UNIVERSITY OF TURKU
Turku School of Economics

Transformer-based deep learning model for stock return forecasting

Empirical evidence from US markets in 2012–2021

Master's thesis
in Accounting and Finance

Author:
Markus Päivärinta

Supervisor:
Prof. Luis Alvarez Esteban

7.6.2022
Turku

Master's thesis

Subject: Accounting and Finance
Author: Markus Päivärinta
Title: Transformer-based deep learning model for stock return forecasting: Empirical evidence from US markets in 2012–2021
Supervisor: Prof. Luis Alvarez Esteban
Number of pages: 86 pages + appendices 10 pages
Date: 7.6.2022

Abstract

A growing number of studies in recent years have deployed various machine learning methods for financial time series analysis. The ability of machine learning methods to deal with complex and nonlinear data sets, as well as the increasing amount of available data and computational capacity, has pushed research further in this direction. While machine learning methods are nowadays widely used for forecasting financial time series, the results have been mixed. The rapid increase in machine learning research has also meant that new and more advanced models are being developed all the time. In many areas where machine learning methods are employed, designs based on the Transformer deep learning model often represent the state-of-the-art. However, the applications of the Transformer model for financial tasks are still in their infancy as only a few studies have been published on the matter.

This study aims to investigate the feasibility of a Transformer-based deep learning model for stock return prediction. The feasibility is tested by predicting the daily directional movements of four different US stock indices on an out-of-sample period from the start of 2012 until the end of 2021. Only historical price data is utilized to predict the directional returns with two sets of explanatory variables. The model performance is tested against benchmarks and evaluated using various performance criteria such as prediction accuracy. Moreover, a trading strategy is carried out to reveal possible profitable attributes of the Transformer-based model.

The reported classification accuracy over the whole empirical sample for the better Transformer model is 52.52% while LSTM, another deep learning model used as a benchmark, achieves an accuracy of 53.87%. However, the Transformer model manages to defeat all the benchmark models in every other performance metric. When the performances are tested using the trading strategy, the best Transformer model is able to generate an annualized return of 15.7% before transaction costs. The best performing benchmark, a simple buy-and-hold strategy, yields a return of 14.2%. The two tested Transformer models also have the highest Sharpe ratios out of the tested models at 1.063 and 1.061. Nevertheless, after transaction costs are taken into account, none of the tested models beat a simple buy-and-hold strategy in terms of profitability.

Although the Transformer model was not able to perform superiorly throughout the sample period, it nevertheless exhibited increased predictive performance over shorter periods. For example, the model seemed to exploit periods of higher volatility as seen during the start of the COVID-19 pandemic. Overall, although the predictive performance of the Transformer model in this study might leave more to be desired, the model undoubtedly has predictive properties which should encourage further research to be executed.

Key words: machine learning, deep learning, stock market, asset return, forecasting

Pro gradu -tutkielma

Oppiaine: Laskentatoimi ja rahoitus
Tekijä: Markus Päivärinta
Otsikko: Transformer-pohjainen syväoppimismalli osaketuottojen ennustamiseen: Empiiristä näyttöä Yhdysvaltojen markkinoilta 2012–2021
Ohjaaja: Prof. Luis Alvarez Esteban
Sivumäärä: 86 sivua + liitteet 10 sivua
Päivämäärä: 7.6.2022

Tiivistelmä

Viime vuosina lisääntynyt määrä tutkimuksia on soveltanut koneoppimismenetelmiä rahoituksen aikasarja-analyyseissä. Koneoppimismenetelmien kyky käsitellä monimutkaisia ja epälineaarisia data-aineistoja, sekä lisääntynyt datan määrä ja laskentakapasiteetti ovat entisestään vauhdittaneet tutkimusta tällä alueella. Vaikka koneoppimismenetelmiä käytetään nykyisin laajalti rahoituksen aikasarjojen ennustamiseen, ovat niiden tuottamat tulokset olleet vaihtelevia. Koneoppimistutkimuksen nopea kasvu on myös tarkoittanut, että uusia ja kehittyneempiä malleja kehitetään kaiken aikaa. Monilla aloilla, joissa koneoppimista käytetään, alan johtavat mallit pohjautuvat usein Transformer-syväoppimismalliin. Transformer-pohjaisten mallien soveltaminen rahoituksen tehtäviin on kuitenkin vielä varhaisessa vaiheessa, sillä alalla on julkaistu vain muutamia tutkimuksia aiheesta.

Tämä tutkielma pyrkii selvittämään Transformer-pohjaisen mallin soveltuvuutta osaketuottojen ennustamiseen. Soveltuvuutta testataan ennustamalla neljän eri yhdysvaltalaisen osakeindeksin päivittäisiä suunnanmuutoksia vuoden 2012 alusta vuoden 2021 loppuun. Tuottojen suunnan ennustamisessa hyödynnetään vain historiallista hintadataa kahdella joukolla muuttujia. Mallin suorituskykyä testataan ja verrataan muihin käytettyihin malleihin monin eri suorituskykymittarein, kuten esimerkiksi ennustustarkkuuden avulla. Lisäksi toteutetaan kaupankäyntistrategia, jotta nähtäisiin mallin tuottamien ennusteiden mahdollinen taloudellinen hyöty.

Raportoitu ennustetarkkuus koko tutkimusotoksen ajalta oli paremmalla Transformer-mallilla 52,52%, kun sen sijaan vertailumallina käytetty LSTM-syväoppimismalli saavutti 53,87%:n ennustetarkkuuden. Kyseinen Transformer-malli onnistui kuitenkin suoriutumaan paremmin kuin vertailumallit kaikkien muiden suoritusmittareiden osalla. Kun mallien suoriutumista vertaillaan kaupankäyntistrategialla, paras Transformer-malli saavuttaa 15,7%:n vuosittaisen tuoton ennen kaupankäyntikustannuksia. Paras vertailukohta, yksinkertainen osta-ja-pidä-strategia tuottaa 14,2%:n tuoton. Kahdella testatulla Transformer-mallilla on myös korkeimmat Sharpen luvut: 1,063 ja 1,061. Kuitenkin, kun kaupankäyntikulut huomioidaan, yksikään testatuista malleista ei suoriudu osta-ja-pidä-strategiaa paremmin tuottojen osalta.

Vaikka Transformer-malli ei pystynyt suoriutumaan selvästi parhaiten läpi koko tutkimusotoksen, se esitti kasvanutta suorituskykyä lyhempinä aikoina. Malli näytti pystyvän esimerkiksi hyödyntämään korkean volatiliteetin ajanjaksoja, kuten COVID-19-pandemian alkuaikaa. Kaiken kaikkiaan, vaikka Transformer-mallin ennustuskyky tässä tutkielmassa saattaa jättää toivomisen varaa, Transformer-malli on epäilemättä kykeneväinen ennustustehtävissä, minkä tulisi edistää lisätutkimusten tekemistä aiheesta.

Avainsanat: koneoppiminen, syväoppiminen, osakemarkkinat, omaisuustuotto, ennustaminen

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

1.1 Background and motivation

Academic research in finance has for long been interested in finding useful information from a plethora of variables, e.g., fundamental and technical, that could help to explain variations in asset returns. The discoveries of such informative variables improve the understanding of underlying mechanisms and relations of the dependent variable. Another goal for academia and industry alike is to utilize these models to forecast the future values of an asset for financial gain. In the past, explanatory modeling has been the primary objective in multiple scientific disciplines and a focus on the predictive approach has often been regarded as unscientific. In the 2000s, predictive modeling has nevertheless gained increasing popularity and it has been shown to, e.g., help uncover complex relationships that newer larger datasets might hold. (Shmueli 2010.) The forecasting of values of financial securities is one of the most challenging issues due to the chaotic and inherently noisy nature of financial time series (Tay & Cao 2001). Although only a narrow degree of predictability can be expected, it can translate into considerable gains with the right tools.

The predictability of financial markets has been of interest to academics for over a century. Already in 1900, Louis Bachelier stated in his thesis that for a speculator of stock price movements the expectation should be zero and therefore the movements cannot be predicted (Courtault et al. 2000). Many early studies have supported the view that stock prices are not forecastable, i.e., they follow a random walk (Cowles 1933; Kendall & Hill 1953). Later, Fama (1970) introduced the efficient market hypothesis based on previous research on the topic. According to the theory, it should be impossible for an investor to generate abnormal returns in the long run. Thus, it should be useless to forecast any asset returns as they behave randomly. There have since been numerous claims for and against the hypothesis and it still remains as one of the most fundamental theories in finance. Financial time series analysis methods are usually utilized in tests of market efficiency and predictability.

The theoretical foundations of financial time series analysis are based on the concepts of linearity and stationarity. A process is said to be weakly stationary if its mean and covariance do not vary with time (Tsay 2010). The linear regression model is perhaps the most used in financial time series analysis, but it requires multiple assumptions to be met for it to provide correct and fitting results. These

assumptions are often too restrictive and it has been widely documented that financial time series can exhibit, e.g., nonlinear dependencies, and leptokurticity (Andreou et al. 2001). These characteristics have made the classical linear models less useful since they are unable to model the true relationships between variables. While many traditional nonlinear models, such as regime-switching models, have been developed to combat the complex characteristics of financial time series, machine learning methods have also been shown to work well with financial data.

Machine learning methods are by design meant to address complex, nonlinear, and noisy data which are all characteristics of financial time series (Grudnitski & Osburn 1993). Overall, machine learning methods have been implemented in financial prediction tasks since the 1980s, but their true potential has only been shown in the last decade. This is a result of the growing computational capabilities, which have allowed more complex models to be developed and fitted for financial time series tasks. According to a survey by Henrique et al. (2019), the most common machine learning methods used lately in financial research are support vector machines and artificial neural networks.

Artificial neural networks are a group of nonlinear models inspired by the nervous system and brains. They can, in theory, approximate any function arbitrarily well given a complex enough network. (Qi & Maddala 1999.) This result is also known as the universal approximation theorem. Neural networks act as a basis for a category of machine learning known as deep learning. The word *deep* comes from the usage of multi-level neural networks. While neural networks have perhaps had more success than some more traditional methods, there is a trade-off between interpretability and flexibility. Neural networks are usually almost impossible to interpret and hence they are generally referred to as black-box models (Géron 2019). Classical linear models, on the other hand, have great interpretability but lack the flexibility of neural networks.

Financial time series also show temporal dependencies that simple neural networks aren't able to model as they have no memory or sense of order. The order of the input is often important in many applications including financial time series. To overcome this hurdle, recurrent neural networks (RNN) were developed for natural language processing (NLP) purposes to memorize previous steps in a sequence. RNNs use hidden states that allow the model to calculate a weighted representation of previous inputs. Thus, the simple recurrent neural networks can be seen as an upgrade of a more simple neural network by using additional internal loops that act as a memory. However, the standard RNN model is only able to memorize a few previous steps of the input sequence. The estimation of

these models is also rather slow because the optimization of the network must be iterated through looped time steps. (Chollet 2017.)

Hochreiter & Schmidhuber (1997) developed a long short-term memory (LSTM) model to address the issues of standard RNN models. LSTM uses self-looped gates to better preserve the information of previous inputs. The model learns to decide how much of the old information it wants to preserve and forget using different gates. Although the LSTM model can capture longer dependencies than RNN, it still has difficulties capturing very long-range dependencies. It also uses the same optimization as the standard RNN, which makes the model slow to be estimated and optimized. Despite these shortcomings, according to a survey by Ozbayoglu et al. (2020), the LSTM model has been the most studied deep learning model in financial applications and most of the studies have been conducted in the past five years. The LSTM and its variants can perhaps be considered the current state-of-the-art method in financial prediction tasks.

The latest improvement in sequence modeling, the Transformer model, was developed by Vaswani et al. (2017) for natural language processing tasks. Unlike RNNs that treat the input as a sequence, Transformer treats the input as a set. This allows for the calculation of gradients to be parallelized which makes the estimation of such a network considerably faster. In some financial prediction tasks, the speed of the networks' fitting and prediction processes can play an important part in its success. Another difference between RNNs and Transformers is the capturing of long-term dependencies. While RNNs can either forget or retain information from earlier points in a sequential manner, Transformer networks can attend to all previous inputs and weight them simultaneously using a mechanism called attention. (Géron 2019.)

Since the release of the original article, Transformer-based architectures have become the state-of-the-art method in many fields such as machine translation and speech recognition, and it has even challenged convolutional neural networks in image recognition (Dosovitskiy et al. 2020). Different forms of the Transformer architecture have also been developed to tackle time series problems (Wu et al. 2020; Zhou et al. 2021). Despite their success in various fields, Transformer-based models have yet to be recognized by the financial research community as the Transformer has been applied in financial context only on a few occasions.

Ding et al. (2020) were possibly the first to provide an implementation of a Transformer-based model to a stock return prediction task and showed great promise as it was able to achieve higher performance than an LSTM model for example. Yoo et al. (2021), on the other hand, predicted six different stock indices using their version of the Transformer, and also find their model superior to other

methods. Moreover, Zhang et al. (2022) show their version of the Transformer to defeat most of the benchmark models in four different datasets. In addition, stock price volatilities have also been studied on a few occasions with Transformer-based models (Lim et al. 2021; Ramos-Pérez et al. 2021). Given the novelty value and the supportive results, further research into this area should be conducted.

1.2 Objectives and structure

As this area of research is still extremely new and lacks comprehensive research, the goal of this thesis is to build a Transformer-based deep learning model and evaluate its predictive performance on financial time series. The results of an empirical analysis will be discussed and analyzed in detail. The model performance is also compared to benchmark methods from the existing literature. The predictions are made with and without technical indicators for each model. Additionally, a simple trading strategy will be implemented to showcase the economic significance and profitability of the models. Another goal is to be able to explain some of the reasons behind the predictions of the deep learning model.

Hence the research questions of the thesis are:

- Is the Transformer-based deep learning model a viable tool for financial time series prediction?
- Does the model achieve higher performance compared to the benchmark models?
- Do the model predictions contribute to any considerable financial gains over other methods?
- Does the inclusion of technical indicators help the model performance?
- Can the black box decisions of the model be interpreted to some degree?

The empirical analysis in the study is conducted using four different stock indices for forecasting: the S&P 500, Nasdaq Composite, Dow Jones Industrial Average, and Russell 2000. These indices are one the most used in financial prediction and some of them could perhaps be considered benchmark data sets when testing different models. The goal of the empirical study is to forecast one-step ahead directional return, i.e., positive or negative, of each index given earlier realizations of chosen variables. The chosen explanatory variables for each index are the opening price, highest price, lowest price, adjusted close price, and trading

volume. In addition, another extended model including a set of technical variables is used to see if the addition of variables can generate even better performance, as the Transformer architecture should benefit from the addition of auxiliary variables. The out-of-sample forecast period is a 10-year window from the start of 2012 to the end of 2021. In this research, the daily frequencies of the variables are used.

Along with the proposed model, other models are also used as a benchmark in performance evaluation. The benchmark models used in this empirical analysis are a logistic regression classifier and an LSTM model. Both models are also using the five main variables as well as the extended model with additional technical indicators. Since the predicted variable is the directional return, the prediction is essentially a binary classification. The models are thus compared using several performance metrics suited for classification, such as accuracy. Additionally, a trading strategy is implemented to find out whether the predictions can be translated into profitable signals. It is also attempted to reveal some possible explainability behind the decisions of the deep learning model.

