	Neurons	Param #
LSTM	209	178620
Attention	0	0
Conv1D	210	25376
LSTM	209	173160
Dropout	None (40% dropout)	0
LSTM	209	304980
Dropout	None (40% dropout)	0
Dense	1	196
Total params:	682.332	
Trainable params	682.332	
Non-trainable params:	0	

Batch Size = 32, Optimiser = Adam(Learning rate = 0.00018), Loss Function = Huber Loss, Validation Split = 0.1, Activation = Tanh

6.2 Model Performance

The performance of the NNs on the prediction of the WMR is displayed in table 6.4. The performance is fairly similar. Yet, the CNN-LSTM exceeds the LSTM and Attention CNN-LSTM in terms of MSE and MAE performance, while all NNs perform equally in terms of Theil's U. The Theil's U < 1, meaning that all NNs outperform guessing when predicting the WMR, albeit marginally. All NNs outperform the benchmark, which is the naive prediction.

Although the WMR is the target variable, we established earlier that we want to predict the difference between the WMR at t and $t+1$, the WMR delta. Measuring performance on the WMR delta also makes more sense, as the predicted WMR at $t+1$ consists in a large part of a known variable, the WMR at t, resulting in only small differences between the predictions of the WMR at $t+1$. The performance of the NNs on the prediction of the WMR delta is presented in table 6.5.

Table 6.4: Model Performance on WMR

Model	MSE (WMR)($*10^{-5}$)	MAE (WMR)	Theil's U WMR
Naive	2.326 213 144	1.812 150 000	1.00011
LSTM	2.240 954 746	1.777 602 249	0.98095
CNN-LSTM	2.240 948 561	1.777 600 047	0.98095
Attention CNN-LSTM	2.240 951 835	1.777 600 283	0.98095

When we analyse the results of the performance metrics on the actual prediction, which is the WMR delta, then the divergences are more pronounced. The LSTM outperforms both the Attention CNN-LSTM and CNN-LSTM on the MSE and

MAE, contrary to the previous results. Still, the CNN-LSTM outshines the Attention CNN-LSTM and LSTM in terms of Theil's U of the prediction and actual values. In terms of Theil's U (delta: up-down), the CNN-LSTM and Attention CNN-LSTM perform equally. This metric is based on if the NN predicts correctly whether the delta is positive or negative and thus if the WMR will go up or down. Both the Theil's U metrics of the hybrid NNs are smaller than 1, indicating that they outperform guessing, while the Theil's U metrics of the LSTM are larger than 1, indicating that it performs worse than guessing. All NNs outperform the naive prediction in terms of MSE, MAE, and Theils's U.

Table 6.5: Model Performance on WMR delta

Model	MSE (delta)	MAE (delta)	Theil's U (delta: up, down)	Theil's U (delta: x,y)	% Correct: up-down
Naive	0.09201	0.24299	*	*	** 51.896% 49.984%
Linear Regression	0.00031	0.00434	2.05059	1.41403	48.915%
Polynomial Regression (4 degrees)	0.00031	0.00434	2.14820	5.04561	43.984%
Random Forest (1000 trees)	0.00025	0.00371	1.11747	23.13992	50.690%
LSTM	0.04926	0.18621	1.00053	1.00004	53.094%
CNN-LSTM	0.05281	0.19255	0.99841	0.99978	52.894%
Attention CNN-LSTM	0.05222	0.19054	0.99841	0.99999	52.894%

*No Theil's U metrics can be calculated for the naive prediction, as the delta is always = 0.
** 51.896% is the % that is correct when the prediction is that the delta is always positive, 49.984% is the % that is correct when the prediction is that the delta is always negative.

We value the outcome of the Theil's U higher than the outcome of the MSE and MAE. The reason being that when predictions are not better than guessing, the MSE and MAE do not contain much value.
Thus, following this logic, the LSTM model performs the worst, even though it resulted in the best MSE and MAE. Consequently, making the CNN-LSTM the best performing model in this particular case.

The naive prediction is based on 0 parameters. As another comparison we fitted a linear model, a 4th degree polynomial model and a 1000 trees random forest model on the data. The linear and polynomial models perform extremely well on MSE and MAE, however, the performance on the Theil's U is significantly worse than guessing, indicating that these models might be overfitted. In terms of predicting a positive or negative movement of the WMR at time $t + 1$, it is better to always predict that the WMR will go up, or always down, than to use the linear or polyno-

mial prediction. The random forest even outperforms all models on MSE and MAE, however, it performs poorly on the Theil's U metrics as well.

6.2.1 CNN-LSTM results

In this subsection, we go more in-depth into the results of the CNN-LSTM as it is selected as the best model. This model seems to only sporadically make large enough deviations to really impact the outcome of the predictions. This is clear to see in figure 6.2.

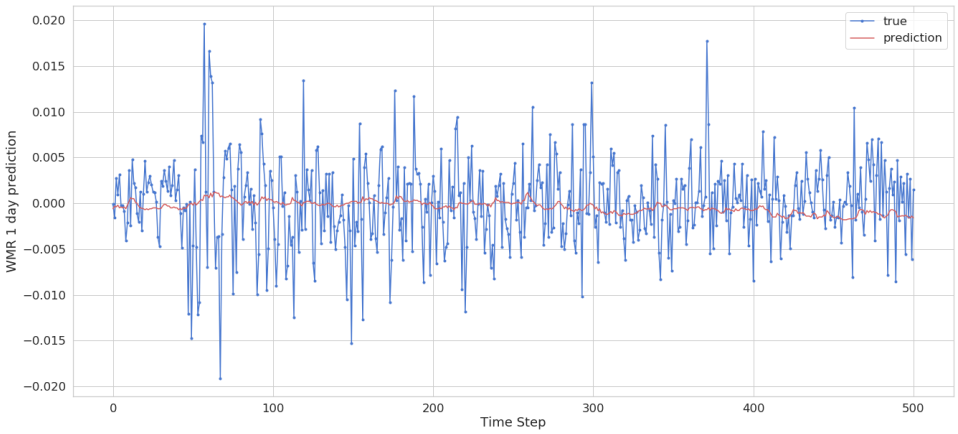

Figure 6.1: Distribution of the predicted and actual values of the CNN-LSTM

It seems that the model is not able to obtain enough meaningful temporal trends to make a significant prediction. The distribution of the predictions and actual values that supports this is shown in figure 6.2. Where the distribution of predictions has a significantly smaller range than the distribution of the actual values. Thus, with everything else equal, a larger standard deviation could indicate a model with more potential, as it better extracts the long-term temporal dependencies.

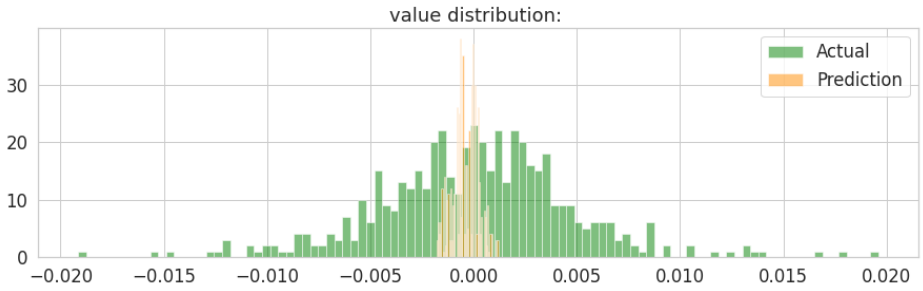

Figure 6.2: Distribution of the predicted and actual values of the CNN-LSTM

It is also interesting to look at the MAE of the predictions and actual values over time. When there is a clear trend in the MAE then the model might require more training. The difference over time is shown in Figure 6.3.

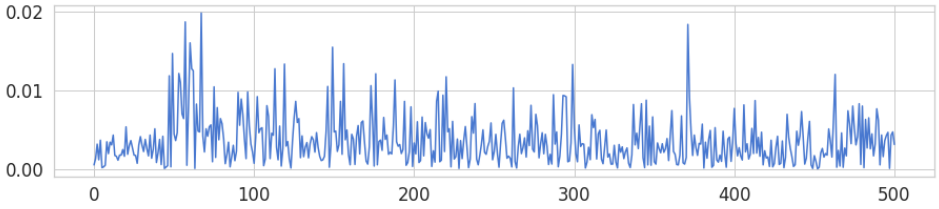

Figure 6.3: MAE over time of the CNN-LSTM

The MAE over time seems to not follow any clear pattern. The ideal situation is when the line is random, otherwise, it may indicate that the model is not sufficiently trained, which seems to not be the case.

6.2.2 Ljung-Box Test

In order to reject if the model is under-fitted we perform a Ljung-Box test on the out-of-sample residuals. This statistic is used to test whether this error is auto-correlated. When the p-values of the Ljung-Box test are < 0.05 we can reject the Null-hypothesis at the 95% confidence level, indicating that the model is under-fitted (Shivarova, 2021). The Ljung-Box test is formulated as:

$$Q(m) = T(T+2)\sum_{l=1}^{m}\frac{\hat{\tau}_l^2}{T-l} \tag{6.1}$$

Where T is the number of observations, $\hat{\tau}_l^2$, are the sample auto-correlations of the residual at lag l, and m is the maximum lag used in the test. This statistic follows an asymptotically chi-squared distribution with m degrees of freedom. The decision rule is to reject the Null-hypothesis if $Q(m) > \chi_\alpha^2$, where χ_α^2 is the $100(1-\alpha)^{th}$ percentile of a chi-squared distribution.
The largest p-value that resulted from the Box-Ljung Test is 0.0405. Hence, we can reject the Null-hypothesis in favour of the alternative hypothesis at a 95% confidence level. Thus, rejecting the hypothesis that the model is under-fitted.

6.3 Black Box

Once a NN is trained, several important issues surface around how to interpret the model parameters. This interpretability is a prominent issue for practitioners in deciding whether to use a NN or to opt for other ML or statistical methods. Even if the latter's predictive accuracy is inferior (Dixon et al., 2020).
In this section we will show the effect that the different parameters have on the result of the NNs, this is more of a 'peek' into the black box, instead of 'opening it up'. Unfortunately, there does not seem to be a good solution to 'opening up' the NNs. In the section on future research 7.4, we will go more in-depth on possible solutions for this problem.
The robustness and sensitivity tests are only performed on the best performing model that was established earlier, the reason being computational and time constraints.

6.3.1 Hyperparamter Range Performance

Below, the outcomes of the hyperparameter tuning jobs are displayed. These are the parameters that the Bayesian optimiser set and the respective validation loss it achieved. It is possible to plot a line in the plots to visualise a possible linear correlation. However, we think that no correlation is prevalent enough to be meaningful, and we do not want to convey the impression of causality. Especially because the results are influenced by confounding parameters, which are not randomly set, but set probabilistically by the Bayesian optimiser.
Keep in mind that a lower *val_loss* on the y-axis, means better performance.

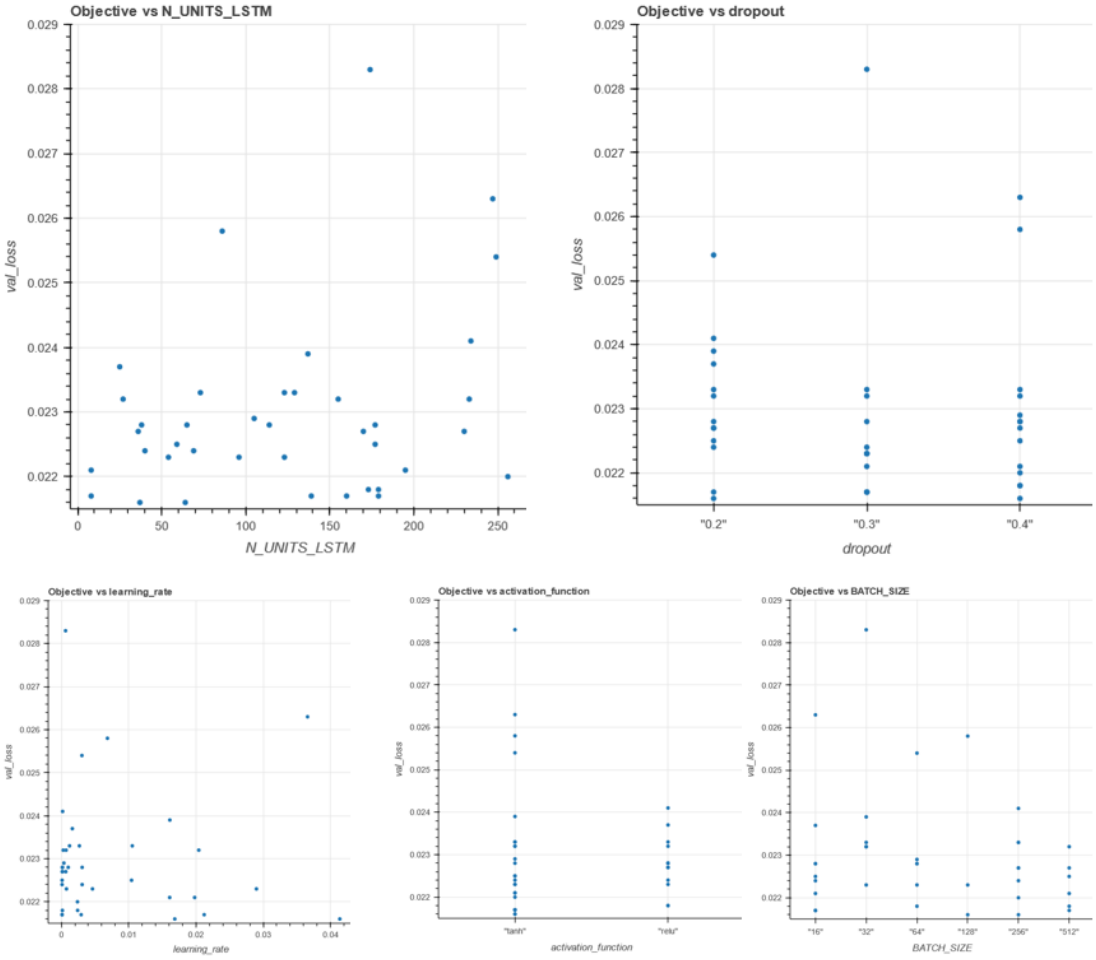

Figure 6.4: Result of the hyperparameter tuning of the LSTM Neural Network

We can only make two conclusions from these results. Mainly being that the tanh activation function produces better results than the ReLU activation function, in our particular case, for every NN. The Bayesian algorithm also uses the tanh function more often, which supports this conclusion.

