LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK

FORSCHUNGSGRUPPE
DATA MINING IN DER MEDIZIN

Bachelor Thesis
in Computer Science

Predictive Capabilities of LSTM Networks: A Case Study of the STOXX Europe 600 Index

Valentin Hasner

Aufgabensteller:	Prof. Dr. Christian Böhm
Betreuer:	Mauritius Klein
Abgabedatum:	14.04.2023

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This paper was not previously presented to another examination board and has not been published.

München, 14.04.2023

.....
Valentin Hasner

Abstract

This Bachelor thesis investigates the predictive capabilities of long short-term memory (LSTM) models for financial markets, focusing on the STOXX Europe 600 index. In addition, the study investigates the effect of including a vector representation of time as a supplementary information source on the networks' prediction performance. In parallel, the effects of different regularisation settings are examined, while the financial performances of the deep learning (DL) models are assessed.

During the experiments, two LSTM-based DL networks were deployed to predict out-of-sample directional movements for each STOXX Europe 600 index constituent from 2005 until 2022. First, an alternative regularisation mechanism was used for both LSTM models compared to the implementation by Fischer and Krauss [14]. Second, a trainable embedding layer that extracts the time feature inherent in the financial data extended one of the two models. For evaluation purposes, two other benchmark models were included in addition to the Fischer and Krauss framework: a buy-and-hold approach (BH) and a decision heuristic based on the short-term reversal effect (STR).

This study builds upon the LSTM framework of Fischer and Krauss [14] to investigate whether their findings extend to more recent periods and the European market. The network performances follow a similar trajectory but are offset in time by a few years. The results reveal that the DL networks initially achieve high accuracy rates of up to 52.52 %, but the performances gradually decline and fall behind the benchmarks from 2017 onwards. These developments could be attributed to the low signal-to-noise environment, increased technology distribution, and lower entry barriers for smaller investors. Moreover, the study highlights that slight differences in the regularisation settings can lead to significant differences in the computation time. Finally, incorporating vector representations of time as additional input features did not improve the network's performance.

Contents

1 Introduction **4**

2 Theory **8**
2.1 Deep Learning 10
2.2 Recurrent Neural Networks 14

3 Related Work **17**
3.1 Overview 17
3.2 Closely Related 19

4 Methodology **21**
4.1 Software and Hardware 21
4.2 Data 22
4.2.1 Raw Data 23
4.2.2 Data Cleaning with R 25
4.2.3 Data Preparation with Python 27
4.3 Benchmark 29
4.4 Deep Learning Architecture 30
4.4.1 Long Short-term Memory 30
4.4.2 Long Short-term Memory with Time2Vec Extension . . . 31
4.5 Forecasting, Ranking, Trading and Evaluation 33

5 Results **36**
5.1 Deep Learning Evaluation 36
5.1.1 Evaluation of Training and Validation Results 36
5.1.2 Evaluation of Different Regularisation Mechanisms . . . 40
5.1.3 Evaluation of Test Results 45
5.2 Investment Evaluation 49

6 Conclusion **58**

A Software Settings **62**

B Clean Data Statistics **64**

C Long Short-term Memory Network Architecture by Fischer and Krauss **65**

D Modified Long Short-term Memory Network Architecture **66**

E Modified Long Short-term Memory with Time2Vec Network Architecture **67**

F Modified Network Settings **68**

Bibliography **69**

Acronyms

AI Artificial intelligence. 8, 9

BH Buy and hold. 1, 5, 6, 29, 50, 52–56

DJIA Dow Jones Industrial Average. 18, 19

DL Deep learning. 1, 4–11, 14, 16–20, 22, 23, 28–31, 33, 34, 36, 39, 45, 47–50, 52, 54–61

LSTM Long short-term memory. 1, 4–6, 16, 18, 19, 30–33, 36–56, 58, 60, 61, 65–68

ML Machine learning. 8, 9, 12, 17, 18, 22, 31, 41, 45

PCA Principal component analysis. 8

PIT Point-in-time. 25–27

RNN Recurrent neural network. 14–16, 18–20

STR Short-term reversal. 1, 6, 29, 34, 45–51, 53–55

T2V Time2Vector. 5, 6, 30–33, 36–41, 43–56, 58, 60, 67, 68

Chapter 1

Introduction

The rapid advancements in the DL industry have led to the development of groundbreaking technologies, including the recently developed finance-related DL network BloombergGPT [38]. Inspired by the remarkable achievements of ChatGPT, BloombergGPT is a new model with 50 billion parameters specially designed to assist with finance-related challenges such as sentiment analysis and binary classification problems [38]. As the topic of DL in finance gains momentum, it becomes apparent that this technology has vast and diverse applications. From predicting market trends to optimising investment strategies, the possibilities are endless. In addition to larger models, smaller, specialised DL networks may also be beneficial, depending on the task and available resources. This thesis aims to investigate the performance of smaller DL networks and their potential contribution to the field of finance.

The last decade has witnessed considerable technical progress in DL research, facilitated by the availability of large and clean datasets, powerful hardware, and the proliferation of open-source and proprietary software [7]. This technological breakthrough has led to a surge in the use of DL networks in various fields, including finance, where they have proven effective in various use cases, such as risk management, portfolio management, and fraud detection, as evidenced by the extensive literature reviews conducted by Kumbure et al., Ozbayoglu et al., and Sezer et al. [25, 28, 31]. In particular, DL models have been employed to predict the price movements of various assets, which has been an area of active research. Among the various DL models, LSTM networks have emerged as particularly suitable for this purpose, owing to their ability to capture long-term relationships within the data. Developed by Hochreiter and Schmidhuber in 1997, the LSTM architecture has since become the foundation of high-performance neural networks for sequential data sets or data sets with medium- to long-term dependencies [21].

One of the recent research projects on LSTM models that has garnered

attention is the 2018 paper by Fischer and Krauss [14]. The study extensively investigated a smaller LSTM network's efficacy in predicting future movements in the stock market. Their work, among others, has inspired the realisation of this thesis. While the Transformer approach based on the 2017 article 'Attention is all you need' by Vaswani et al. [34] has introduced entirely new network architectures that replace previous industry standards [7], smaller innovations also regularly emerge to extend and improve the currently used models, such as the Time2Vector (T2V) concept proposed by Kazemi et al. in their paper 'Time2Vec: learning a vector representation of time' [27]. Since most financial data are time series with short- to long-term dependencies, the influence of time as a vectorised input factor is additionally examined. For this purpose, the T2V principle, according to Kazemi et al., is implemented and tested [27].

This research project investigates the extent to which LSTM models can provide sustainable predictions in a complex environment with a low signal-to-noise ratio like the financial markets. Building on the research of Fischer and Krauss, the study examines to what extent their observations also apply to more recent data in the European market. Furthermore, this study explores the impact of incorporating a vectorised form of time as an additional source of information on the performance of LSTM models in financial prediction tasks. The potential benefits of incorporating temporal information have been shown in other fields, but the impact on the financial market is still largely unexplored. Finally, the impact of different regularisation mechanisms on accuracy and computation time will be assessed. By addressing these questions, this thesis aims to contribute to a better understanding of the applicability of LSTM models and their potential to improve financial prediction accuracy.

The thesis methodology is based on a quantitative approach and involves using the STOXX Europe 600 index stock data from 2002 until 2022 as the primary database [30]. The time series data of each index member underwent thorough cleansing and preparation, after which two smaller LSTM networks were implemented and tested. Apart from a slightly modified LSTM network based on Fischer and Krauss' framework [14], an additional layer implementing the T2V concept proposed by Kazemi et al. was also incorporated [27]. Following the implementation and experimental phase, the experiment's results were extensively analysed and evaluated, focusing on DL performance and finance-related analyses. Additional experiments were conducted to better assess the influence of the deviations from Fischer and Krauss's original model settings [14]. Finally, the results were compared with two benchmark models to get a better perspective on the effectiveness of the DL models compared to two straightforward decision heuristics. The benchmark models are built upon two different approaches, on the one hand, the BH approach and,

on the other hand, a decision heuristic based on the STR effect.

The present thesis provides a number of important contributions to the field of DL and finance that are listed below:

1. **First investigation of DL models on STOXX Europe 600 index data:** To the present knowledge, STOXX Europe 600-related data have only been used once for the assessment of DL models, whereby this dataset was comprised of ad hoc announcements on STOXX Europe 600 stocks and not price data [13, 25].

2. **Performance evaluation of LSTM and T2V extended LSTM networks:** In the study, two slightly modified LSTM (LSTM with and without T2V layer) network variants based on the Fischer and Krauss [14] approach are implemented and tested on recent and European data. The model performances show a similar evolution on the European data as the Fischer and Krauss framework on US-American data, with the developments in the European region occurring delayed by a few years. The DL networks initially performed very well, significantly outperforming the simpler benchmark heuristics (BH and STR), whereby the LSTM model without T2V extension achieved the best results. Nevertheless, the performance steadily deteriorated over the years and could not show a significant advantage over the benchmark models from 2017 onwards.

3. **Assessment of the performance impact of the T2V vector representation of time:** The integration of the T2V approach as an additional trainable embedding layer was tested in the course of the experiment. The aim of the additional layer was to extract the time components from the price data as a synthetic feature. However, incorporating the vectorised form of time as an additional source of information did not lead to a sustained improvement in the model's performance.

4. **Analysis of performance differences between the modified and the LSTM network by Fischer and Krauss:** In addition to the fundamental performance analysis, the study investigates the impact of minor adjustments to the dropout regularisation mechanism on loss, accuracy, and computation time. The results suggest that Fischer and Krauss's approach [14], which uses a recurrent dropout within the LSTM layer, achieves slightly better results. However, the modified version with an extra dropout layer, rather than the recurrent dropout feature, significantly reduces the training time.

The thesis is structured into six chapters. Chapter 1 serves as an introduction and motivates the research topic. Chapter 2 introduces the leading DL

concepts and explains the idea behind recurrent neural networks. Chapter 3 provides an overview of current research and takes a closer look at the papers that have significantly inspired the work of this Bachelor thesis. In Chapter 4, a detailed description of the entire experimental setup and implementation is provided. The results of the experiments are presented in Chapter 5 and evaluated from a DL perspective and then from a financial perspective. Finally, Chapter 6 summarises the main results and offers an outlook for future research.

Chapter 2

Theory

To provide a solid foundation of knowledge for the subsequent chapters, this section introduces the fundamental concepts of DL. DL forms a subfield of ML and thus also belongs to the overarching collective term artificial intelligence (AI). Following Goodfellow et al. and Chollet, Figure 2.1 illustrates the hierarchical relationship between AI, ML, and DL and names an exemplary representative for each [7, 15]. AI represents the broader field of computer science focused on creating intelligent machines, ML is a subset of AI that involves training algorithms to learn from data and make predictions, and DL is a subset of ML that uses neural networks with multiple layers to extract high-level features from data. In ML, an additional distinction is made between unsupervised and supervised learning. The former includes methods such as k-means for clustering or principal component analysis (PCA) for dimension reduction. By contrast, the DL approaches examined below are to be located in the area of supervised learning.

This approach uses artificial neural networks with multiple layers (hence the 'deep' in deep learning) to model and solve complex problems. Compared to classical rule-based systems, DL algorithms are inspired by the structure and function of the human brain and follow the idea of automatically identifying complex relationships between inputs and outputs from large amounts of data and deriving operating rules from them. Figure 2.2 tries to capture this conceptual difference between classical programming and ML [7]. Self-learning DL algorithms have achieved top performance in many applications, such as image classification, speech recognition, natural language processing etc. Therefore, for a better understanding of the work, the central core concepts of DL are highlighted below.

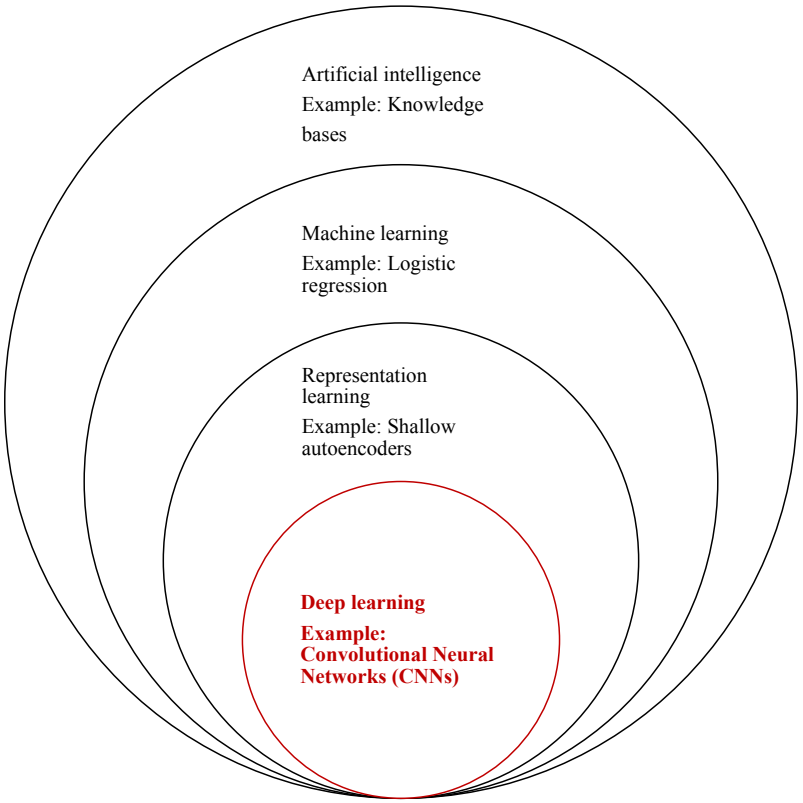

Figure 2.1: Hierarchical relationship between AI, ML and DL [7, 15]

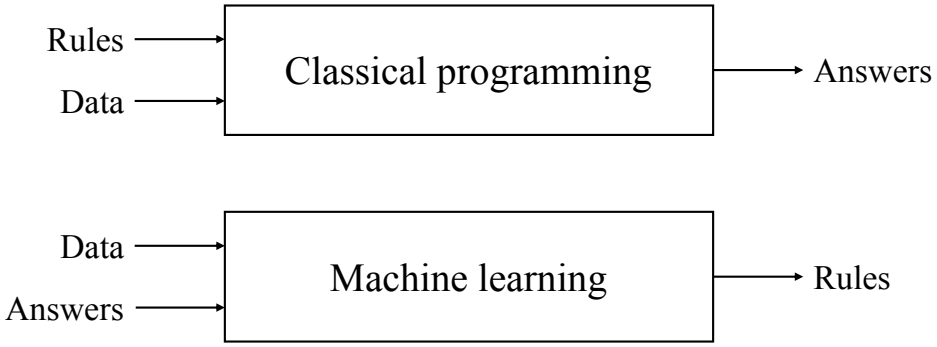

Figure 2.2: Conceptual difference between classical programming and ML [7]

2.1 Deep Learning

The following elaboration of essential DL concepts is mainly based on the information provided by F. Chollet and I. Goodfellow et al. [7, 15].

DL models are often characterised by their depth, which refers to the number of layers in the model. This depth is crucial to the model's ability to learn valuable representations from data as each layer processes and transforms the input data received from the previous layer. In contrast to shallow learning approaches that only aim to extract certain representations, modern DL models can now consist of hundreds of layers.

DL models are called neural networks due to their layered structure, similar to neurons in a brain. The individual nodes in a layer correspond to the neurons, and their connections can vary depending on the model architecture. The input data is usually fed into the model in the form of higher dimensional tensors via the first layer, called the visible layer, and is passed on to the hidden layers, of which there can be hundreds. These hidden layers form the core of any DL model and extract meaningful representations or features from the data using learned parameters (matrices of weights and biases). The output of a layer forms the input of the subsequent layer, allowing for the extraction of increasingly complex and hierarchical data characteristics.

The last layer, the output layer, computes a prediction, which can be a probability, a concrete value, or even a sequence of values, depending on the problem. In summary, the depth and hierarchical structure of DL models allow for extracting complex and valuable representations from data, leading to its high predictive capabilities in various domains, including the financial markets. Following Chollet and Goodfellow et al., Figure 2.3 illustrates a DL model's previously described schematic structure [7, 15].

In DL, the term 'learning' primarily refers to the training process of the models, where they learn to extract meaningful representations from the data. The following explanation of the learning process is conceptually illustrated in Figure 2.4, thus summarising the key relationships. This process can be divided into several iterations, each consisting of a forward and backward pass. During a forward pass, a predefined portion of the input data X is fed into the model through the visible layer, processed by the model, and produces a prediction Y' at the output layer. The deviation between the prediction Y' and the true target Y, the actual value, is then calculated using a loss function. The lower the loss value, the better the prediction aligns with reality and the better the model is trained.

The backward pass involves updating the weights and biases in the model layers to reduce the loss value. This is accomplished through the process of backpropagation, which is the central algorithm in DL. The optimisation

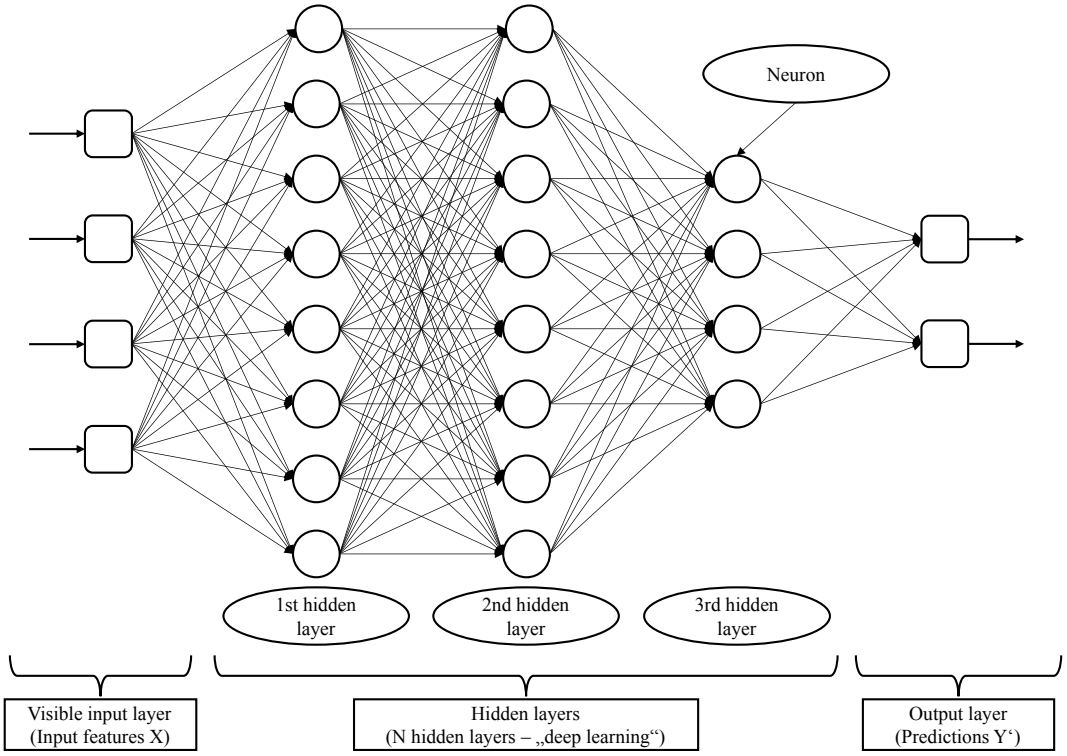

Figure 2.3: Illustration of the schematic structure of a DL neural network [15]

process is implemented using an optimiser, which adjusts the values of the layer weights only slightly in each iteration, leading to a slight decrease in the loss score.