The rest of this thesis is organized as follows. Section 2 focuses on the theoretical framework of the study. First, it covers concepts such as return predictability and financial time series analysis. Then it introduces the necessary machine learning structures and the development of neural network models for prediction tasks. The rest of the chapter is devoted to the analysis of previous research on financial time series in a machine learning context. Next in Section 3, the research methods are introduced. These include, e.g., the description and the preprocessing of the data, the proposed variant of the original Transformer model used in this research as well as the metrics that are used to evaluate the performance of the models. Section 4 includes the analysis and interpretation of the results. Finally, Section 5 concludes and considers possible future research implementations.

The deep learning models are developed using Python 3.7.13 within Google Colaboratory, which provides the user with an external GPU for free that alleviates the computational burden. The deep learning models, LSTM and Transformer, are developed within the Keras library that runs on Tensorflow 2.8.0 (Chollet et al. 2015; Abadi et al. 2016). The logistic regression is conducted using the scikit-learn library (Pedregosa et al. 2011). The scikit-learn library is also used to compute the performance metrics that are used to evaluate the models. The technical indicators are constructed with a pandas-ta library (Johnson 2021). Most of the visualizations and some of the data processing are done within Microsoft Excel and the rest in Python. A simplified and shortened version of the code used in this thesis is displayed in Appendix V.

2 THEORETICAL FRAMEWORK

2.1 Financial economics

2.1.1 Stock market predictability

The predictability of financial markets has always been tightly linked to the concept of efficient markets. While the market efficiency and predictability had been studied before, Fama (1970) formalized the theory of efficient markets by redefining and expanding the previous framework. The efficient market hypothesis (EMH) states that financial markets, where the prices fully reflect all available information at any point in time, are considered efficient. Consequently, the security prices also adjust to any new information instantly. The idea behind the efficient market hypothesis is closely related to the random walk hypothesis, which claims that the future price change of an asset is random and therefore unpredictable (Shiller 2000).

According to Fama (1970), the empirical research on the informational efficiency of the markets can be classified into three different levels of available information: weak, semi-strong and strong form tests. Most studies on the matter are focused on weak, and semi-strong form efficiency. For the weak form efficiency to be true, it requires the price of an asset to reflect all relevant information of the asset's trade history, e.g., historical prices, short interest, and trading volume (Bodie et al. 2003). Semi-strong form efficiency of the markets also includes other publicly available information, e.g., stock splits or announcement of earnings to be reflected in the price. Therefore, not even the use of fundamental analysis should help to generate abnormal returns. Strong form efficiency asserts that all public and private information is included in the current price. Hence, even people with inside information cannot systematically generate higher risk-adjusted returns. It also follows that if the markets are efficient, a market participant cannot systematically generate risk-adjusted excess returns on the market using the available information. (Knüpfer & Puttonen 2018.)

The EMH is one of the most debated theories in finance and it has had a lot of advocates and critics. Despite numerous attempts on both sides, no clear consensus has been reached. One notable problem is that the test of EMH is always a joint hypothesis between efficiency and an asset pricing model capturing investors' risk preferences. If any evidence against efficient markets can be found, it can always be argued that the asset pricing model is not correctly specified.

(Fama 1991.) Although the efficiency of the markets is not testable per se, the predictability of the markets can indeed be tested. These tests can also be formed based on the same three informational efficiencies depending on the explanatory variables used.

The academic literature has identified hundreds of possible explanatory variables that can potentially explain the price variations of financial assets. However, as Welch & Goyal (2008) find in their study, many of these variables that were found to be a good fit in the literature were only tested with in-sample data in their respective studies. When these variables were introduced to new out-of-sample data, the variables lost their statistical significance and thus lacked any generality. In fact, in the latter part of their research period, no model had superior performance out-of-sample against the benchmark of the prevailing mean, and most models were inferior even in-sample. Rapach & Zhou (2013) later point out that more recent studies that use more advanced strategies, e.g., diffusion indices and regime shifts, have shown statistical and economic significance even in out-of-sample. These variables however were mostly using semi-strong efficient information.

Some challenging arguments against efficiency and predictability have been demonstrated by various pricing irregularities that only utilize weak form information, e.g., momentum, mean reversion, and calendar anomalies (De Bondt & Thaler 1985; Haugen & Jorion 1996; Jegadeesh & Titman 1993). These anomalies are usually interpreted and explained using theories of behavioral finance such as bandwagon and overreaction effects. Although, Fama (1998) argues that these irregularities are consistent with EMH since the overreactions and underreactions seem to be as common and are thus the result of chance. Malkiel (2003) also reviews some earlier studies on these anomalies and argues that the markets are a lot less predictable and more efficient than many studies have claimed. He notes that while momentum can be statistically significant, its economical significance is questionable and earlier studies have not beaten a buy-and-hold strategy when transaction costs are taken into account. Similar arguments are also shown against mean reversion and January anomalies. He also argues that the problem with predictable patterns is that they are not dependable through time which makes them hard to be exploited. Some of these anomalies, and other found predictors alike, have also more or less disappeared soon after they have been published (Gu 2003; McLean & Pontiff 2016).

One possible way to improve the weak form predictability of the markets is to utilize a set of tools known as technical indicators. Technical analysis methods use certain rules to anticipate future movements based on historical price data.

Relative strength index (RSI) and moving average (MA) are examples of these methods. While these methods are largely used in the industry, they have faced a lot of criticism from scholars and some have even called it 'voodoo finance' (Lo et al. 2000). Despite the status of technical analysis amongst some academics, a lot of studies have focused on technical trading strategies.

Park & Irwin (2007) review 95 studies that use technical strategies and record positive results in 56 of these. However, they point out several deficiencies, e.g., data snooping, that these studies are subject to and thus the results are inconclusive. Hsu et al. (2010) study 16,380 trading rules for three different US indices and six emerging market indices. Before the US indices had exchange-traded funds (ETF) following them, 269 rules were statistically significant in their mean return. For the post-ETF period in the three US indices, no trading rules proved to be significant. After ETFs were introduced to the studied emerging markets, in two out of the six markets significant rules were still detected. On the contrary, Neely et al. (2014) find that technical indicators match or even exceed the predictive power of macroeconomic variables in the US markets. Authors point out that the technical indicators exhibit greater predictive power, especially near peaks of business cycles.

Another less studied approach to overall return predictability is to forecast the sign of the movement instead of the exact magnitude. Leitch & Tanner (1991) are one of the first to point out that accurately forecasting the direction of an asset return is in fact related to profitability, unlike any conventional point estimation method. Leung et al. (2000), on the other hand, are one of the first to empirically examine the predictability of directional asset returns using probability-based classification approaches such as logit models and neural networks. Their findings support the previous literature that models based on classification, instead of level estimation, outperform both in terms of maximizing the returns from trading, as well as in prediction accuracy of the return sign. They also argue that the idea of minimizing the forecast error in point estimation models does not necessarily contribute to economic significance, and classification-based trading rules might improve capital gains. This is also rational from an individual investor's point of view since an investment decision is also a binary decision in real life – either an investment decision, buy or sell, is made or not.

Furthermore, Christoffersen & Diebold (2006) prove theoretically that asset returns can display conditional sign dependence even if the returns have no conditional mean dependence. This is possible if the directional return correlates with the conditional volatility dependence. Mathematically, an asset return R_{t+1} exhibits conditional mean dependence if $\mathbb{E}[R_{t+1}|\Omega_t]$ alters with the information

set Ω_t. In other words, the return has a time-varying mean. Conditional directional dependence or conditional sign dependence on the other hand is true if $\mathbb{E}[\mathbb{1}_{[0,\infty)}(R_{t+1})|\Omega_t]$ varies with Ω_t. Lastly, conditional volatility dependence can be formulated as $\sigma^2_{t+1|t} = \mathrm{Var}(R_{t+1}|\Omega_t)$. Assuming the returns of an asset are distributed as

$$R_{t+1}|\Omega_t \sim N(\mu, \sigma^2_{t+1|t}),$$

the probability for an upside movement is

$$\mathbb{E}\big[\mathbb{1}_{[0,\infty)}(R_{t+1})|\Omega_t\big] = 1 - \mathbb{P}[R_{t+1} < 0|\Omega_t] = \Phi\left(\frac{\mu}{\sigma_{t+1|t}}\right),$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. While this distribution is symmetric and concentrated around the constant conditional mean, the return sign is forecastable if the probability is time-varying and above 0.5 if the conditional mean μ is positive. Thus, there is a reverse relationship between the sign and volatility of the return: when volatility increases, the probability of a positive return gets smaller and vice versa. Nevertheless, if the expected return is zero, then the sign of the return would be unforecastable, as well as if the volatility of the asset return is constant. These results are also true for non-Gaussian distributions, and therefore the direction of asset returns is forecastable as long as the volatility is dynamic, i.e., time-varying, and the mean return is non-zero. Christoffersen et al. (2007) also point out that even if the mean return is zero, possible variation in higher moments might introduce directional predictability if the distribution is asymmetric.

The predictability of the directional return should be most prominent in monthly and weekly frequencies and less so in daily or annual frequencies (Christoffersen & Diebold 2006). Nevertheless, some previous studies have found directional predictability in daily asset returns. For example, Bekiros & Georgoutsos (2007) study the daily predictability of the Nasdaq and Nikkei indices and report a directional accuracy of 54.3% in the Nasdaq index. However, the accuracies were significantly below 50% for the Nikkei index. Skabar (2013), on the other hand, reports overall daily accuracies of 52.5% when comparing multiple models over a 20-year forecasting period. Furthermore, Zhong & Enke (2019) utilize hybrid machine learning algorithms using 60 different explanatory variables and report daily directional accuracies in quite a high range of 54.9%–57.5%.

Pesaran & Timmermann (1995) examine the predictability of US stock returns from 1954 until 1992 and find that the predictability of the stock market is fairly small during periods of low volatility. In comparison, the predictability of stock returns increased during higher periods of volatility, and they note that the pre-

dictability might be more distinct during regime switches in the market. Similarly, Krauss et al. (2017) show that during the global financial crisis of 2008–2009, the tested models are able to generate Sharpe ratios as high as 4.46 after transaction costs following a specific trading strategy. They point out that especially long-short strategies should be able to exploit this phenomenon. Fischer & Krauss (2018) consider two possible explanations for the strong performance during financial crises based on previous literature. Firstly, they argue that it is fair to assume that investors just get disoriented during market turmoil which creates arbitrage opportunities. Secondly, limits to arbitrage are extraordinarily high so possible arbitrage opportunities created by the first reason can't possibly be exploited, or short-selling might even be banned overall.

2.1.2 Financial time series analysis

Financial time series analysis often focuses on valuing assets over time from both theoretical and empirical aspects. The most fundamental theories in financial time series analysis are based on the assumptions of linearity and stationarity of the data generating process. As many of the financial time series exhibit nonlinear and otherwise complex behaviour, more advanced models could also be used. While more advanced statistical models have been developed, another alternative is to use machine learning methods instead.

In time series analysis, a time series of a dependent variable y can be expressed as $\{y_t\}_{t\in T}$, where T is the number of observations. In financial forecasting, the future values of the dependent variable are usually estimated using a series of past observations of one or multiple explanatory variables, $\{x_{it}\}_{t\in T}$. A general formula for forecasting the future values of a given dependent variable y_t can be represented as

$$y_{t+\tau} = F(\mathbf{X}; \boldsymbol{\theta}|\Omega_t) + \epsilon_{t+\tau}, \qquad t = 1, ..., T, \tag{1}$$

where $F(\cdot)$ represents any arbitrary linear or nonlinear function used to model the time series, $\mathbf{X}$ is a set of explanatory variables at given time intervals, $\boldsymbol{\theta}$ is the corresponding parameter vector, τ represents the forecast horizon, ϵ is the random disturbance and Ω_t is the information set at time t. Quite often the interest is only in one-step ahead predictions where $\tau = 1$. If for example $\mathbf{X}$ only includes $T + 1$ lagged values of the dependent variable itself, the prediction function illustrated in Equation 1 can be expressed as

$$y_{t+1} = F(y_t, y_{t-1}, ..., y_{t-T}; \boldsymbol{\theta}) + \epsilon_{t+1}. \tag{2}$$

As acknowledged earlier, the stationarity of the time series is often a necessary condition for traditional models. Stationary time series processes can be divided into two groups: weak and strong stationarity. However, the strong assumptions are difficult to test empirically and therefore financial literature is usually interested only in weak stationarity (Tsay 2010). Therefore, here stationarity always implies weak stationarity. A time series process Y_t is then said to be stationary if it has the following characteristics:

$$\begin{aligned} \mathbb{E}[Y_t] &= \mu \qquad \text{for all } t \\ \mathrm{Cov}(Y_t, Y_{t-s}) &= \gamma_s \qquad \text{for all } s, t, \end{aligned}$$

where γ_s denotes the autocovariance function of the process with lag s. A time series process is therefore stationary if its first and second moments are finite and time-invariant. In other words, if the expected return μ is constant through time and the covariance function is not dependent on s and t, but on their relative distance $t - s$, the process is weakly stationary. Also if $s = t$, the covariance is the variance of the process $\mathrm{Var}(Y_t)$. The downside of these assumptions is that financial time series often exhibit time-varying means and thus they are not always stationary. Nonstationary time series can nevertheless usually be transformed to stationary forms through transformation, such as taking a first difference of the values. (Tsay & Chen 2019.)

On top of stationarity, classical time series models often assume a linear relationship between the dependent and independent variables. Probably the most common form of the model function $F(\cdot)$ is the linear regression model

$$F(\mathbf{X}; \boldsymbol{\theta} | \Omega_t) = \beta_0 + \boldsymbol{x_t'}\boldsymbol{\beta},$$

where $\boldsymbol{x_t}$ is a vector of explanatory variables, $\boldsymbol{\beta}$ is the corresponding parameter vector and β_0 is a constant. One very simple and popular approach that utilizes this linear form is the family of autoregressive moving average (ARMA) models. In fact, applying the linear model as the function $F(\cdot)$ in Equation 2 yields an autoregressive AR(p) model, where $p = T + 1$ as it includes $T + 1$ previous observations of its time series.

The assumptions of these traditional models are nevertheless rather restrictive and they might not be able to uncover the possibly complex relationships between variables. To tackle these issues, some more sophisticated traditional models have been developed, such as the threshold autoregressive model (TAR) which introduces nonlinear regime switches. The model uses two or more AR models and utilizes one model at a given time step depending on which regime the data comes

from. As the model introduces the idea of separate regimes it allows for some variability in the mean of the time series. (Tsay 2010.)

After all, most of these traditional models are developed to estimate continuous dependent variables that have always been the dominant area of research. Yet, the focus of this thesis is to forecast a qualitative response variable y. In the case of a directional return of an asset, we can use the following notation to introduce the binary response variable to be estimated as

$$y_t = \begin{cases} 1, & \text{if } R_t > 0, \\ 0, & \text{if } R_t \leq 0, \end{cases}$$

where R is the simple return of an asset. In these types of applications, the use of linear models is not encouraged. Instead, when the dependent variable has discrete values, qualitative response models should be used since ultimately the task is to estimate the conditional probability of a binary outcome (Horowitz & Savin 2001). It is also shown by Leung et al. (2000) that probability-based models are superior to linear models in this regard.