Secondly, as mentioned before, there does not seem to be a clear correlation and causality between certain hyperparameters and the performance on the Validation Loss of the NN. This implies that we cannot simply improve the neural network by increasing or decreasing a parameter, but the validation loss depends on the

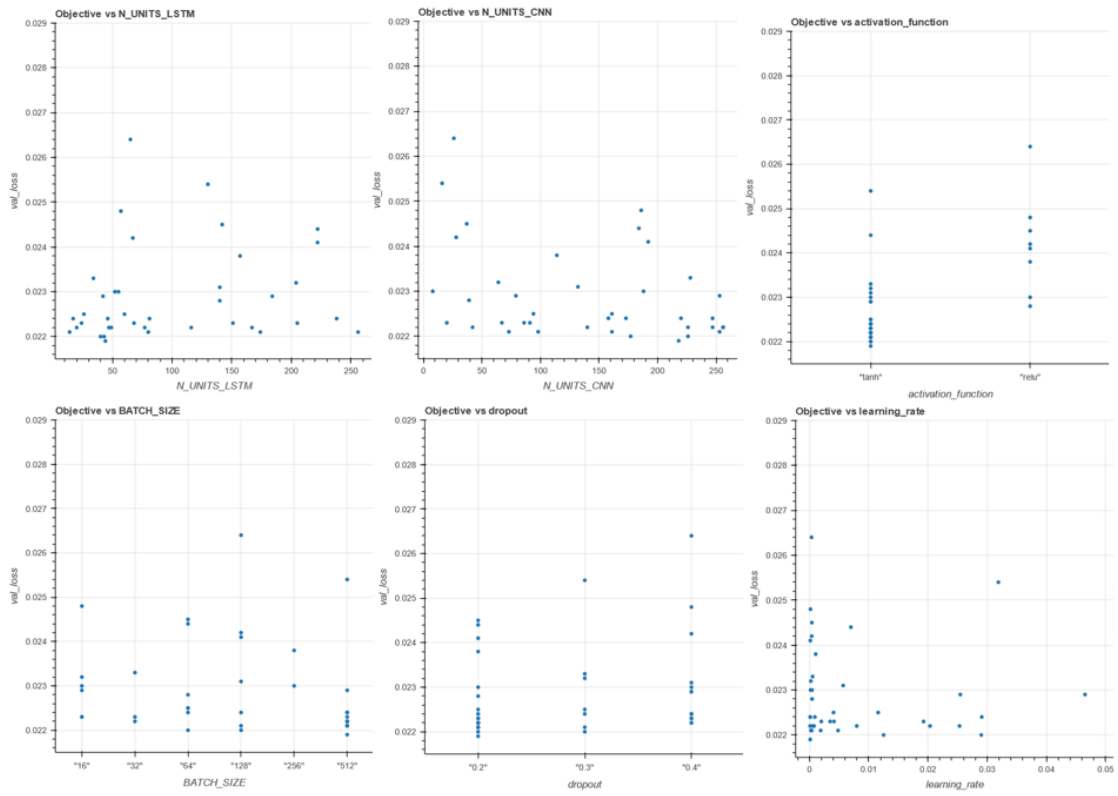

Figure 6.5: Result of the hyperparameter tuning of the CNN-LSTM Neural Network

combination of hyperparameters. The parameters are also much intertwined, Smith (2018) for example, correlates the optimal batch-size to the learning rate, which makes it difficult to isolate causality.

6.3.2 Performance End-of-Period Features

It is difficult to precisely test our hypothesis that the NN performs better towards month-end, the reason being that the model is trained, not just on month-end periods, but on all data. There are also too few end-of-month data points to train an NN on. A possibility to test the hypothesis, is to remove the end of month features, and see how the models compare[1]. The results are shown in table 6.6.

The model performance without the hedge rebalancing features is better for the MSE and MAE metric, however, the model performs worse on the Theil's U of the actual predictions. We, again, argue that the CNN-LSTM with the hedge rebalancing features outperforms the CNN-LSTM without, using the same arguments as before.
The predictions of the model without the features also have a lower standard deviation, meaning that predictions are centered closer to the mean of the WMR delta distribution.

[1]This results in the removal of the following features: 'Net_hedge_flow', 'end_month_estimated_hedge_flow', 'month_end_value', 'quarter_end_value', 'CUMU_US_inflow', 'year_end_value', 'CUMU_EU_inflow', which are described in appendix A.1.

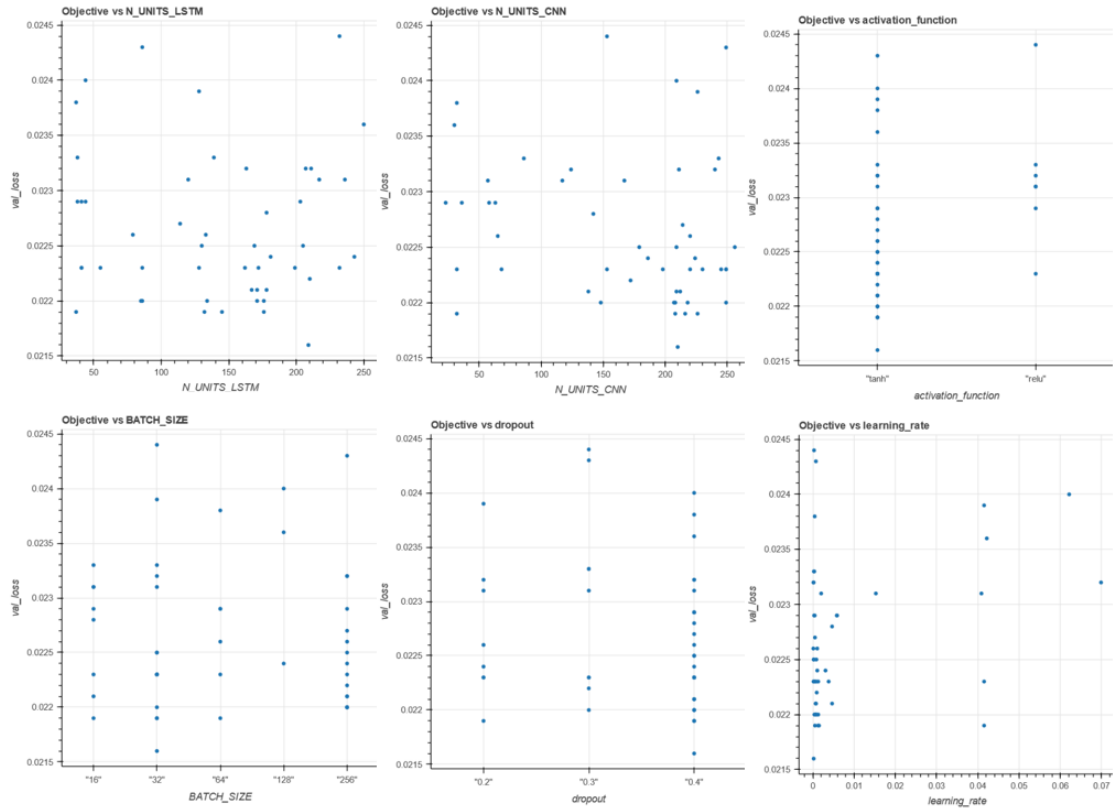

Figure 6.6: Result of the hyperparameter tuning of the Attention CNN-LSTM Neural Network

Table 6.6: Model Performance without End-of-Period Features on WMR delta

Model	MSE (delta)	MAE (delta)	Theil's U (delta: up, down)	Theil's U (delta: x,y)
CNN-LSTM w/ hedge variables	0.05281	0.19255	0.99841	0.99978
CNN-LSTM w/o hedge variables	0.05089	0.18968	0.99841	0.99998

Table 6.7: Model Performance without End-of-Period Features on WMR delta

Model	Prediction Mean	Prediction Standard Deviation
CNN-LSTM w/ hedge variables	0.48288926	0.042286146
CNN-LSTM w/o hedge variables	0.49669522	0.025146397

6.3.3 Robustness

Neural networks weights are initialised randomly. In order to make the training reproduce-able and the outcome (almost) deterministic, a seed, which is normally

random, can be set. These seeds are created in programming languages to avert randomness in a random process, making it reproducible. The weights initialisation of the NN is determined by this 'random' seed. In order to test the robustness of the model, we make use of this randomness, by training the CNN-LSTM 25 times, using different seeds. This results in different weight initialisation, so we can test if the model performance is stable, or if results are widely divergent. The model performance is measured and compared in validation loss. In table 6.8 the results are displayed, while in appendix B.6 these results are plotted.

Table 6.8: Model Robustness Validation Loss

	Seed	Validation Loss
Original CNN-LSTM	-	0.0219
11	317.0	0.0219
24	72.0	0.0220
1	156.0	0.0220
18	389.0	0.0221
4	284.0	0.0221
5	194.0	0.0221
15	334.0	0.0221
13	331.0	0.0221
23	350.0	0.0221
20	333.0	0.0222
12	226.0	0.0222
7	272.0	0.0222
10	488.0	0.0222
8	362.0	0.0223
16	399.0	0.0223
17	381.0	0.0223
2	480.0	0.0223
22	59.0	0.0223
0	178.0	0.0223
9	38.0	0.0224
14	174.0	0.0224
21	185.0	0.0224
6	202.0	0.0225
3	412.0	0.0225
19	292.0	0.0236

Only one of the 25 trained models actually performs just as well as the original model on validation loss. The other 24 models that were trained, with exactly the same hyperparameters, perform worse. From this, we can conclude that the model has trouble with convergence. This conclusion is backed up by figure 6.7. This Figure shows the training and validation loss of the CNN-LSTM per epoch. What stands out is that the validation loss barely goes down after a few epochs, which indicates that the model has trouble finding fitting weights for convergence.

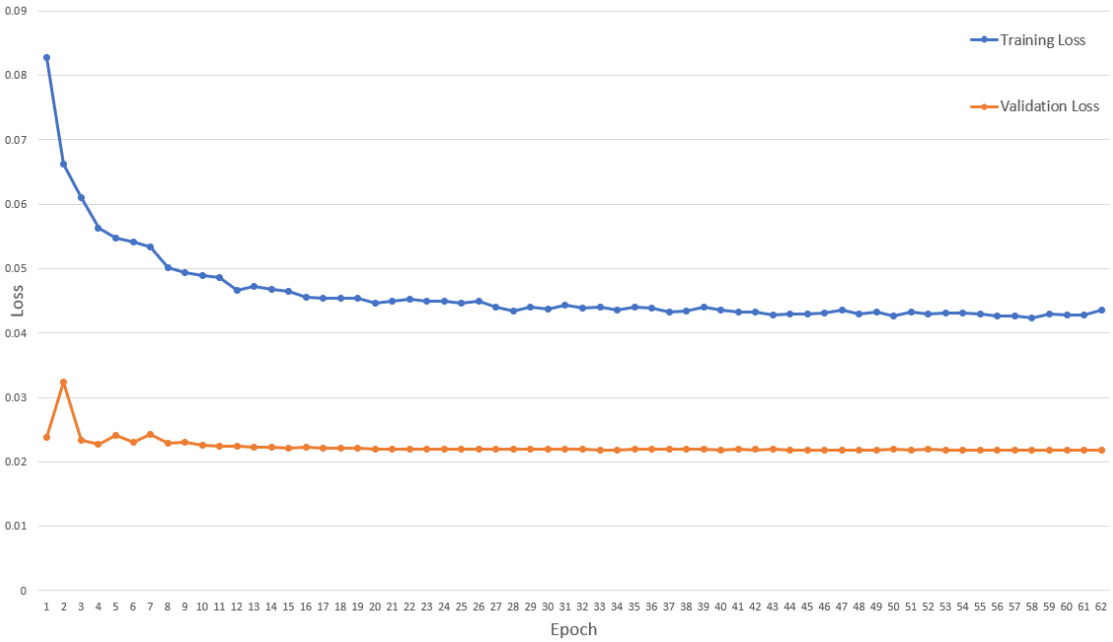

Figure 6.7: Loss per Epoch

6.3.4 Parameter Sensitivities

To investigate how different hyperparameters affect the variance of the validation loss, we train the CNN-LSTM 15 times per hyperparameter. In each of these 15 model training rounds, this hyperparameter value is chosen randomly, while all others remain unchanged. The result is a range of outcomes for every hyperparameter, from which we can decompose the sensitivity towards the validation loss.