After all the training data has been fed through the model and the weights and biases in the layers have been adjusted accordingly, the model is considered trained and theoretically capable of making accurate predictions with high probability. The training process involves multiple iterations of the forward and backward passes until a satisfactory loss value is achieved. The training process optimally results in a model that has learned to extract relevant features from the data, enabling it to make informed predictions. However, it should be noted in advance that there is always a risk that the trained model cannot make reasonable predictions despite intensive training. This can have several reasons, which will be explained later.

The backpropagation process, the central algorithm in DL, plays a crucial role in training DL models. This process uses gradient descent, a powerful and modern optimisation method. The development of this method has significantly contributed to the success of DL models and is considered a cornerstone of the field.

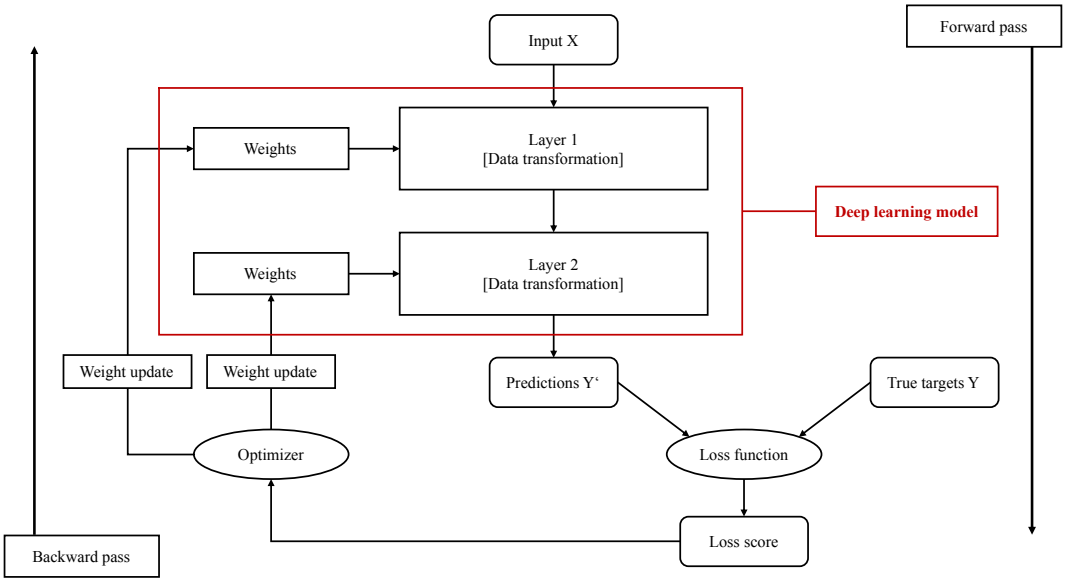

Figure 2.4: Conceptual incorporation of the neural network into the training and prediction process, considering the loss function, optimiser, and flow direction of the data [7]

Gradient Descent can be compared to a mountain climber trying to find the global low point in a valley. The idea is that one starts randomly at a point on the curve and approaches the global minimum with small steps. The objective is to find the curve's lowest point, representing the optimal solution. Figure 2.5 provides a clear visual representation of the concept.

However, the process of gradient descent has its challenges. For example, if the learning rate is too low, the optimisation process can get stuck in a local minimum and not converge to the global minimum. On the other hand, if the learning rate is too large, the steps taken might be too big, making it often impossible to find the global minimum.

The training process is typically performed several times to reduce the impact of randomly bad starting points. This helps ensure the model converges to the global minimum with higher probability and can make accurate predictions.

Training a ML model is an iterative process that aims to balance optimisation and generalisation. The problem with ML is that the models are trained on a limited set of data, which can lead to overfitting. In this scenario, the model becomes too specialised in recognising patterns in the training data that may be absent in new, unseen data.

Generalisation, on the other hand, refers to the ability of a model to make accurate predictions based on new data. To achieve good generalisation, the

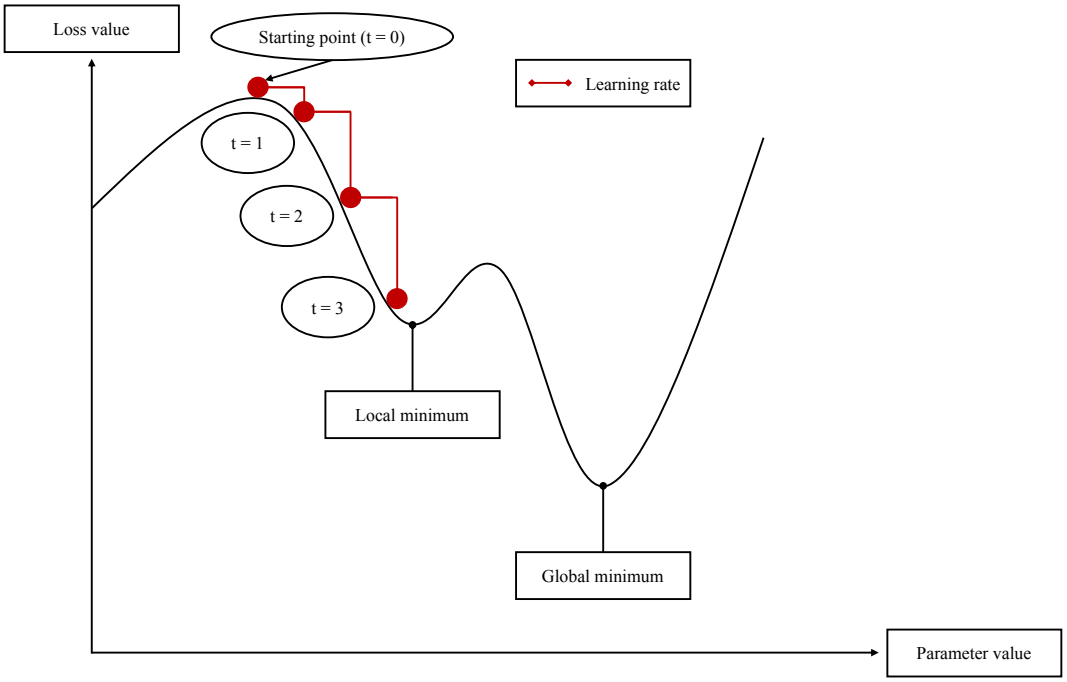

Figure 2.5: Schematic 2D representation of the gradient descent process [7]

model must be optimised during the training process and avoid overfitting.

Figure 2.6 illustrates the relationship between underfitting and overfitting and optimisation and generalisation. During the early stages of training, the loss of the training data and the validation data (data unknown to the model and not yet seen) typically decreases at the same rate. This is called underfitting, where the model is optimised and learning meaningful generalisations simultaneously.

However, as the training process continues, at a certain point, the validation loss starts to increase while the training loss continues to decrease. This is known as overfitting, where the model is still being optimised, but the generalisation properties of the model are reducing. Overfitting can negatively affect the model's predictions and must be avoided. To identify the optimal trade-off between underfitting and overfitting, it is common practice to deliberately overfit the model initially, thus obtaining a clear indication of the point at which the balance between underfitting and overfitting is achieved. Afterwards, the training process starts again and interrupts at the right moment to achieve a robust fit. Overfitting can be attributed to various data quality issues, such as noise or certain rare features which can promote overfitting.

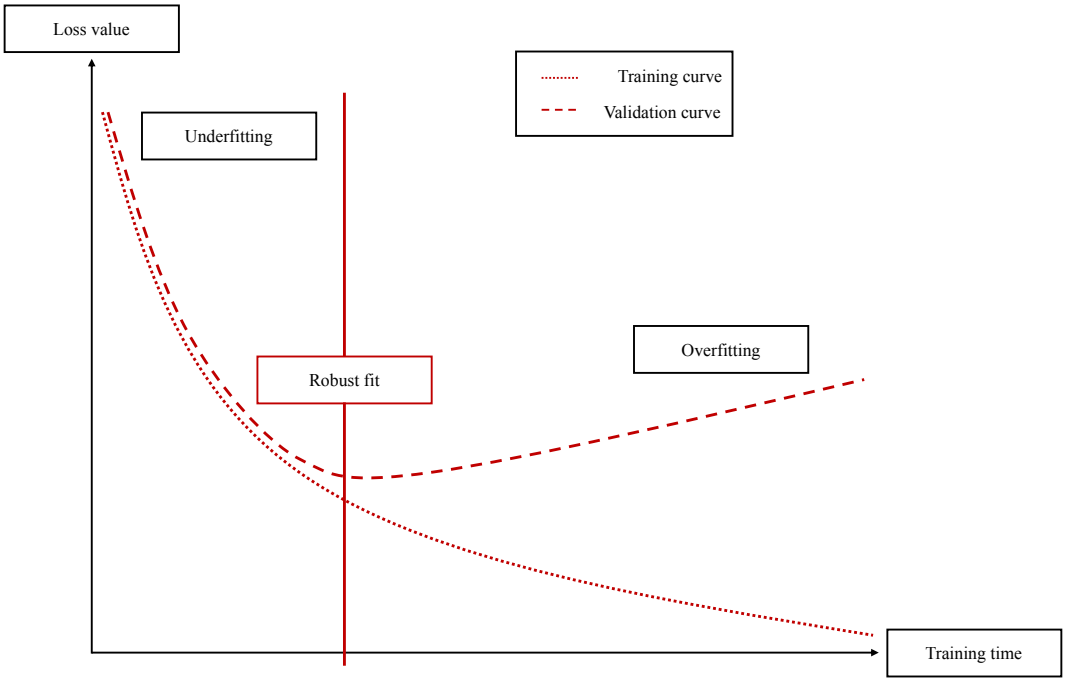

Figure 2.6: Underfitting vs Overfitting - The continuous conflict between optimisation and generalisation [7]

2.2 Recurrent Neural Networks

Thus far, the DL models presented in this text have been of the feed-forward variety, such as Convolutional or Densely connected networks, which lack the capability to retain and carry forward prior results for subsequent computations. Consequently, these models are not well-suited to effectively process datasets that contain sequential or temporal dependencies, such as those present in financial time series or languages and texts. On the other hand, RNNs provide a solution to this challenge through their ability to cache previous computations and integrate them into the current calculation. This enables the model to consider short to medium-term dependencies, allowing it to process entire sentences, for example, by analysing each word within the context of the whole sentence, much like the human brain processes information. A simple illustration of this mechanism can be seen in Figure 2.7, which showcases the effectiveness of this intermediate memory in facilitating the learning of temporal dependencies. Furthermore, Figure 2.8 depicts the structure of a basic RNN layer over time, highlighting how the output of each computation is made available as an information source for the subsequent calculation.

While capable of accommodating short-term dependencies in sequences, the

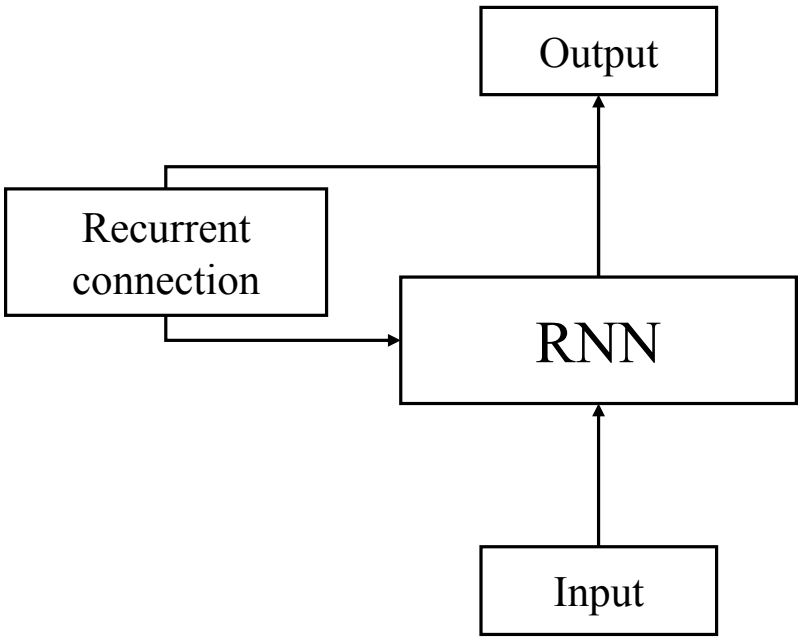

Figure 2.7: Basic concept of an RNN: intermediate memory via loop [7]

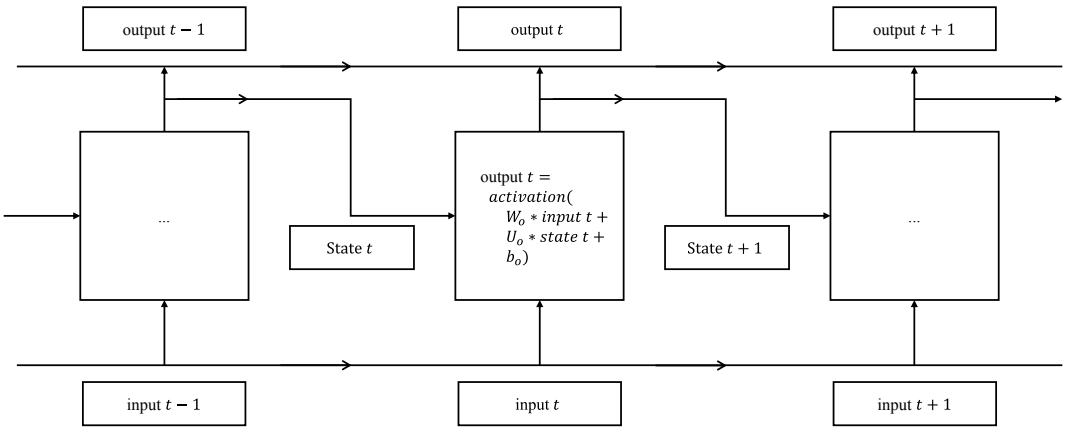

Figure 2.8: Simple RNN architecture rolled out over time [7]

traditional recurrent neural network (RNN) architecture is often insufficient in learning longer-term dependencies. This is due to the computational gradient tending to either vanish or explode in particularly deep networks, which makes learning these dependencies a formidable challenge. Several researchers independently identified the vanishing and exploding gradient problem [4, 21], highlighting the difficulties that arise when trying to capture long-term dependencies in RNNs. However, a solution to this problem has emerged in the form of gated RNNs, particularly the LSTM layers, which have proven to be highly effective in many applications such as handwriting recognition [17], speech recognition [16, 18], image captioning [24, 26, 36], and parsing [35].

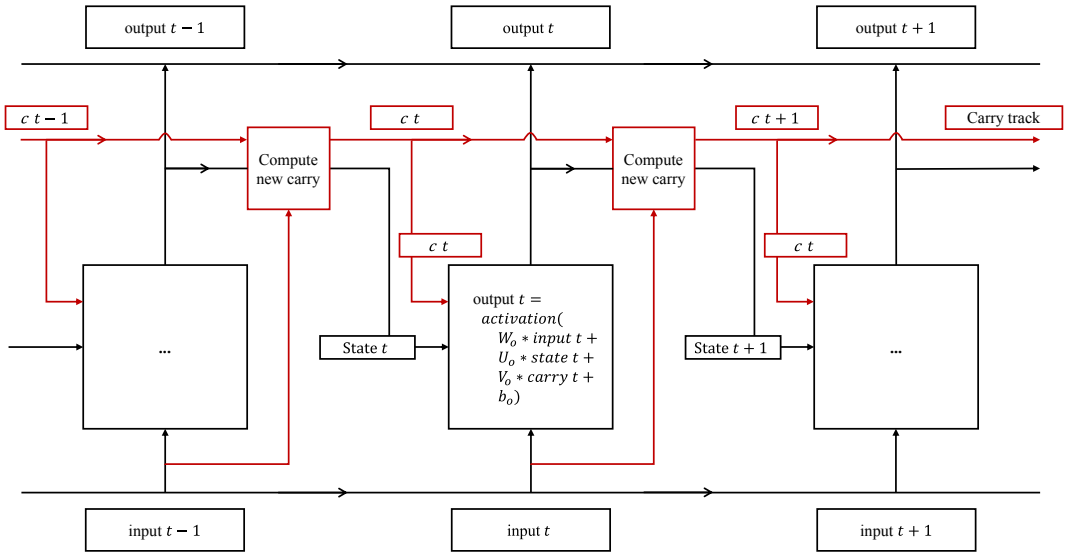

Figure 2.9: Advanced LSTM architecture rolled out over time [7]

Hochreiter and Schmidbauer developed the basic algorithm of the LSTM in 1997 and had previously delved into the vanishing gradient problem [21]. The LSTM layer differs significantly from the superficial RNN layer, as it provides a mechanism for storing information for extended periods of time, making it available for subsequent computations. The information can be thought of as being carried along a conveyor belt, releasing it as needed, and taking up new information again. The extension of the simple RNN architecture to an LSTM architecture is depicted in Figure 2.9 (highlighted in red), where it can be seen that the previously computed output is used as input for the subsequent computation. Still, long-term information is stored and carried away for longer periods, as illustrated by the carry mechanism in the upper part of the figure. The LSTM is highly successful in many applications and continues to play a central role in developing DL models.

Chapter 3

Related Work

The body of literature on machine learning (ML) and particularly DL for finance applications, specifically for financial time series forecasting, has grown dramatically in recent years. This is evidenced by the extensive literature reviews of Ozbayoglu et al., Sezer et al. and Kumbure et al. [28, 31, 25]. First, a rough overview of the current research directions is given to put this work in a better context. Then, the main content of the papers on which this thesis is based is outlined, before explaining the thesis' methodology.

3.1 Overview

In addition to the different DL architectures, the input data for training, testing and evaluating the models play a central role. The authors rely on a variety of different features and time intervals.

Besides the classical price and volume data of individual stocks and stock indices, variables such as fundamental data (e.g., year-end reports of companies, interest rates or unemployment rates), macro data, technical indicators, commodity prices and exchange rates, such as well as other market data such as volatility values are becoming increasingly important [25, 31]. On top of the numerical data, some authors rely on text-based information sources, such as Twitter feeds, ad hoc news, or press releases [25]. While all these data types carry individual information, some of which overlap, there has yet to materialise a clear consensus on which data are crucial for market forecasting.