These qualitative response variables also possess different characteristics than the commonly used continuous variables. A binary-valued dependent variable conditional on prior information can be shown to have a Bernoulli distribution

$$y_{t+1}|\Omega_t \sim \text{Ber}(p_{t+1}),$$

where p_t is the conditional probability and thus

$$p_{t+1} = \mathbb{E}(y_{t+1}|\Omega_t) = \mathbb{P}(y_{t+1} = 1|\Omega_t).$$

The conditional probability, p_t, can be modeled using a linear model π_t which is linked to the conditional probability via $p_t = G(\pi_t)$, where $G(\cdot)$ is a link function. The linear function π_t is usually of the form

$$\pi_t = \omega + \boldsymbol{x'\theta},$$

where ω is the constant term, $\boldsymbol{x}$ is a set of explanatory variables and $\boldsymbol{\theta}$ includes the corresponding parameters. This type of generalized linear model allows for linear representations of the data to be implemented in a probabilistic environment. The simplest probability model, the linear probability model, uses an identity link function, and thus the probability can be expressed as $G(\pi_t) = \pi_t$. This model has theoretical disadvantages because the modeled function π_t has no constraints to be in the probability range of [0,1]. While it can be corrected by fixing the values to $F = 1$ if $F(\pi_t) > 1$ and $F = 0$ if $F(\pi_t < 0)$, it introduces unrealistic kinks

at the truncation points. Thus, this model is often not recommended to be used. (Amemiya 1981.)

Probit and logit models are examples of traditional generalized linear models that are used in financial time series analysis. In these cases, the link function $G(\cdot)$ is the cumulative distribution function of the standard normal distribution or the logistic distribution, respectively. Both of these models are identical at $\pi = 0$ and thus yield similar results in binary classification tasks. There are slight differences in the models, as the coefficients of the logit model are approximately 1.6 times larger than the ones in probit. (Agresti 2015.) This does not matter in pure directional prediction tasks where the coefficients' interpretations aren't the main focus. In terms of their provided probabilities, a static probit model for instance can be expressed as

$$p_t = \Phi(\pi_t) = \int_{-\infty}^{\pi_t} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) dx.$$

Alternatively, we can express the logit model in terms of probabilities, when $\Phi(\cdot)$ is replaced with the logistic function as follows

$$p_t = \Lambda(\pi_t) = \frac{1}{1 + \exp(-\pi_t)} = \frac{\exp(\pi_t)}{1 + \exp(\pi_t)}. \tag{3}$$

Financial time series analysis has mostly utilized binary response models to predict recessions or similar occasional events. Nonetheless, a few studies focusing on stock market returns have been conducted with promising results (Nyberg 2011; Chevapatrakul 2013; Nyberg & Pönkä 2016). Even so, the use of traditional statistical models for directional return prediction has not received wider popularity. Part of the reason might be the more complex interpretation of the model coefficients for causal inference.

Nonetheless, the predictive modeling research has seen a rise of multiple different machine learning methods tackling the question of directional predictability in financial time series. Previous research work that utilizes machine learning methods for stock return prediction is analyzed in Section 2.3. These methods include, e.g., support vector machines, k-nearest neighbors, decision trees, random forests, and neural networks. While numerous machine learning techniques have been accepted as proper alternatives to traditional statistical models, the evidence of their performance has been mixed.

For instance, Makridakis et al. (2018b) compare the out-of-sample accuracy of statistical models such as ARIMA models against different machine learning models like k-nearest neighbors and LSTMs. The data in the article is from Makrida-

kis's M-3 competition of which 25% is finance-related data. The Makridakis competitions are famous time series forecasting contests of which the first was held in 1982 (Makridakis et al. 1982). Makridakis et al. (2018b) observe traditional statistical models to be superior in all forecasting horizons and accuracy measures in the M-3 competition. The authors also encourage future machine learning research to incorporate more traditional statistical models as benchmarks in order to show more realistic results of the capabilities of these methods. Opposingly, some researchers like Siami-Namini et al. (2018) have shown LSTMs to have preferable results against ARIMA models.

Next Makridakis competition, the M-4, also saw similar trends as most of the top models continued to have traditional statistical qualities. All six models submitted to the competition that were purely based on machine learning methods performed poorly. The best model however was based partly on recurrent neural networks. (Makridakis et al. 2018a.) The most recent M-5 competition was split between accuracy and uncertainty forecasting. In both competitions, four out of the top five methods utilized a machine learning method called LightGBM. The other top method in the accuracy challenge placed third and it used a combination of 43 deep learning models combining LSTM layers. Similarly, in the fourth place in the uncertainty challenge was an LSTM-based neural network. (Makridakis et al. 2021a; Makridakis et al. 2021b.)

2.2 Neural network models

2.2.1 Feed-forward networks

Neural networks are gaining interest in multiple scientific areas, including financial academic research. Neural networks are a great tool to model complex and nonlinear dependencies between variables. This makes them very suitable for financial modeling tasks since financial data is complex, dynamic, and often nonlinear in its nature. In addition, neural networks allow for an analysis of financial time series without the restrictions of traditional statistical methods. (Nagel 2021.)

The perceptron, introduced by Frank Rosenblatt in 1958, was the first algorithmically represented neural network, and still today it is the cornerstone of most modern neural networks. The following presentation of a perceptron loosely follows the description of Haykin (1999). A basic perceptron is composed of three main elements: synapses or connecting links, adders, and activation functions.

Synapses represent the connections between two different neurons, or nodes, each of which has its own synaptic weight, or connection strength. The term adder symbolizes the linear combination between the inputs and their synaptic weights. Activation functions are used to limit the output amplitude and they are also the only nonlinear part of a neural network. Mathematically, we can formulate the basic perceptron as

$$\mathbf{y} = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi(\mathbf{w}'\mathbf{x} + b)\,,$$

where $\varphi(\cdot)$ is usually a nonlinear activation function, $\boldsymbol{w}$ is the weight vector i.e. the synaptic weights, $\boldsymbol{x}$ is the input vector, $\boldsymbol{y}$ is the output vector and b is the bias. The part inside the activation function, $\varphi(z)$, is known as the activation or activation potential z. The corresponding computational graph of a perceptron is shown in Figure 1. The perceptron can also be represented in form of the general forecast function in Equation 1, where the function $F(\cdot)$ is replaced with $\varphi(\cdot)$, and the parameter vector $\boldsymbol{\theta}$ includes the estimated parameter weights $\boldsymbol{w}$ and the weight of the bias term b. The bias is used in neural networks to apply an affine transformation to the linear combination that is similar to a constant in linear functions, which hopefully allows the model to have a better performance.

Originally, the perceptron was used with an activation function called the Heaviside step function, $\varphi(z) = \mathbb{1}_{(0,\infty)}(z)$, that tried to imitate the human brain by simply either firing or not firing a specific neuron. Nowadays, many more advanced activation functions are common in practice. The choice of an activation function is dependent on the problem and the model you are solving it with.

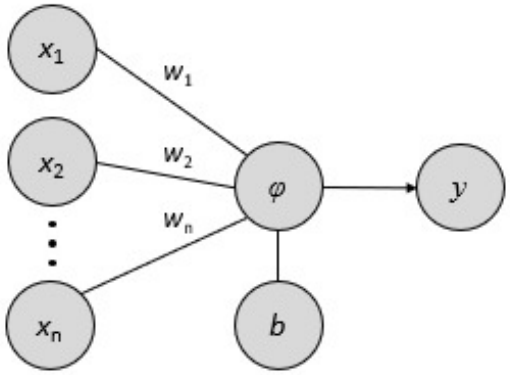

Figure 1: Computational graph of a single-layer perceptron

Some of the most common activation functions, $\varphi(z)$, are the following:

$$\begin{aligned} \sigma(z) &= \frac{1}{1+e^{-z}} \quad \text{(sigmoid)} \\ \text{softmax}(z_i) &= \frac{e^{z_i}}{\sum_j e^{z_j}} \quad \text{(softmax)} \\ \tanh(z) &= \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad \text{(hyperbolic tangent)} \\ \text{ReLU}(z) &= \max(0, z) \quad \text{(rectified linear unit).} \end{aligned} \tag{4}$$

The three first activation functions in Equation 4 are all from the same exponential family. The logistic sigmoid function is one of the most commonly used activation functions. When replacing the $\varphi(\cdot)$ with a sigmoid $\sigma(\cdot)$, it is equivalent to the logistic regression formulated in Equation 3. Consequently, logistic regression can also be thought of as a machine learning approach, which is how it is used in this thesis. Because the output range of values provided by the sigmoid is [0,1], it is often used as an activation function in the output layer of binary classification tasks, i.e., $\mathbb{P}(y = 1|x)$. If it is used as an activation function in the inner layers of a network, the function might saturate when its inputs are highly positive or negative, and thus the function becomes insensitive to minor changes (Goodfellow et al. 2016). The generalization of the sigmoid is the softmax function. It can be used to solve multi-class classification tasks, i.e., $\mathbb{P}(y = i|x)$. The resulting vector gives out the probabilities to each class i in the task.

The hyperbolic tangent is another very commonly used activation function. The hyperbolic tangent is just a scaled and shifted form of the sigmoid function as $\tanh(z) = 2\sigma(2z) - 1$. The hyperbolic tangent usually works better than the sigmoid and does not saturate as easily inside a network. It also behaves quite linearly near the origin, which makes the model estimation easier. Rectified linear unit (ReLU) in the fourth formula is the most common type of activation function nowadays. ReLUs are piecewise linear functions and therefore very easy to optimize by using differentials. Piecewise linearity means that it is partly linear. In the case of ReLU, the function is linear when the input is positive and zero otherwise. This makes the optimization through gradient-based methods a lot easier. However, if the activation z goes to zero at any point, the model cannot use those observations in the estimation anymore. Many modifications to the original ReLU have been implemented to allow the model to keep optimizing the values even when the input is negative. (Goodfellow et al. 2016.)

While the perceptron is the basis for most neural networks, a single-layer perceptron is not very useful as it can only act as a linear binary classifier. However,

by using a stack of perceptrons a lot more complex structures can be achieved. In addition to input and output layers, a multilayer perceptron (MLP) has one or more hidden layers in between. MLP is also known as a feed-forward neural network (FNN). MLP with one hidden layer can be expressed mathematically as

$$\mathbf{h} = \varphi\big(\mathbf{W}^x\mathbf{x} + \mathbf{b}^h\big)$$
$$\mathbf{y} = \mathbf{W}^h\mathbf{h} + \mathbf{b}^y,$$

where $\mathbf{W}^x \in \mathbb{R}^{n\times T}$ is the weight matrix used to map the inputs to the hidden layer via a linear transformation, $\mathbf{W}^h \in \mathbb{R}^{o\times n}$ is the weight matrix used to map the hidden layer to the output layer, $\mathbf{b}$ is the bias vector and $\varphi(\cdot)$ is an arbitrary activation function. The n and o respectively denote the number of nodes in the hidden and output layer, and T is the length of the input vector. Also, models, where every neuron of one layer is connected with every neuron of another are called fully-connected layers, or dense layers.

The weight matrices in MLPs are often initialized randomly with values within the range (0,1). More efficiently, the weight parameters of the network can be, for example, sampled from the uniform distribution $U(-1/T^{1/2}, 1/T^{1/2})$ capping the maximum total size of weights to 1. (Marsland 2015.) Another commonly applied parameter initialization strategy is to use the normalized initialization, also known as Glorot initialization, developed by Glorot & Bengio (2010).

Artificial neural networks, such as MLPs, are usually fitted and optimized by minimizing a loss function $J(\boldsymbol{\theta})$ via a method called gradient descent. The loss function used in binary classification tasks is the binary cross-entropy loss

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\widehat{y_i}) + (1 - y_i)\log(1 - \widehat{y_i})),$$

where $\widehat{y_i}$ is the predicted output, y_i is the actual output, and N is the number of predictions. The minimization of the cross-entropy loss, or equivalently, the negative of the log-likelihood, can also be interpreted as a maximum likelihood estimation (Goodfellow et al. 2016). The most popular way to calculate the gradient $\boldsymbol{g}$ of the loss function, also notated as $\nabla J(\boldsymbol{\theta})$, is to use the backpropagation technique. Backpropagation is the use of repeated chain rules of partial derivatives. The goal is to differentiate the loss function with respect to the weights of the model so that the parameter weights could be optimized. (Graves 2012.)

Firstly, for every node j in a layer h, the output is defined here as

$$o_j^h = \varphi\big(z_j^h\big) = \varphi\left(\sum_{i=1}^{n} w_{ij}^h o_i^{h-1}\right),$$

where o_i^{h-1} are the outputs from the previous layer, w_{ij}^h is the synaptic weight from i to j, z_j^h is the activation potential and φ is an arbitrary activation function.

Backpropagation uses the chain rule as follows

$$g = \frac{\partial J}{\partial w_{ij}^h} = \frac{\partial J}{\partial o_j^h} \frac{\partial o_j^h}{\partial z_j^h} \frac{\partial z_j^h}{\partial w_{ij}^h}, \tag{5}$$

where g is the gradient and J is the chosen loss function. Redefining parts of the formula as

$$\delta_j^h = \frac{\partial J}{\partial o_j^h} \frac{\partial o_j^h}{\partial z_j^h}$$
$$o_i^{h-1} = \frac{\partial z_j^h}{\partial w_{ij}^h}$$

will allow the Equation 5 to be expressed as

$$\frac{\partial J}{\partial w_{ij}^h} = \delta_j^h o_i^{h-1}.$$

All the values of δ_j^h can then be recursively iterated for each layer H by

$$\delta_j^h = \varphi'\big(z_j^h\big) \sum_{k \in H+1} w_{jk}^h \delta_k^{h+1},$$

where $\varphi'(\cdot)$ is the derivative of the activation function. Thus, the weighting formula update can be calculated from values of δ and multiplied by the output of the previous layers o^{h-1}. The obtained loss gradients of the weights are then stored into a gradient vector $\boldsymbol{g}$. It is then used to update the parameter weights of the network layers as follows

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \boldsymbol{g},$$

where η is an adjustable learning rate. There are numerous different methods to improve the optimization process of these neural networks. For example, the learning rates can be constant, decaying or cyclical, and even have certain restarts that improve the optimization. Similarly, different adaptive learning rate optimization algorithms, such as RMSprop or Adam, are often employed in model training, i.e., model estimation. (Goodfellow et al. 2016.)

2.2.2 Recurrent neural networks

Recurrent neural networks (RNN) are neural networks that use internal feedback loops to retain information from previous inputs. This kind of parameter sharing, using earlier outputs as the next inputs, is the core reason for RNN's ability to

generalize across time. (Goodfellow et al. 2016.) This idea can also be seen from the mathematical formulation of the network:

$$\mathbf{h}_t = \varphi\big(\mathbf{W}^x\mathbf{x}_t + \mathbf{W}^h\mathbf{h}_{t-1} + \mathbf{b}^h\big)$$
$$\mathbf{y}_t = \mathbf{W}^y\mathbf{h}_t + \boldsymbol{b}^y,$$

where $\mathbf{W}^x$ is the weight matrix applied to the input, $\mathbf{W}^h$ is the weight matrix used to map values from the previous hidden layer to the next one and $\mathbf{W}^y$ is the weight matrix that maps the values of the current hidden layer to the output layer. The unfolded recurrent structure of the RNN network can be seen in Figure 2.