To compare and visualise the loss, we use the percentage change of the hyperparameter in comparison to the mean of the hyperparameter and equivalent for the validation loss. Then we use the linear correlation, R^2 and STD to quantify the strength of the relationship. These results can be found in table 6.9. A logarithmic scale is used for the learning rate parameter since the range is significant. Consequently, the negative values are displayed in red.

Table 6.9: Parameter Sensitivity Coefficients

	Batch Size	Neurons LSTM	Neurons CNN	Learning Rate	Dropout	Validation Loss
Mean	181.8418	87.9242	99.7430	0.0039	0.2475	0.02229
Correlation	-0.4274	0.1687	-0.2313	-0.2778	-0.0178	-
R^2	0.1826	0.0285	0.0535	0.0772	0.0003	-
STD	0.8428	0.6285	1.5713	1.4018	1.0554	-

The standard deviation of the realised validation losses, gives an indication of the sensitivity that the validation loss has towards a hyperparameter. Although, this metric does not present information about the direction and explainability of affect of the parameter. The correlation and R^2 give a better view in terms of these

The number of CNN neurons and the learning rate most significant affect the validation loss in this test, based on the standard deviation. The batch size has the largest correlation with the produced validation loss, thus best explaining the linear direction of the change in validation loss. While the correlation between the CNN neurons and dropout might be better explained by a convex function, as can be seen in figure 6.8. Another notable outcome is that the number of neurons of the LSTM is the only hyperparameter that is positively correlated with the validation loss. Meaning that increasing the neurons in this layer, actually results in worse outcomes.

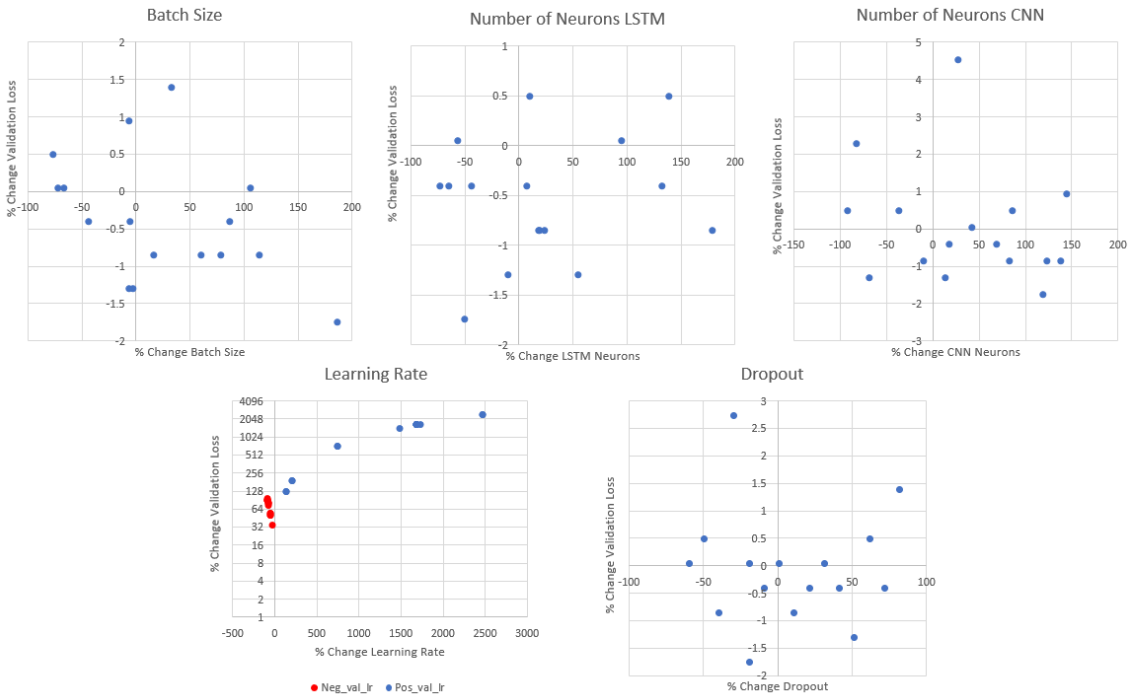

Figure 6.8: Hyperparameter Sensitivity Plots

6.4 Results Conclusion

Returning to sub-question two:

Sub-question 2: "What is the performance and robustness of the models on predicting the FX spot market movements?"

We have prepared data, added features, and defined the performance metrics. Hereafter, we have constructed, trained, validated, and tested the three models we defined earlier. Based on the Theil's U metric we conclude that the CNN-LSTM model performs the best, as it resulted in a Theil's U of 0.99978 on the predictions. This means that this model outperforms guessing, but only marginally. From these results, we can conclude that the hybrid architectures outperform the regular neural network architectures in this study.

We have evaluated the robustness of CNN-LSTM model through training 15 models with different random seeds. We found that the model convergence is inconsistent, resulting in divergent model performance. We think that the reason for this inconsistency is the highly noisy target variable we try to predict. Due to this inconsistency, the model architecture and hyperparameters need substantially more testing and tuning, before we can conclude that this is the absolute best performing model for this situation.

Coming back to our third sub-question:

Sub-question 3: "How do the different hyperparameters affect the model performance?"

We have trained 15 models per hyperparameter, changing it's value randomly, while keeping everything else equal. We then decomposed the variability of the validation loss towards different hyperparameters and ranked them according to their significance.

The number of CNN neurons and the learning rate have the most significant affect on the validation loss, based on the resulting standard deviation of respectively 1.5713 and 1.4018. The batch size has the largest correlation with the produced validation loss, thus best explaining the linear direction of the change in validation loss. All parameters, except for the number of LSTM neurons are negatively correlated with the validation loss. Where the dropout has the least significant linear correlation of only -0.0178.

7. Discussion

This chapter discusses the results of this thesis, answers the main research question and we go over the implications for machine learning on highly noisy financial time series data. The chapter is closed by the recommendations for MN and suggestions for future research.

7.1 Conclusion of the Main Research Question

In previous chapters, we have seen how the relevant data is primed for the models and how some extra features are engineered. We have also looked at the best fitting models according to the theory, and constructed and evaluated them. Moreover, we have answered the sub-questions, so returning to the main question:

Main Question: "What is the forecast performance of the proposed machine learning models on WMR Fix one day in advance, when using historical FX, bonds, and equity market data?"

To start, based on our results, it seems that hybrid-NNs outperform the sole LSTM model. The LSTM model performs dreadfully on the Theil's U metrics. It does, however, have the lowest MAE and MSE, and also predicts the direction of the change correctly the most often (53.094%). We argue that the MSE and MAE only should be looked at when the Theil's U is below 1.0 or when this metric is equivalent between models.

It is debatable as well if adding the Attention layer to an CNN-LSTM architecture improves predictions. Again, we argue that it does not, as the Theil's U of the predictions performs worse. Hence, making the CNN-LSTM the best performing model in this particular case.

Thus, we propose a hybrid neural network consisting of one CNN layer and one LSTM layer with a dropout layer in between for regularisation. The best performing model that we found during tuning consists of 218 neurons in the CNN layer, a dropout of 20%, and 44 neurons in the LSTM layer, resulting in 68,133 parameters. It also has a batch size of 512 and a learning rate of 0.00012. We can confirm that the proposed tanh activation function outperforms the conventional ReLU function for every model.

Coming back to our hypothesis:
We believe that the machine learning model, using FX, bonds, and equity market data, will be able to predict price movements, especially when using end-of-month FX hedge re-balancing data of asset managers and pension funds.

We can confirm the hypothesis, as far as the results from this study go. The model is able to predict price movements, as the Theil's U of both the direction and prediction are respectively 0.99841 and 0.99978, marginally outperforming random selection. However, marginally outperforming random selection can result in significant outperformance over time. In a hypothetical scenario, where we start with 1 Million Dollars, assume no transaction costs, and assume that we can always trade on the WMR fixing rate. In this scenario we buy the EUR/USD at the WMR FIX when the model predicts a higher WMR tomorrow, and sell when the model predicts a lower WMR tomorrow. This results in 1.08 Million at the end of the out-of-sample test set. This means that the model achieves a return of 8.15% during the test set, which consists of around 2 years of observations. When we look at 2020 and 2021 separately, the returns are -4.73% and 13.5% respectively. Assuming no transaction costs is more reasonable than it seems, as the goal of the algorithm is to improve the timing of the transactions that MN has to make, regardless of the prediction.

Although the direction of the movements are predicted decently, when we compare the predictions to the actual values, then it seems that the model has issues extracting temporal dependencies of the relevant driving series. The predictions are much more centered around zero than the actual values, thereby resulting in predictions with toned-down volatility. This might not be surprising, as there might be very few relevant dependencies in historical data.

In our hypothesis we mentioned the end-of-month hedge rebalancing. We tested, and we can confirm, that the model performs better with the end-of-month hedge rebalancing estimations on these on the Theil's U. Yet, it performs worse on the MAE and MSE. We argue that the reason is that the standard deviation of the predictions of the proposed CNN-LSTM is larger.

We tested the proposed CNN-LSTM on robustness, by training 25 identical models, using different initialisation seeds. This resulted in widely divergent outcomes, where only one of the 25 trained models performed as good as the original CNN-LSTM. This means that the model has trouble with optimal convergence.

To increase the explainability of the neural network we performed a sensitivity analysis by training 15 models per parameter, testing a range of each parameter, while keeping all others equal. In this analysis the neurons of the CNN and the learning rate came forward as having the most influence on the validation loss. While the number of neurons in the LSTM has a positive correlation with the validation loss. Meaning that an increase in the neurons in this layer results in a worse outcome.

7.2 Limitations

This study into the application of deep learning methods for forecasting the WMR fix has several limitations. First, the hedge rebalancing flows are, according to the literature, an important predictor for the WMR. Yet, we had to make some significant assumptions to make an estimation of these flows, resulting in a potentially error-prone feature. It is expected that a more complete estimate of these flows will increase model performance.

Another limitation of this study is the lack of convergence of the model we argue has the best performance. The consequence of this is that tests on the importance of the hedge rebalancing feature, and the sensitivity analysis can be influenced by chance.

We also observed possible saturation in two initial architectures for the LSTM and CNN-LSTM networks. This could be the result of the neural networks having trouble learning from the data, as the data is extremely noisy. Indicating that there might be a limited amount to learn from this data to base a prediction upon.

The last limitation is time. Although we were fortunate enough to be able to use extensive computing power and resources, with more time, the outcomes might be improved by testing significantly more architectures and performing more tuning jobs. We were only able to test three main architectures recommended by the literature, still, many more architectures with potential can be tested. We also focused on the main parameters during tuning, yet, there are many more parameters that can be tuned to increase model performance.

7.3 Recommendations

Although the results of the CNN-LSTM are significant, there is much to test and improve upon this model. The recommendations are aimed at improving the current model, as implementation is still far-fetched.

A first step to improve the models proposed in this research, is to formulate a more extensive estimation model on the end-of-month hedge re-balancing flows. The hedge re-balancing model that is used as input for the deep-learning models is relatively simple and can be much improved upon. This model uses limited data on AUM, risk appetite, and hedging ratios, by improving this data quality, the model consequently increases in quality as well. We also did not take possible corporate hedge contracts into account, which could have an important impact on the estimations.

A second step is to perform more model tuning and test a wide range of neural network architectures. When a model is performing satisfactorily, it can be validated in live testing, after which a transaction cost analysis (TCA) can provide more insights into the added values of implementing a DL enhanced method. TCA is not only performed to provide insights into the transaction costs, but also for execution

performance pertaining to the benchmark, which is more relevant here.

In this study, only three possible architectures are tested, and many more should be tested. This thesis, however, provides a framework and a new benchmark to beat in terms of performance. It also shows that a NN can outperform a naive benchmark and simply guessing, which is promising.

7.4 Future Research

The future research is similar to the recommendations, as both go over further developments on top of this study. The suggestion for further research is more general and is going back on the choice to use neural networks. The choice for neural networks was made due to their predictive ability, however, in finance, a crucial part of model creation is also model intelligibility. In previous research into stock and FX prediction researchers often use NNs or basic machine learning models, yet, in this research we want to suggest moving towards models such as a Temporal Fusion Transformer (TFT). This model is still a high achiever, while it is also less of a 'black box' method. A TFT provides the researcher more insight into how predictions are made, for example, which time steps impact the result the most, which is shown in Figure 7.1. Where time index 0 is the current day, and time index -260 is 260 observations back. The Attention variable on the y-axis is the importance certain days have for the prediction. For example, in Figure 7.1, everything more than -140 days back is much less important than more recent observations. And the data point at t-100 is the most important data point for the current predictions. This data point is the observation made 100 days prior to the prediction we want to make. There is no explanation why the TFT assigns a larger significance to certain data points over others.