In terms of time intervals, daily data is the preferred choice, presumably because of its greater availability and relatively low acquisition cost. Some of the daily data can even be obtained for free via Yahoo Finance [3]. The more granular the data is, the more expensive the acquisition and the more complex the data preparation can be. In turn, the length of input sequences ranges from a few minutes to a whole year. Usually, a few days of lag are used within

a series.

Most of the research relates to US stock indices such as the S&P 500, NASDAQ 100 or the Dow Jones Industrial Average (DJIA). Individual stocks and indices from China and Taiwan are in second and third place. European stocks are rarely used for such studies. According to Kumbure et al., the use of the STOXX Europe 600 index is mentioned only once in the literature [25, 31]. Overall, Kumbure et al. provide a detailed overview of the data used and their characteristics.

Before the widespread implementation of ML approaches for predicting stock market movements, financial market analyses were primarily based on fundamental data and technical indicators. Over time, these methods were supplemented by statistical time series analyses. Only recently have ML resp. DL architectures gained increasing attention. On the one hand, this is because technical innovations in software, hardware and data processing have made the widespread use of these systems possible in the first place. On the other hand, ML or DL networks can process large amounts of data very effectively compared to classical analysis methods and recognise complex and non-linear correlations despite noisy data characteristics. This sustainably leads to more efficient predictions [25].

All three literature reviews unanimously state that DL models are not only better than classical approaches but also superior to other ML approaches [28, 31]. This performance advantage of DL models has led to a research focus primarily on DL implementations [28, 31]. As per Kumbure et al.'s findings, in 2019, the majority of papers published on ML techniques in finance primarily concentrated on the applications of DL [25].

Examining the overall research findings, LSTM based models emerge as the dominant model architecture among the various models. The LSTM architecture belongs to the Recurrent neural networks (RNN) and is able to identify temporal relationships in time series. Thus, LSTM models are ideally suited for the analysis of financial time series. A more detailed look at the reviews indicates that there is an emerging trend regarding the development of hybrid models of different architectural components. In many cases, they perform better than native models and are, therefore, the preferred choice. However, there is also the danger of creating more complex hybrid models that are not easy to build, and their interpretation also might be difficult [28, 31].

In addition to application areas such as risk management, portfolio management or fraud detection, most authors aim to make predictions about the price movements of certain assets (e.g. stocks, commodities, cryptocurrencies, etc.). Two main groups can be distinguished. According to Kumbure et al., 55.0% of authors define their problem as a regression and 44.3% as a classification problem [25]. The categorisation as a regression problem refers to

predicting a concrete price value, such as tomorrow's close price of a certain stock. By contrast, a classification problem aims at determining the direction of price movements, i.e. only the future trend of a stock. The latter group can again be divided into a binary classification problem, and a classification problem with three categories (Up, Neutral, Down) [25, 28, 31]. In this thesis, the assignment was defined as a binary classification problem. The following section will examine the papers that form the intellectual starting point for this Bachelor's thesis.

3.2 Closely Related

Zou and Qu present a study using LSTM networks to predict future stock prices and develop a quantitative trading strategy [39]. The study found that all LSTM models performed better than a non-DL benchmark system, whereby the LSTM network with an attention mechanism performed best regarding prediction accuracy.

Fischer and Krauss' [14] 2018 research paper, which is also covered by the literature review by Kumbure et al. [25], presents a study on the use of LSTM networks. The authors define their problem as a binary classification problem, meaning that they attempt to train their model to predict whether the price of a particular stock will rise or fall the next day. To train, validate, and test the LSTM network, Fischer and Krauss use historical stock market data from the US stock index S&P 500. The data range from December 1989 to September 2015 and consist of open, high, low, and close price data, as well as daily volume values. In terms of data granularity, the experiment relies on daily time intervals. Ultimately, the authors use only daily returns as input values and feed sequences of 240 days into the models. The LSTM network is compared to traditional statistical methods such as ARIMA and GARCH. It turns out that the LSTM network outperforms these conventional methods in terms of predictive accuracy and can also handle the non-linear and dynamic nature of financial data [14].

Fabbri and Moro explored the application of RNN architectures for market movement prediction, particularly LSTM models [12]. Three different LSTM network sizes (small, medium, and large) were compared to a DL feed-forward model that lacks the ability to identify temporal dependencies in the data. The authors used publicly available historical price data of the DJIA stock market index from 2000 to 2017, with the first half used for training and the second half for testing. They found that all LSTM sizes outperformed classical feed-forward approaches in prediction accuracy and trading performance. The authors focused on the index data and did not utilise the data of the individual

index constituents, as previously done by Fischer and Krauss [14].

Despite the substantial increase in publications on this topic over the past years and the sometimes very similar model implementations, it is too risky to make generalised statements based on these single experiments. RNNs provide good performance data across different data sets and mostly outperform classical approaches. However, the very dynamic research situation on DL shows that new insights are continuously being gained and that it is still worthwhile to implement and test different data sets and model architectures.

Chapter 4

Methodology

In the subsequent sections, the methodology employed in this thesis is elaborated. To provide readers with a brief overview of the process, Figure 4.1 presents a graphical representation of the experimental setup.

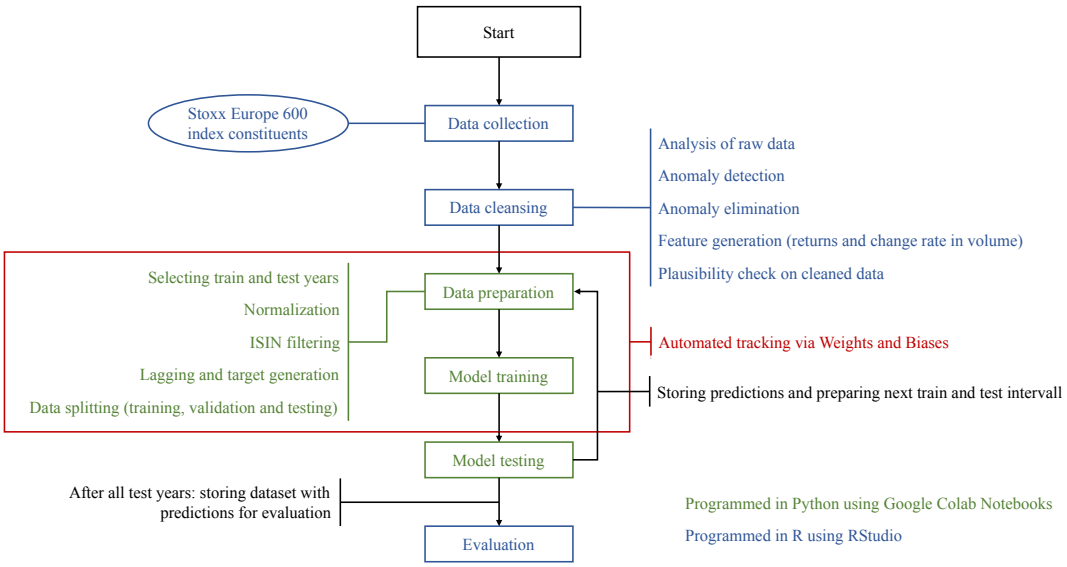

Figure 4.1: Overall workflow of the conducted experiments

4.1 Software and Hardware

During the experiment, different software packages and hardware configurations were used for the various experimental phases. For the analysis and cleaning of the raw data from anomalies, the statistical programming language

R [33] was used, with RStudio as the IDE. Regarding the software packages, Data.table [11] was mainly used since column-based operations are particularly efficient and straightforward to implement. After successfully preparing the data with R, the dataset was adapted using Python for training, validating and testing the model architectures. Python was deliberately chosen for this purpose because much of the research relies on Python. Many different packages make it easy to customise an implementation to meet individual needs [28, 31]. In the course of further data preparation, the packages Pandas, Numpy and Datatable were primarily used [9, 19, 29]. The DL models were implemented, trained and tested using Keras and TensorFlow [6, 10]. According to authors Ozbayoglu et al. and Sezer et al., Keras and TensorFlow are the industry standard among Python-based ML packages, with the ever-growing number of open-source libraries and frameworks making the research and development process more accessible than ever before [28, 31]. All performance data generated during the training and testing of the models was tracked seamlessly using the API of Weights and Biases [5]. Finally, R with Data.table was used again for the evaluation, and all visualisations were implemented using ggplot2 [37].

Regarding hardware, most of the data processing and all of the training and testing of the models were handled through Google Colab Pro, as Google Colab provides access to powerful GPUs with a few setting changes. The premium GPUs Google Colab Pro offers are from international graphics card manufacturer Nvidia. They are optimised to perform DL specific computational operations such as matrix multiplication for large data sets. Tables A.1, A.2 and A.3 in Appendix A summarise all software settings compactly for reproducibility reasons.

4.2 Data

An essential part of the work is the preparation and adaptation of the data so that they can not only be processed by the models but also have a high quality and purity. F. Chollet [7] emphasises several times that it is worth investing more time in data preparation than in training and testing models and optimising hyperparameters. Only data with sufficient representative power allow the model to recognise relationships and learn new meaningful rules [7]. The following section is divided into three parts. First, the essential characteristics of the raw data are described before the second step explains in more detail which anomalies are eliminated with the help of the programming language R. Finally, the processing steps in Python are examined in more detail, whereby the data are modified for feeding into the ML models.

4.2.1 Raw Data

For the empirical tests, the data of the STOXX Europe 600 index constituents are used. The index was chosen because most of the research in the review papers is based on US, Chinese, and Taiwanese indices and stocks [25]. Only Feuerriegel and Gordon use the STOXX Europe 600 as a data basis. However, the authors use ad hoc news and not price data for their forecasts [13]. Thus, according to the current state of knowledge, this paper is the first to investigate the predictive capabilities of DL models regarding the price and volume data of the STOXX Europe 600. The STOXX Europe 600 is a stock index similar to the DAX 40 or S&P 500 and includes around 600 companies. The Swiss index provider STOXX Ltd. published the index, which is now part of Qontigo and thus belongs to the umbrella group Deutsche Börse AG [30]. The raw data are obtained from Bloomberg, ranging from the beginning of January 2000 to the beginning of April 2022. Regarding the granularity of the data, the decision was made to use daily data since, on the one hand, this reduces the noise of intraday trading. On the other hand, higher availability of high-quality data can be expected. For each day and each share, the data set contains the critical price points of the day (open, high, low and close) and the daily volume in the number of traded shares. Prices, in turn, are reported in national currencies so that no exchange rate effects affect the price data over time. The data obtained from Bloomberg have already been adjusted for corporate action, i.e. stock splits, for example, have been considered. In the following, the key points of the raw data and a small exemplary section of the data set are shown in Table 4.1 and Table 4.2.

Table 4.1: Raw data characteristics

Characteristics	Data
Time period	2000 - 2022
Index sectors*	19
Total different stocks	1,377
Total data points (rows)	7,940,913
NAs in open series	1,125,925
NAs in high series	1,125,926
NAs in low series	1,125,927
NAs in close series	1,124,226
NAs in volume series	1,128,098

Note: *See all sector names in Table B.1

Table 4.2: Data sample of the raw dataset

	ISIN	Date	Open	High	Low	Close	Volume
0	GB00B0NH0079	2000-01-03	277.1470	282.7150	272.1930	282.4180	212716.0
1	GB00BVYVFW23	2000-01-03	NaN	NaN	NaN	NaN	NaN
2	DE000A1YCMM2	2000-01-03	97.2592	100.2542	94.2592	98.2182	4.0
3	DE000A2AADD2	2000-01-03	NaN	NaN	NaN	NaN	NaN
4	GB0001528156	2000-01-03	110.7290	112.1950	109.4810	111.0660	5892.0
5	IE00B00MZ448	2000-01-03	72.1929	72.1929	72.1929	72.1929	617.0
6	DK0010287234	2000-01-03	49.1850	52.4490	47.4670	51.1050	140348.0
7	FI0009005987	2000-01-03	7.5033	7.9537	7.4796	7.6692	563912.0
8	CH0012138605	2000-01-03	71.3200	72.3800	69.1200	71.6312	94220.0
9	NO0010081235	2000-01-03	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...

4.2.2 Data Cleaning with R

To obtain a replication of the STOXX Europe 600 index that is as close to reality as possible and of high quality at all times, the raw data received from Bloomberg must undergo several processing steps [2]. In the course of this, anomalies of various kinds are identified and eliminated. In the following, the essential stages of the data preparation are examined in more detail, and the core characteristics of the resulting data set are presented.

First, a quarterly check is made over the entire period to determine which stocks were members of the index at this point in the past. All data points of a stock that lie in the period of non-membership are filtered out. This yields a data set that reflects the exact composition of the index at any point in time. This is also referred to as point-in-time (PIT) data. This adjustment serves the realistic replication of the index, eliminates the so-called survivorship bias, and ensures a realistic implementation of the subsequent experiments.

During the data cleansing, it also became apparent that a large proportion of the NAs are in the non-membership periods of the individual stocks, so in the next step, only a few thousand data points need to be deleted. In the subsequent step, the remaining NAs are eliminated and not replaced. Padding the data using interpolation makes little sense, as relatively few data points are lost.

After closer examination of the data, it becomes clear that some data series are frozen over a certain period of time, i.e. the values are carried forward unchanged for a certain period. These frozen sequences usually appear at the end of an asset-specific time series. However, they can also be found at the beginning and middle. The causes can be of various kinds. For example, insolvencies, takeovers, trading stops, public holidays, ISIN changes or simply low liquidity are possible causes. Since the data are quoted in the respective national currencies, as mentioned in the previous section 4.2.1, possible exchange rate fluctuations do not affect the data. The frozen time series can be identified without any problems.

The frozen sequences at the beginning and end of the respective time series are deleted without further ado. Only the last or first appearance of the carried forward price and volume values are retained. As for the sequences that lie in the middle of a time series, those that are longer than four days were identified. The threshold of four days was deliberately chosen because it covers most weekends, bank holidays, and national vacations. The resulting list of stocks with frozen data values in the middle is manageable with 16 different ISINs.

Reviewing the 16 shares was conducted manually, focusing on anomalies primarily related to takeovers, liquidations, changes of ISIN, or low trading

volumes. Through careful evaluation, it was possible to distinguish which data points were erroneous and which were simply unusual, such as a day with no trading. The incorrect periods were then identified, and the corresponding time frames were removed from the dataset. The Greek sovereign debt crisis resulted in a high frequency of anomalies for Greek stocks, leading to the decision to exclude the period from May 2010 to April 2015 for all Greek stocks. This decision was made to minimise investment risks, considering the possibility of trading restrictions in real-time during this crisis period.

Before the next step, the daily log returns R_t^s and the log growth rate of the volume values dV_t^s are calculated for each given stock $s \in S_t$ at a time t whereby S_t stands for all stocks resp. index members at a certain Point-in-time. The calculations of the additional features are based on the following Formulas 4.1 and 4.2, where C_t^s corresponds to the close price and V_t^s corresponds to the volume.

$$R_t^s = log\left(\frac{C_t^s}{C_{t\ 1}^s}\right) \tag{4.1}$$

$$dV_t^s = log\left(\frac{V_t^s}{V_{t\ 1}^s}\right) \tag{4.2}$$

This gives us a preliminary data set with seven features - price data (open, high, low and close), volume values, daily log returns and logarithmic change in volume.

The presence of stocks leaving and re-entering the stock index can result in abnormal price jumps, causing extreme values when calculating daily log returns. To preserve the integrity of the data set and prevent these erroneous values from affecting the training of the model later on, all values for the respective stock are set to NA upon re-entry. This adjustment ensures the data set's accuracy and simplifies the later stages of data preparation in Python.

Before transferring the data to Python, a final plausibility check was conducted, which did not raise any other red flags. Overall, it can also be noted that the period under consideration has been shortened from January 2002 to April 2022 due to the anomaly removals. This provides nearly 20.5 years of data for training and testing the models. The essential information regarding the calculation steps is summaised below in below in tabular form - Tables 4.3. For more information on the core properties of the dataset, a tabular summary is provided in Appendix B.

Table 4.3: Data cleansing steps with R

Calculation steps	Data points	Delta
Before PIT filter (raw data)	7,964,139	—
After PIT filter	3,149,990	4,814,149
Removing NA's	3,139,731	10,259
Removing back-freezes	3,131,310	8,421
Removing front-freezes	3,131,202	108
Manual adjustments - non Greeks	3,127,300	3,902
Manual adjustments - Greeks	3,121,187	6,113

4.2.3 Data Preparation with Python

The subsequent section outlines the necessary steps to prepare the previously cleaned data for model training and testing using Python. According to Fischer and Krauss [14], three consecutive years are allocated for training the model, with the following year utilised for testing or trading purposes. This results in a total of four years per training and testing iteration. With almost 18 complete test periods starting from the first test year, 2005, each iteration utilises an untrained model. The 18 test periods are independent of each other, allowing for a comparison of models based on the selected test year.

After careful consideration, the decision was made to use only the logarithmic daily returns as input features for the final experiment, despite the preparation and calculation of other features. On the one hand, the experimental results thus remain more comparable with the results of Fischer and Krauss. On the other hand, the reduced amount of data allows faster training and test iterations as well as a more flexible allocation of the limited computing resources.

As seen in Equation 4.3, the mean and standard deviation of the train interval is used for the normalisation of each feature F_t^s. The formula is also known as Z-score.

$$\tilde{F}_t^s = \frac{F_t^s \quad \mu_{train}}{\sigma_{train}} \tag{4.3}$$

The normalisation process is essential for preparing the data for training the models. To maintain the integrity of the test data, only the training data is used to calculate the mean and standard deviation. This ensures that no information from the test interval influences the training data, avoiding a look-ahead bias and producing accurate results. Normalisation scales the data so that all values are of similar magnitude, allowing the model to better identify

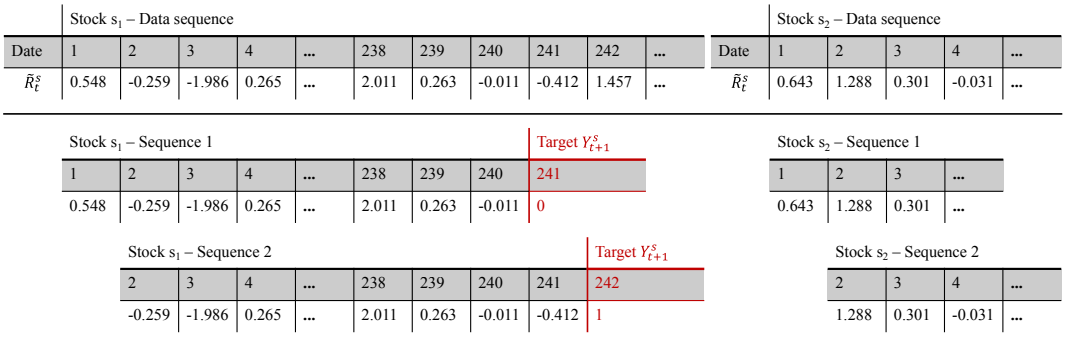

Figure 4.2: Schematic illustration of the sequence construction process for DL networks based on the feature vectors

relationships without being influenced by significant differences in magnitude [7]. Although normalisation can result in the loss of information about price volatility, it is usually still deemed worthwhile. Alternatively, one could incorporate the information on volatility as an additional feature in the data set.