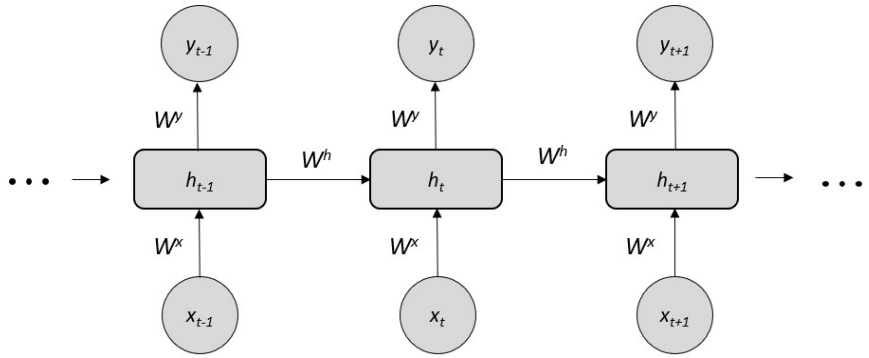

Figure 2: Unfolded computational graph of RNN

The parameters in an RNN network are shared across time and thus inputs at each time t influence the model parameters. This makes the optimization very expensive as the gradients of the model have to be calculated through the sequence using a method called backpropagation through time (BPTT). (Goodfellow et al. 2016.) This makes the model unsuitable for some tasks that require fast convergence in optimization. Another downside of the standard RNN is also related to the long backpropagation sequences. When BPTT is used, the gradients eventually tend to vanish or blow up as the sequence gets longer. Vanishing gradients can lead to parameters not being updated and thus the estimation of the model finishes, leading to non-optimal results. Exploding gradients on the contrary might lead to oscillating weights that can make the model unable to learn anything useful. (Hochreiter et al. 2001.) Thus, standard RNNs are only suitable for tasks with a limited number of past inputs where the gradient is controllable.

To overcome the problem of vanishing and exploding gradients of standard RNNs, Hochreiter & Schmidhuber (1997) proposed the long short-term memory (LSTM) model as the solution. The gradients are kept more stable by introducing paths through time that retain or forget information in a certain way. This pathway

allows the accumulation of information over longer durations. The single hidden layer function from simple RNN, where previous hidden values are looped forwards, is now reconstructed to a significantly more complex structure utilizing different gates. (Goodfellow et al. 2016.) The whole LSTM architecture can be expressed as the following composite function

$$\begin{aligned}
\mathbf{f}_t &= \sigma\big(\mathbf{U}^f \mathbf{x}_t + \mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{b}^f\big) \\
\mathbf{i}_t &= \sigma\big(\mathbf{U}^i \mathbf{x}_t + \mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{b}^i\big) \\
\mathbf{o}_t &= \sigma(\mathbf{U}^o \mathbf{x}_t + \mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{g}_t &= \tanh(\mathbf{U}^g \mathbf{x}_t + \mathbf{W}^g \mathbf{h}_{t-1} + \mathbf{b}^g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{y}_t &= \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)\,,
\end{aligned} \tag{6}$$

where $\odot$ represents the Hadamard product, the $\mathbf{f}$, $\mathbf{i}$, $\mathbf{o}$, $\mathbf{g}$, $\mathbf{c}$, and $\mathbf{h}$ respectively denote the forget gate, the input gate, the output gate, the classic RNN gate, the cell state, and the hidden state. Also, $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{b}$, respectively denote the input weight matrices, recurrent weight matrices, and bias vectors of different gates of the model. The LSTM cell and the flow of information through the cell can be seen visually in Figure 3.

The different components in Equation 6 can be seen to have different activation functions and roles in the network. The most important of these is the cell state c_t, or alternatively the internal state unit, that acts as the accumulated long-term memory where the previous cell state is updated with new information at each time step. The input gate layer, i_t, and the classic RNN gate, also known just as the tanh layer, g_t, are responsible for adding new information from the current input to the accumulated memory. The amount of past information to forget at a given time is controlled by the forget gate, f_t, which outputs a value between 0 and 1. Finally, the output or hidden state of each time step is the accumulated information held in c_t multiplied by the output gate layer.

Although LSTM enables longer memory capacity for the network, it still has a few disadvantages. While the gates allow the network to skip unnecessary information and thus the small multiplication terms in the gradient calculation, it still has the same sequential structure. This means that it has to use the same BPTT algorithm that is computationally expensive. Also, for example, Khandelwal et al. (2018) show that in an NLP setting, the LSTM is capable of using the context of approximately the last 200 inputs but clearly distinguishes about 50 of the most recent inputs.

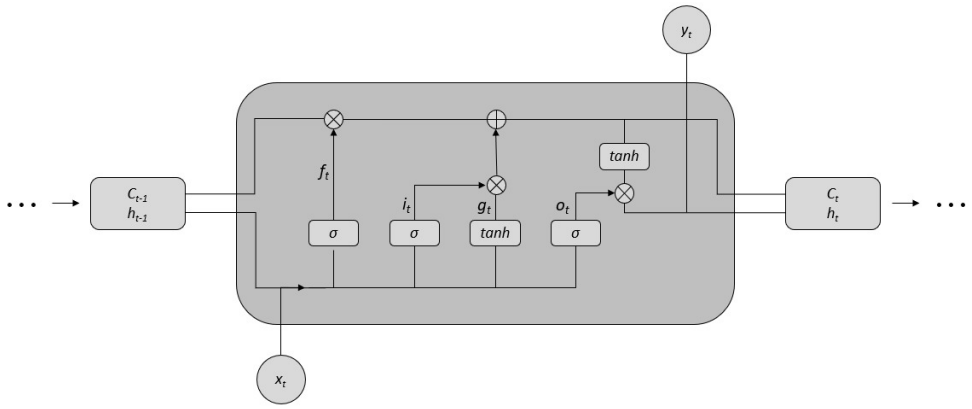

Figure 3: Long short-term memory (LSTM) cell

Gers et al. (2000) also improve the model by adding peephole connections, $V^i c_{t-1}$, where V^i are the peephole matrices for different gates i. These connections can be added to one or more of the forget, input, and output gates. These connections let different gates use the information of the cell state when the model is fitted. Another well-known variant of the gated RNNs is the gated recurrent unit (GRU) by Cho et al. (2014), which is simpler yet almost as efficient as the standard LSTM model. Many other variations of LSTM models have been proposed and they have also been applied as hybrid methods with an attention technique. The attention mechanism is also the core idea behind the Transformer network.

2.2.3 The Transformer network

The latest substantial advancement in sequence processing, and especially in natural language processing (NLP), has been the development of the Transformer architecture and its variants. The Transformer architecture was introduced by Vaswani et al. (2017) in a groundbreaking paper called "Attention Is All You Need" which has since generated a whole new family of deep learning models. As the paper's title suggests, a concept called attention is utilized to overcome many of the shortcomings of RNNs. The goal of the technique is to only pay attention to relevant parts of the input. The idea behind attention can be shown to arise from our biology. For example, given an image, we automatically attend our focus on a certain part of the image we feel is important. (Zhang et al. 2021.)

The attention technique proposed by Bahdanau et al. (2014) allows a model to attend to all observations of the input sequence simultaneously. This is a clear difference from RNNs that only have access to the values through the last hidden

state. The attention calculates a learnable weighted representation of the input in order to highlight the most important observations in it by giving more weight to them. This is done using queries, keys, and values that are compared to each other. The concept of queries, keys, and values is used in search engines for example. Generally, for a given query it has a matching key in a database, and a value corresponding to the key is returned. Here, the concept is similar, but instead of a single match, the query matches against all the keys to a certain degree based on their dot product. The dot product score is therefore a similarity metric between the queries and the keys. The similarity scores are then used to weigh the corresponding values to produce a new attention-weighted representation of the input sequence. (Chollet 2021.)

Earlier attention networks were usually combined with recurrent or convolutional networks and the Transformer model was the first to solely use the attention mechanism in its architecture. Since RNN models go through the input sequentially, the last hidden state has to capture all the relevant information of the input and therefore acts as a memory bottleneck even when attention is used with it. However, when there are no models it is combined with, the ability to attend to all observations of the input sequence makes it possible to utilize information from arbitrarily long sequences. Attention can instantly access all states in the input sequence, which helps with vanishing gradients. It might also allow for some interpretability because the attention similarities, i.e., attention scores, can be visualized. The attention mechanism is also bidirectional by its design, i.e., it can go through the sequence from start to end and from end to start. (Vasilev 2019.)

Although the Transformer was originally developed for NLP purposes, it has been applied in many other areas as well (Parmar et al. 2018; Dong et al. 2018; Huang et al. 2018). However, Transformer-based models have been applied only in a few financial applications and there is no unified representation of a Transformer for financial time series tasks. Therefore, the original Transformer model developed for NLP-related tasks is presented here. The proposed model of this thesis with its modifications for financial time series data is presented in Section 3.2.

The original Transformer model of Vaswani et al. (2017) is shown in Figure 4. These types of encoder-decoder architectures are very common in language tasks where the source and target sequences can be of different lengths and languages. The encoder layer is on the left, and the decoder layer is on the right. The encoder provides an encoded representation of the source sequence and uses it together with the decoder to predict the next target output. (Chollet 2021.) Most of the sub-layers are similar in both the encoder and decoder. The encoder contains two main sub-layers, a multi-head self-attention layer and a simple position-wise fully

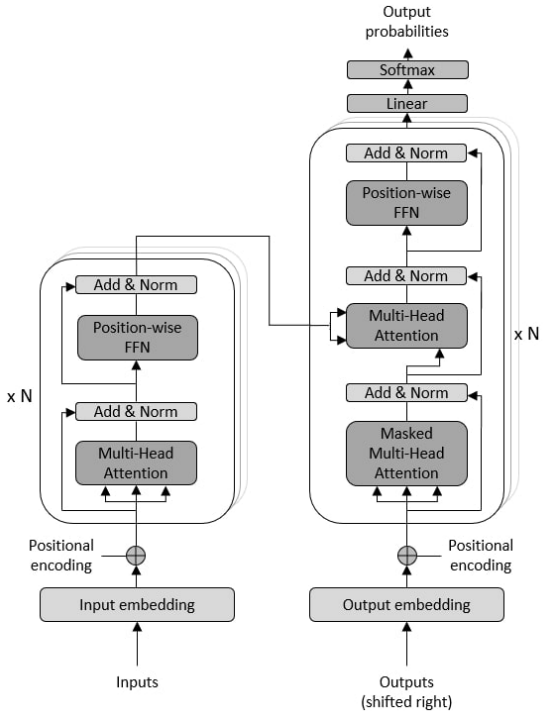

Figure 4: The Transformer architecture (Vaswani et al. 2017)

connected FFN, with some skip connections and normalizations. The decoder is mostly similar, except a masked self-attention is first applied to avoid any look-ahead bias by masking future values. The multi-head attention in the decoder also uses values from both encoder and decoder. As the encoder and decoder are mostly comparable, only the encoder block is presented here in detail.

The fundamental part of the Transformer model is the multi-head self-attention sub-layer. The operations within the layer are visualized in Figure 5. The Transformer utilizes dot-product attention with an additional scaling term, $\sqrt{d_k}$. The formula for this scaled dot-product attention, also visualized on the right side of Figure 5, can be written as

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{\mathbf{QK}'}{\sqrt{d_k}}\right)\mathbf{V},$$

where d_k is the hidden dimension of the keys, and $\mathbf{Q} \in \mathbb{R}^{T \times d_k}$, $\mathbf{K} \in \mathbb{R}^{T \times d_k}$ and $\mathbf{V} \in \mathbb{R}^{T \times d_v}$ are the matrices for queries, keys and values, respectively. The dimension of the values, d_v, can differ from the dimension of queries and keys. The softmax is used to turn each row of the similarity matrix $\mathbf{QK}' \in \mathbb{R}^{T \times T}$ into probabilities. As the variance of dot products in the similarity matrix increases as a function of d_k, the values are scaled by the square root of d_k to ensure a non-vanishing gradient (Vaswani et al. 2017). The values that are used to weigh the value vector are

known as attention scores which can be visualized in a $T \times T$ matrix to showcase each connection between the values of the input sequence with length T.

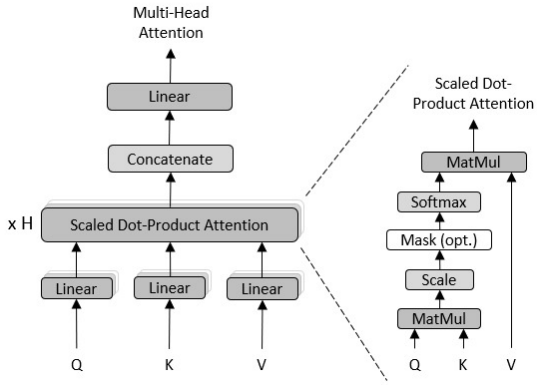

Figure 5: Multi-head self-attention layer visualized

The self-attention in the multi-head attention means that the queries, keys, and values are calculated using the input $\mathbf{X} \in \mathbb{R}^{T \times d_{model}}$ itself, hence the term self-attention. Three different weight matrices are used for three different linear transformations of the input sequence. (Zhang et al. 2021.) The three produced matrices are:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}^K$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^V,$$

where $\mathbf{W}^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d_{model} \times d_k}$ and $\mathbf{W}^V \in \mathbb{R}^{d_{model} \times d_v}$. All of these weights are learned and optimized in the estimation process of the model in order to produce the best attention weighting possible.

The multi-head self-attention itself is an expansion from the single self-attention calculation. The idea is to use multiple attention score mappings, i.e., multiple attention heads, to learn different kinds of relationships from the input sequence. Mathematically, the multi-head attention can be expressed as

$$\text{MultiHead}(Q, K, V) = \text{concat}(head_1, ..., head_h)\mathbf{W}^O,$$

where

$$head_i = \text{attention}(Q, K, V).$$

Given the number of attention heads, h, the outputs of the attention heads, $head_i \in \mathbb{R}^{T \times d_v}$, are first concatenated after which they are multiplied by $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$ to achieve the original shape of the input.

In addition to the multi-head attention, the Transformer architecture includes residual connections with a layer normalization that helps to maintain gradient flow through the model. Layer normalization means that the normalization of the values is calculated across the explanatory variables, i.e., the feature dimension of the model. The method is advantageous in many models as it allows for a fast calculation independently at each time step. (Géron 2019.) This procedure can simply be formulated as $\text{LayerNorm}(x + \text{SubLayer}(x))$, where the $\text{SubLayer}(x)$ is the output of the given sub-layer. The usage of residual skip connections between layers helps to maintain the positional information throughout the network. (Rothman 2021.) Introducing these skip connections to deep networks can also make the loss surface much smoother and therefore improve the estimation of the model (Li et al. 2018).

The other sub-layer in the Transformer encoder is the position-wise feed-forward layer. This means that a fully connected feed-forward network is applied identically and separately to each position (Vaswani et al. 2017). This is achieved using a ReLU activation function between two linear transformations as follows

$$\text{FFN}(x) = \text{ReLU}(x\mathbf{W}_1)\mathbf{W}_2,$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{model} \times 4d_{model}}$ and $\mathbf{W}_2 \in \mathbb{R}^{4d_{model} \times d_{model}}$.