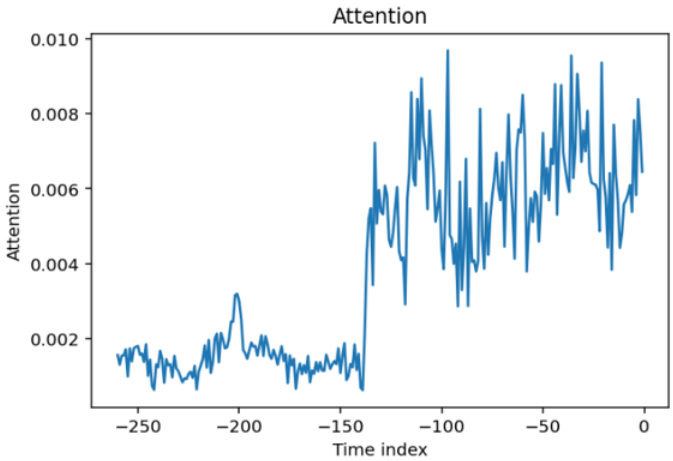

Figure 7.1: TFT attention over time index

The TFT also provides information on which features are most important to come up with the current prediction, as shown in Figure 7.2 and 7.3. It is important to keep in mind that the outcome is not necessarily which features are most important for WMR fix prediction in general, just which features are most important for the

current model to predict the WMR fix. Descriptions of the variables can be found in appendix A.1.

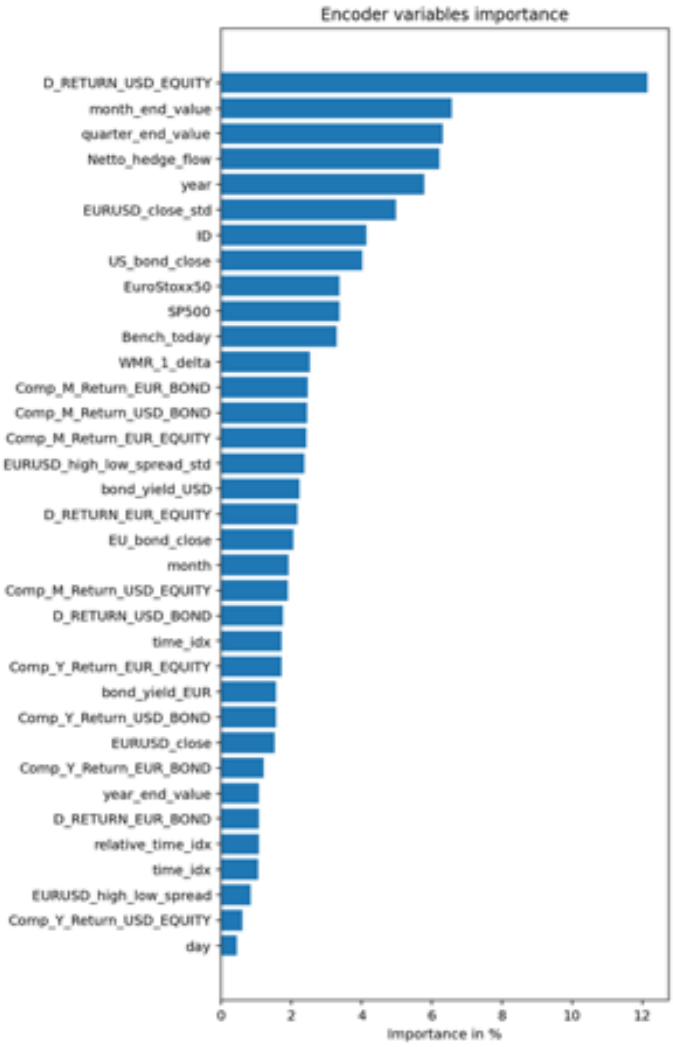

Figure 7.2: TFT encoder variable importance

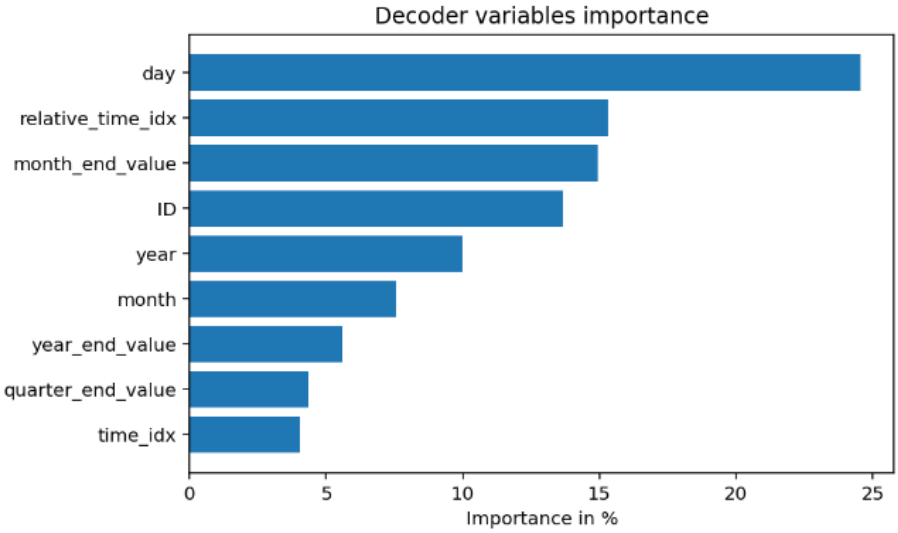

Figure 7.3: TFT decoder variable importance

8. Conclusion

This thesis set out to use deep learning techniques on the prediction of highly noisy financial time series data. According to the literature review, the end-of-month hedge rebalancing flows can have an important impact on the performance of this deep learning model. Hence, we designed a model to capture and prime this data, from which we can confirm that the end-of-month hedge rebalancing data increases the deep learning model performance.

Another finding in the literature is that hybrid neural networks will outperform regular neural networks in most cases. We can confirm this finding, in the case of our application, however, adding an attention layer does not improve the performance further.

The proposed model outperforms the naive benchmark, linear, polynomial, and random forest model, and has a Theil's U score below 1, for both direction and prediction, indicating superiority over guessing. When we assume no transaction costs, this translates to a return of 8.15% over two years when we would buy when the model predicts the WMR will go up and sell when the model predicts that the WMR will go down. These returns consist of -4.73% over 2020 and a 13.5% return over 2021.
But, the black box of neural networks remains a problem, and, essentially, no causality between the hyperparameters and the model performance on validation loss can be found.

This model can easily be applied into practice at MN and PGGM. They have to trade significant flows of EUR/USD, and this model can help the traders in their decision making process regarding the most efficient trading strategy for executing the required flow on a given day.

To conclude, the FX market is a notoriously difficult market to forecast. During this thesis, we developed a model that outperforms both the naive benchmark and random selection, on both direction and predictions, which is a big step in the right direction. The model also achieves a significant return over the two year out-of-sample test set. This study provided evidence of the added value that deep learning can bring to financial time series predictions, and provided a framework on which future developments can be built.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others (2016). Tensorflow: A system for large-scale machine learning. In *12th usenix symposium on operating systems design and implementation (osdi 16)* (pp. 265–283).

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Andonov, A., Bauer, R., & Cremers, M. (2012). Pension fund asset allocation and liability discount rates: camouflage and reckless risk taking by us public plans?

Ang, A., & Chen, J. (2010). Yield curve predictors of foreign exchange returns. In *Afa 2011 denver meetings paper.*

Beardsley, B., Donnadieu, H., Fages, R., Hapelt, C., Heredia, L., Morel, P., ... et al. (2017). *The innovators advantage - boston consulting group.* The Boston Consulting Group. Retrieved from `https://image-src.bcg.com/Images/BCG-The-Innovators-Advantage-July-2017_tcm9-163905.pdf`

Cho, J.-W., Choi, J. H., Kim, T., & Kim, W. (2016). Flight-to-quality and correlation between currency and stock returns. *Journal of Banking & Finance*, *62*, 191–212.

Chung, H., & Shin, K.-s. (2018). Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability*, *10*(10), 3765.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). Stl: A seasonal-trend decomposition. *J. Off. Stat*, *6*(1), 3–73.

Cohen, B. H. (2005). Currency choice in international bond issuance. *J. Payment Sys. L.*, *1*, 473.

D'Agostino, R., & Pearson, E. S. (1973). Tests for departure from normality. empirical results for the distributions of b 2 and b. *Biometrika*, *60*(3), 613–622.

Dixon, M. F., Bilokon, P., & Halperin, I. (2020). *Machine learning in finance: From theory to practice.* Springer.

Djeutem, E., & Dunbar, G. R. (2018). *Uncovered return parity: Equity returns and currency returns* (Tech. Rep.). Bank of Canada Staff Working Paper.

Evans, M. D. (2018). Forex trading and the wmr fix. *Journal of Banking and Finance*, *87*, 233–247.

Evans, M. D., O'Neill, P., Rime, D., Saakvitne, J., et al. (2018). Fixing the fix? assessing the effectiveness of the 4pm fix benchmark. *FCA Occasional Paper*(46).

Fages, R., Beardsley, B., Donnadieu, H., Macé, B., Pardasani, N., Schmitz, L., ... Xu, Q. (2018). The Boston Consulting Group. Retrieved from `https://image-src.bcg.com/Images/BCG-The-Digital-Metamorphosis-July-2018-R_tcm30-197509.pdf`

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, *3*(Aug), 115–143.

Govani, K. (2020, May). *Time series analysis on multivariate data in tensorflow.* Towards Data Science. Retrieved from `https://towardsdatascience.com/time-series-analysis-on-multivariate-data-in-tensorflow-2f0591088502`

Gunduz, H., Yaslan, Y., & Cataltepe, Z. (2017). Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, *137*, 138–148.

Hau, H., & Rey, H. (2004). Can portfolio rebalancing explain the dynamics of equity returns, equity flows, and exchange rates? *American Economic Review*, *94*(2), 126–133.

Hau, H., & Rey, H. (2006). Exchange rates, equity prices, and capital flows. *The Review of Financial Studies*, *19*(1), 273–317.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Heredia, L., Bartletta, S., Carrubba, J., Frankle, D., Kurihara, K., Macé, B., ... et al. (2020). The Boston Consulting Group. Retrieved from `https://web-assets.bcg.com/img-src/BCG-Global-Asset-Management-2020-May-2020-r_tcm9-247209.pdf`

Heredia, L., Bartletta, S., Carrubba, J., Frankle, D., McIntyre, C., Palmisani, E., ... et al. (2021, Jul). BCG Global. Retrieved from `https://www.bcg.com/publications/2021/global-asset-management-industry-report`

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

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

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Hoseinzade, E., & Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, *129*, 273–285.

Hu, Z., Zhao, Y., & Khushi, M. (2021). A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, *4*(1), 9.

Husselmann, C., & Kasikov, K. (2020). Trend-following market behaviour at the 4pm london time bfix and wmr fixing windows. *Quantitative Finance*, *20*(1), 1–8.

Islam, M., Hossain, E., Rahman, A., Hossain, M. S., & Andersson. (2020). A review on recent advancements in forex currency prediction. *Algorithms*, *13*(8), 186.

Jain, S., Gupta, R., & Moghe, A. A. (2018). Stock price prediction on daily stock data using deep neural networks. In *2018 international conference on advanced computation and telecommunication (icacat)* (pp. 1–13).

Jung, G., & Choi, S.-Y. (2021). Forecasting foreign exchange volatility using deep learning autoencoder-lstm techniques. *Complexity*, *2021*.

Kandel, S., Heer, J., Plaisant, C., Kennedy, J., Van Ham, F., Riche, N. H., ...

Buono, P. (2011). Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, *10*(4), 271–288.

Khurana, U., Samulowitz, H., & Turaga, D. (2018). Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

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

Kramer, K., Beardsley, B., Donnadieu, H., Kumar, M., Maguire, A., Morel, P., ... Tang, T. (2011). *G a m. building on success.* The Boston Consulting Group. Retrieved from `https://silo.tips/download/g-a-m-building-on-success`

Li, C., Zhang, X., Qaosar, M., Ahmed, S., Alam, K. M. R., & Morimoto, Y. (2019). Multi-factor based stock price prediction using hybrid neural networks with attention mechanism. In *2019 ieee intl conf on dependable, autonomic and secure computing, intl conf on pervasive intelligence and computing, intl conf on cloud and big data computing, intl conf on cyber science and technology congress (dasc/picom/cbdcom/cyberscitech)* (pp. 961–966).

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Melvin, M., Pan, W., & Wikstrom, P. (2020). Retaining alpha: The effect of trade size and rebalancing frequency on fx strategy returns. *Journal of Financial Markets*, *51*, 100545.