The stocks considered for the test period are restricted to only those members of the index, which are part of the index on the final day of the training period. This approach ensures that the model has been exposed to sufficient data in the training phase for most of the actively traded stocks, increasing the probability of accurate results. Furthermore, this approach simulates the real-world scenario of a fund manager or investor who invests in a specific set of stocks for a given period. This method simplifies the process and eliminates the need for complex and time-consuming adjustments during the trading phase, reducing the risk of errors.

After normalisation and selecting the stocks for testing, the data is ready to be transformed into sequences, which is essential for effectively training the model. The data is shifted several times and added to the dataset to create sequences of 240 days in length, nearly covering a trading year of 252 days. This enables the model to identify long-term dependencies. The advantage of having set values to NA when stocks re-enter the index in the data preparation process in R is that these NAs are carried through the time-delayed sequences, making it easy to remove the input sequences that are not complete. This ensures that the model is only fed with sufficient data from the affected stock once it is available after re-entry.

Finally, the calculation of the ground truth respectively the targets is carried out for each data series. For this purpose, the median log return is the dividing line between outperformers and underperformers. Stocks whose log daily return is above the median are considered outperformers, while stocks

below are considered underperformers. No further feature engineering is done, leaving seven different features available for the subsequent experiments. Figure 4.2 schematically displays the sequence generation process.

The data preparation process is complete, and the dataset is divided into training, validation and testing packages. The training and testing intervals have already been defined using concrete annual units, with the training data further divided into 80% for training and 20% for validation.

4.3 Benchmark

The performance of the DL models is evaluated by comparing them against two benchmark models, namely a BH approach and a STR strategy. The BH system is a passive investment strategy where the investor buys and holds onto a stock for an extended period regardless of market fluctuations. This strategy is often used as a benchmark to evaluate the performance of other investment strategies, as it represents a simple and low-cost approach to investing.

The STR strategy is based on the well-known phenomenon in financial markets where stocks that have performed poorly over the short term tend to rebound in the near future, while stocks that have performed well tend to underperform. This effect has been extensively studied in the academic literature and has been shown to persist across different markets and time periods [8]. In this study, the reversal strategy involves examining the stock performance over the previous five days, respectively, one trading week, to determine whether to sell or buy the stock. If the stock has performed worse, it is bought; if it has performed better, it is sold. The investments' performance is assessed daily in accordance with the predefined strategy and rebalanced as needed.

For several reasons, comparing the DL models against these two benchmark models is reasonable. Firstly, the BH approach provides a baseline for comparison that is easy to understand and implement. Secondly, the STR strategy represents a well-established and widely studied investment strategy, which allows us to evaluate the performance of the DL models against a well-known benchmark. Furthermore, comparing the DL models against these two benchmark models provides insight into how well they perform compared to a passive and an active investment strategy, allowing us to better understand their strengths and weaknesses.

4.4 Deep Learning Architecture

After successfully processing the data and establishing relevant benchmark models for later comparison, it is imperative to advance to implementing the DL models. In this endeavour, a deliberate choice was made to adopt the basic model proposed by Fischer and Krauss in their seminal work [14]. This basic model, a relatively straightforward LSTM network featuring a single LSTM layer, serves as a starting point and provides a solid foundation for further exploration and extension. To provide the basic LSTM model with additional information about the temporal dependencies and thus potentially improve the system's performance, an additional layer has been introduced whose architecture is inspired by the research on vectorisation of time by Kazemi et al [27]. The following section will delineate the precise specifications of both implementations, emphasising the distinctive characteristic of the T2V layer.

4.4.1 Long Short-term Memory

The first of the two tested DL models is established based on Fischer and Krauss's work [14]. By doing so, the experiments undertaken connect to current research and build upon it. Fischer and Krauss utilise a simple model with few hidden layers and a limited number of trainable parameters. This approach has both benefits and drawbacks. Larger models can identify more complex relationships and infer patterns; however, a simpler architecture is more suitable for binary problem classification. This is due to larger models with more parameters having a greater likelihood of identifying unwanted patterns and relationships. In contrast, the simple version is less prone to learning undesirable patterns and is thus more effective for more straightforward problems. Additionally, a simplified problem places fewer demands on the system than computing a precise value. Smaller architectures are also easier to train and necessitate less computing power, as fewer parameters require adjustment after each iteration. Therefore, it is necessary to determine the best-suited architecture for the task-based explicitly on the problem being addressed. For this task, a simple modelling approach is more practical.

For the most part, the architecture of Fischer and Krauss is adopted in the following experiments. Only two adjustments were made [14]. Firstly, in the LSTM layer, the recurrent dropout was set to zero to enable the use of faster GPU compute units during training, as required by Keras. However, to better assess the consequences of the alternative architecture implementation in terms of differences in computational performance and model accuracy, training and testing were performed with the original recurrent dropout setting (recurrent dropout = 0.1) for selected test years. The results of the various implementa-

tions are presented in Chapter 5.1. Besides, the activation function of the last layer, a dense layer, uses the sigmoid function instead of the softmax function utilised in Fischer and Krauss's implementation. The major model settings are presented in Appendix F. Furthermore, the original implementation and the modified version are depicted graphically in Appendix C and D. The LSTM layer situated in the centre of the model represents the central element of the pure LSTM network.

4.4.2 Long Short-term Memory with Time2Vec Extension

An additional embedding layer extending the previous LSTM model is tested during the experiments. The functionality of the embedding layer is built on the approach of Kazemi et al. to the vector representation of time. [27]. The purpose of this layer is to generate a synthetic feature based on the input data that encodes the temporal structure of the data. The trainable layer is designed to provide the model with additional vectorised information about the temporal dependencies in the data beyond what can be captured by the LSTM layer. The core elements of the T2V approach are explained below, and the system architecture is presented concisely.

The challenge of encoding time in ML models, especially in the context of time series data, has been addressed by Kazemi et al. [27]. Traditional methods of representing time as a numerical value, such as a timestamp or datetime object, have limitations in their ability to capture the complex patterns and relationships in time series data. In contrast, T2V is a flexible and adaptive approach that learns a vector representation of time without needing a static definition. T2V can be implemented as an additional layer in a DL model. It can capture periodic and non-periodic patterns in time series data, such as daily, weekly, or yearly cycles. The resulting vector representation of time can be used as input to subsequent DL layers or models to improve their ability to capture the temporal relationships and patterns within the data.

The T2V approach of Kazemi et al. is based on the idea that time can be represented as a non-periodic and a periodic function [27]. Their paper presents the following Formula 4.4.

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ F(\omega_i \tau + \varphi_i), & \text{if } 1 \quad i \quad k. \end{cases} \tag{4.4}$$

In this equation, τ represents the input time series, while $\mathbf{t2v}(\tau)$ denotes the resulting T2V representation, with a vector length of $k+1$. $\mathbf{t2v}(\tau)[i]$ refers to the ith element of the resulting vector. The periodic activation function F

is chosen individually, while ω_i and φ_i represent the trainable parameters.

Upon closer examination, the two partial terms of the T2V function resemble the general mathematical formula for a linear function of the form $y = m \quad x + t$. Precisely, ω_i corresponds to the slope of the linear function, and φ_i represents the intersection of the straight line or the time series with the y-axis. Figure 4.3 graphically illustrates this relationship, suggesting that the non-periodic component of time is captured in this manner.

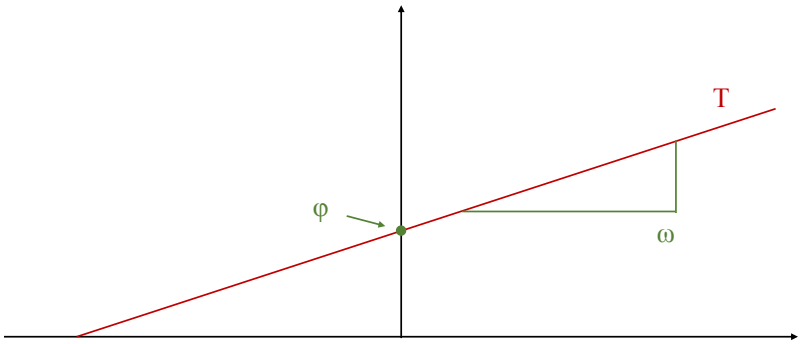

Figure 4.3: Non-periodic or linear time component

The periodic properties of the time series are captured by the second term of $\mathbf{t2v}(\tau)$, with the inner term corresponding to the aforementioned linear function and F being a periodic activation function. The authors have experimented with various periodic activation functions, such as sigmoid, tanh, ReLU, mod, and triangle, but ultimately concluded that the sine function provided the best and most stable results. Henceforth, the sine function will be used in the rest of the analysis.

In the context of a simplified two-dimensional representation of the resulting function, ω_i determines the wavelength of the sine curve, while φ_i determines the displacement of the curve along the x-axis. Graph 4.4 graphically depicts the periodic acquisition of the time series. Those interested in further details about the results obtained by Kazemi et al. can refer to their paper [27].

The T2V approach offers several advantages compared to other techniques for processing time series data. Firstly, it is more flexible and can capture more complex patterns within the data. Secondly, it is computationally efficient, requiring fewer parameters than other approaches. Finally, the T2V representation of time provides a more interpretable output as the resulting vector representation can be easily visualised and analysed to gain insights into the underlying temporal patterns within the data.

For the conducted experiments, the LSTM model, according to Fischer and Krauss, is being supplemented by an individually implemented embed-

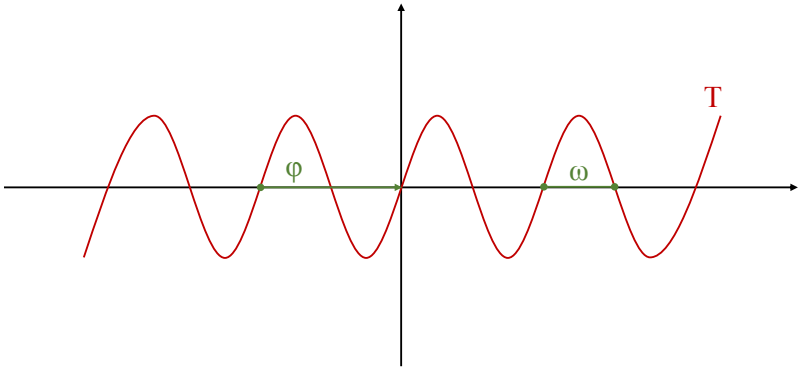

Figure 4.4: Periodic time component

ding layer T2V. The additional layer is based on the previously discussed T2V approach and is inserted directly before the LSTM layer. Since the layer computes an additional feature, the intermediate result is connected to the input data set before the data can pass through the remaining model layers. Therefore, another Concatenate layer is inserted after the T2V layer to implement this operation. The exact model structure is again graphically captured in the appendix E. To better compare the simpler LSTM model and the extended version, the same model settings are used for the extension and the base model - see Appendix F.

4.5 Forecasting, Ranking, Trading and Evaluation

After successfully processing the data, defining the benchmark and DL models in detail, and training them, the final step is to predict the stock movements, rank and trade them, and evaluate the models' performances.

During each iteration over a test year, the models that have been trained previously receive test data that is entirely unknown. Like in training, the models use the past 240 data points as input to predict the future movement for each stock s. Based on the limited input data, the models calculate the probability of each stock s overperforming or underperforming the median at time $t + 1$. A clear separation between each training / testing period, without reusing previously trained models, guarantees comparability between the iterations analysed and enables one to draw conclusions about the impact of real-world events.

After predicting the probability of over- or underperformance of each stock s over the entire available test period, the probabilities are then ranked by

day in descending order of likelihood. The model identifies stocks with high probability values as those that seem to have performed poorly previously or are undervalued and thus are expected to perform better than average at time $t + 1$. Conversely, stocks with very low probabilities are expected to perform worse than average and are thought to be overvalued. The model is more uncertain about a stock's future performance when its probability value is closer to 50%.

The evaluation of the prediction quality involves two main aspects. Firstly, all prediction values are considered to assess the models' overall accuracy. Secondly, the accuracy values for specific portfolio sizes are analysed. To achieve this, portfolios consisting of $2k$ individual securities are formed, where $k \; 2 \; f10, 50, 100, 150, 200, 0.5 \quad jS_tjg$, with jS_tj denoting the total number of securities at time t. Thus, the top or bottom k prediction probabilities are considered in the formation of the portfolios.

A similar approach is used for the benchmark model STR. The difference, however, lies in the criterion used to determine the order of the securities in the portfolio. Unlike the DL models, where the order is determined by the prediction probabilities, the order of securities in the STR portfolio is based on their returns over the past five trading days. The more extreme the return, the higher the expected correction movement in the opposite direction. This methodology provides a standardised approach for comparison across different models and portfolio sizes.

During the evaluation, the experimental results are analysed from two perspectives. Firstly, the main effects are examined from a technical standpoint, in which execution time and DL specific parameters like loss and model accuracy are evaluated. These DL specific performance metrics are founded upon the implementations of Keras, and their mathematical expressions can be elucidated through the following formulas [6]. The first of the two equations describes the prediction accuracy of the model. The second function is the binary cross-entropy loss function, which measures the dissimilarity between the true and predicted probability distribution of binary classification tasks.

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions} \tag{4.5}$$

$$Loss = \quad \frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 \quad y_i) \log(1 \quad \hat{y}_i) \tag{4.6}$$

In this formula, N represents the total number of samples, y_i represents the true label (either 0 or 1) for the ith sample, and $\hat{y}_i$ represents the predicted probability of the positive class (i.e., the probability that the label is 1) for the ith sample. The formula calculates the binary cross-entropy loss by taking the

negative of the average of the sum of two logarithmic terms. The first term is the logarithm of the predicted probability if the true label is 1 ($y_i = 1$), and the second term is the logarithm of the predicted probability if the true label is 0 ($y_i = 0$).

Once the technical aspects have been scrutinised, the second part of the analysis concentrates on the financial outcomes. The model's performance is assessed primarily in terms of profitability. To better understand the results, they are compared with the benchmark performance and various financial sector valuation ratios, such as the Sharpe ratio [32].

The Sharpe ratio is a financial metric that measures the risk-adjusted return of an investment or a trading strategy. It was introduced by Nobel laureate William F. Sharpe and is calculated by dividing the excess return of an investment or a trading strategy by its standard deviation [32]. The excess return is the difference between the actual return and the risk-free rate, while the standard deviation is a measure of the volatility of the investment or the trading strategy. A higher Sharpe ratio indicates that the investment or the trading strategy has generated higher returns per unit of risk and is, therefore, more attractive to investors who are risk-averse. The following equations summarise the Sharpe ratio mathematically - Equations 4.7 - 4.10.

$$D_t = R_{Pt} \quad R_{Ft} \tag{4.7}$$

$$\bar{D} = \frac{1}{T}\sum_{t=1}^{T} D_t \tag{4.8}$$

$$\sigma_D = \sqrt{\frac{\sum_{t=1}^{T}(D_t \quad \bar{D})^2}{T \quad 1}} \tag{4.9}$$

$$SR_{ex \ \ post} = \frac{\bar{D}}{\sigma_D} \tag{4.10}$$

Where:

- D_t represents the average daily return of the portfolio
- R_{Pt} represents the realized portfolio return
- R_{Ft} represents the risk-free rate of return (i.c. $= 0$)
- $\bar{D}$ represents the average yearly return
- σ_D represents the standard deviation of the portfolio return
- $SR_{ex \ \ post}$ represents the ex-post Sharpe ratio

Chapter 5

Results

After describing the methodology used for the experiments in the previous chapter, the central results are presented in this section. The evaluation of the results is broadly divided into two sections. The first section examines key figures related to DL 5.1, which can be further divided into three main areas. The first area evaluates the data associated with the training and validation for the two DL models - subsection 5.1.1. The second area compares the two DL implementations with the original implementation by Fischer and Krauss in a spot-check fashion for selected years to examine the impact of the implementation changes - subsection 5.1.2. Finally, the third area looks at the testing phase results - subsection 5.1.3.

The second section of the evaluation assesses the results from a financial perspective - section 5.2. Different comparison models are used depending on the section and evaluation variable to ensure the results are discussed meaningfully. In some cases, including all comparison models in the evaluation may not make sense. In the further course, it is made clear in each section which models are being considered.

5.1 Deep Learning Evaluation

5.1.1 Evaluation of Training and Validation Results

This section examines the training and validation results of the two DL models, LSTM and LSTM with T2V, in detail without using any other models for comparison. The Graphics 5.1, 5.2, and 5.3 summarise the main results in a compact way. It is noteworthy that no other models were included for comparison in this section. The comparison models that will be used later do not require a training or validation phase, as they are based on simple decision heuristics.