The downside of the model so far is that it does not utilize the order of the input in any way since there are no recurrent or convolutional layers. The attention treats the input as a set and is permutation invariant as such (Chollet 2021). Since the order of inputs is often an important factor, the positional information of the inputs has to be encoded. Vaswani et al. (2017) experimented using different methods to encode the positional information and found a fixed sinusoidal positioning to be the preferred method for their purpose. They utilize different frequencies of the sine and cosine functions as follows

$$\begin{aligned} \text{PE}_{pos,2i} &= \sin(pos/10000^{2i/d_{model}}) \\ \text{PE}_{pos,2i+1} &= \cos(pos/10000^{2i/d_{model}}), \end{aligned}$$

where pos is the position in the input sequence and i is the model dimension. The positional encodings are then added to the inputs, $\bar{\mathbf{X}} = \mathbf{X} + \text{PE}(\mathbf{X})$. A visualization of some positional encoding values can be seen in Figure 6. The sequence length is chosen to be 200 and the dimension of the model is 50.

One obvious downside of most deep learning models is their lack of interpretability. While some model-agnostic methods for neural networks have been developed, e.g., LIME and SHAP, they are not very suitable for time series prob-

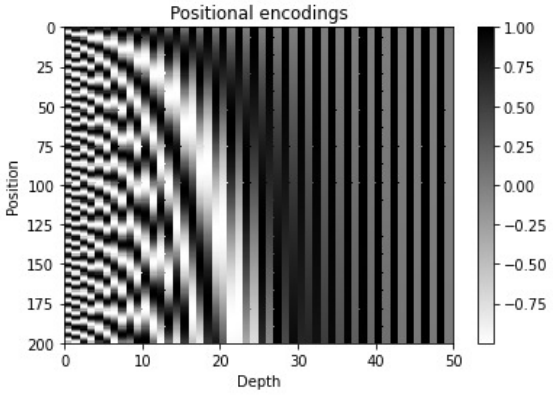

Figure 6: Fixed sinusoidal positional encodings over model dimensions

lems as they do not account for the order of time (Lim et al. 2021). Even though the goal of the thesis is to showcase the possible predictive capabilities of the Transformer model, any found explainability might contribute to the model's use in a broader context. Unlike standard RNN architectures like LSTMs or GRUs, Transformers might hold insights into the importance of different timesteps. The theoretical aspects are explored here, while the visualizations using the empirical data are shown as part of the results.

The Transformer is based on the attention technique which uses the outcomes of two different linear transformations of the input. However, the usage of attention as a realistic explanation is debated and the following should be approached with some scepticism (Jain & Wallace 2019; Wiegreffe & Pinter 2019). When the input $\mathbf{X}$ is multiplied with the linear matrices $\mathbf{W}^Q$ and $\mathbf{W}^K$ to produce $\mathbf{Q}$ and $\mathbf{K}$, the model loses the cross-sectional information of the independent predictors. Technically, the input vector could also be transposed so that $\mathbf{X} \in \mathbb{R}^{d_{model} \times T}$, where the variable importance is retained instead of temporal information. In the more common setting, where $\mathbf{X} \in \mathbb{R}^{T \times d_{model}}$, the temporal information is still retained as each row still represents the individual time steps. Thus, we can interpret the importance of previous observations and their relationships with each other for a given prediction. When the Transformer has multiple heads, a comprehensive interpretation is not as straightforward, as different heads tend to learn different patterns. Therefore, e.g., simple averaging of the attention scores may lose important information. (Chefer et al. 2021.) For simplicity, only one head will be assumed in the following.

The temporal relationships can be represented by visualizing the attention scores, $\boldsymbol{\alpha} \in \mathbb{R}^{T \times T}$, which can be attained from the probability-weighted scaled dot-product

$$\boldsymbol{\alpha} = \text{softmax}\left(\frac{\mathbf{QK}'}{\sqrt{d_k}}\right). \tag{7}$$

As the softmax is applied row-wise, each row of length T has a total value of 1. If any day k can be interpreted as being more important, its attention values tend to be larger vertically in the same kth column when it is compared to each ith day. Since there are no limitations to the value on the vertical column, the summed attention scores of that kth day can exceed 1. The downside of the attention score is that it is local, in the sense that it only provides information regarding the current prediction for $t + \tau$ using T previous time steps as the explanatory variables. Though if certain time lags exhibit attentive persistence throughout the series, one could average the score of each individual value, $a(i, j)$, over the length of the estimated time series N as

$$\bar{\alpha}(i, j) = N^{-1} \sum_{t=1}^{N} \alpha(t, i, j). \tag{8}$$

Another possibility regarding the attention scores is to use them to identify regime shifts, as experimented by Lim et al. (2021). Even though the authors apply the concept for volatility time series, it might be possible to exploit their idea to simple asset returns as well. In order to calculate the metric, one row has to be fixed. In the empirical analysis $i = 1$ will be used and thus its notation is excluded from the following. Now, the average attention scores for each jth lag at first row can be written as $\bar{\boldsymbol{\alpha}}(j) = [\bar{\alpha}(1), ..., \bar{\alpha}(T)]^T$. These averages of the jth lags will be compared against the jth lags of each individual predicted time step, $\boldsymbol{\alpha}(t, j) = [\alpha(t, 1), ..., \alpha(t, T)]^T$, and possible significant deviations from the average can be construed as regime shifts. The deviations are calculated as a distance metric for each predicted time step t as follows

$$\text{dist}(t) = \kappa(\bar{\boldsymbol{\alpha}}(j), \boldsymbol{\alpha}(t, j)), \tag{9}$$

where κ signifies the Hellinger distance expressed as

$$\kappa(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{1 - \sum_{i=1}^{T} \sqrt{p_i q_i}}.$$

The value of dist(t) can now be visualized against a price or a return series to see whether the metric captures any significant regime changes. The results from the Equations 7, 8 and 9 will be represented in Section 4.4.

2.3 Previous research

The empirical analysis in this thesis focuses on machine learning methods that are employed in financial time series analysis. Therefore in this section, previous research in this area is examined and analysed. Machine learning methods have been deployed in financial applications since the 1980s (White 1988). Back then, computers were of course far from their current computational capabilities, which limited the early usage and development of models. Since then, multiple different machine learning methods have been applied to asset prediction tasks. These methods include, e.g., support vector machines (SVM), k-nearest neighbours (kNN), tree-based classifiers (TBC), neural networks (NN) and random forests (RF).

A comprehensive study on machine learning methods for financial prediction was constructed by Ballings et al. (2015) who evaluated multiple models on a large dataset consisting of 5767 public companies from Europe. The models used in the study were AdaBoost, Kernel Factory, neural network, logistic regression, support vector machine, k-NN and a random forest. They use a 2-fold cross-validation five times and as such, there is no clear out-of-sample data available. The random forest was a clear winner, achieving an AUC score of 0.904 while the second-best model, SVM, only attained a score of 0.840. The third is the Kernel Factory followed by AdaBoost. The logistic regression was the weakest with an AUC score of 0.661.

Another study consisting of various machine learning models was executed by Nevasalmi (2020) who studies the predictability of the S&P 500 index from the 12th of February 1990 until the 5th of October 2018 by using a multinomial response variable. The multinomial variable in question has three different categories in which the bounds are the upper and lower quartile of S&P 500 index returns in the in-sample. The applied methods include the k-nearest neighbour, gradient boosting machine (GBM), random forest, neural network and a support vector machine. The out-of-sample accuracies of the models were close, as the worst model, k-nearest neighbour, reached a 54.57 % accuracy, and the best model, GBM, achieved an accuracy of 55.97 %. A trading strategy with a starting value of 100 was also implemented in the analysis and all the methods defeated a simple buy-and-hold strategy even when accounting for transaction costs. As with the accuracies, GBM proved to be the winning model by a clear margin as it reached a final wealth of 351.54, while the second-best model, neural network, reached a wealth of 244.51.

Krauss et al. (2017) analyze the performance of neural networks, gradient-boosted trees, random forests and an equally-weighted ensemble model of the previous methods in predicting the daily directional returns of the S&P 500 index constituents between 1992 and 2015. The authors apply the models to predict stocks that outperform the cross-sectional median of the returns instead of the more common threshold of zero. The reported accuracies were all in the range of 54% and 55%. The ensemble model was the best method producing an annualized return of 72.96 % after transaction costs, while the neural network was the worst with 26.85 % return. All methods easily beat the market return of 9.25 %. They find, however, that most of the advantage was gained before 2001 after which the returns started to diminish. The authors suspect that the increase of ML techniques and the simultaneous surge in computing power are the cause of these findings.

Later, Fischer & Krauss (2018) broaden the scope of the earlier study by including an LSTM model and a logistic regression in the study. They report the LSTM model to have the best accuracy at 54.3% while the logistic regression reaches an accuracy of 52.2%. They find that the LSTM model is able to achieve an annualized return of 82.29 % after transaction costs, topping all previous methods. In contrast, the logistic regression is only able to produce an annualized return of 7.11 % which is even below the market return. The authors note that after the financial crisis of 2008, the cumulative profits generated by the LSTM stay fairly constant suggesting there is no economic significance in the latter period.

Nelson et al. (2017) study the intraday performance of the LSTM model by using 15-minute data from five stocks on the Brazilian stock exchange. They use a total of 175 different technical indicators and five predictors formed from price data. The LSTM model is compared to the baseline models of multi-layer perceptron, random forest and a pseudo-random approach. The LSTM has the highest median accuracy in all five stocks by a rather clear margin. They note that the standard deviation of the accuracy, however, is quite high.

Jiang et al. (2018) are one of the first to implement an attention-based recurrent neural network in financial time series. They estimate the models on the currency and stock markets of China, India and USA and further split their study period into two parts. First is the crisis period from 2007 to 2009, and then a non-crisis period from 2010 to 2017. The proposed attention-based model achieves significantly higher AUC scores than the competing models. For example, on the stock market data sets, its AUC score is in the range 0.547–0.566. A normal bi-directional LSTM model, on the other hand, is only able to achieve an AUC score in the range 0.512–0.538. Similar results were true for currency markets, where the

attention-based model attain an AUC in range 0.553–0.772, while the second-best model, cross-domain attention network, reached a AUC score of 0.520–0.628.

Ding et al. (2020) are probably the first ones to apply a Transformer model in financial time series prediction. They compare the performance of a convolutional neural network, LSTM, attention LSTM, original encoder-only Transformer and their own Transformer variant using daily Nasdaq data and 15-minute data from China A-shares. They also introduce modified thresholds to make the data more balanced. Their variant of the Transformer is able to produce a 57.3 % accuracy, while the original Transformer achieves an accuracy of 56.01 %. The LSTM model, on the other hand, was able to attain an accuracy of only 53.81%, and similar results in terms of the model performances were also presented in the Chinese data set. The out-of-sample period in the Nasdaq data was only one year long, and thus the result cannot be considered very robust.

A second article regarding Transformers in stock return prediction is published by Yoo et al. (2021). They also apply their own variant of the Transformer, called Data-Axis Transformer, to forecast six different data sets of various lengths including Nasdaq 100 and Nikkei 225 indices. They compare their model against benchmark models such as LSTM and attention LSTM. Their proposed model is able to achieve superior performance in each of the six time series. For example, the reported accuracy of the model for Nasdaq is 54.06 % whereas LSTM and attention LSTM for example reached accuracies of 52.63 % and 52.60 %, respectively. One implementation can be found in Zhang et al. (2022) who deploy a Transformer-based hybrid model that uses both textual and numerical information in the estimation. They also report their results using only historical price data. Authors find that on four different modified datasets the model predictions achieve accuracies in the range 58.61 %–59.99 %, which beats all non-Transformer models, even if the other models also employ text data.

3 DATA AND METHODOLOGY

3.1 Data and preprocessing

The goal of the empirical analysis in this thesis is to predict one-step ahead directional returns of different assets. The empirical analysis is conducted using four different US stock indices. The indices used in the study are the S&P 500, NASDAQ Composite (Nasdaq), Dow Jones Industrial Average (DJIA), and Russell 2000 indices (Russell). The first three indices have been chosen to represent the most efficient and well-known market indices in the world. Russell 2000 is a small-cap stock market index and thus its predictability might slightly differ from other indices in this study. The data set of the study includes daily observations of the selected variables: opening price (open), highest price (high), lowest price (low), adjusted close price (close) and volume. Instead of the more traditional approach of using only the adjusted close prices, the other weak-form efficient variables are also selected, as the Transformer should benefit considerably from additional variables. The aforementioned variables will also be used to calculate 12 different technical indicators.

The full research window is the period from 1st of January 2007 to 31st of December 2021. This includes the 10-year prediction period as well as five years of data to generate the technical indicators and to fit the models. The empirical analysis, including the predictions and trading strategies, is performed with daily observations. In total, 3777 daily observations are used in the analysis, of which 2517 are also used as out-of-sample data. Asset returns have been shown to pose a real challenge in prediction tasks since they have a very low signal-to-noise ratio. This is even more true in the case of daily returns. Neural networks also have tendencies to overfit easily when the noise is high relative to the signal (Abu-Mostafa et al. 2012). Yet, daily returns are still utilized as some predictability has been found in the previous literature. Furthermore, as the Transformer can simultaneously attend to all observations, in theory, possible predictable dynamics should be found.

The dependent variable in this study is the directional sign of the asset return. Thus the objective is to forecast the conditional probability that the return is positive or negative, given lagged observations of the explanatory variables. This type of classification task can be indicated with a binary indicator: 1 if the asset return is positive and 0 if the return is negative. The simple daily returns of assets are calculated from the adjusted close prices for each day. Excess returns are not

used, as the risk-free rate for daily observations is practically zero for most of the period and averages 0.0004% in the out-of-sample period.

Two different variations with a different number of explanatory variables will be examined in this study. The first variation, hereinafter the main model (M), includes the variables high, low, open, close and volume. The second one, hereinafter the extended model (E), will include all the variables in the first case, but also a variety of additional technical indicators presented in Table 1 along with their mathematical formulations. The lengths, k, of the technical variables are the default values used for the indicators in the pandas-ta library (Johnson 2021). In addition to the default values, also the 50-day moving average (MA_{50}) is used. The chosen variables were influenced by previous studies on the matter (Neely et al. 2014; Qiu & Song 2016; Bao et al. 2017). Having multiple variables might introduce highly correlated predictors to the model. Neural networks can deal with multicollinearity and as the task is focused on forecasting instead of interpretation, the multicollinearity shouldn't cause the forecasts to be any worse for the logistic regression either (Hyndman & Athanasopolous 2018).

Table 1: The selected technical indicators

Here C_t denotes the adjusted closing price, L_t the lowest price and H_t the highest price of an asset at time t. The σ_k is the standard deviation over the last k days and m is the number of standard deviations. The RS is the average gain divided by the average loss over k periods. The $TP = (H_t + L_t + C_t)/3$ is the typical price at time t and $MD = k^{-1} \sum_{i=1}^{k} |TP_i - MA_i|$ which is the mean deviation from the moving average. PS is the average sum of standard deviation on positive days over k periods and NS is the average sum of standard deviation on negative days.