Melvin, M., & Prins, J. (2015). Equity hedging and exchange rates at the london 4 pm fix. *Journal of Financial Markets*, *22*, 50–72.

Michelberger, P. S., & Witte, J. H. (2016). Foreign exchange market microstructure and the wmreuters 4 pm fix. *The Journal of Finance and Data Science*, *2*(1), 26–41.

Ouyang, Z., Ravier, P., & Jabloun, M. (2021). Stl decomposition of time series can benefit forecasting done by statistical methods but not by machine learning ones. *Engineering Proceedings*, *5*(1), 42.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Pojarliev, M. (2018). *Some like it hedged*. CFA Institute Research Foundation.

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

Reddi, S. J., Kale, S., & Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Reuters, T. (2017). *Wm/reuters fx benchmarks–spot & forward rates methodology guide.* November.

Ritchie, C. (2021). Banks scrap or alter unprofitable fx algos targeting month-end. *Bloomberg*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Rundo, F. (2019). Deep lstm with reinforcement learning layer for financial trend prediction in fx high frequency trading systems. *Applied Sciences*, *9*(20), 4460.

Ryll, L., & Seidens, S. (2019). Evaluating the performance of machine learning algorithms in financial market forecasting: A comprehensive survey. *arXiv preprint arXiv:1906.07786*.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, *6*(12), 310–316.

Shivarova, A. (2021). *Matthew f. dixon, igor halperin, and paul bilokon: Machine learning in finance from theory to practice.* Springer.

Shub, G., Bartletta, S., Beardsley, B., Hapelt, C., Donnadieu, H., Macé, B., ... Tang, T. (2014). *Global asset management 2014 steering the course to growth.* BCG Global. Retrieved from `https://www.bcg.com/publications/2014/financial-institutions-global-asset-management-2014-steering-course-growth`

Shub, G., Beardsley, B., Donnadieu, H., Kramer, K., Kumar, M., Maguire, A., ... Tang, T. (2012). The Boston Consulting Group. Retrieved from `https://www.fund-academy.com/assets/Uploads/library/2012-BCG-Global-Asset-Management-Growth.pdf`

Shub, G., Beardsley, B., Donnadieu, H., Macé, B., Mogul, Z., Schwetlick, A., ... et al. (2016). The Boston Consulting Group. Retrieved from `https://www.agefi.fr/sites/agefi.fr/files/fichiers/2016/07/bcg-doubling-down-on-data-july-2016_tcm80-2113701.pdf`

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

Smales, L. A., & Kininmonth, J. N. (2016). Fx market returns and their relationship to investor fear. *International Review of Finance*, *16*(4), 659–675.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Turkington, D., & Yazdani, A. (2020). The equity differential factor in currency markets. *Financial Analysts Journal*, *76*(2), 70–81.

van der Meulen, M., Vermeulen, S., & Tetereva, A. (2021). Deep learning methods with attention mechanism for financial market prediction.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Yang, C., Zhai, J., & Tao, G. (2020). Deep learning for price movement prediction using convolutional neural network and long short-term memory. *Mathematical Problems in Engineering*, *2020*.

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

A. Solution Design

A.1 Activation Functions

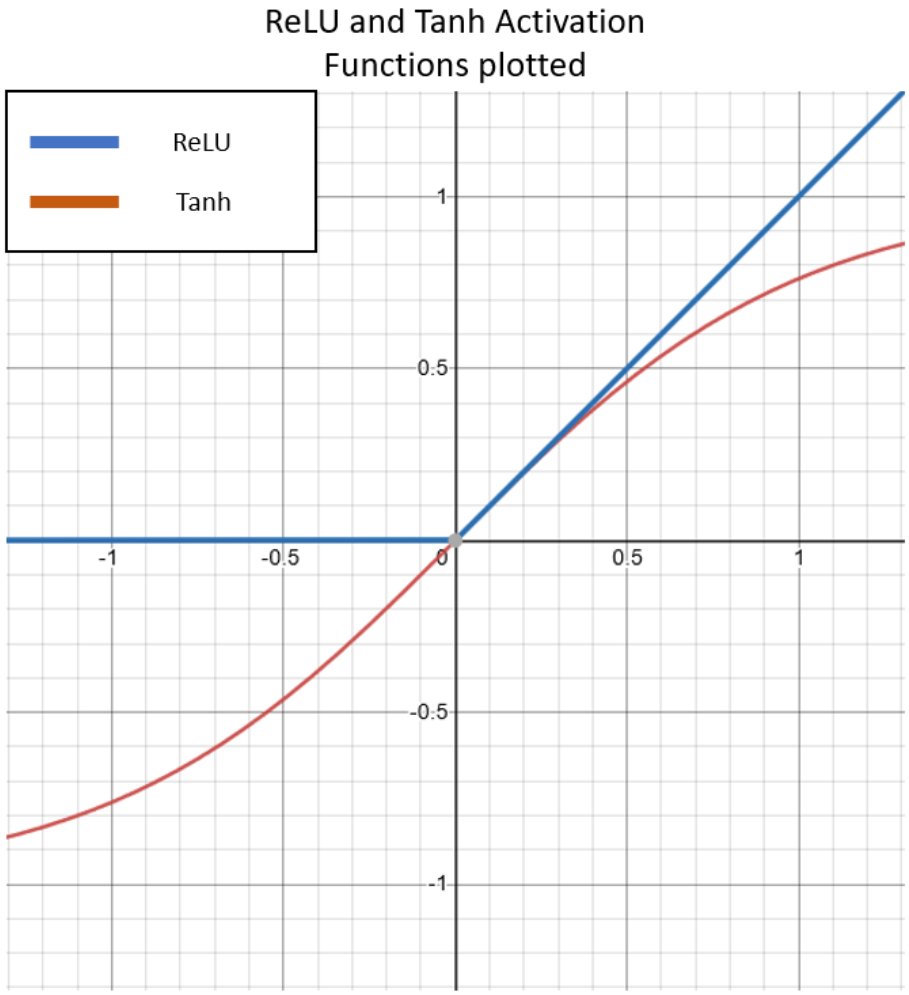

Figure A.1: ReLU and Tanh activation functions

A.2 Feature Order and Meaning Description

When ordering the variables according to the linear correlation with the target variable, as described in 4.2.1, it results in the order that is shown in Table A.1. Where the first feature has the lowest correlation with the target variable.

Table A.1: Feature Order and Meaning Description

Feature name	PPMCC in absolute value	description
year_end_value	0.000442	The last five days of each year get the end-of-period value of respectively 0.2, 0.4, 0.6, 0.8 and 1
EuroStoxx50	0.000821	The daily close of the EuroStoxx50 index
Comp_Y_Return_EUR_BOND	0.003281	The compounded yearly returns of the I02513EU index
quarter_end_value	0.001134	The last five days of the quarter get the end-of-period value of respectively 0.2, 0.4, 0.6, 0.8 and 1
CUMU_US_inflow	0.002200	The estimated cumulative inflow of the US AUM
Comp_Y_Return_EUR_EQUITY	0.003281	Compounded yearly return of the EuroStoxx50 index
EURUSD_high_low_spread_std	1 0.004407	The standard deviation of the daily EURUSD high low spread
Comp_Y_Return_USD_EQUITY	0.004514	Compounded yearly return of the s&p500 index
month_end_value	0.006116	The last five days of each month get the end-of-period value of respectively 0.2, 0.4, 0.6, 0.8 and 1
Comp_M_Return_USD_BOND	0.010554	Compounded monthly return of the LUATTRUU index
CUMU_EU_inflow	0.011551	Cumulative estimated inflow of the EU AUM
month	0.012913	Variable that indicates each month
bond_yield_USD	0.013238	The daily yield of the US 10 year generic government bond
day	0.014245	Variable that indicates each day
SP500	0.015128	The daily close of the s&p500 index
Comp_Y_Return_USD_BOND	0.016123	The compounded yearly return of the LUATTRUU index
US_bond_close	0.017856	The daily close of the LUATTRUU index
EU_bond_close	0.018497	The daily close of the I02513EU index
end_month_estimated_hedge_flow	0.018603	The estimated hedge flows that are expected at the end of the month, distributed over the last three days
D_RETURN_EUR_EQUITY	0.018691	The daily return of the EuroStoxx50 index
bond_yield_EUR	0.019254	The daily yield of the EU 10 year generic government bond
year	0.019720	Variable that indicates each year
EURUS_close_std	0.021758	Standard deviation of the daily EURUSD close
Comp_M_Return_EUR_BOND	0.022666	Compounded monthly return of the I02513EU index
EURUSD_high_low_spread	0.024658	The daily high-low spread of the EURUSD
EURUSD_close	0.030118	The daily mean price of the EURUSD
Bench_today	0.034290	WMR fix at time t
Comp_M_Return_EUR_EQUITY	0.035197	Compounded monthly return of the EuroStoxx50 index
D_RETURN_EUR_BOND	0.045929	Daily return of the I02513EU index
D_RETURN_USD_BOND	0.052042	daily return of the LUATTRUU index
Comp_M_Return_USD_EQUITY	0.065739	Compounded monthly return of the s&p500 index
Net_hedge_flow	0.068200	The estimated net hedge flows accumulating over each month
D_RETURN_USD_EQUITY	0.108225	Daily return of the s&p500 index
Following Variables Are Only Used In THe TFT Model		
ID	-	ID given per certain time period as possible indication of structural breaks
Time$_idx$	-	Counter that starts at zero, and counts up for every observation

B. Results

This is the Appendix of the results chapter. The first two sections go over alternative architectures of the LSTM and CNN-LSTM NNs, as they resulted in a (almost) constant prediction. The hypothesis is that this is caused by weight saturation.

B.1 LSTM Architecture with Saturation

The result of the three layer LSTM architecture shown below is completely constant, even after hyperparameter tuning. This resulted in a MAE of 0.188836 and MSE of 0.05021.

Table B.1: LSTM Architecture Results with Saturation

Layer (type)	Neurons	Param #
LSTM	68	178620
Dropout	None (0.4% dropout)	0
LSTM	68	304980
Dropout	None (0.4% dropout)	0
LSTM	68	304980
Dropout	None (0.4% dropout)	0
Dense	1	196
Total params:	788.776	
Trainable params	788.776	
Non-trainable params:	0	

Batch Size = 32, Optimiser = Adam(Learning rate = 0.03156), Loss Function = Huber Loss, Validation Split = 0.1, Activation = Tanh

The eventual result for the prediction of the WMR delta:

B.2 CNN-LSTM Architecture with Saturation

Although the saturation is not as prominent as in the LSTM architecture, the outcome is still relatively constant.
The eventual result for the prediction of the WMR delta:

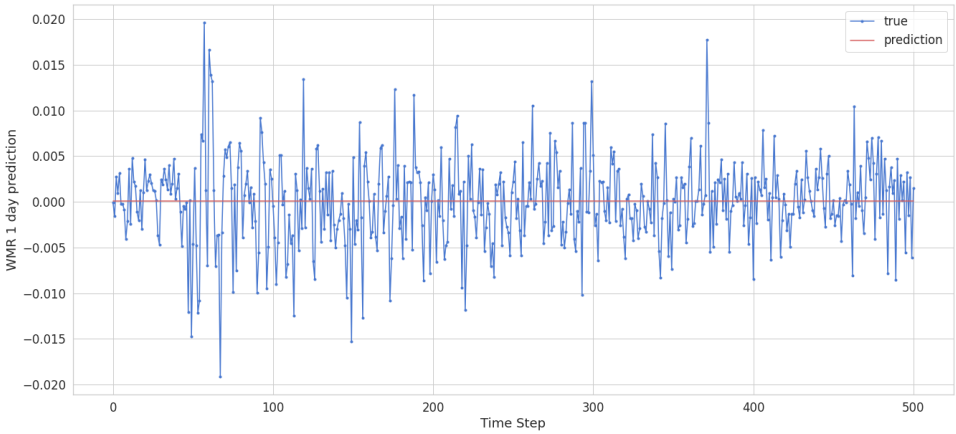

Figure B.1: Result of a possibly saturated LSTM network

Table B.2: Scalar comparison CNN-LSTM

Layer (type)	Neurons	Param #
Conv1D	256	2600
MaxPooling1D	None (Pool size 2)	0
Dropout	None (20% dropout)	0
Conv1D	256	2054
MaxPooling1D	None (Pool size 2)	0
Dropout	None (20% dropout)	0
LSTM	26	173160
Dropout	None (20% dropout)	0
LSTM	26	304980
Dropout	None (20% dropout)	0
LSTM	26	304980
Dropout	None (20% dropout)	0
Dense	1	196
Total params:	787.970	
Trainable params	787.970	
Non-trainable params:	0	

Batch Size = 256, Optimiser = Adam(Learning rate = 0.00010), Loss Function = Huber Loss, Validation Split = 0.1, Activation = Tanh

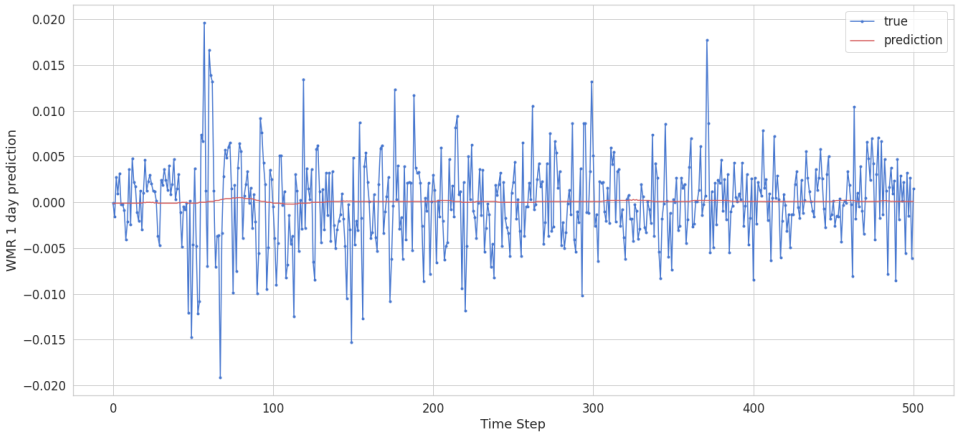

Figure B.2: Result of a possibly saturated CNN-LSTM network

B.3 CNN-LSTM results

This section displays the results of the CNN-LSTM model against the actual and naive prediction values, as alternative to the zoomed-in version in the main body of the thesis.