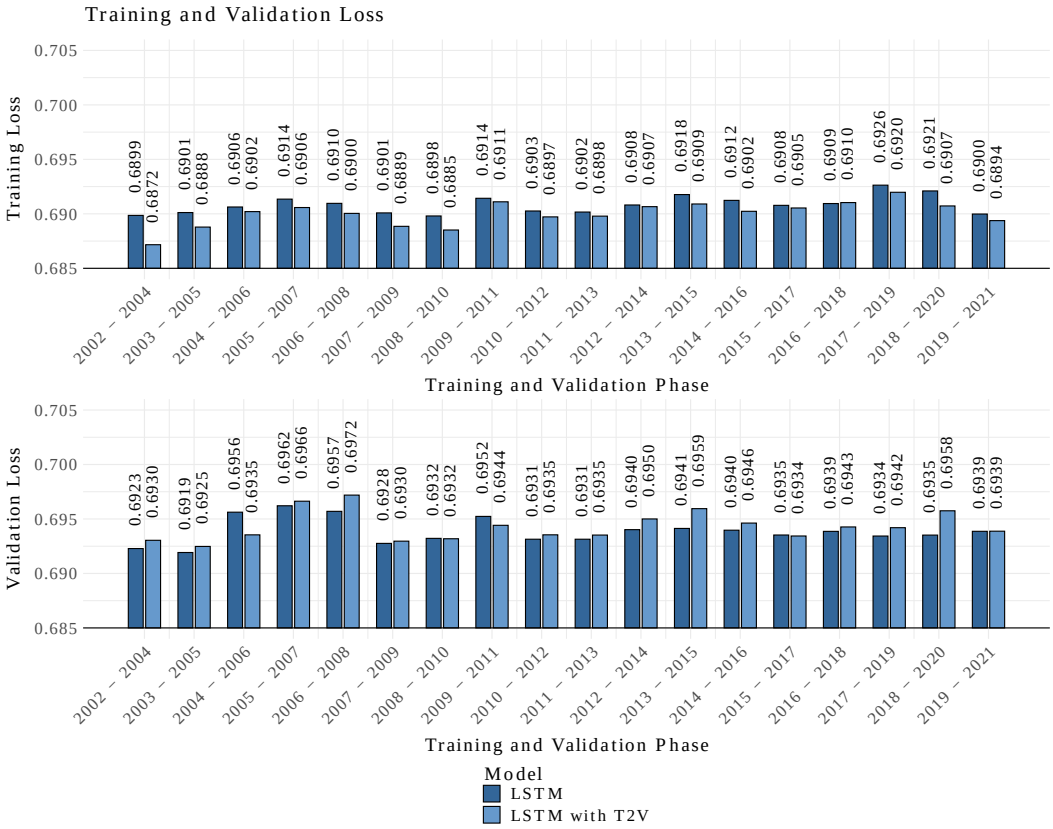

Figure 5.1: Annual loss values for LSTM and LSTM with T2V during training and validation phases, calculated using the Formula 4.6

Figure 5.1 displays the annual loss values of the training and validation. A slight upward trend can be observed in training and validation, indicating that the model's performance worsens over time. The T2V model performs better than the LSTM model in training, while the LSTM network performs better during validation. However, several significant deviations are noticeable in the validation period between the LSTM and T2V-modified model, with the T2V model consistently performing worse. Incorporating time as a synthetic feature and vectorising it in the T2V-modified LSTM network provides more degrees of freedom, resulting in improved training results. However, this additional degree of freedom could potentially lead to poorer performance during validation, as evidenced in the subsequent analysis. Although there are some striking outliers, it is essential to note that all values are in a similar order of magnitude and changes only become apparent from the third decimal place onwards. Herefore, these slight differences may be insignificant.

As previously highlighted in section 4.5 (forecasting, ranking, trading and

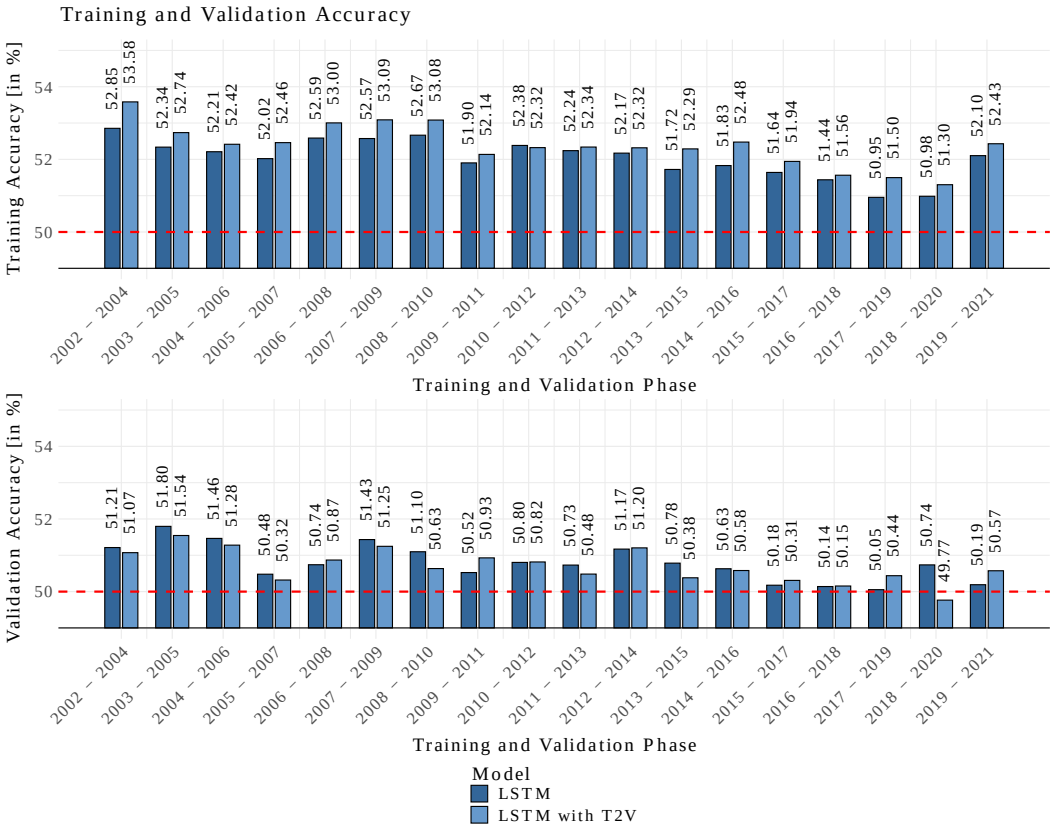

Figure 5.2: Annual accuracy values for LSTM and LSTM with T2V during training and validation phases, calculated using the Formula 4.5

evaluation), the independent examination of observation periods enables conclusions to be drawn regarding the impact of actual events. This section also highlights two upward swings in the loss values. In the training data, both models performed worst from 2017 to 2019 before the Corona crisis. However, this deviation is not evident in the validation data. In the validation periods, the 2007 and 2008 financial crisis stands out as both models achieve their worst validation performance.

These differences can be seen more clearly when looking at the accuracy values. The same tendencies can be seen in Figure 5.2. Over the years, the accuracy of both models' training values and validation data has continued to decrease. This is consistent with the observations in Figure 5.1. The more significant deviations can also be seen accordingly.

In addition, it is noticeable that most accuracy values are only a few points above the vital mark of 50%. A success rate exceeding 50% is crucial because a positive profit forecast is expected over a sufficient number of iterations.

The law of large numbers applies in this context, which stipulates that the probability of a random event tends to approach its mathematical probability as the number of trials increases [20]. This assumption is based on the premise that absolute gains and losses are distributed equally. This implies that small gains are not outweighed by a single large loss and that gains and losses balance each other out in a nearly 1:1 ratio. An excellent example of this would be the roulette game in a casino. The bank has only a slight statistical advantage. But the longer the banker plays, the higher the chances of winning in the long run. An accuracy of just over 50% may sound like little, but it can be enough to manage money successfully.

In the finance sector, the acceptance criteria for the accuracy of DL models differ from those in other domains. Whereas a minimum accuracy threshold of 90% or higher is often targeted in other fields like speech recognition or image classification, an accuracy of just over 50% is generally considered sufficient for finance-related applications like investing or trading. This is due to the inherent randomness and unpredictability of financial markets, which makes it challenging to achieve a high level of accuracy. As a result, it is often more important to have a model that can identify opportunities with a slightly better than random chance of success rather than a model that is highly accurate but may miss many profitable opportunities. However, it is important to note that the specific accuracy requirements may vary depending on the use case and the associated risk tolerance.

Only in the test year 2021 or training and validation period 2018 to 2020 does the validation accuracy of the T2V model temporarily fall below this critical threshold. This indicates that the model may struggle to perform well in unseen data or during a period of higher market volatility. It is crucial to consider the potential limitations and weaknesses of the models when interpreting the results.

As expected, the level of values drops when moving from training to validation. Although the validation data are not entirely unknown, i.e. certain information from the training data is already in the validation data. They are the first indicator of how well the models can handle anonymous data. It is important to note that the validation accuracy is a more realistic measure of the model's performance since it is evaluated on data that the model has not seen before.

In conclusion, the evaluation of the training and validation results of the LSTM and T2V models shows that both models achieve very similar results in terms of loss and accuracy. Although the T2V model performs better in training, the basic LSTM model performs better in validation. The differences in performance regarding the loss are insignificant and only become apparent when looking at the third decimal place. Nevertheless, the validation accuracy

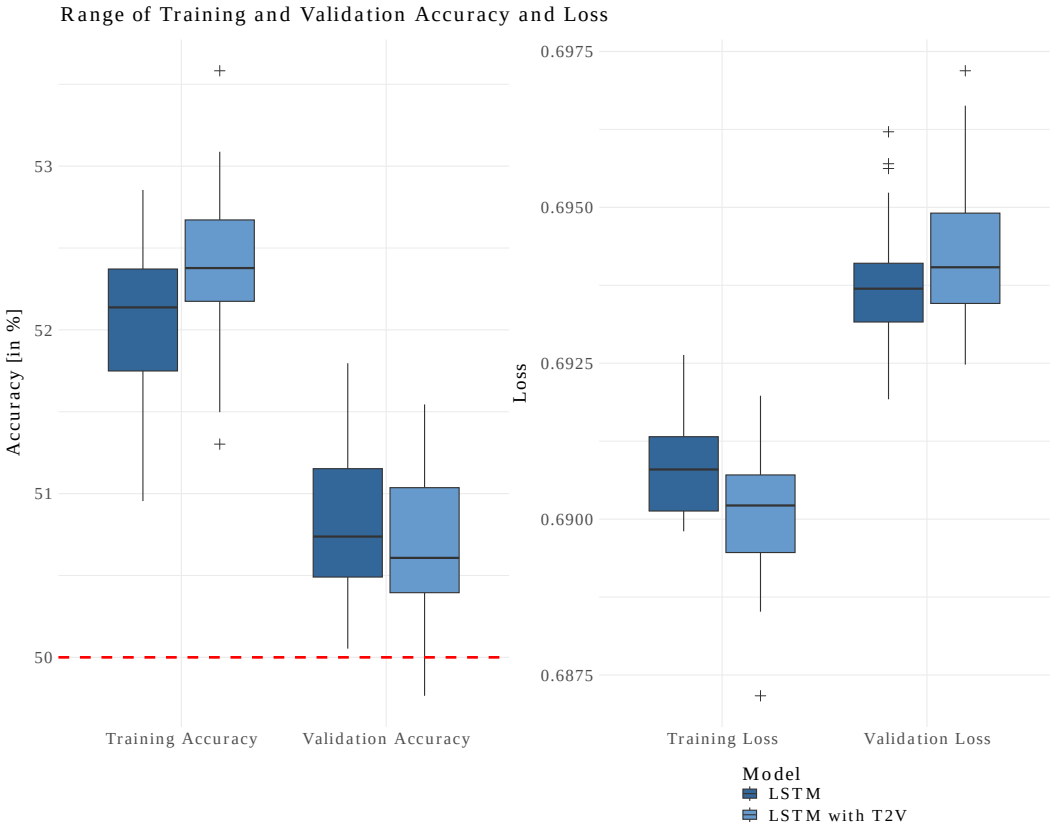

Figure 5.3: Range and rough distribution, with outliers included, of training and validation values for LSTM and LSTM with T2V

of both models is above 50%, which is a critical threshold for a positive profit outlook. The boxplot analysis shown in Figure 5.3 also illustrates apparent differences in the distribution of the loss and accuracy values between the models. Additionally, a clear diametrical relationship between accuracy and loss is evident. Specifically, as loss increases, accuracy decreases and vice versa. These findings provide a basis for further analysis and suggest that the simpler LSTM model may achieve better predictions on the test data than the T2V-modified LSTM network. The following subsection will discuss the differences between the original implementation of Fischer and Krauss and the modified version used in this study.

5.1.2 Evaluation of Different Regularisation Mechanisms

This section compares the performance of three LSTM models: the original LSTM network, a slightly modified LSTM model, and the modified LSTM

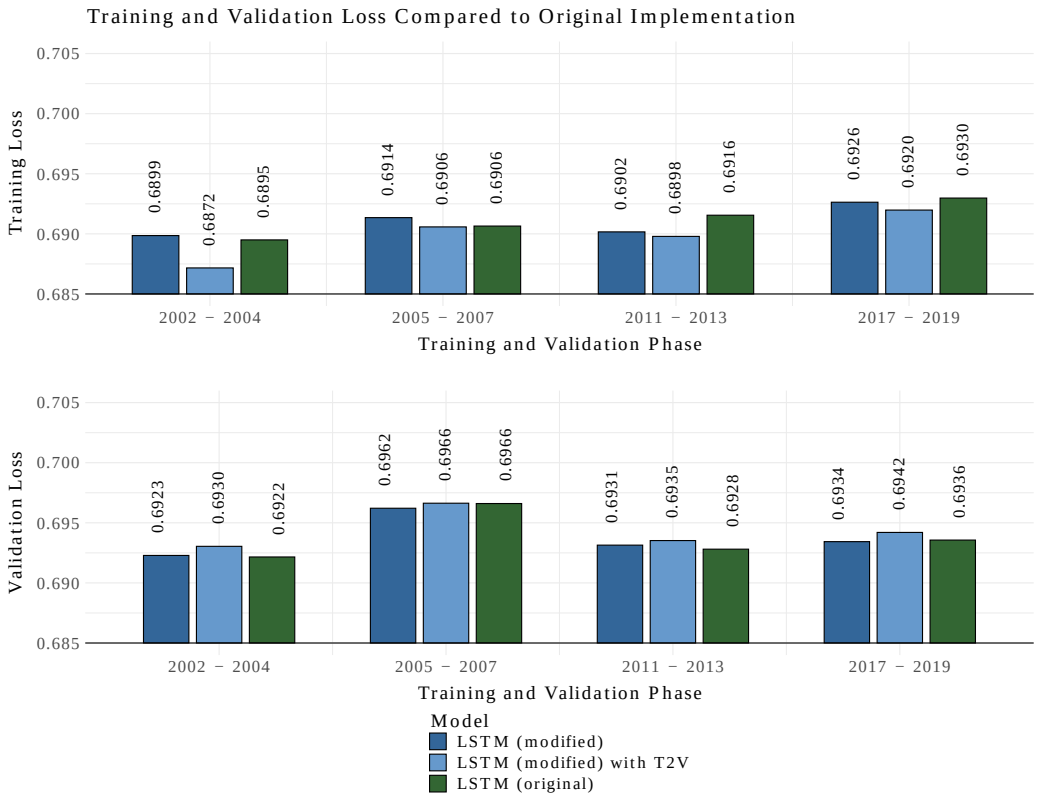

Figure 5.4: Comparison of training and validation loss between Fischer and Krauss's original LSTM network implementation and the modified network variants proposed over a selected sample of training and validation periods

network with an additional T2V layer. The comparison is based on the loss, accuracy, and computing time values during the training and validation phases for selected time intervals to better assess the influence of the deviations in the implementations. According to Fischer and Krauss, the LSTM layer implementations differ only in terms of the recurrent dropout, which is set to 0.1 in the original network settings [14]. Google Colab's GPUs optimised for ML are used for the experiments. The recurrent dropout in the modified networks is set to 0.0 per Keras' requirements to reduce computing time. To compensate for this change, an additional dropout layer is added directly after the LSTM layer, initialised with a value of 0.1. Details of the LSTM networks used in the experiments can be found in the appendices D and E.

The original LSTM network performs similarly to the modified model versions regarding loss and accuracy. However, the original LSTM model outperforms the revised LSTM networks in most cases. Despite the overperformance

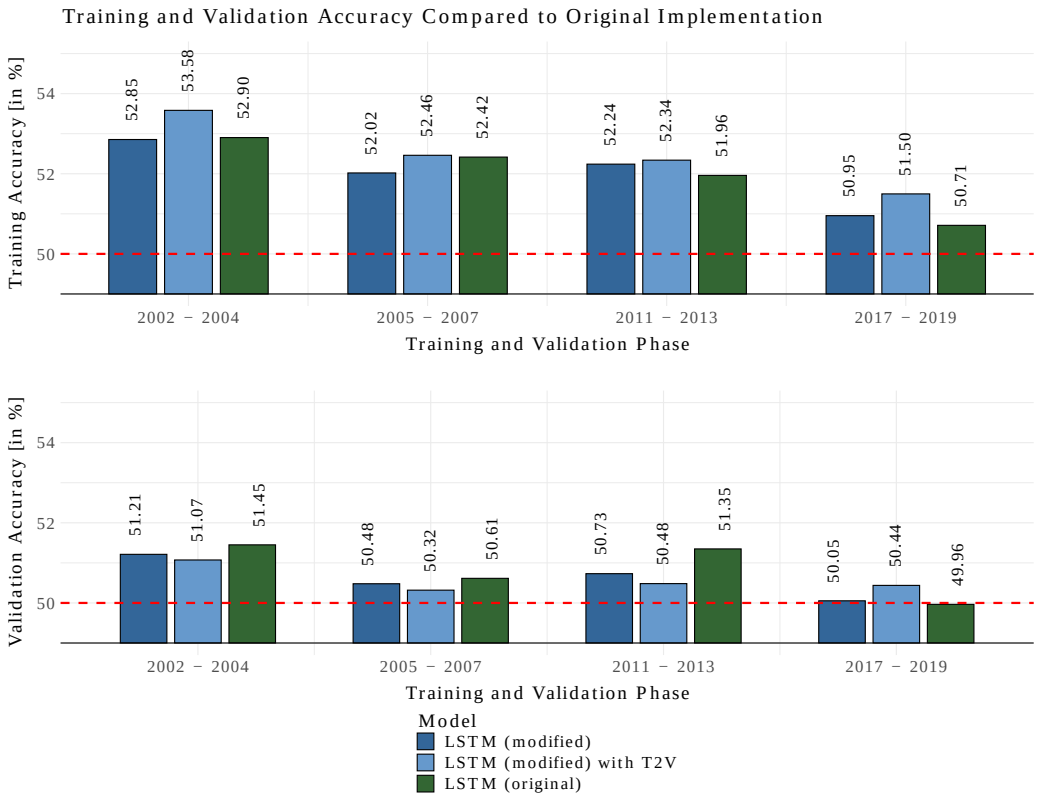

Figure 5.5: Comparison of training and validation accuracy between Fischer and Krauss's original LSTM network implementation and the modified network variants proposed over a selected sample of training and validation periods

of the original version, the modified models perform significantly better in computing time than the original model. This is due to the optimised use of Google Colab's GPUs. The performance differences are presented in detail in the following.

In Figure 5.4, the loss values for training and validation gradually increase over the years, consistent with the previous section 5.1.1. This trend also applies to the original LSTM network, with value levels changing to a similar extent as the other two models. Notably, while the basic model of Fischer and Krauss performs worse in training loss, its validation values are sometimes better than the comparison values. This suggests that the recurrent dropout within the LSTM layer produces slightly better results than the alternative variant using the dropout layer.

The accuracy values support this assumption in Graph 5.5, which shows the same relationship as the loss values but the magnitude of the differences

is somewhat higher. The accuracy values of the original LSTM network are among the worst during the training phase but deliver the best results for three of the four selected periods during the validation phase. However, the Fischer and Krauss model performs worst in the last period from 2017 to 2019. Interestingly, the LSTM model with the T2V layer performs largely between the values of the two simpler models, possibly because the additional layer generates a synthetic feature that partially compensates for the missing recurrent dropout through the additional information flow.