Abbreviation	Technical indicator	Formula
MA_k	Moving Average of k days	$k^{-1} \sum_{i=1}^{k} C_{t-i+1}$
MOM_k	Momentum	$C_t - C_{t-k}$
$STO\%K_k$	Stochastic %K	$(C_t - L_k)/(H_k - L_k) \times 100$
$STO\%D_k$	Stochastic %D	$k^{-1} \sum_{i=1}^{k} \%K_i$
RSI_k	Relative Strength Index	$100 - 100/(1 + RS)$
$MACD_{k,n}$	Moving Average Convergence Divergence	$EMA_k - EMA_n$
CCI_k	Commodity Channel Index	$(TP_t - MA_t)/(0.015MD_t)$
$PPO_{k,n}$	Percent Price Oscillator	$(MACD_{k,n})/EMA_n$
UB_k	Upper Bollinger Band	$MA_k(TP) + m \times \sigma_k(TP)$
LB_k	Lower Bollinger Band	$MA_k(TP) - m \times \sigma_k(TP)$
RVI_k	Relative Volatility Index	$PS/(PS + NS) \times 100$

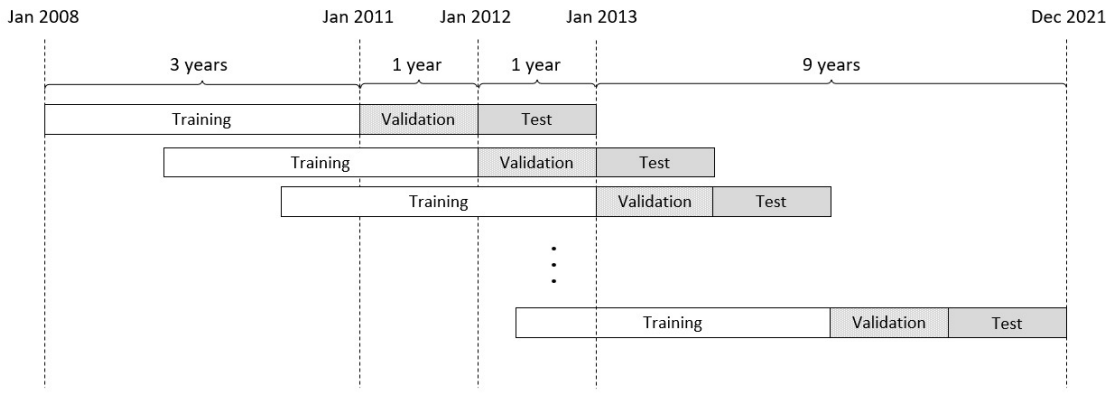

Figure 7: The rolling window approach

The empirical analysis is carried out using a rolling window approach shown in Figure 7. The forecasts are made for a 10-year out-of-sample period from the 1st of January 2012 until the 31st of December 2021. The out-of-sample period is divided into ten non-overlapping prediction windows, each one year long. At the start of each 1-year period, the model is refitted and optimized. The rolling estimation method ensures that the model is more attentive towards more recent information, as previous data is excluded as the estimation window changes. The split for the in and out-of-sample period is 80–20 and thus for each predicted out-of-sample year, the model is estimated and further optimized using the previous four years of data.

The in-sample period can be further divided into two separate parts called training and validation data. The training data is used to fit the deep learning model and its parameters based on the gradient of the loss function. The validation data is then used to further optimize and fine-tune the model in order to make it generalize well on new unseen data. The out-of-sample data, also known as test data, is left for model evaluation. (Goodfellow et al. 2016.) The evaluation metrics for the out-of-sample data are discussed in Section 3.3. Of the in-sample data, the training sample covers the first three years, while the validation is the fourth and final year.

The estimation also requires the lagged observations of the explanatory variables. In this analysis, the selected lag window, also known as sequence length, is chosen to be 252 observations. Thus, the earliest observations used are the lagged values of the variables one year prior to the start of the training period, i.e., 1st of January 2007. The lag window of 252 observations is chosen as it covers approximately one year of observations and should be sufficient for the model to learn any possible seasonal effects within a year. The data is then rolled forward one day after each day's prediction. This also means that, for example, the training data

for the main model with five variables has the input dimensions of approximately 756x252x5, slightly changing through the years. The validation and test samples have input dimensions of approximately 252x252x5.

Neural networks usually require the input variables to be preprocessed in order to speed up and improve the optimization of the network. The variable preprocessing loosely follows the way of Fischer & Krauss (2018). Firstly, the simple daily return is defined as

$$R_{it} = \frac{P_{i,t}}{P_{i,t-1}} - 1,$$

where P_i is the adjusted close price of the asset i on a given day. This return is also used to construct the dependent binary response variable. All the other independent variables in both models are also changed to returns to remove any possible trends.

A usual way to further preprocess a time series is to standardize the variables or use a min-max rescaling where the values are scaled to any arbitrary range, typically within the range of 0 and 1 (Chollet 2021). In this study, all the explanatory variables are standardized by first subtracting their mean from the values and then dividing the result by their standard deviation. To avoid any possible look-ahead bias, only observations from the training set are used to standardize the variables. The obtained values from the training sample are then also used to standardize the validation and out-of-sample data sets. Thus the standardized daily value of any independent variable, x_i, can be written as

$$x_{it} = \frac{x_{it} - \mu_{i,train}}{\sigma_{i,train}},$$

where $\mu_{i,train}$ is the mean return of ith variable in the training sample and $\sigma_{i,train}$ is the standard deviation of ith variable in the training sample. This procedure is calculated separately for each of the ten estimation windows.

Neural networks are also stochastic as they initialize the preliminary weight matrices with different values each time. This means that each training instance gives a different result as the model finds different local minimums in the high-dimensional loss surface of the network. (Géron 2019.) Thus, in this study, the network is run through three times for each separate occasion and the average performance metrics are reported unless stated otherwise. The same random initializations, also known as random seeds, are used for each run and each model to make the results replicable.

3.2 Model configurations

Here the configurations of the models of this thesis are represented. First, the modifications for the original Transformer are described. The original Transformer architecture described in Section 2.2.3 was developed mainly for NLP-related tasks and as such, it will be modified to fit the needs of financial time series prediction. Most importantly, only the Transformer encoder will be utilized in this model. The autoregressive decoder is not needed since the estimations are done using a rolling window approach. Numerous different configurations were considered during the construction of the architecture and only the selected final modifications are presented here. The final Transformer-based model architecture used in the empirical analysis is shown in Figure 8. The illustration only shows a general overview of the model and a more detailed description of the model layers, including the activation functions and regularizations, is shown in Appendix I.

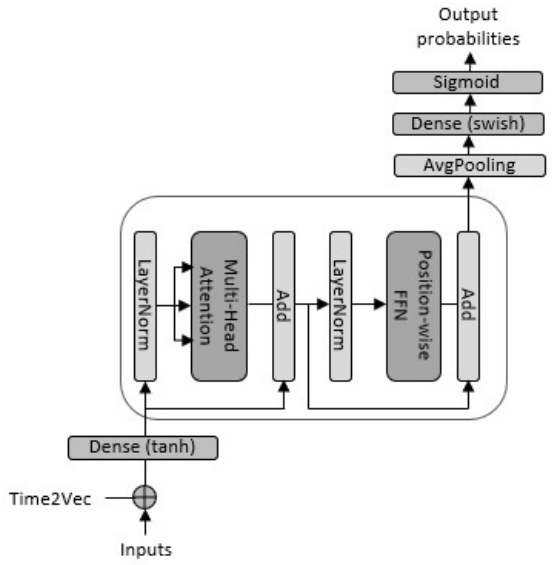

Figure 8: The proposed model architecture

Firstly, the temporal dimension of the input has to be presented to the model using positional encodings. Instead of the original fixed positional encoding, a learnable vector representation called Time2Vec, is utilized (Kazemi et al. 2019). Time2Vec represents the passage of time in both periodic and linear patterns. The formula for Time2Vec is

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

where τ is the time series input, F is a periodic activation function, ω and φ are learnable weight and bias parameters respectively, and k is the number of periodical representations. Based on the experiments of Kazemi et al. (2019), a sine function is chosen to be the periodic activation function F. Only one periodical representation is used in this study. Theoretically, multiple representations could be used to potentially capture different seasonal patterns. The temporal representations are added to the input matrix as two additional features instead of adding together the input and the encoding, as in the original Transformer.

Additionally, following Ding et al. (2020), a linear layer with tanh activation is applied to the original input as well as to the positional encodings before the Transformer encoder. Thus, the modified input $\bar{\mathbf{X}}$ can now be represented as

$$\bar{\mathbf{X}} = \tanh\left(\left[\mathbf{t2v}\left(\mathbf{X}\right)\right]\mathbf{W}^{I}\right),$$

where $\mathbf{W}^{I} \in \mathbb{R}^{d_{model} \times d_{model}}$ is the input weight matrix.

The encoder itself has the same sub-layers, multi-head attention and feed-forward layer, as the original. However, numerous tweaks have been implemented to further improve the model. Xiong et al. (2020) show that applying the layer normalization before the sub-layers improves the optimization procedure significantly. In their implementation, layer normalizations were added before the two sub-layers, but also a third normalization was implemented on the output with added residuals. However, in this architecture, the final normalization is not included. The activation functions in the model are changed from ReLU to a more recent activation function called Swish, which has achieved improved performance in multiple datasets (Ramachandran et al. 2017). Mathematically, Swish can be written as

$$\text{Swish}(x) = x\sigma(x),$$

where $\sigma(x)$ is the sigmoid activation function.

After the encoder, the data dimensions are reduced from $T \times d_{model}$ to $T \times 1$ by using a method called global average pooling (Lin et al. 2013). It averages across the values of explanatory variables to produce a single value for each time step. The values are then multiplied by a fully-connected network with a Swish activation function. Lastly, these values are fed into another fully-connected layer with a sigmoid function that transforms the values into probabilities.

One of the hardest problems in neural networks is their tendency to learn noisy representations of the in-sample data too well, and thus fail at generalizing to new unseen data. Because of this, the networks are often punished for overfitting to the in-sample data by using different techniques known as regularization methods.

An often-applied form of regularization is a technique called dropout. In linear regression the dropout is equal to the ridge regression, also known as $L2$ regularization. Nevertheless, in neural networks, it has been shown to be more effective. It is also computationally cheap and works well with almost all model architectures. (Goodfellow et al. 2016.) The dropout removes some of the outputs, y_{ij}, of layer h randomly with a probability p as follows

$$\bar{\mathbf{Y}}^h = \mathbf{D}^h \odot \mathbf{Y}^h, \qquad D_{ij}^h \sim \text{Ber}(p),$$

where $\odot$ is the Hadamard product. The probability p is one of the most important hyperparameters of any machine learning model. In the proposed Transformer-based model of the study, there are three different dropout layers utilized and their placements can be seen from the full model architecture in Appendix I.

Hyperparameters, one of which is the probability p in Dropout, are a set of variable settings that are not learned by the model and have to be specified by the user of the network. As neural networks are often complex structures, the latter part of the in-sample data, i.e., the validation data, is used to optimize the hyperparameters. (Goodfellow et al. 2016.) The hyperparameter choices can either be chosen based on some specified grid of possible values or by a rule-of-thumb (Claesen & De Moor 2015).

For this empirical study, the hyperparameter optimization is conducted using a trial-and-error based approach accompanied by results from experiments with the Keras Tuner library (O'Malley et al. 2019). The Keras Tuner includes a random search that randomly combines different model configurations of a pre-specified grid and tests them to find the optimal hyperparameters that minimize the loss on the validation set. The tuning is done on the validation sets of the assets for the years 2010 and 2011 before the out-of-sample data, and the same configurations are used throughout the ten years for all the assets. This makes the models non-optimal for the full period as each asset and year differs from one another. However, it gives a more general representation of the model's ability to forecast time series. The chosen hyperparameters of all the models are listed in Table 2. In addition, a search grid of values that have been tried is reported as well as the total number of trainable parameters in each model.

The chosen hyperparameters differ for both main and extended models in Transformer and LSTM. In the main Transformer model, referred to as TF(M), two different heads are used in the multi-head attention layer as opposed to only one in the extended model, referred to as TF(E). The number of stacked encoder blocks in both models is one. The number of hidden units refers to units within

Table 2: Hyperparameters of the different model configurations

Model	Hyperparameter	Search grid	Final value
TF(M)	Number of heads	[1, 2, 3, 4]	2
	Number of encoders	[1, 2, 3, 4]	1
	Number of hidden units	[8, 16, 24, ..., 64]	32
	Dropout	[0, 0.1, 0.2, ..., 0.5]	0.4
	Key dimension (d_k)	[7, 126, 252, 504]	504
	Total trainable parameters		25,993
TF(E)	Number of heads	[1, 2, 3, 4]	1
	Number of encoders	[1, 2, 3, 4]	1
	Number of hidden units	[8, 16, 24, ..., 64]	64
	Dropout	[0, 0.1, 0.2, ..., 0.5]	0.4
	Key dimension (d_k)	[7, 126, 252, 504]	504
	Total trainable parameters		41,624
LSTM(M)	Number of hidden layers	[1, 2, 3]	1
	Number of hidden units	[4, 6, 8, ..., 32]	16
	Dropout	[0, 0.1, 0.2, ..., 0.5]	0.2
	Total trainable parameters		1,425
LSTM(E)	Number of hidden layers	[1, 2, 3]	2
	1st hidden layer units	[4, 6, 8, ..., 32]	6
	2nd hidden layer units	[4, 6, 8, ..., 32]	6
	Dropout	[0, 0.1, 0.2, ..., 0.5]	0.1
	Total trainable parameters		895

the last dense layer with a swish activation function. The main Transformer model seemed to work the best using 32 hidden units in the layer, whereas the extended Transformer model required 64 units. For both Transformer models, the dropout probabilities and dimensions of the key matrices were the same at 0.4 and 504, respectively. The dimension of the key matrix allows the model to learn a 504-dimensional representation of each of the 252 lagged observations instead of the original features.

The main LSTM model, LSTM(M), only uses a single hidden layer in it, while the extended model, LSTM(E), uses two hidden layers. The number of hidden units and layers in LSTMs is often quite small in financial applications (Fischer & Krauss 2018; Siami-Namini et al. 2018). The number of hidden units in the main model is selected to be 16, while the amount of hidden units in the extended model is selected to be six in both layers. Bengio (2012) also states that layers of matching sizes generally have seemed to work better. The dropout probabilities of the main and extended models are 0.2 and 0.1, respectively.

The total number of trainable parameters in the Transformer greatly surpasses the amount used by the LSTM model, as expected. The main models, TF(M) and LSTM(M), have 25,993 and 1,425 variables, respectively. The contrast between the number of variables for the extended models is even greater. The logistic regression also has one controllable parameter, the lag window length. If the model is given the full 252 lagged observations with multiple predictors, the model

predictions will become extremely unstable. Thus, lag windows between 1–60 days were tested using the same validation sets as for the neural networks. For the main logistic model, LOG(M), a lag window of 22 observations is used, while the extended model, LOG(E), uses the last 18 observations.

Deep learning models are also often sensitive when it comes to the selection of the optimizer and its learning rate. The original Transformer uses an Adam optimizer that has a warm-up period with exponential decay. (Vaswani et al. 2017.) The only difference in this thesis compared to the original is the usage of cosine decay instead of exponential decay. In the warm-up, the learning rate is first increased rather fast until it reaches a set maximum after which it decays over a longer period. The changes in the learning rate over 150 training epochs are visualized in Figure 9 to give a more intuitive explanation of the warm-up and decay. A learning rate warm-up is often essential for Transformer-based models as otherwise, the gradients might become large and thus make the estimation unstable (Xiong et al. 2020).