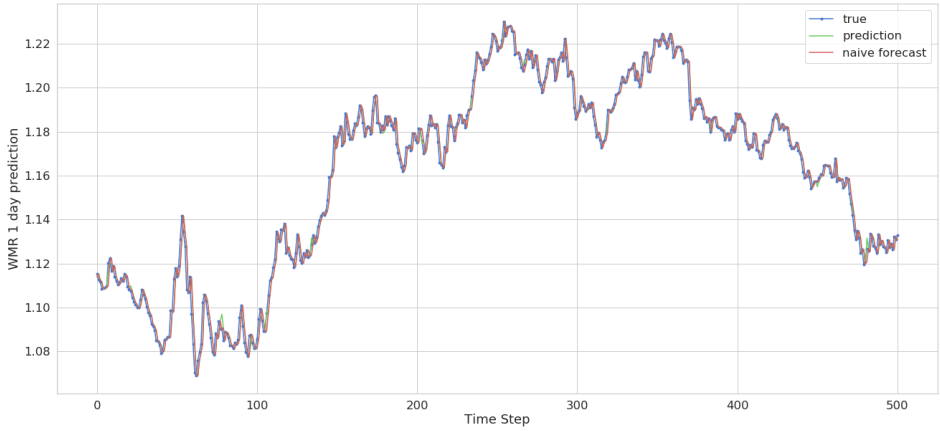

Figure B.3: CNN-LSTM WMR prediction results against the actual and naive prediction values

B.4 Outcomes of the Hyperparameter Tuning

Below the exact results of the hyperparameter jobs are shown. This includes the different models that were trained, and in which order the Bayesian optimiser tried the models.

Table B.3: LSTM tuning results

	BATCH_SIZE	N_UNITS_LSTM	activation_function	dropout	learning_rate	TrainingJobStatus	FinalObjectiveValue	TrainingElapsedTimeSeconds
22	"128"	37.0	"tanh"	"0.2"	0.016847	Completed	2.160000e-02	272.0
30	"256"	64.0	"tanh"	"0.4"	0.041372	Completed	2.160000e-02	298.0
27	"16"	139.0	"tanh"	"0.3"	0.000100	Completed	2.170000e-02	499.0
29	"16"	160.0	"tanh"	"0.2"	0.000107	Completed	2.170000e-02	483.0
9	"16"	8.0	"tanh"	"0.3"	0.021216	Completed	2.170000e-02	373.0
35	"512"	179.0	"tanh"	"0.3"	0.002939	Completed	2.170000e-02	313.0
18	"64"	179.0	"relu"	"0.4"	0.000149	Completed	2.180000e-02	1438.0
0	"512"	173.0	"relu"	"0.4"	0.002409	Completed	2.180000e-02	486.0
4	"256"	256.0	"tanh"	"0.4"	0.002389	Completed	2.200000e-02	318.0
7	"16"	8.0	"tanh"	"0.3"	0.019800	Completed	2.210000e-02	346.0
21	"512"	195.0	"tanh"	"0.4"	0.016089	Completed	2.210000e-02	273.0
39	"128"	96.0	"tanh"	"0.3"	0.000748	Completed	2.230000e-02	298.0
31	"64"	123.0	"tanh"	"0.3"	0.028978	Completed	2.230000e-02	333.0
38	"32"	54.0	"relu"	"0.3"	0.004614	Completed	2.230000e-02	1576.0
8	"256"	40.0	"tanh"	"0.2"	0.000100	Completed	2.240000e-02	333.0
34	"16"	69.0	"relu"	"0.3"	0.003081	Completed	2.240000e-02	2232.0
23	"16"	59.0	"tanh"	"0.4"	0.000105	Completed	2.250000e-02	403.0
24	"512"	177.0	"tanh"	"0.2"	0.010422	Completed	2.250000e-02	293.0
47	"256"	230.0	"relu"	"0.2"	0.000159	Completed	2.270000e-02	622.0
49	"512"	170.0	"relu"	"0.2"	0.000128	Completed	2.270000e-02	635.0
12	"256"	36.0	"relu"	"0.4"	0.000647	Completed	2.270000e-02	570.0
19	"64"	177.0	"relu"	"0.4"	0.000160	Completed	2.280000e-02	1067.0
2	"64"	38.0	"tanh"	"0.2"	0.000135	Completed	2.280000e-02	288.0
33	"16"	65.0	"relu"	"0.3"	0.003081	Completed	2.280000e-02	2907.0
10	"16"	114.0	"relu"	"0.4"	0.001014	Completed	2.280000e-02	2206.0
26	"64"	105.0	"tanh"	"0.4"	0.000367	Completed	2.290000e-02	334.0
48	"32"	233.0	"tanh"	"0.3"	0.020414	Completed	2.320000e-02	368.0
43	"32"	155.0	"relu"	"0.4"	0.000694	Completed	2.320000e-02	1710.0
40	"512"	27.0	"tanh"	"0.2"	0.000270	Completed	2.320000e-02	339.0
41	"256"	123.0	"tanh"	"0.3"	0.001211	Completed	2.330000e-02	329.0
37	"32"	129.0	"tanh"	"0.4"	0.010532	Completed	2.330000e-02	343.0
45	"256"	73.0	"relu"	"0.2"	0.002670	Completed	2.330000e-02	635.0
36	"16"	25.0	"relu"	"0.2"	0.001624	Completed	2.370000e-02	2760.0
17	"32"	137.0	"tanh"	"0.2"	0.016110	Completed	2.390000e-02	341.0
46	"256"	234.0	"relu"	"0.2"	0.000182	Completed	2.410000e-02	620.0
32	"64"	249.0	"tanh"	"0.2"	0.003045	Completed	2.540000e-02	345.0
15	"128"	86.0	"tanh"	"0.4"	0.006840	Completed	2.580000e-02	323.0
25	"16"	247.0	"tanh"	"0.4"	0.036580	Completed	2.630000e-02	469.0
44	"32"	174.0	"tanh"	"0.3"	0.000616	Completed	2.830000e-02	394.0
11	"16"	112.0	"relu"	"0.4"	0.001086	Completed	3.190000e-02	2578.0
42	"128"	152.0	"relu"	"0.4"	0.007007	Completed	1.435024e+15	635.0
28	"128"	111.0	"relu"	"0.3"	0.000951	Completed	2.808448e+17	851.0

Table B.4: CNN-LSTM tuning results

	BATCH_SIZE	N_UNITS_CNN	N_UNITS_LSTM	activation_function	dropout	learning_rate	TrainingJobStatus	FinalObjectiveValue	TrainingElapsedTimeSeconds
20	"512"	218.0	44.0	"tanh"	"0.2"	0.000125	Completed	2.190000e-02	293.0
26	"128"	226.0	40.0	"tanh"	"0.3"	0.029006	Completed	2.200000e-02	228.0
41	"64"	177.0	43.0	"tanh"	"0.2"	0.012512	Completed	2.200000e-02	238.0
35	"512"	253.0	80.0	"tanh"	"0.3"	0.001882	Completed	2.210000e-02	243.0
45	"512"	73.0	174.0	"tanh"	"0.2"	0.004822	Completed	2.210000e-02	248.0
38	"128"	98.0	256.0	"tanh"	"0.2"	0.000230	Completed	2.210000e-02	263.0
12	"512"	161.0	14.0	"tanh"	"0.2"	0.000366	Completed	2.210000e-02	287.0
32	"512"	256.0	49.0	"tanh"	"0.2"	0.000760	Completed	2.220000e-02	232.0
17	"512"	42.0	20.0	"tanh"	"0.2"	0.020344	Completed	2.220000e-02	263.0
30	"32"	226.0	47.0	"tanh"	"0.4"	0.000456	Completed	2.220000e-02	301.0
29	"512"	256.0	116.0	"tanh"	"0.2"	0.007975	Completed	2.220000e-02	217.0
40	"512"	140.0	167.0	"tanh"	"0.2"	0.000100	Completed	2.220000e-02	283.0
44	"512"	247.0	77.0	"tanh"	"0.4"	0.025329	Completed	2.220000e-02	255.0
8	"32"	67.0	205.0	"tanh"	"0.2"	0.003483	Completed	2.230000e-02	243.0
24	"16"	91.0	151.0	"tanh"	"0.4"	0.019253	Completed	2.230000e-02	269.0
46	"512"	86.0	68.0	"tanh"	"0.2"	0.004135	Completed	2.230000e-02	268.0
28	"16"	20.0	24.0	"tanh"	"0.4"	0.001989	Completed	2.230000e-02	235.0
34	"64"	173.0	17.0	"tanh"	"0.4"	0.000909	Completed	2.240000e-02	246.0
43	"512"	247.0	81.0	"tanh"	"0.4"	0.029081	Completed	2.240000e-02	239.0
21	"512"	220.0	46.0	"tanh"	"0.2"	0.000116	Completed	2.240000e-02	314.0
11	"128"	158.0	238.0	"tanh"	"0.3"	0.000131	Completed	2.240000e-02	258.0
33	"64"	94.0	60.0	"tanh"	"0.3"	0.004062	Completed	2.250000e-02	252.0
31	"64"	161.0	26.0	"tanh"	"0.2"	0.011590	Completed	2.250000e-02	253.0
0	"64"	39.0	140.0	"relu"	"0.2"	0.000432	Completed	2.280000e-02	609.0
19	"16"	253.0	42.0	"tanh"	"0.4"	0.025454	Completed	2.290000e-02	328.0
42	"512"	79.0	184.0	"tanh"	"0.4"	0.046506	Completed	2.290000e-02	258.0
14	"16"	188.0	55.0	"relu"	"0.4"	0.000180	Completed	2.300000e-02	1674.0
48	"256"	8.0	52.0	"tanh"	"0.2"	0.000424	Completed	2.300000e-02	303.0
49	"128"	132.0	140.0	"tanh"	"0.4"	0.005688	Completed	2.310000e-02	293.0
39	"16"	64.0	204.0	"tanh"	"0.3"	0.000227	Completed	2.320000e-02	269.0
18	"32"	228.0	34.0	"tanh"	"0.3"	0.000519	Completed	2.330000e-02	278.0
47	"256"	114.0	157.0	"relu"	"0.2"	0.001048	Completed	2.380000e-02	454.0
36	"128"	192.0	222.0	"relu"	"0.2"	0.000143	Completed	2.410000e-02	479.0
6	"128"	28.0	67.0	"relu"	"0.4"	0.000372	Completed	2.420000e-02	534.0
37	"64"	184.0	222.0	"tanh"	"0.2"	0.006992	Completed	2.440000e-02	248.0
1	"64"	37.0	142.0	"relu"	"0.2"	0.000403	Completed	2.450000e-02	685.0
16	"16"	186.0	57.0	"relu"	"0.4"	0.000168	Completed	2.480000e-02	2491.0
27	"512"	16.0	130.0	"tanh"	"0.3"	0.031888	Completed	2.540000e-02	213.0
7	"128"	26.0	65.0	"relu"	"0.4"	0.000347	Completed	2.640000e-02	403.0
4	"512"	226.0	109.0	"relu"	"0.4"	0.000232	Completed	3.160000e-02	288.0
5	"512"	224.0	107.0	"relu"	"0.4"	0.000217	Completed	3.810000e-02	319.0
13	"128"	135.0	163.0	"tanh"	"0.2"	0.048971	Completed	4.210000e-02	288.0
10	"256"	254.0	136.0	"relu"	"0.3"	0.003277	Completed	5.833514e+10	360.0