After comparing the original LSTM models with modified implementations in terms of performance, the question now arises as to whether the gains in performance justify the additional computing time required for training the original models. In other words, how high is the price one must pay for using the original network architecture?

To answer this question, the assumption is made that the experimental environment in which the models were evaluated stays the same. The experiments were carried out in the same way, with only the model architectures differing according to their configurations. Under this assumption, the computing times required to train the different model implementations are compared and assessed below.

In a real-world scenario, the importance of a robust and reliable implementation cannot be overstated, and in certain cases, it may even take precedence over a faster model architecture. The significance of this factor varies depending on the specific use case, and hence, generalisations are subject to limitations. However, for the case of trading and investing, the following observations can be broadly made. If the investment horizon is relatively longer, a slower but more secure implementation may be acceptable. For instance, investors who aim to adjust their portfolios weekly or monthly have greater flexibility and are not constrained by time pressures. In contrast, stock traders engaged in high-frequency trading require a model that is both robust and fast and can be quickly adapted to new market conditions.

Furthermore, the declining cost of computing resources over the past two to three decades must also be taken into account while interpreting the following results. According to the AI Impacts website [1], the cost per GFLOPS (giga floating point operations per second) has reduced from hundreds of US dollars to a few cents. This trend has made it feasible for a broader population to undertake such projects. This aspect will be revisited in subsequent analyses.

To highlight the striking difference in computing power, Figure 5.6 shows the computing time in minutes for the entire cycle of an iteration, including training, validation, and testing. Although the original LSTM models tend to deliver better results in terms of loss and accuracy, they require many times more computing time for execution. The difference is most extreme from 2011

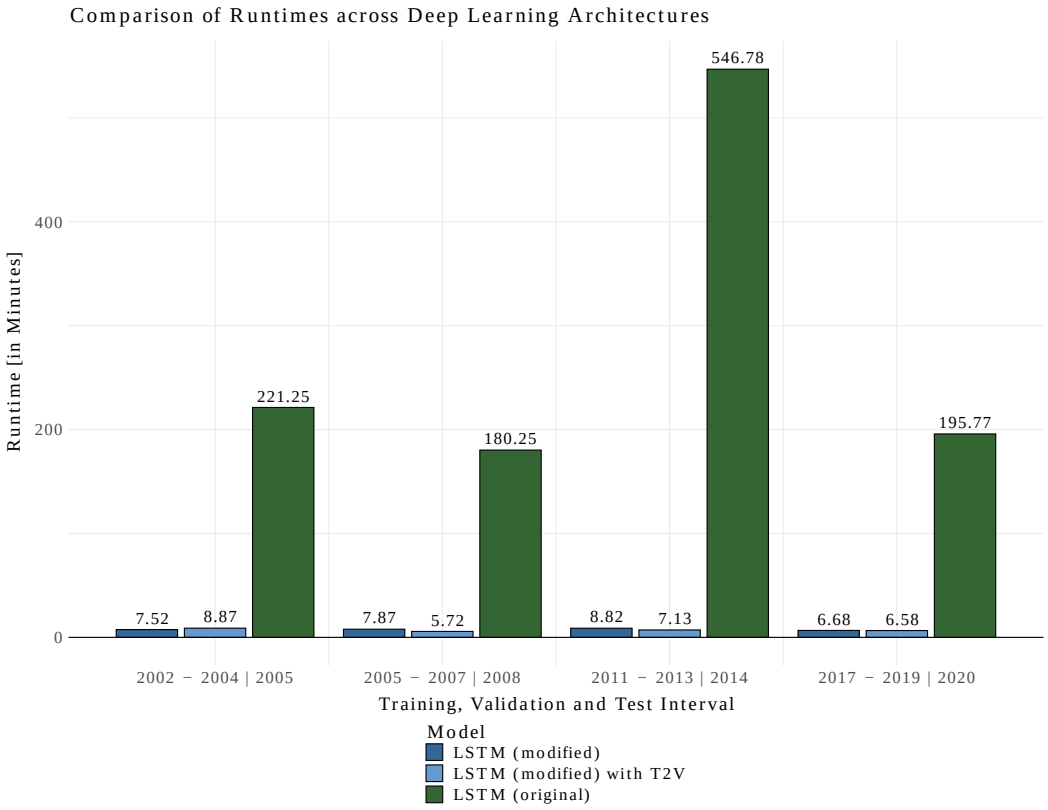

Figure 5.6: Comparison of computing times between Fischer and Krauss's original LSTM network implementation and the modified network variants proposed over a selected sample of training and validation periods

to 2014, with the Fischer-Krauss implementation taking almost 62 times as long compared to the LSTM with the T2V extension and nearly 77 times as long in contrast to the modified LSTM model. The differences in the other study periods are not quite as dramatic, but they are still striking.

This is an excellent example that it is always necessary to check individually whether the implementation corresponds to the requirements catalogue. If additional time is available, only little can be said against using the original network implementation. However, if a reasonable compromise between computing time and model performance has to be found, the modified model implementation offers a suitable alternative. Therefore, the choice between the original LSTM models and the modified implementations depends on the specific requirements and constraints of the application.

5.1.3 Evaluation of Test Results

The next part of the evaluation focuses on the test values of the LSTM model, LSTM with T2V, and the STR benchmark. These test phase results are critical indicators of the successful deployment of DL models since they assess how well the models would perform in real-world applications. To analyse the accuracy values of different models, this section exclusively looks at their performance during testing. The STR System discussed earlier is also considered. Furthermore, accuracy values for various portfolio sizes and the accuracy values for all stock predictions are examined. This approach extends the understanding of the presented results and provides a conclusive transition and link between the DL related and the finance-related results. To aid comprehension, corresponding illustrations will be used to explain the results further.

Figure 5.7 illustrates the annual accuracies for the LSTM model, LSTM with T2V, and the STR benchmark. As one can observe, most of the accuracy values are above the significant 50% mark. This result suggests that the models would have made good investment decisions, especially in the first test years. However, it is noticeable that there is a downward trend similar to the training and validation phases. The data also shows drastic stress phases in the financial markets, such as the financial crisis from 2007 to 2009. In this period, the accuracy of all three models declines significantly. However, the performance of the DL models improves again in the years after that.

The test years 2017 and 2018 can be considered a turning point, as the accuracy values of the ML models fall below the 50% threshold for the first time. Moreover, from then on, the DL models perform worse than the simple benchmark heuristic STR. Although the benchmark model is consistently slightly worse than the DL variants over the years, it steadily delivers good results. It can improve towards the end of the entire test period. Overall, the accuracy values of the DL models are promising, but their performance is subject to market fluctuations and crises, similar to other investment models.

There are a few potential explanations for the paradigm shift observed in the performance of DL models versus the benchmark heuristic. One possibility is that the barriers to successfully implementing these models have decreased substantially in recent years, leading to broader adoption and better performance. However, in the context of finance, this may actually work against the models, as more market participants using similar technology can rapidly disappear exploitable market anomalies. Some large companies now have entire departments dedicated to this area. On the other hand, it's also possible that the market anomaly underlying the benchmark heuristic is too large or difficult to eliminate. Thus it remains in the market despite the increased adoption of DL models. One potential reason for this anomaly's persistence

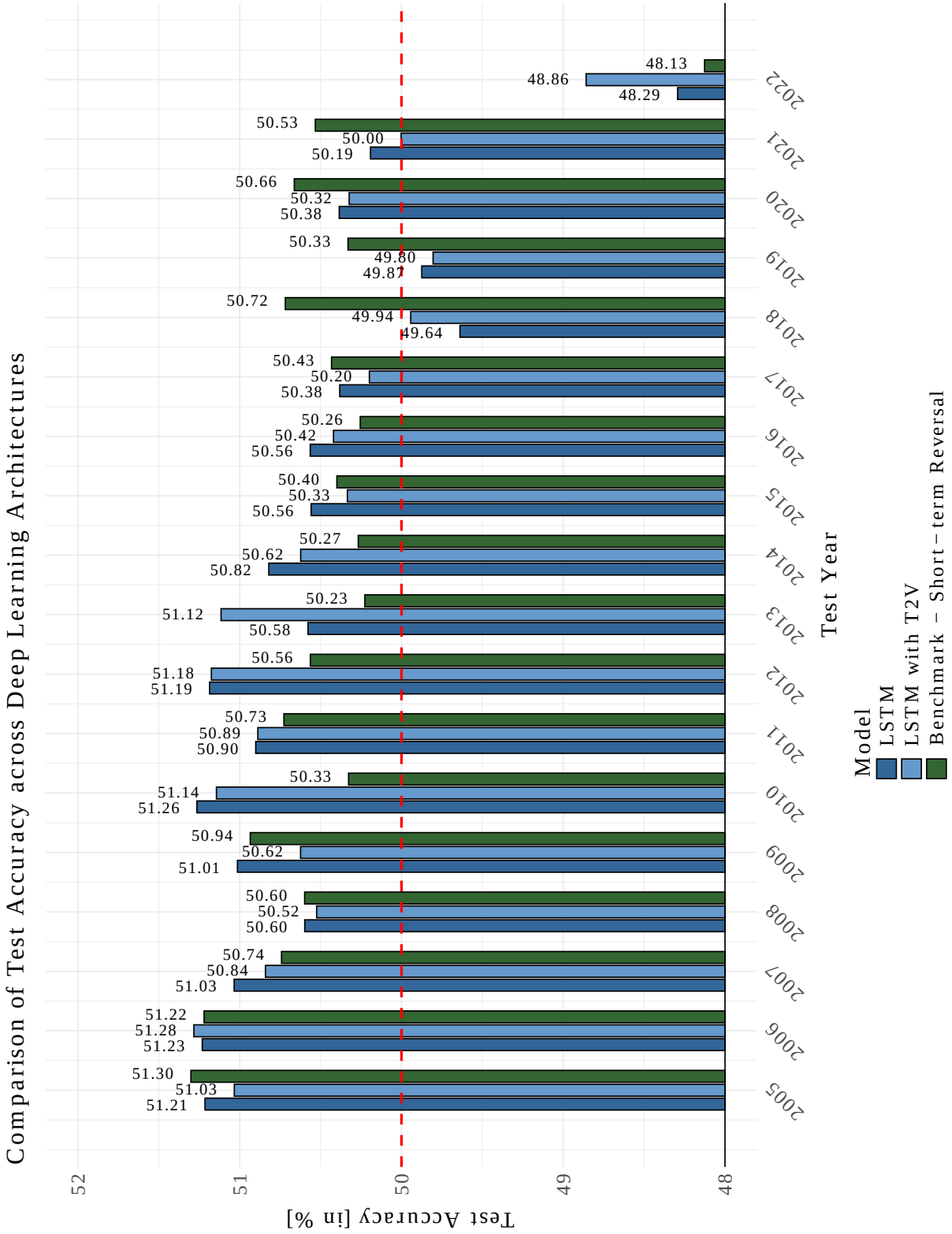

Figure 5.7: Annual accuracy values for LSTM, LSTM with T2V and Benchmark STR

may be excessively high transaction costs, which render the investment strategy unprofitable in practice.

Furthermore, it is essential to note that the test year 2022 falls outside the main test period and thus may only be partially meaningful. This value may be skewed because only the first few months of 2022 were included in the test phase. Overall, these factors highlight the complex and multifaceted nature of evaluating the performance of DL models in finance and the need to carefully consider a wide range of factors in interpreting the results.

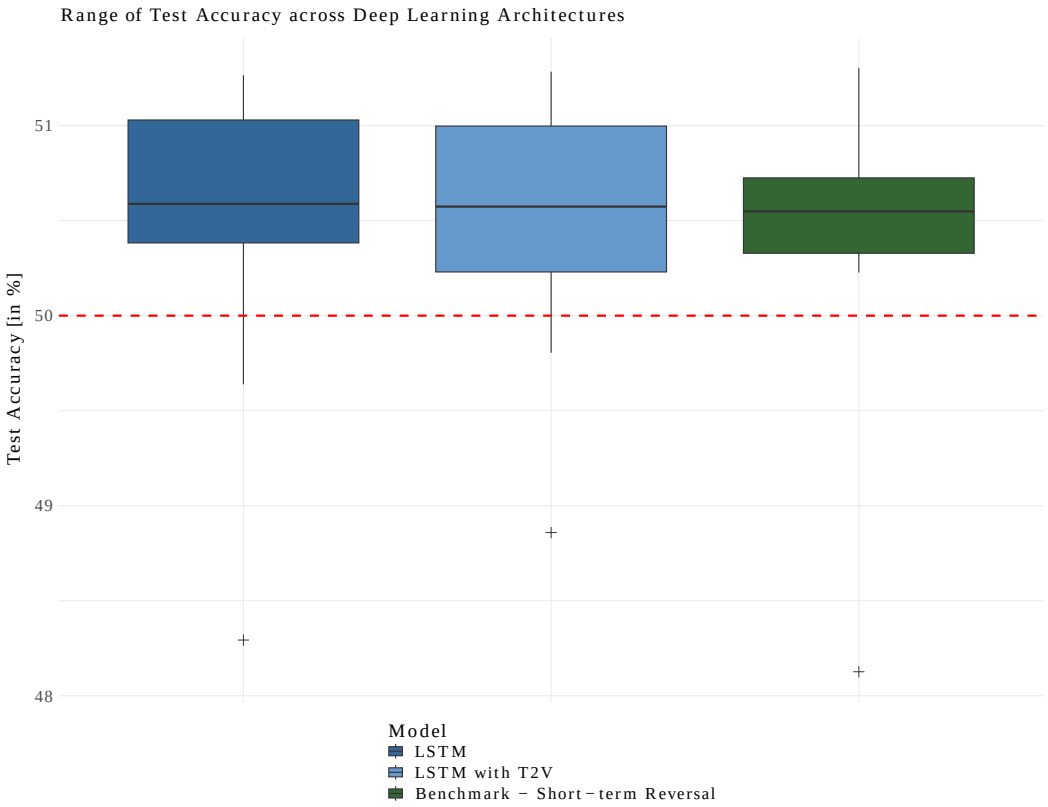

Figure 5.8: Range and rough distribution, with outliers included, of test values for LSTM, LSTM with T2V and Benchmark STR

The boxplot depicted in Figure 5.8 provides valuable insight into the distribution of the test values for the three models. It reveals that most values for all three models are above the 50% limit. However, the three outliers at the bottom of the graph, representing the test values for the year 2022, can be considered insignificant and disregarded for further analysis. Another noteworthy observation is that the median of all three models is at the same level, despite the distributions of the remaining values being very different. The boxes of the

DL models cover a more comprehensive range and are larger than that of the benchmark model, with the test values of the LSTM networks being subject to greater variance. This confirms the previous figure's assessment that the DL models initially perform better but exhibit a decline over time, whereas the benchmark model consistently delivers a solid performance. A trade-off between better performance with higher variance and moderate accuracy with lower variance is evident. The benchmark model has the highest lower limit, not considering the outlier, and is above the limit of 50%. Next, the accuracies for various portfolio sizes will be examined, offering a more detailed analysis of the performance of the models.

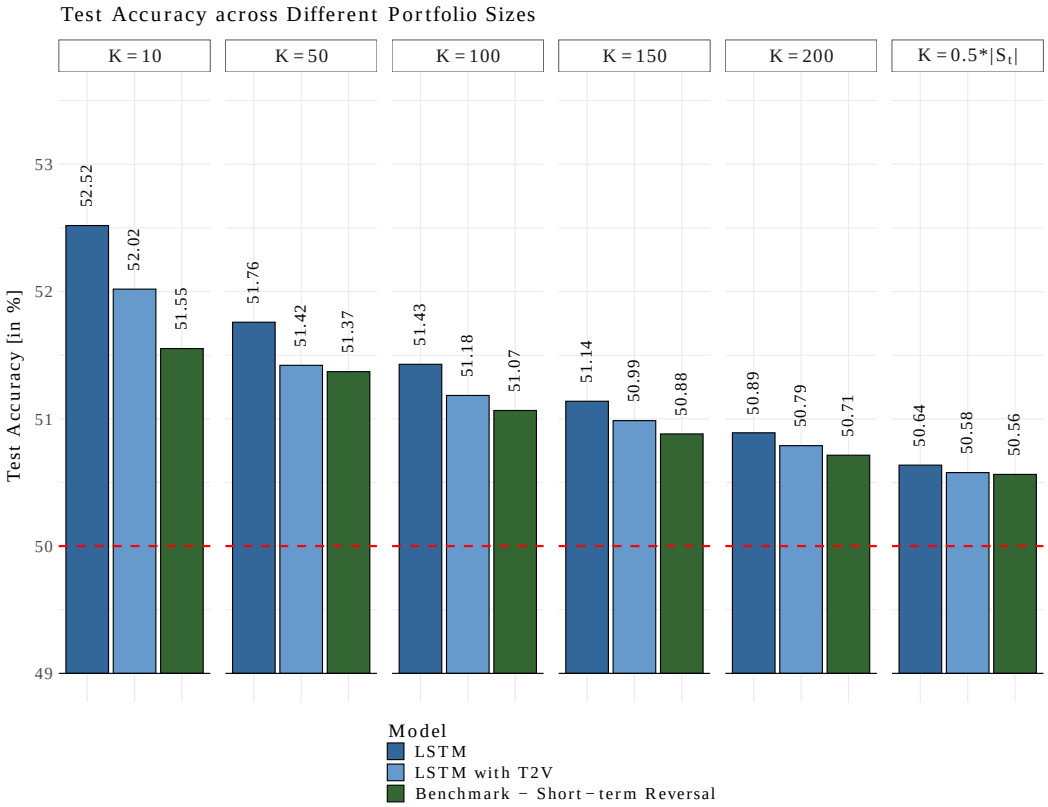

Figure 5.9: Aggregated test accuracy across different portfolio sizes for LSTM, LSTM with T2V and Benchmark STR

Before completing the evaluation of the DL metrics, the accuracy values for different portfolio sizes are examined in preparation for the next section. Figure 5.9 compares the model accuracies across different draw sizes $k \in \{10, 50, 100, 150, 200, 0.5 \cdot |S_t|\}$, whereby the daily portfolio composition consists of $2k$ stocks. The expression $0.5 \cdot |S_t|$ denotes half of all selectable

stocks in S_t at a point in time t. This mathematical term depending on the cardinality of S_t was used because the number of total stocks in S_t fluctuates around 600 due to certain stocks entering or leaving the index without correction.

It is evident that all values are not only above 50% but also an increasing restriction of the portfolio size leads to better accuracy values. This makes sense as it limits the selection to the stock values with the best forecast values. On the other hand, the larger the selection size k, the lower the accuracy. Overall, the results indicate that regardless of the draw size, the LSTM model exhibits superior predictive performance, followed by the T2V modified variant, while the benchmark model demonstrates the weakest predictive power.