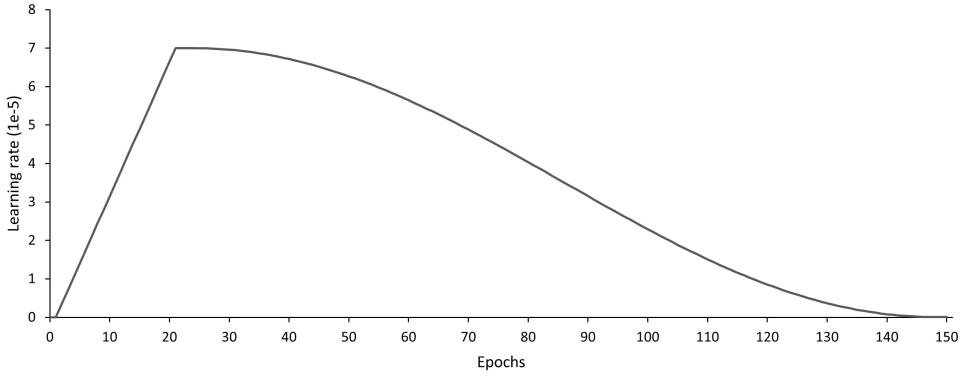

Figure 9: Learning rate warm-up with cosine decay

3.3 Performance metrics

The out-of-sample performance of classification models can be evaluated using several different criteria. The most common performance metrics in the literature include accuracy, precision, recall, and F1 score which can all be derived from the confusion matrix. F1 score may also be construed as the harmonic mean of recall and precision. The confusion matrix, represented in Figure 10, is often used to summarize the forecasting results of a classification task.

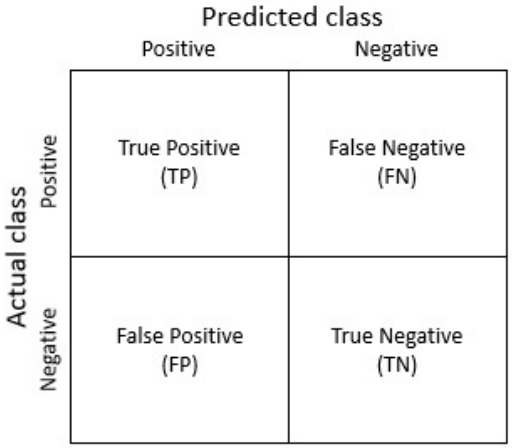

Figure 10: The confusion matrix

The above-mentioned performance evaluation metrics are calculated from the confusion matrix as follows:

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\ \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \\ \text{F1 score} &= \frac{2TP}{2TP + FP + FN}. \end{aligned} \tag{10}$$

The metrics shown in Equation 10 assume that the data is evenly distributed. However, on average there are more days with positive returns than negative and thus there is a clear class imbalance. While these variables are often used in the literature, they can sometimes depict a false image of the model performance, especially with unbalanced data sets. This can easily lead to poor generalization as the classifier usually only predicts the class with the largest size. This poor performance is not always visible in the classical metrics. As an example, given 100 values of which 90 are positive, always predicting true will yield an accuracy of 90 % and an F1 score of 0.947.

Nowadays, a growing number of studies in several fields have addressed these problems. Two recommended metrics for imbalanced data are the Matthew's correlation coefficient (MCC) and the area under the ROC curve (AUC). The MCC, also known as the phi coefficient in statistics, has been shown to be one of the best options in multiple studies (Boughorbel et al. 2017). According to Chicco &

Jurman (2020), the MCC is reliable since it has to have relatively good results in all four confusion matrix categories. The formula for MCC based on the confusion matrix can be written as

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{11}$$

The ROC curve is a common graphical way to evaluate binary classifications and it is based on two values from the confusion matrix: sensitivity and specificity. The plot shows their tradeoff as the values of the two metrics are plotted against each others. The area under the ROC curve (AUC) is a single value that can be used to simplify the performance in the graphical ROC. It can be defined from the ROC as

$$AUC = \int_0^1 ROC(t)dt. \tag{12}$$

A value of 0.5 can be interpreted as a coin toss and a value of 1 as a perfect predictor. Halimu et al. (2019) even find the AUC to be a better measure than MCC because of its discriminancy, though both are statistically consistent. Based on the previous, the main reported evaluation metrics throughout the results are accuracy, MCC and AUC. In addition, the F1 score is reported for the main results of the empirical analysis. These metrics are used to analyze the differences between model predictions of the different indices. Also, since Transformer should be relatively fast to train compared to its size, the time needed to fit the model network is reported.

As the estimations and predictions are done separately for each index it is also possible to conduct the one-tailed test by Pesaran & Timmermann (1992). The null hypothesis states that there is no directional predictability and as such, the realizations, y_t, and predictions, $\hat{y}_t$, are distributed independently. This gives an opportunity to evaluate the predictability of each index separately. Given that $\hat{P}$ is the share of correctly predicted signs of y_t, it has a sample success probability of

$$\hat{P}_* = \hat{P}_y \hat{P}_{\hat{y}} + (1 - \hat{P}_y)(1 - \hat{P}_{\hat{y}}),$$

where $\hat{P}_y = \mathbb{P}(y_t > 0)$ and $\hat{P}_{\hat{y}} = \mathbb{P}(\hat{y}_t > 0)$ are the sample means. The test statistic under the null can now be presented as

$$PT = \frac{\hat{P} - \hat{P}_*}{\left(\hat{\mathrm{Var}}(\hat{P}) - \hat{\mathrm{Var}}(\hat{P}_*)\right)^{1/2}} \overset{as.}{\sim} N(0, 1),$$

where $\hat{\text{Var}}(\hat{P}) = n^{-1}\hat{P}_*(1-\hat{P}_*)$ and

$$\hat{\text{Var}}(\hat{P}_*) = n^{-1}(2\hat{P}_y - 1)^2\hat{P}_{\hat{y}}(1-\hat{P}_{\hat{y}}) + n^{-1}(2\hat{P}_{\hat{y}} - 1)^2\hat{P}_y(1-\hat{P}_y) + 4n^{-2}\hat{P}_y\hat{P}_{\hat{y}}(1-\hat{P}_y)(1-\hat{P}_{\hat{y}}).$$

Diebold & Mariano (2002) have also introduced a common test to evaluate the performance of different forecasts. This test can be used to see whether the predictions of one forecast are significantly different from another. Under the null hypothesis, the predictions are equal and the test statistic is asymptotically $N(0,1)$ distributed. The alternative hypothesis states that the predictive performance of the first model is superior to the second model. The Diebold-Mariano test statistic can be expressed as

$$DM_{12} = \frac{\bar{d}_{12}}{\hat{\sigma}_{d_{12}}},$$

where $\bar{d}_{12}$ is the sample mean of the loss differential between the forecasts, $L(\epsilon_1) - L(\epsilon_2)$, and $\hat{\sigma}_{d_{12}}$ is an estimate of the standard deviation. Following Fischer & Krauss (2018), a vector of zeros and ones are used as the classification errors $L(\epsilon_i)$. Zero is assigned if the forecast is correctly classified and otherwise, the assigned value is one.

3.4 Trading strategy

The possible economic significance of the forecasts is also studied as it is one of the most important factors behind a successful prediction model. Even though higher predictability does support the possibility of economic success, it is not always the case. Only a small percentage of the returns often yield most of the profits, and thus the misclassification of those observations would adversely affect the performance of the model. Since all the models in the thesis provide conditional probabilities of the predictions, these probabilities can be converted into a trading strategy to showcase the economic significance. All the models are compared in terms of profitability against each other and a simple buy-and-hold method. The portfolios built based on the trading strategy are reviewed with and without transaction costs.

Daily observations often hurt the trading performance as the number of transactions in daily frequencies is fairly high. Thus, a method for reducing transactions is highly sought after. Nevasalmi (2020) uses the multinomial response variable of his study to split the trading strategy implementation into multiple option strategy

where the thresholds are the upper and lower quartiles. This approach also reduces the number of transactions. The following method used here is somewhat similar to the study. As most of the probabilities around the median are the noisiest, the lower quartile is considered as the trading threshold instead. This might also benefit the trading strategy as probabilities further away from the median should reflect greater confidence. The conditional prediction based on the probabilities can then be equated as

$$\hat{B}_t = \begin{cases} 0, & \text{if } \bar{\mathbb{P}}(y_{t+1} = 1) < Q_1, \\ 1, & \text{if } \bar{\mathbb{P}}(y_{t+1} = 1) \geq Q_1, \end{cases}$$

where $\hat{B}_t$ is the bin indicator showing whether the asset is included in the portfolio or not, $\bar{\mathbb{P}}$ is the average conditional probability of the three random out-of-sample forecasts for each index and Q_1 is the lower quartile of the estimated probability. The value of the lower quartile Q_1 is the average of the three results on the first validation set of 2011 for each index. As logistic regression only has one estimation, the probabilities and quartiles of the single estimation are used.

Implementing the above-mentioned strategy in the year 2011 reduced the number of transactions by 28.2% and 41.9% for the TF(M) and LOG(M) model, respectively. However, the LSTM(M) model seemed to favor more positive probabilities and as such, the implementation of the lower quartile would increase the transactions by 84.4%. Thus, LSTM models will be evaluated using the average probability of 0.5.

The long-only trading portfolio is constructed from all n indices. To avoid daily weight adjustments that would become extremely costly in a combined portfolio, the assets are handled separately and each is given a starting amount of 25. Consequently, the total starting value of the portfolio is indexed to 100. The value of the trading portfolio, V, at any given time t is

$$V_t = \sum_{i=1}^{4} V_{it},$$

and

$$V_{it} = V_{i,t-1} \times \left(1 + \left(R_{it}\hat{B}_{it} - ca_{it}\right)\right),$$

where R_{it} is the simple daily return of the ith asset, c is the fixed transaction cost and a_{it} is a binary indicator indicating if an action, i.e., buying or selling, in asset i took place at time t. The transaction cost c is fixed to be 0.05 percent per one-way transaction based on the academic literature on the topic (Hsu et al. 2010; Krauss et al. 2017; Fischer & Krauss 2018).

4 RESULTS

4.1 Performance evaluation

The empirical part of this research is focused on evaluating the predictive performance of the Transformer model and its benchmarks on four different US indices: the S&P 500, Dow Jones Industrial Average, Nasdaq Composite, and Russell 2000. The out-of-sample period in this empirical study is from the 1st of January 2012 until the 31st of December 2021. The predictions for the out-of-sample period are made using the main and extended versions for each of the models. Thus, the evaluated models are the main Transformer model TF(M), the main LSTM model LSTM(M), the main logistic regression model LOG(M), the extended Transformer model TF(E), the extended LSTM model LSTM(E) and the extended logistic regression model LOG(E).

Table 3 represents the main results of the empirical study where the performance metrics of the full 10-year out-of-sample predictions are displayed. The reported results are averaged across the assets as well as across the three different random runs for each asset. The performance metrics that are used to evaluate the models include the directional accuracy, area under the ROC curve (AUC), Matthew's correlation coefficient (MCC), F1 score, Pesaran-Timmermann value, and the time used to fit the model. The statistical significance of the Pesaran-Timmermann test is also reported.

Table 3: Performance results of the models 2012–2021

The table represents the main findings of the study. The classification accuracy, area under the ROC curve (AUC), Matthew's correlation coefficient (MCC), F1 score, Pesaran-Timmermann test (P-T) and execution time (ET) are reported. The performance metrics reported are the average results across the four individual indices. The reported Pesaran-Timmermann scores are the average scores over the three random out-of-sample forecasts. The execution time is the average time (in seconds) required to fit the model for the 10-year estimation window. The bolded values represent the top values for each metric. The symbols *, ** and *** denote the statistical significance at the 10%, 5%, and 1% levels, respectively.

Metrics	TF(M)	LSTM(M)	LOG(M)	TF(E)	LSTM(E)	LOG(E)
Accuracy (%)	52.52	**53.87**	50.65	52.33	52.82	50.22
AUC	**0.507**	0.503	0.494	0.506	0.501	0.496
MCC (%)	**1.54**	1.03	-1.25	1.30	0.53	-0.75
F1 score	0.612	**0.679**	0.578	0.610	0.651	0.549
P-T	**1.730****	0.398	-1.043	1.387*	-0.451	-0.246
ET	281.2	**99.5**	-	164.1	175.6	-

It can be seen from the table that the LSTM models, LSTM(M) and LSTM(E), provide the highest directional accuracies out of the models. The LSTM(M) managed to achieve an accuracy of 53.87% across the assets during the ten-year out-of-sample period, while the extended LSTM(E) model reached an accuracy of 52.82%. The main and extended Transformer models attained slightly lower accuracies at 52.52% and 52.33%, respectively. The accuracies of the logistic regression models were close to a coin toss of 50%. From the accuracy perspective, the addition of technical indicators in the extended versions did not improve the performance of any model. In fact, the results show decreasing accuracies across the models. The reported accuracies are below those reported in some previous studies, like the ones by Zhong & Enke (2019) and Ding et al. (2020), but are mostly in line with, e.g., Skabar (2013).

Although the accuracies of the Transformer models were below the ones of LSTMs, they were able to beat the benchmark models on the two metrics that measure predictive performance on unbalanced data. In terms of AUC, the TF(M) and TF(E) managed to get scores of 0.507 and 0.506, respectively. Those are somewhat higher than the scores achieved by LSTM(M) and LSTM(E) models of 0.503 and 0.501, respectively. The logistic regression model showed scores under the coin toss value of 0.5. The MCC scores also show similar results across the six models. While the Transformer models performed better on these metrics, the differences from a coin toss are minimal. Thus, the performance does not necessarily indicate any predictive capabilities. Altogether the reported values were far below the results of previous studies that report AUC or MCC statistics (Jiang et al. 2018; Ding et al. 2020; Yoo et al. 2021).

The one-tailed Pesaran-Timmerman test of directional accuracy shows that the null hypothesis of having no directional predictability is accepted at the 5% level for five out of the six models despite the models having higher accuracies than 50%. The directional predictability of the main Transformer model on the other hand is statistically significant at the 5% level as the test score was 1.730 with a p-value of 0.042. The TF(E) model is nevertheless significant at the 10% level. The fact that LSTM models fail in terms of the more robust predictability metrics can likely be attributed to the fact that the models had a higher number of positive predictions on average, i.e., higher recall. As the classes were unbalanced, this resulted in a higher accuracy even though the model might not actually have any predictive power.

In terms of the F1 score, the LSTM also shows superior performance. As the F1 score is the harmonic mean between recall and precision, the high score can be mainly attributed to the high recall rate of the model. The metric can also be

seen as misleading in terms of real predictive accuracy. Thus, while slightly losing on the normal accuracy metrics, Transformer models showed the best predictive performances out of the tested models when the imbalanced data was taken into account. As such, the main Transformer model seems to have some predictive capabilities, although small.

When comparing execution times of the model fitting process, the times of the main models differ quite significantly as LSTM(M) is almost three times faster to fit than TF(M) at only 99.5 seconds for ten model estimations. The TF(M) on the other hand has approximately 18 times as many trainable parameters which shows the power of parallelized calculations. The TF(E) is even slightly faster to estimate than LSTM(E), even though the TF(E) has approximately 47 times more parameters. Overall, the models are fast to train as refitting the model ten times during a ten-year period only takes a maximum of 300 seconds.

The Table 4 reports the performance metrics of the models for each individual index used in the empirical study. The classification accuracy, AUC and MCC scores as well as the significance of the Pesaran-Timmermann test with asterisks next to accuracies is reported. The models show various predictive capabilities on an index-specific level. The Transformer models achieve the highest AUC and MCC scores for three out of the four predicted indices. However, the main LSTM model attains higher scores in the Russell index. In terms of accuracy, the main LSTM model reaches the highest accuracies across all four different indices. The logistic regression classifiers show the worst performance out of the models also on individual indices.