Table B.5: Attention CNN-LSTM tuning results

	BATCH_SIZE	N_UNITS_CNN	N_UNITS_LSTM	activation_function	dropout	learning_rate	TrainingJobStatus	FinalObjectiveValue	TrainingElapsedTimeSeconds
20	"32"	210.0	209.0	"tanh"	"0.4"	0.000179	Completed	0.0216	40803.0
25	"16"	208.0	145.0	"tanh"	"0.4"	0.001253	Completed	0.0219	31783.0
23	"32"	226.0	132.0	"tanh"	"0.2"	0.041518	Completed	0.0219	14276.0
34	"32"	216.0	176.0	"tanh"	"0.4"	0.001466	Completed	0.0219	24475.0
15	"64"	32.0	37.0	"tanh"	"0.4"	0.000494	Completed	0.0219	8479.0
29	"256"	207.0	134.0	"tanh"	"0.4"	0.000580	Completed	0.0220	2861.0
2	"256"	249.0	86.0	"tanh"	"0.3"	0.000828	Completed	0.0220	2912.0
46	"256"	148.0	85.0	"tanh"	"0.4"	0.000355	Completed	0.0220	3490.0
42	"256"	208.0	171.0	"tanh"	"0.4"	0.000746	Completed	0.0220	3799.0
36	"32"	218.0	176.0	"tanh"	"0.4"	0.001277	Completed	0.0220	16073.0
4	"16"	138.0	178.0	"tanh"	"0.4"	0.004643	Completed	0.0221	32752.0
43	"256"	209.0	167.0	"tanh"	"0.4"	0.000650	Completed	0.0221	4643.0
41	"256"	212.0	171.0	"tanh"	"0.4"	0.000746	Completed	0.0221	3140.0
13	"256"	172.0	210.0	"tanh"	"0.3"	0.000914	Completed	0.0222	2488.0
38	"16"	249.0	55.0	"tanh"	"0.4"	0.003812	Completed	0.0223	8510.0
16	"32"	245.0	199.0	"tanh"	"0.4"	0.000297	Completed	0.0223	14600.0
17	"32"	153.0	232.0	"relu"	"0.3"	0.000237	Completed	0.0223	18446.0
35	"32"	220.0	172.0	"tanh"	"0.4"	0.001277	Completed	0.0223	18078.0
10	"64"	32.0	41.0	"tanh"	"0.4"	0.000431	Completed	0.0223	9084.0
22	"32"	230.0	128.0	"tanh"	"0.2"	0.041518	Completed	0.0223	8335.0
1	"256"	249.0	86.0	"tanh"	"0.3"	0.000790	Completed	0.0223	2874.0
30	"32"	198.0	162.0	"tanh"	"0.4"	0.000148	Completed	0.0223	19024.0
24	"32"	68.0	41.0	"tanh"	"0.2"	0.000395	Completed	0.0223	6030.0
47	"128"	186.0	243.0	"tanh"	"0.2"	0.003047	Completed	0.0224	4884.0
31	"256"	224.0	181.0	"tanh"	"0.4"	0.001046	Completed	0.0224	3159.0
37	"256"	179.0	130.0	"tanh"	"0.4"	0.000234	Completed	0.0225	3943.0
8	"32"	256.0	169.0	"tanh"	"0.4"	0.000826	Completed	0.0225	18357.0
21	"32"	209.0	205.0	"tanh"	"0.4"	0.000179	Completed	0.0225	44420.0
49	"64"	65.0	133.0	"tanh"	"0.2"	0.001044	Completed	0.0226	8614.0
28	"256"	220.0	79.0	"tanh"	"0.4"	0.000102	Completed	0.0226	2678.0
26	"256"	214.0	114.0	"tanh"	"0.4"	0.000489	Completed	0.0227	2780.0
0	"16"	142.0	178.0	"tanh"	"0.4"	0.004643	Completed	0.0228	29515.0
33	"64"	23.0	203.0	"relu"	"0.4"	0.005861	Completed	0.0229	10448.0
14	"16"	58.0	44.0	"tanh"	"0.4"	0.000282	Completed	0.0229	1942.0
11	"64"	36.0	41.0	"tanh"	"0.4"	0.000431	Completed	0.0229	7474.0
40	"256"	63.0	38.0	"tanh"	"0.4"	0.005785	Completed	0.0229	2131.0
48	"16"	57.0	217.0	"relu"	"0.3"	0.015260	Completed	0.0231	32777.0
32	"32"	167.0	120.0	"tanh"	"0.4"	0.001998	Completed	0.0231	8566.0
45	"16"	117.0	236.0	"relu"	"0.2"	0.040870	Completed	0.0231	36724.0
5	"32"	211.0	207.0	"tanh"	"0.4"	0.000167	Completed	0.0232	24943.0
12	"256"	240.0	211.0	"tanh"	"0.4"	0.000133	Completed	0.0232	2911.0
44	"256"	124.0	163.0	"relu"	"0.2"	0.069883	Completed	0.0232	4048.0
6	"32"	86.0	139.0	"relu"	"0.3"	0.000228	Completed	0.0233	15954.0
39	"16"	243.0	38.0	"tanh"	"0.3"	0.000323	Completed	0.0233	7356.0
18	"128"	30.0	250.0	"tanh"	"0.4"	0.042171	Completed	0.0236	4828.0
9	"64"	32.0	37.0	"tanh"	"0.4"	0.000431	Completed	0.0238	7305.0
27	"32"	226.0	128.0	"tanh"	"0.2"	0.041518	Completed	0.0239	12667.0
19	"128"	209.0	44.0	"tanh"	"0.4"	0.062154	Completed	0.0240	4731.0
3	"256"	249.0	86.0	"tanh"	"0.3"	0.000721	Completed	0.0243	3342.0
7	"32"	153.0	232.0	"relu"	"0.3"	0.000272	Completed	0.0244	17482.0

B.5 Visualised Results of the Models

The figures below display the visualised results from the LSTM and Attention CNN-LSTM, similarly to how the results of the CNN-LSTM were displayed previously.

B.5.1 LSTM

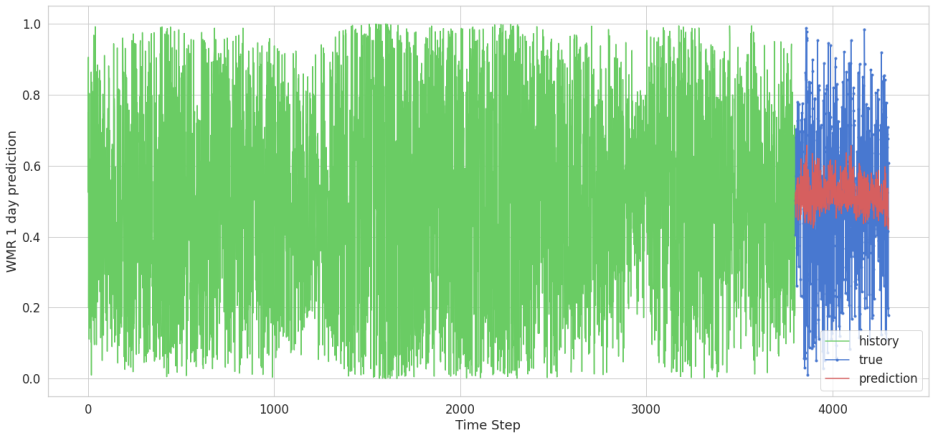

Figure B.4: LSTM delta prediction transformed

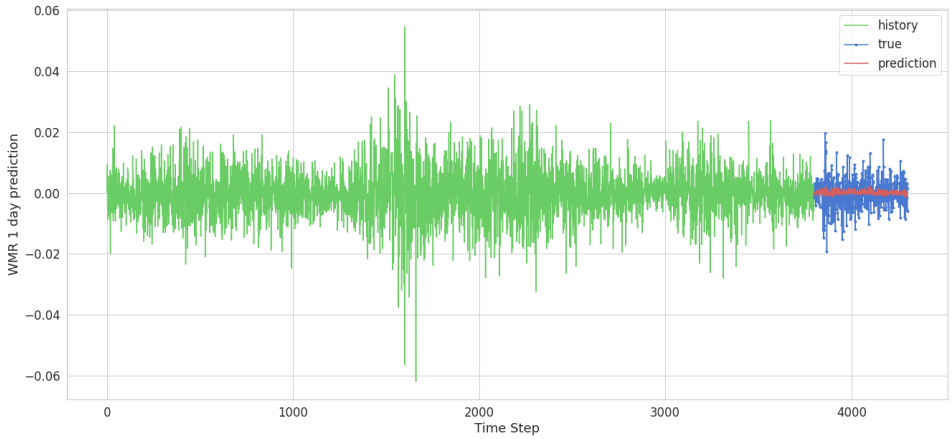

Figure B.5: LSTM delta prediction with training

B.5.2 CNN-LSTM

This subsection shows some plots of the CNN-LSTM that were not displayed in the main study.