However, the difference between the values decreases with increasing portfolio size, with the values converging more and more to one level. The differences are most remarkable in the $k = 10$ category, where the LSTM model with 52.52% is clearly above the threshold value of 50%. The T2V model, while better than the benchmark model, is closer to the benchmark's performance than that of the LSTM network. Overall, the values suggest that the models can make reasonable predictions, but a significant degree of inaccuracy in the predictions is still expected as accuracy decreases with increasing k. The following section will demonstrate the suitability of the model predictions for making investment decisions on the financial market.

In conclusion, the results showed that restricting the portfolio size led to better accuracy values, while larger selection sizes decreased accuracy. The LSTM basic model made the best predictions across all draw sizes, followed by the T2V modified variant, and the benchmark model performed the worst. However, all models had a significant degree of inaccuracy, which decreased with increasing portfolio size. These findings suggest that the models are capable of making reasonable predictions, but a considerable degree of uncertainty remains. The next chapter will assess the models' suitability for making investment decisions in the financial market.

5.2 Investment Evaluation

This chapter builds upon the previous section's DL analysis and shifts the focus to evaluate the financial performance aspects of the models. In the first part of this chapter, an evaluation is conducted on the financial implications of the LSTM, LSTM with T2V and the benchmark model based on the STR heuristic. The analysis focuses on key financial ratios for different portfolio sizes, such as the average daily return, annualised standard deviation, and annualised Sharpe ratio (see Equation 4.10 in section 4.5). The performance

of these models is compared to each other to gain insight into their potential use in investment decisions.

In the second section of the chapter, the investment performance of a portfolio with a draw size $k = 10$ is analysed in more detail. In addition to the LSTM, LSTM with T2V and the benchmark model based on the STR heuristic, the simple BH strategy is also included as a benchmark. The analysis covers the average daily and cumulative returns for different periods, along with the annualised Sharpe ratio by year. Comparing the performance of these models and strategies can provide a comprehensive view of their financial implications.

The first part of the performance analysis is summarised in Figure 5.10, which presents the results for different portfolio sizes. The figure shows that the best-performing model for all ks is the LSTM network with an average daily return of 0.09%, followed closely by the T2V-supplemented LSTM model with 0.07%. In contrast, the benchmark model based on the STR heuristic has the lowest return, with an average daily return of just 0.04%, less than half that of the LSTM network. This ranking remains consistent for all other ks, but the distances between the models change. Notably, the returns of the DL models decrease significantly with a larger k, with the LSTM model's daily average return falling to 0.06% for $k = 50$. However, this value still represents a good return. The LSTM model with a T2V layer comes close to the benchmark performance and is slightly above the benchmark return. The benchmark model can maintain its level of return and even achieves an average daily return of around 0.04% per day for $k = 50$. The different developments between $k = 10$ and $k = 50$ may be related to the fact that, on the one hand, the probability increases with increasing k that selected individual stocks were misclassified for the DL models. On the other hand, the benchmark model may benefit from a certain degree of diversification. Across the remaining ks, the average daily returns for all three models decrease equally without any significant deviation. Notably, despite a huge K (e.g., $k = 200$), the returns remain positive for all models and the benchmark strategy, indicating their potential for different investment approaches.

Moving on to the risk evaluation, the standard deviation plotted in the line directly below the average daily return in Figure 5.10 is analysed. Beginning at $k = 10$, it becomes apparent that the benchmark model yields a lower return and involves higher risk compared to the DL models. While the LSTM and T2V-supplemented LSTM models exhibit standard deviations of around 0.73% and 0.69%, respectively, the benchmark heuristic incurs a risk of almost 1.10%, approximately 40% higher than the risk level of the DL models. This significant difference in the level of risk is consistent across all k values and even increases further between the models with increasing k. The discrepancy in the percentage difference within each draw size between the models rises

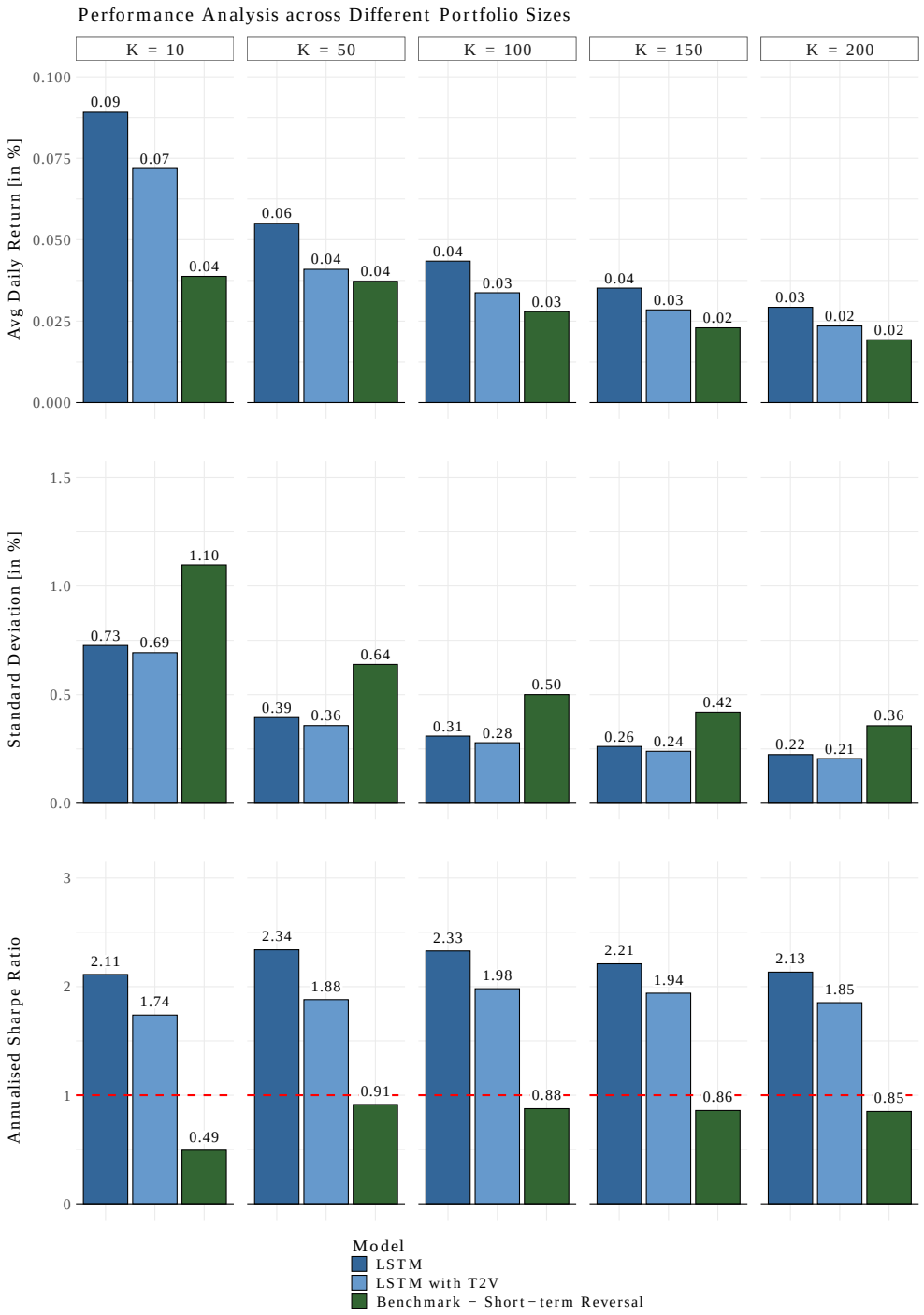

Figure 5.10: Aggregated performance analytics before trading costs (average daily return, standard deviation and annualised Sharpe ratio) across different portfolio sizes for LSTM, LSTM with T2V and Benchmark STR

to over 70%, which is greater than the decrease in the values of all models with increasing k. These observations, combined with the analysis of average daily returns, suggest that DL models not only yield higher returns but also achieve them at a lower risk. This indicates that DL models offer a promising alternative to the more straightforward model-based approach.

The Sharpe ratio is a crucial metric to assess the risk-adjusted performance of a portfolio. It combines insights into returns and risks and is defined as the ratio of the annualised average return to the annualised standard deviation. As explained earlier in section 4.5, a Sharpe ratio value above 1 is positive because it indicates that you can expect more return than risk. In Figure 5.10, the Sharpe ratio for each model is plotted against the drawdown size k. The last section of the figure again clearly illustrates the massive difference in performance between the DL models and the benchmark model. Although the LSTM network does not achieve the best Sharpe ratio for a draw size of $k = 10$, it is clearly above the other two comparison approaches. The LSTM model with T2V addition does not perform poorly either. Only the benchmark approach is clearly below the limit of 1.

As the draw size k increases, the Sharpe ratio increases slightly for all models, with the benchmark Sharpe ratio, in particular, increasing enormously. However, it remains below the hurdle of 1. The Sharpe ratios of the individual models reach the following values for drawdown sizes of $k = 50$ or $k = 100$. This suggests that a larger portfolio with broader diversification is more conducive to the portfolio's total return than a more concentrated investment approach. The Sharpe ratios also remain at their respective levels and only fluctuate slightly by a few percentage points. Based on the results obtained, if one looks ahead and thinks of a possible productive set-up, a portfolio of about 60 to 100 individual stocks, i.e. the k is between 30 and 50, probably makes the most sense. This portfolio can be expected to generate high returns and requires taking less risk than a portfolio with $k = 10$, so the Sharpe ratio is close to the corresponding maximum. In addition, transaction costs, which were not considered in this study, would be significantly lower than for a portfolio with 300 to 400 stocks.

To better understand the profitability of the models over time, the cumulative returns, yearly average daily returns and yearly Sharpe ratios for a portfolio with $k = 10$ are displayed in the following chart 5.11. In addition to the three previously mentioned DL models, the comparison will include the commonly used and straightforward BH strategy. This analysis will provide further insight into the effectiveness of each model in generating returns over an extended period.

Three different periods, namely 2005-2009, 2010-2015, and 2016-2022, are considered, following the approach of Fischer and Krauss [14]. The first period

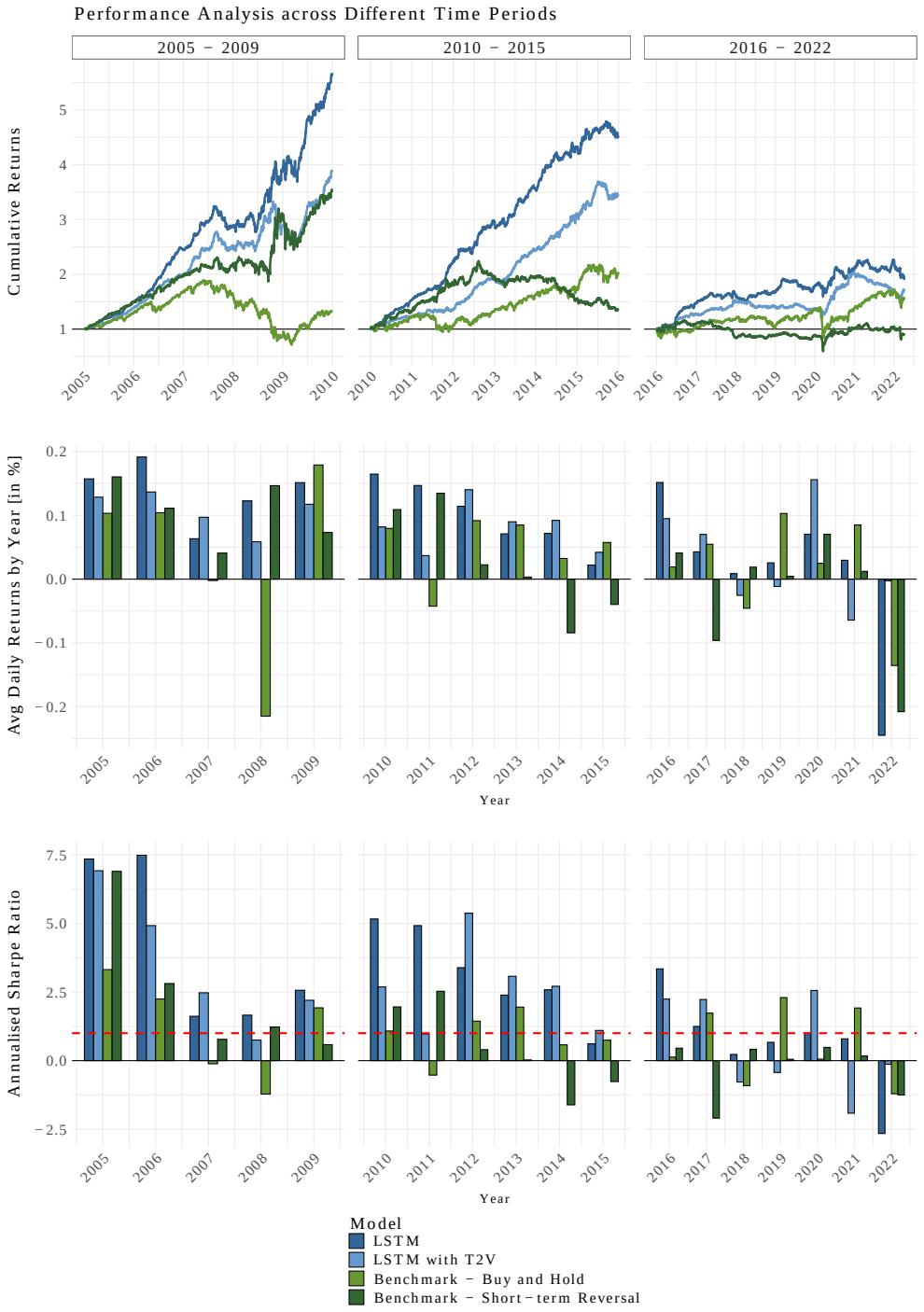

Figure 5.11: Performance analytics before trading costs (cumulative return, average daily return by year and annualised Sharpe ratio by year) over time for a portfolio size of $k = 10$ for LSTM, LSTM with T2V and Benchmarks STR and BH

covers the entire financial and economic crisis in the 2000s, while the second period coincides with the financial market's recovery and new all-time highs. Finally, the period from 2016 to 2022 is examined, which also sheds light on the behaviour of the models at the beginning of the Corona crisis. To better understand the profitability over the years, the following chart 5.11 looks at the cumulative returns and the Sharpe ratio over these three different periods. In addition to the three previous models, the BH approach is used as a further comparative model, as this strategy is straightforward and widespread.

The plot of the cumulative return in the first row of Figure 5.11 shows very different growth curves for the four models over the three periods. In the first period of 2005-2009, the simple LSTM model achieved the best return with a total return of almost 500% in 5 years. Still impressive, but much more moderate, is the performance of the LSTM network with T2V extension, with the final result of the benchmark model STR only just behind. Both values are in the range of an increase of 250% to 300%. In contrast, the benchmark model BH comes in last place with not even 50% total return. The performance of the models in the first 1.5 years suggests that they are similarly efficient. However, some disparities become noticeable in the years leading up to mid-2007. The LSTM model can outperform the other models by a significant margin. The second LSTM model, which employs time vectorisation, shares second place with the benchmark model STR for a considerable time and ultimately finishes slightly ahead of its peer. On the other hand, the BH approach experienced a decline from mid-2007, with total returns plummeting in the following year and hitting rock bottom with negative returns in early 2009.

Although the BH approach shows an evident decline from mid-2007 onwards, the other three models can maintain their previously established level and incur significantly smaller losses than the BH concept. From mid-2008 onwards, all models can catch up with the BH approach or even display favourable growth rates, despite the increased volatility.

The second row of the table provides additional insight into the models' annual average daily returns in percentage. The years 2005 and 2006 are identified as outstanding trading years. In 2005, the LSTM network and the STR approach delivered almost equivalent results at approximately 0.16%. The DL model performed even better in 2006 and came close to achieving a daily return of 0.2%. The investment strategy based on the STR approach registers a slightly lower value of just above 0.1%. The values of the other two models remain almost the same and range between 0.1% and 0.15%.

The onset of the financial and economic crisis resulted in the market becoming more active, leading to difficulties for the models in maintaining their growth rates. In 2007, all models experienced a significant decline, with the LSTM models suffering the most significant setback. However, the T2V model

performs relatively well and maintains a high level of 0.1%. The BH concept declines to just below 0%, further underlining the superiority of the models over the simple investment strategy.

The analysis of the performance of DL models in the stock market from 2005 to 2009 has revealed several interesting findings. In the first 1.5 years, all models performed similarly, but from mid-2007, differences in their performance became evident. The LSTM model showed a significant improvement and outperformed the other models. The BH approach showed apparent weaknesses from mid-2007 onwards, while the other models could hold their ground at their established level and accept significantly smaller losses.

In 2008, the LSTM model made significant gains, which could be attributed to the first data from the crisis being used in training. The STR heuristic approach had unbeatable gains this year, while the LSTM with the T2V network extension continued to decline. In 2009, a certain recovery phase occurred, and all models generated positive returns. The BH approach saw the most growth despite the LSTM model appearing to be growing the most.

The annual Sharpe ratio values from 2005 to 2009 revealed that those from 2005 and 2006 stood out. All values were above the limit of 1, with the three other models reaching outstanding values above 7. In the change to 2006, the values reduced significantly but remained at an attractively high level. During the crisis, the Sharpe ratio values dropped significantly before recovering slightly in 2009.

Overall, the DL models were more profitable than their peers and performed well during the crisis period without major losses. These findings demonstrate the potential of DL models in the stock market and suggest that further research could lead to even more significant advancements in financial modelling.

From 2010 to 2015 was the most favourable for DL models from an investment perspective, as they consistently delivered good results with limited fluctuations, making them less risky for investors. Although high-return years like 2005 or 2006 are desirable, investors also value stability and lower risks, which DL models were able to provide. The LSTM model with the T2V layer and the two DL models performed exceptionally well during this period, except in 2011 for the LSTM model with the T2V layer and 2015 for both models. The DL models remained profitable despite the slight decrease in the total return compared to the previous observation period.

Compared to the two benchmark models, the BH model performed better than the STR approach during this period, but the DL networks outperformed both. The STR model performed similarly to the BH system in the previous period, but its total return started falling steadily from mid-2012 onwards.

From a risk-return perspective, the LSTM model performed the best in the

first two years, but it was later replaced by the T2V-enhanced LSTM network, which dominated the years from 2012 to 2015. However, an overall trend can be observed where the performance of the DL models gradually decreases over the years while the values of the benchmark models seem to remain almost the same. This decline in the performance of DL models is likely due to the increasing proliferation of software and hardware, their increased performance, and the growing amounts of data that enable better training of the models. These developments were already hinted at in the previous chapter on DL-related evaluation. The following observation period will further validate this trend, and it will be interesting to see how DL models perform in the face of these challenges.

The deterioration trend in the performance of DL models continues at an accelerated pace from 2016 to 2022. While the benchmark models perform similarly to previous periods, both DL models can no longer match their previous successes. The LSTM model still realises the best total return at around 100%, but it suffers a drop in returns of 70% to 80% compared to previous years. The LSTM network with T2V extension is also unable to continue its initial successes, although the total return declines less in relative terms than is the case with the basic LSTM model. Moreover, the behaviour of the models during the Corona crisis in the early 2020s is also interesting. While the DL networks cannot match their previous performance, they still convince with their relatively robust behaviour during this critical phase. The growth curve of the BH model, in particular, declines sharply at the beginning of 2020 but can recover a little. In this precarious phase, the LSTM model with the T2V layer is the most convincing.

As the growth curves indicate, the daily average returns only record moderate values and can only occasionally show higher returns for individual years. It is also the first time the DL models have had to accept negative average returns. The year 2022 is again particularly conspicuous here, although, as already mentioned, this period is only of limited significance as it does not cover the entire year 2022. The Sharpe ratios of the DL models have also suffered a severe dent and are mostly even below the vital limit of 1. While the values in 2016 and 2017 were still above 1, they are only below this for the remaining period and even fall into the negative value range. Only the benchmark models, especially the BH principle, can convince with higher Sharpe ratios several times even after 2017.

In summary, while DL models showed consistent and impressive performance in the early years, their performance has deteriorated significantly in recent years. Although these models perform relatively better than benchmark models during periods of economic turmoil, their average returns and Sharpe ratios have decreased considerably. The continued development of software

and hardware and the increasing availability of data may have contributed to the decline in the performance of DL models. Future research will need to investigate these trends further and explore new approaches to improve the performance of DL models in financial forecasting.

It is essential to consider certain aspects that put the results into perspective and an appropriate context. First, the financial markets experienced a remarkable upswing in the phase following the financial crisis of 2007-2008, starting around 2010. For years, share prices rose steadily, enabling solid returns. However, this trend weakened from 2016-2017 onwards, leading to an era of higher volatility and uncertainty, exacerbated by the Corona Crisis in 2020 and the current difficulties in the financial markets. Second, it is essential to recognise that the amount of data available and hardware and software capabilities have increased significantly. At the same time, the barriers to entry have decreased due to numerous open-source resources and the enormous decline in computing costs as already mentioned in section 5.1.2. These lower barriers to entry and the widespread adoption of the technology have led to a surge in research in this area, as explained in 3. The increasing number of better approaches, combined with the previously mentioned factors, suggests that a potential advantage based on a well-trained DL model can only generate good returns for a limited time. In this context, it is not surprising that overall returns have declined for all approaches, but especially for DL networks.

Chapter 6

Conclusion

This final chapter presents a summary of the major discoveries before addressing the questions raised at the start of this thesis and providing a glimpse of potential future research projects. The study found that the LSTM model and the LSTM model with T2V performed well, but the T2V model was slightly better during the training phase, while the simpler LSTM model performed better during validation. A plausible explanation for this phenomenon is that the simpler LSTM network is better equipped to learn general patterns. In contrast, the LSTM model with T2V extension may be more prone to learning new but unwanted relationships during the training phase. Additionally, the research uncovered that during periods of higher market volatility, both models exhibited better training and validation scores than during low volatility periods. One possible explanation for this correlation is that during stressful market phases, otherwise rational and prudent investors have limited focus due to euphoria or fear. In behavioural finance, this is also called bounded rationality [23, 22]. These behavioural patterns guided by emotions can lead to changed market situations, which the DL models can better recognise and learn from. Overall, the networks achieved solid results during the training and validation in the first years. Nonetheless, performance has slowly deteriorated and visibly dropped from 2017 onwards.

The comparison between the modified DL networks and the framework of Fischer and Krauss [14] has revealed some interesting insights. In the initial years, the implementation by Fischer and Krauss with the recurrent dropout as a regularisation mechanism outperforms the modified network variations using a dropout layer. This suggests that the recurrent dropout within the LSTM layer suppresses the learning of unwanted patterns better than a standalone dropout layer, thus yielding model performance. However, the network's performance with recurrent dropout also deteriorates towards the end of the observation period. Regarding computing time, there is a significant difference

between the modified models and the framework of Fischer and Krauss [14]. The modified models usually require only a few minutes to complete an iteration over one of the observation periods, while the original version takes several hours. Although the difference in computing cost may not be significant nowadays, it can still cause problems if there is a need for quick model training. The significant increase in computing time is due to the recurrent dropout in the original network settings of Fischer and Krauss. One way to mitigate this issue is to use the Keras software package and set the recurrent dropout to 0, which allows GPUs to use their full computing power. On the other hand, it may be possible to circumvent this problem by using a different software package or by custom programming. However, this requires advanced programming skills and results in considerable time and effort. Such an implementation may not be economical.

While examining the test phases, it became evident that DL models not exposed to volatile market data characterised by uncertainty in their training phases did not perform well in stressful crisis periods such as 2008/2009. On the contrary, the models trained using data incorporating such phases exhibited superior performance. As previously discussed, the concept of bounded rationality is probable to be a contributing factor [23, 22]. In line with the training and validation results, the DL networks performed very well until 2017 and deteriorated significantly afterwards. Overall, over the entire observation period, a steady decline in performance can be seen compared to the simpler benchmark models. Nonetheless, the examination revealed a general trend of declining performance in DL networks compared to simpler benchmark models, possibly due to technological advancements that have made the market structure more challenging for DL models. This trend underscores the importance of acknowledging the limitations of DL systems and developing alternative approaches to tackle the dynamic and complex nature of financial markets.

From a financial perspective, the previously discovered insights are also reflected by the financial analysis. With Sharpe ratios of up to 7.5 at times and average daily returns of approx. 0.2%, the performance of the DL networks exceeds the results of the comparison models considerably, especially at the beginning of the reviewed period. The healthy returns and notably superior risk-return ratio for numerous investors constitute an initial positive indicator. However, from 2017 onwards, their performance increasingly declined, and the returns were only slightly above those of the comparison models. On the other hand, the benchmarks consistently delivered solid returns. This performance degradation from the DL evaluation is also emerging in the direct use case, indicating that the performance drop also has a negative impact on a possible productive deployment of the network. F. Chollet also points out

that models require regular re-training and, in the worst case, existing network architectures can be replaced by new, more powerful ones [7].

Overall, the findings suggest that LSTM networks have the potential to make accurate predictions in complex low signal-to-noise environments like financial markets. The results show that the framework proposed by Fischer and Krauss, which uses a smaller LSTM model, leads to comparable results for both newer and European market data - despite minor adjustments in the network architecture [14]. Although the outcomes obtained by Fischer and Krauss show a lag of several years in the European region, the results also imply that DL networks produce favourable results despite regional differences [14]. Nevertheless, the error tolerance remains relatively high, and the performance of DL models can deteriorate over time due to the increasing spread of the technology and changing market conditions.

To be successful in the long term, DL models need to be constantly adapted and improved, either through fundamental structural changes or minor enhancements. Hybrid models, which combine different base architectures, are emerging as a powerful approach in this context, and this trend is likely to continue.

With regard to the T2V extension, it does not appear to have led to any sustainable improvement in performance in the experiment conducted. Several factors could contribute to this, such as the small size of the model, the insignificance of the informational value added and the chaotic nature of the financial data. Nevertheless, it is not possible to draw a conclusive assessment due to the black box-like nature of DL models.

The investigation of the different regularisation mechanisms shows that even minor adjustments can have a major impact on the computation time despite only slight changes in the actual model performance. The research results suggest that Fischer and Krauss' approach, which uses a recurrent dropout within the LSTM layer, achieves slightly better results [14]. However, the modified version with an additional dropout layer instead of the recurrent dropout function reduces the training time significantly. Nonetheless, it should be noted that the aforementioned observations are specific to the implementation using Keras and TensorFlow [6, 10]. Thus, other software packages or custom implementations may offer ways to mitigate this issue. Ultimately, the primary objective should be to optimise the training procedure, particularly for intricate models with considerably more layers and parameters, and to leverage the full potential of available computing resources.

In conclusion, while DL frameworks have shown promise in financial prediction tasks, they are not a guaranteed solution. Careful consideration should be given to the development and implementation of DL systems in financial applications, and hybrid approaches that combine different base architectures

should be explored. Further research is needed to understand the factors contributing to DL models' performance in financial applications and to develop methods to improve their accuracy and reliability over time.

This bachelor thesis has demonstrated the potential of DL networks in predicting financial market movements. Nevertheless, it also highlighted the challenges of using these models under dynamic and complex market conditions. The following are potential avenues for further research in this field:

- **Model Configurations:** This study examined the performance of a single-layer LSTM model with 25 units in predicting the stock market. Future research can explore alternative model configurations, such as a larger variant with more layers and units or hybrid models that combine different basic architectures. Additionally, hyperparameter tuning can be conducted to find optimal model settings that achieve the best results.

- **Additional Data Sources:** Including more data sources such as inflation, interest rate, or text information can provide models with additional information about current market conditions. Future research can explore the benefits of using such data sources in combination with DL networks.

- **Continual Learning:** The networks were reinitialised with each observation period without reusing the already trained models. This improves the comparability between the observation periods, but the networks are trained on much fewer data than is available. Continuous learning could improve the predictive capabilities of the networks, providing the models with more historical data and better capturing the long-term patterns of the stock market.

- **Explainability and Interpretability:** Another area of research that requires further exploration is the interpretability of DL models. Due to the black box-like properties of these models, it is often difficult to understand how they arrived at their predictions. Researchers can explore methods to improve the transparency and interpretability of DL systems in the context of financial market prediction.

In summary, this study offers a solid foundation for further research on DL models in a financial environment. Future research can build upon these findings to explore new model configurations, alternative data sources, vectorisation methods, and methods to improve model interpretability.

Appendix A

Software Settings

Table A.1: R Packages

Packages	Versions
stargazer	5.2.3
ggplot2	3.3.6
data.table	1.14.2
qilibrary	0.1.3
zoo	1.8.10
R	4.0.2
stargazer	5.2.3
rstudioapi	0.14
magrittr	2.0.3
tidyselect	1.1.2
munsell	0.5.0
lattice	0.20.41
colorspace	2.0.3
R6	2.5.1
quadprog	1.5.8
rlang	1.0.6
fansi	1.0.3
dplyr	1.0.8
tools	4.0.2

Note: *Packages that are loaded indirectly via namespace

Table A.2: R Packages

Packages	Versions
xts	0.12.1
PerformanceAnalytics	2.0.4
grid	4.0.2
gtable	0.3.1
utf8	1.2.2
cli	3.4.1
withr	2.5.0
ellipsis	0.3.2
digest	0.6.29
tibble	3.1.6
lifecycle	1.0.2
farver	2.1.0
purrr	0.3.4
vctrs	0.4.1
glue	1.6.2
labeling	0.4.2
compiler	4.0.2
pillar	1.8.1
forcats	0.5.2
generics	0.1.3
scales	1.2.1
pkgconfig	2.0.3

Note: *Packages that are loaded indirectly via namespace

Table A.3: Python Packages

python_packages	python_versions
python	3.8.10
wandb	0.13.9
tensorflow	2.9.2
keras	2.9.0
data.table	1.0.0
pandas	1.3.5
numpy	1.21.6

Appendix B

Clean Data Statistics

Table B.1: Clean data statistics

Industry	No of stocks	Mean return	Standard deviation	Skewness	Kurtosis
Automotive Parts	12.93	0.13	0.96	-0.07	1.77
Banks	51.32	0.17	3.80	-0.24	3.43
Basic Resources	22.39	0.19	1.69	-0.65	5.02
Chemicals	19.82	0.20	1.09	-0.63	1.70
Construction Materials	25.40	0.26	1.53	-0.48	2.89
Financial Services	33.09	0.28	1.86	-0.44	1.21
Food & Beverages	29.17	0.29	1.07	-0.91	1.99
Health Care	40.22	0.41	1.74	-1.01	3.49
Industrial Goods & Services	91.30	0.88	4.82	-0.90	3.08
Insurance	32.83	0.28	2.00	-0.44	3.96
Media	21.82	0.11	1.34	-0.35	3.05
Oil & Gas	27.57	0.18	1.89	-1.19	6.26
Personal & Household Goods	41.61	0.39	1.75	-0.51	1.19
Real Estate	24.23	0.20	1.28	-0.85	6.84
Retail	14.45	0.06	0.89	-0.16	3.25
Technology	25.31	0.17	1.90	-1.08	8.73
Telecommunication	28.95	0.14	1.49	-0.82	7.78
Travel & Leisure	21.13	0.14	1.26	-0.49	1.87
Utilities	27.13	0.25	1.11	-0.80	1.57
All	590.67	0.25	1.76	-0.63	3.64

Appendix C

Long Short-term Memory Network Architecture by Fischer and Krauss

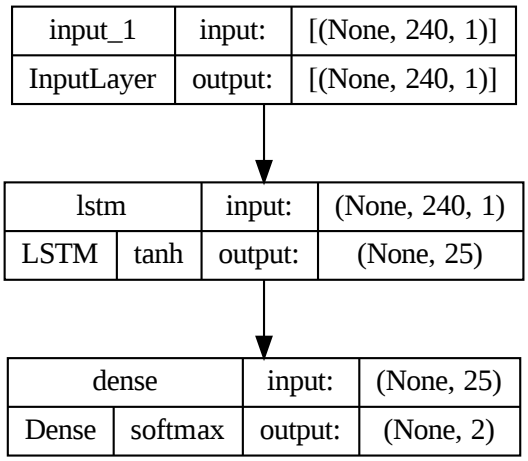

Figure C.1: Original LSTM architecture implementation by Fischer and Krauss using Keras [14]

Appendix D

Modified Long Short-term Memory Network Architecture

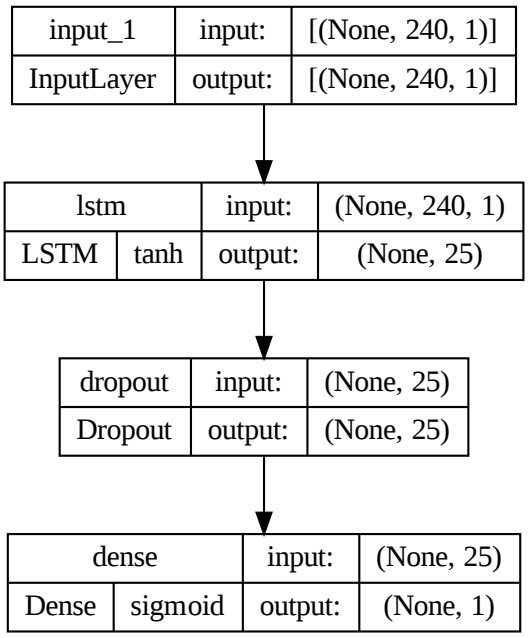

Figure D.1: Modified LSTM architecture implementation using Keras

Appendix E

Modified Long Short-term Memory with Time2Vec Network Architecture

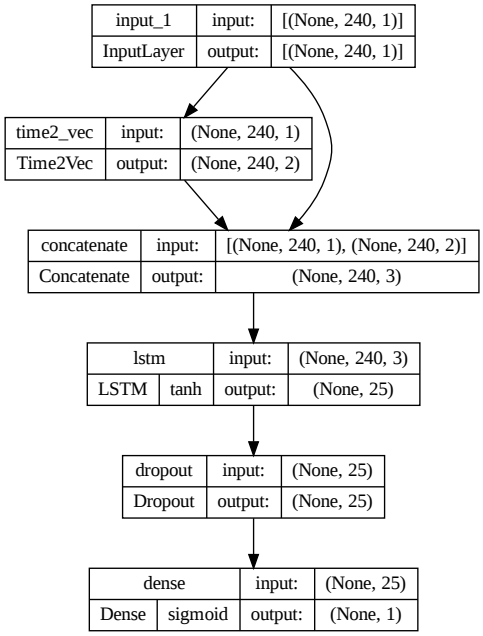

Figure E.1: Modified LSTM architecture implementation combined with the T2V layer using Keras

Appendix F

Modified Network Settings

Table F.1: LSTM and LSTM with T2V network architecture settings

Architecture settings	Configuration
Learning rate	0.005
Epochs	1000
Batch size	256
Loss function	Binary cross-entropy
Optimizer	Rmsprop
Recurrent dropout	0.0
Dropout	0.1
LSTM units	25
Early stop patience	10
Train split	0.8
Validation split	0.2
Sequence length	240
Number of features	1
Number of training years	3
Number of test years	1

Note: TBD

Bibliography

[1] Ai impacts.

[2] Bloomberg professional services.

[3] Yahoo finance - stock market live, quotes, business finance news.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[6] François Chollet et al. Keras. https://keras.io, 2015.

[7] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[8] Wilma De Groot, Joop Huij, and Weili Zhou. Another look at trading costs and short-term reversal profits. *Journal of Banking & Finance*, 36(2):371–382, 2012.

[9] Datatable Developers. Datatable, 2021.

[10] TensorFlow Developers. Tensorflow, May 2022.

[11] Matt Dowle and Arun Srinivasan. *data.table: Extension of `data.frame`*, 2021. R package version 1.14.2.

[12] Mirco Fabbri and Gianluca Moro. Dow jones trading with deep learning: The unreasonable effectiveness of recurrent neural networks. In *Data*, pages 142–153, 2018.

[13] Stefan Feuerriegel and Julius Gordon. Long-term stock index forecasting based on text mining of regulatory disclosures. *Decision Support Systems*, 112:88–97, 2018.

[14] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, 270(2):654–669, 2018.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[16] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1764–1772, Bejing, China, 22–24 Jun 2014. PMLR.

[17] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[19] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[20] Norbert Henze. *Stochastik für Einsteiger: eine Einführung in die faszinierende Welt des Zufalls*. Springer-Verlag, 2011.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Bryan D Jones. Bounded rationality. *Annual review of political science*, 2(1):297–321, 1999.

[23] Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5):1449–1475, 2003.

[24] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[25] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. Machine learning techniques and data for stock market forecasting: a literature review. *Expert Systems with Applications*, page 116659, 2022.

[26] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[27] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: learning a vector representation of time. *arXiv e-prints*, pages arXiv–1907, 2019.

[28] Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer. Deep learning for financial applications: A survey. *Applied Soft Computing*, 93:106384, 2020.

[29] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[30] Qontigo. Qontigo - stoxx europe 600. https://qontigo.com/index/SXXP/. Accessed: 2023-01-10.

[31] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[32] William F. Sharpe. The sharpe ratio. *The Journal of Portfolio Management*, 21(1):49–58, 1994.

[33] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[35] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. *Advances in neural information processing systems*, 28, 2015.

[36] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[37] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[38] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

[39] Zhichao Zou and Zihao Qu. Using lstm in stock prediction and quantitative trading. *CS230: Deep Learning, Winter*, pages 1–6, 2020.