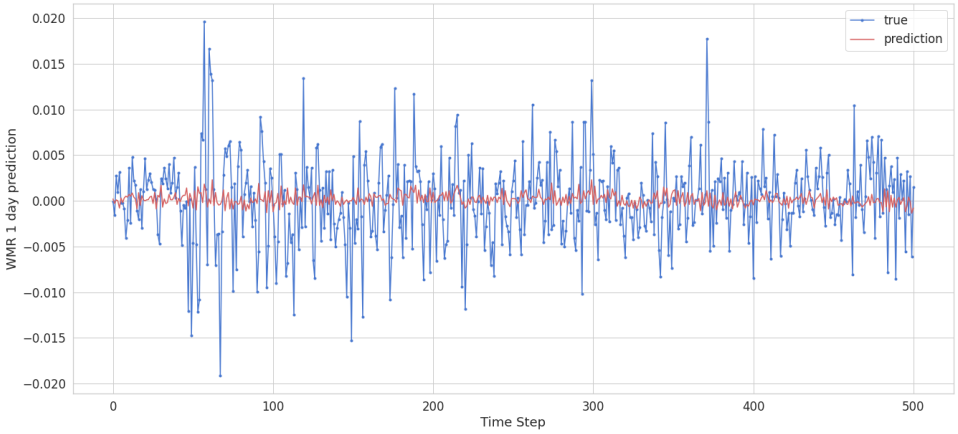

Figure B.6: LSTM delta prediction

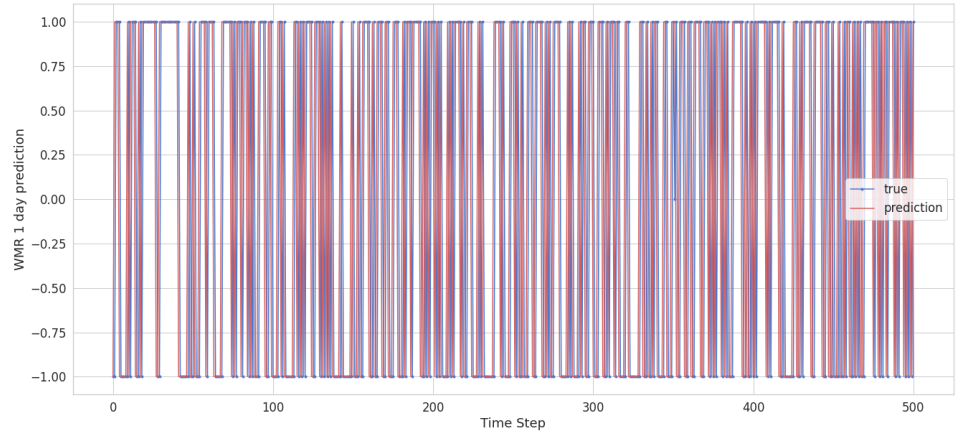

Figure B.7: LSTM Direction Plot

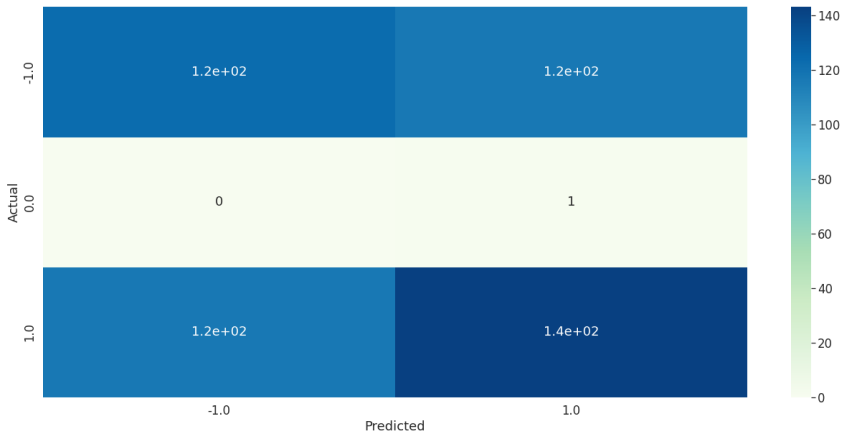

Figure B.8: LSTM Confusion Matrix

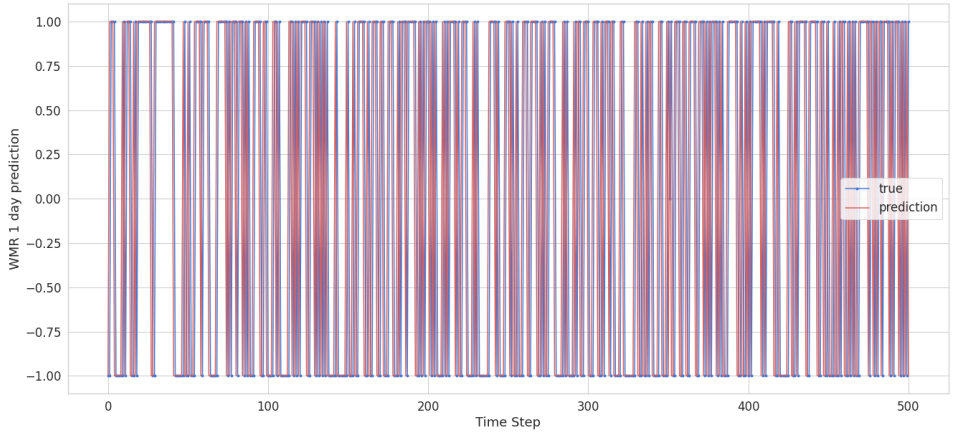

Figure B.9: CNN-LSTM Direction Plot

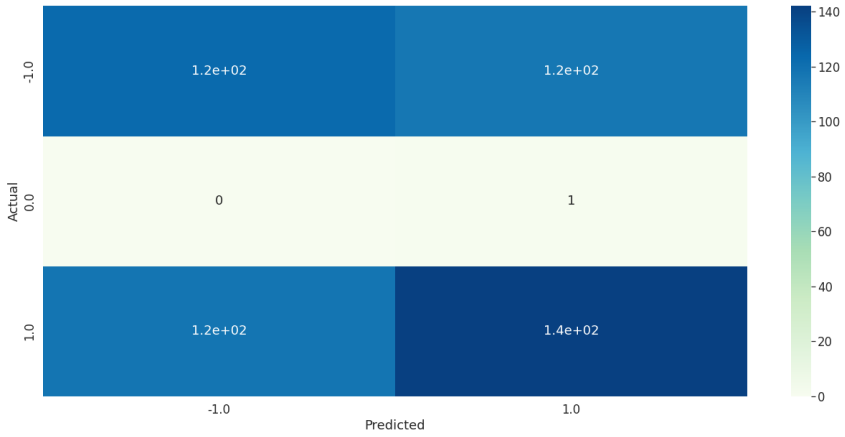

Figure B.10: CNN-LSTM Confusion Matrix

B.5.3 Attention CNN-LSTM

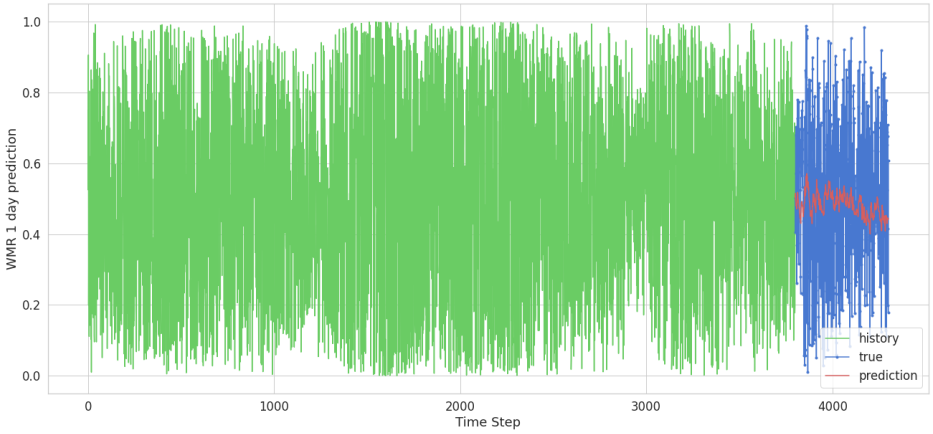

Figure B.11: Attention CNN-LSTM delta prediction transformed

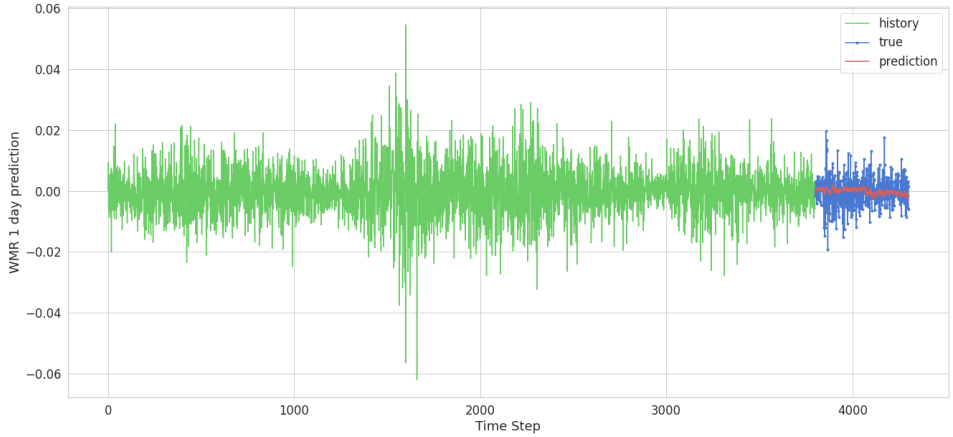

Figure B.12: Attention CNN-LSTM delta prediction with training

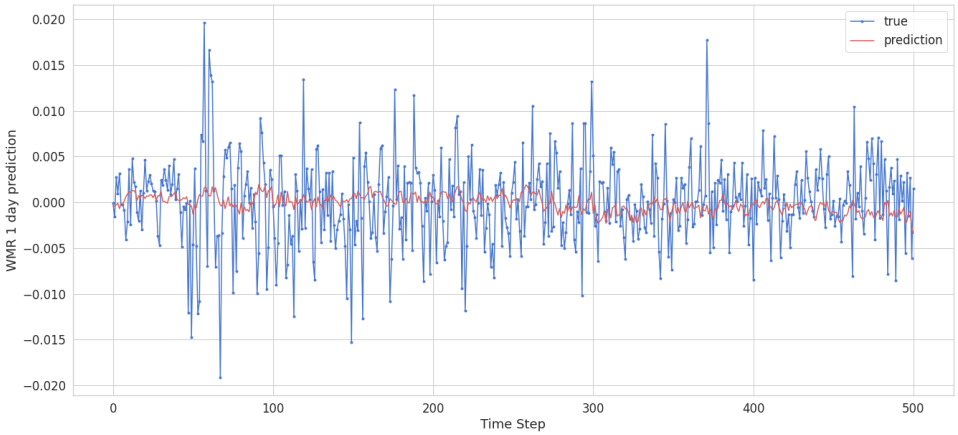

Figure B.13: Attention CNN-LSTM delta prediction

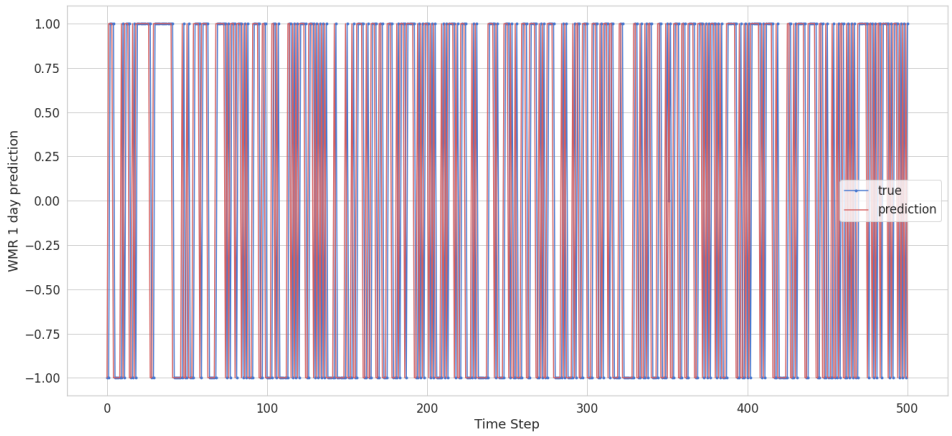

Figure B.14: LSTM Direction Plot

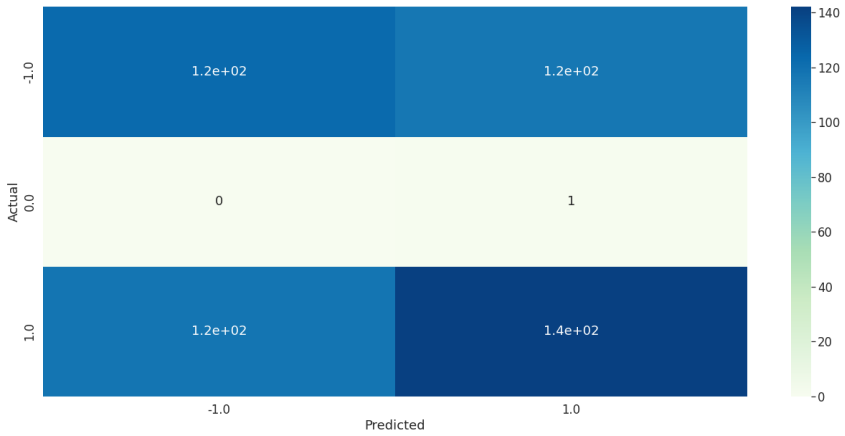

Figure B.15: Attention CNN-LSTM Confusion Matrix

B.6 Robustness

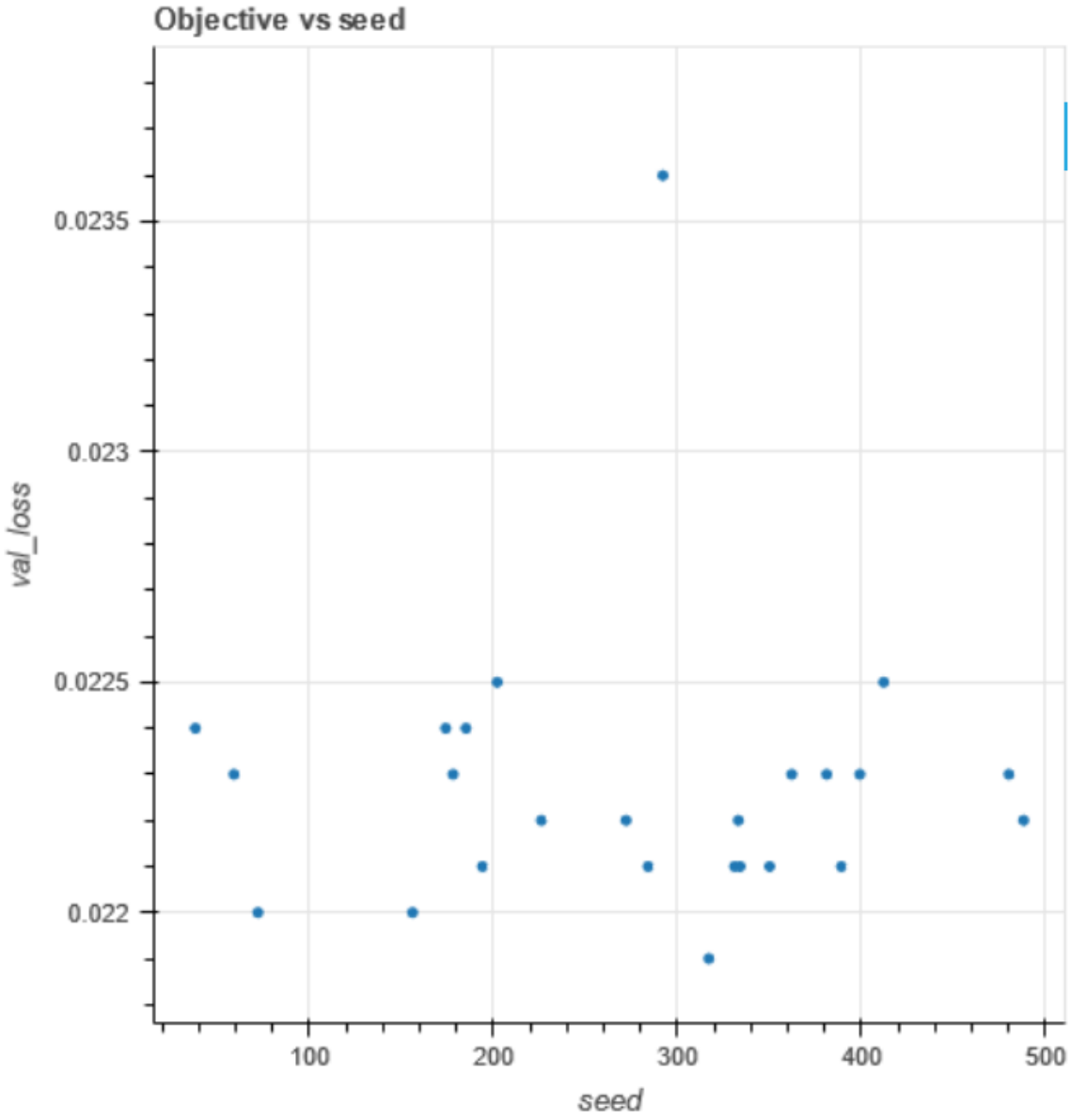

Figure B.16: Robustness of CNN-LSTM performance