The performance of the TF(M) model seems to be relatively stable across the indices in each metric, and all the AUC and MCC scores remain positive. For example, the accuracies range between 51.61% and 53.58%, and the MCC scores are in the range of 0.67%–2.30%. The extended Transformer model also shows similar aspects, though the performance is slightly lower in most cases. However, the TF(E) has the best performance on S&P 500 across any model or any index as it achieves an AUC of 0.513 and MCC of 3.12%. All the AUC scores are also positive, but the MCC for Russell index is practically zero.

The main LSTM model has the highest accuracy of all the models for each of the four indices. In addition, the AUC and MCC scores are the highest in the Russell index. The AUC score is positive for three of the indices overall, while the MCC is positive for only two. As with the extended Transformer model, the LSTM(E) model shows slightly lower performance on average than the main model. Only the AUC and MCC scores in the Nasdaq index beat those of the LSTM(M) model.

Table 4: Performance metrics of the models for different indices 2012–2021

The table represents selected performance metrics for the individual forecasted indices. The classification accuracy, area under the ROC curve (AUC) and Matthew's correlation coefficient (MCC) are reported. The bolded values represent the top values for each metric in each of the individual assets. Also, the significance of the Pesaran-Timmerman test is reported. The symbols *, ** and *** denote the statistical significance at the 10%, 5%, and 1% levels, respectively. The reported performance metrics are the average results of three random estimations for each index over the ten-year out-of-sample period from 1st of January 2012 until 31st of December 2021.

	S&P500			DJIA			Nasdaq			Russell		
Model	Acc. (%)	AUC	MCC (%)	Acc. (%)	AUC	MCC (%)	Acc. (%)	AUC	MCC (%)	Acc. (%)	AUC	MCC (%)
TF(M)	53.12**	0.510	2.30	51.79	**0.503**	**0.67**	53.58*	**0.508**	**1.63**	51.61*	0.508	1.55
LSTM(M)	**54.28**	0.507	1.99	**53.26**	0.501	-0.17	**54.99**	0.498	-0.56	**52.93**	**0.508**	**2.86**
LOG(M)	49.92	0.485	-3.27	49.88	0.489	-2.30	52.24	0.502	0.36	50.56	0.501	0.19
TF(E)	53.16***	**0.513**	**3.12**	51.44	0.502	0.65	53.65	0.506	1.45	51.05	0.501	0.00
LSTM(E)	53.30	0.503	1.11	52.41	0.499	-0.19	54.59	0.501	1.02	51.00	0.499	0.19
LOG(E)	51.20	0.507	1.39	49.64	0.489	-2.30	50.60	0.498	-0.37	49.42	0.491	-1.73

The logistic regressions show varying performances and seem to perform well in totally different indices. While the LOG(M) has an MCC of -3.27% on the S&P500 index, the LOG(E) reaches a positive score of 1.39%. This indicates that the models are not consistent at all. Altogether, both logistic regressions report accuracies of over 50% in two of the four indices. The MCC and AUC scores are positive in two of the four cases in the LOG(M) model and positive in one of the four indices for the extended model.

In terms of the significance of the directional accuracies measured by the Pesaran-Timmermann test, only the Transformer models were able to reach statistical significance even at the 10% level. The LSTM(M) model was close, however, as the accuracy for the Russell index had a p-value of 0.106. Both the Transformer models were statistically significant at forecasting the directional accuracy of the S&P 500 index. The TF(M) model was statistically significant at the 5% level while the TF(E) achieved significance at the 1% level. The TF(M) model also had significant directional accuracy at the 10% level both in Nasdaq and Russell indices.

The results at the index level point out that in terms of accuracy, the Nasdaq model produces the highest accuracies followed by the S&P 500. The slightly less-studied indices, DJIA and Russell, report lower accuracies and for the Dow Jones only the Transformer models are able to have positive results in terms of MCC. The accuracy and MCC score achieved with the Transformer model for the Nasdaq index fall significantly behind those reported by Ding et al. (2020). Although, the LSTM(M) model seems to reach higher levels of accuracy than their benchmark LSTM. The reported accuracies for the Nasdaq index by Bekiros

& Georgoutsos (2007) and Yoo et al. (2021) are 54.30% and 54.06%, respectively, which are relatively close to the accuracies attained here.

The individual results of the TF(M) model for each year, index, and for all of the three random trials can be seen in Appendix II. From there it can be observed that the variation between years and different trials is quite high. Therefore, the decision to use multi-year and multi-trial analysis is important when comparing the predictive abilities of stochastic neural networks. This way the results are a lot more robust and representative of the true behavior of the models. For example, the highest and lowest accuracies for a single year are 61.35% and 46.40%, respectively. This corresponds somewhat with the varying performance of the accuracies in Bekiros & Georgoutsos (2007). Furthermore, the highest and lowest reported MCC scores are 17.61% and -11.08%, respectively. One random trial in the Nasdaq index during quite the same period as in Ding et al. (2020) does also correspond well with their results, however, the two other trials show far worse performance.

The results for the Diebold-Mariano test between the model predictions are reported in Table 5. The table reports the p-values of the test. A rejection of the null hypothesis indicates that the predictions made by the model on the row index of the table are superior to the model on the column index. The comparisons are made pairwise and a p-value below 0.05 indicates statistical significance at the 5% level which is considered the accepted confidence level in this comparison. The results concerning the TF(M) model can be seen to show no superiority against the LSTM models or the extended Transformer model. The predictions show statistical significance over the performance of the main logistic model but the null hypothesis of equal predictions is also accepted against the LOG(E) model.

The LSTM(M) model seems to show a lot better performance in terms of the Diebold-Mariano test. The model predictions are better at the 5% significance level than any of the Transformer and logistic regression models. Moreover, the predictions against both the logistic regressions and the TF(E) model are statistically significant at the 1% level. However, the predictions are not superior to the extended LSTM model. The main version of the logistic regression showed no statistical significance over any other model.

Of the extended models, TF(E) and LOG(E), are not shown to have significantly better predictions compared to any other model in the study. The LSTM(E), on the other hand, also has superior performance over the two logistic regressions at the 1% level and also defeats the performance of the TF(E) model at the 5% level. When comparing the overall performance of the main model and extended models, it seems that the addition of technical indicators has negative effects, if

Table 5: Diebold-Mariano test for the models

The table represents the p-values obtained from the pairwise Diebold-Mariano test. The null hypothesis states that the predictions of the two models are equal. Alternative hypothesis is that the predictions of the method represented on the row index of the table is superior to the one in the column index of the table. The predictions are the binary predictions of all the indices from the out-of-sample period from the start of 2012 to the end of 2021.

	TF(M)	LSTM(M)	LOG(M)	TF(E)	LSTM(E)	LOG(E)
TF(M)	-	0.976	0.014	0.192	0.759	0.152
LSTM(M)	0.024	-	0.000	0.006	0.172	0.000
LOG(M)	0.986	1.000	-	0.884	0.997	0.4605
TF(E)	0.808	0.994	0.116	-	0.969	0.096
LSTM(E)	0.241	0.828	0.003	0.031	-	0.001
LOG(E)	0.985	0.999	0.540	0.904	0.999	-

any. While the results indicate superior performance for the LSTM model, the Diebold-Mariano test, as used by Fischer & Krauss (2018), is only interested in the portion of correct predictions, i.e., accuracies, which partly explains the results here.

The predictive classification performance of the models also varies over time as seen in Appendix II. Concerning the variations, the 126-day rolling accuracies of the models are presented in Figure 11. The upper plot shows the rolling accuracies of the main models while the lower plot displays the accuracies of the extended models. Rolling accuracies of 126 days, i.e., approximately half a year, are used to smooth out the volatile accuracies in shorter periods and show a more full representation of the real predictive power. The illustrated time series contains the period from the 1st of July 2012 until 31st of December 2021.

The Transformer and LSTM models show similar performance for most of the time series. The logistic regression on the other hand seems to have almost contradictive behavior as the accuracy usually rises when the other models show lower accuracies and vice versa. This can also be attributed to the shorter lag window since it only has information from a relatively short period. As the two neural network models show similar attributes, it can be seen that there is some time-varying predictability in these financial time series. The LSTM models seem to exhibit similar characteristics as their peaks happen approximately at the same time, although in different volumes. The Transformer models instead have less comovement as the TF(E) model seems to have a much more stable accuracy throughout most of the period than the TF(M).

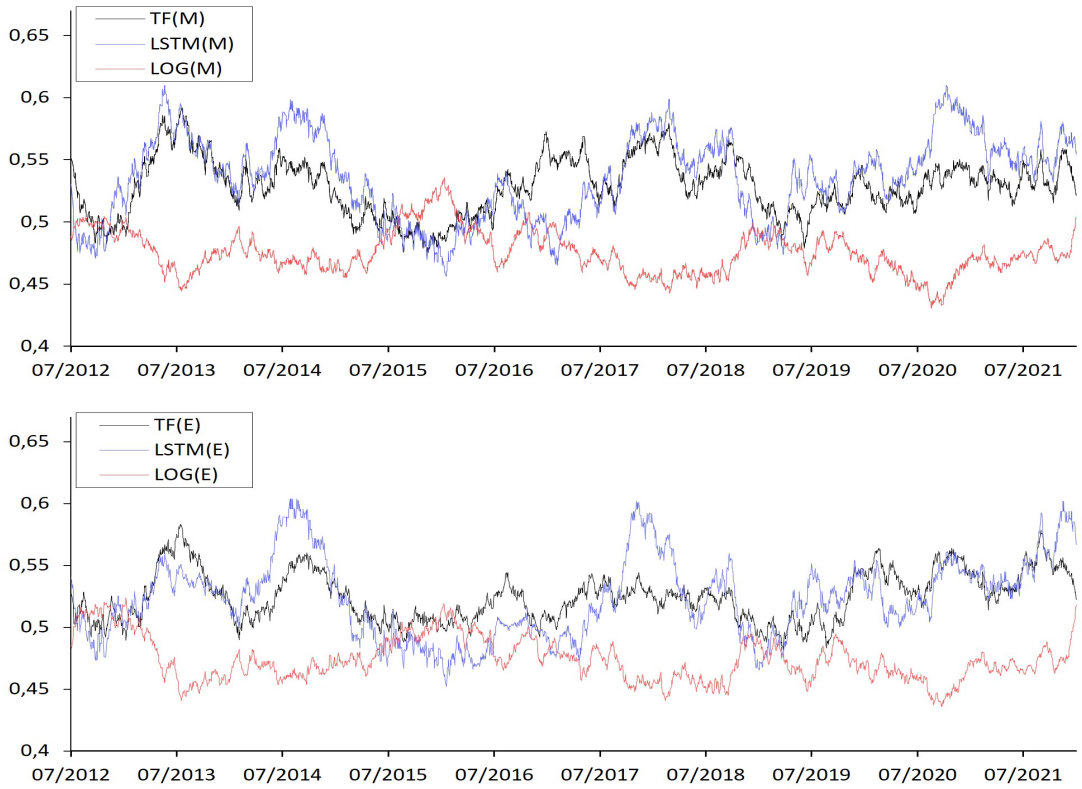

Figure 11: The 126-day rolling classification accuracies of the models

The neural network models seem to have a varying performance during the first two years of the time series. The models experience elevated levels of accuracy during 2017–2018, except for the more stable TF(E). The extended models for Transformer and LSTM seem to have similar performances across the year 2020 and right until the end of 2021. Oppositely, the LSTM(M) reached higher levels of accuracy than any other model in late 2020, while the TF(M) model shows the worst performance out of the four neural networks. Even though previous literature has found more volatile periods to exhibit better predictability, there does not seem to be any major shifts in accuracy during the beginning of the COVID-19 crisis for example. If anything, the directional accuracy seems to drop during the early stages of COVID-19. A longer period of poor performance can be seen from the start of 2015 until the mid of 2016. This corresponds quite well with the stock market sell-off of 2015–2016 that occurred around the same period. Another shorter decline in accuracy happened in the second half of 2018.

4.2 Trading strategy implementation

The results of the trading strategy examined in Section 3.4 are presented here. The trading strategy was implemented to showcase any possible economic significance of the models. Reporting the average results of three runs is not practical, as in reality the investment decision is made only once and on the basis of current information. Thus, the average probabilities across the three predictions are used to construct the strategy. The summary statistics of the trading strategy portfolios before (after) transaction costs are reported in the upper (lower) part of Table 6. The full trading period spans from the 1st of January 2012 until the 31st of December 2021. The buy-and-hold portfolio (B&H), highlighted on the right hand side of the table, is constructed by buying the same amount of each index at the start of the period. Similarly, in the trading portfolio, each of the four assets is given the same starting amount. No further adjustments on the amounts are made besides those resulting from trading the assets.

In part a) of the table, the summary statistics are reported without transaction costs. While the accuracies and even the other metrics like MCC and AUC were positive for most models, almost every model lost to the simple buy-and-hold strategy in terms of returns. The main Transformer model was able to achieve an annualized mean return of 15.7%, while the B&H came second with a return of 14.2%. The worst performing model was the main logistic regression with an annualized return of 8.2%. The logistic regression was the only model that benefitted from the usage of technical indicators in terms of the trading returns. When adjusting for the transaction costs, it can be seen from part b) that no model was able to beat a simple buy-and-hold strategy. Now the LSTM models are close to TF(M) model in terms of returns and have surpassed the TF(E) model that was ahead of them before transaction costs. This can be attributed to the fact that the number of transactions reported at the bottom of the table is a lot smaller for the LSTM models.

As the LSTM models use the 0.5 probabilities to reduce transactions, a confirmation check was conducted using a lower quartile threshold in order to see if it would provide better results. The number of transactions experiences an increase of 106%, while simultaneously the annualized mean return decreases from 13.4% to 9.49%, even before transaction costs. Thus, the decision to use the method with lower transactions proved to be beneficial.

The standard deviations of all the models are smaller than that of the B&H strategy. The logistic regressions have the smallest standard deviations while the

Table 6: Summary statistics of the trading strategy portfolios

The table illustrates the summary statistics of all trading portfolios before and after transaction costs for the out-of-sample period from 1st of January 2012 until 31st of December 2021. The annualized mean return, annualized standard deviation, skewness, excess kurtosis, maximum drawdown (MDD), historical Value-at-Risk (VaR), historical approximated expected shortfall (ES) and Sharpe ratio are reported. In addition, the number of transactions made in the out-of-sample period is reported for all the models.

a) Before transaction costs							
	TF(M)	LSTM(M)	LOG(M)	TF(E)	LSTM(E)	LOG(E)	B&H
Return	0.157	0.134	0.082	0.139	0.125	0.094	0.142
St.dev.	0.148	0.157	0.139	0.130	0.137	0.111	0.172
Skewness	-0.338	-0.700	-2.182	-1.233	-0.909	0.004	-0.836
Kurtosis	11.716	13.112	25.096	40.805	25.843	12.796	16.029
MDD	0.221	0.318	0.391	0.299	0.306	0.277	0.346
$VaR_{0.95}$	-0.014	-0.015	-0.013	-0.011	-0.012	-0.010	-0.016
$VaR_{0.99}$	-0.026	-0.028	-0.025	-0.021	-0.025	-0.022	-0.030
$ES_{0.95}$	-0.022	-0.024	-0.021	-0.018	-0.020	-0.017	-0.026
$ES_{0.99}$	-0.036	-0.039	-0.037	-0.033	-0.035	-0.029	-0.044
Sharpe ratio	1.061	0.855	0.589	1.064	0.910	0.852	0.827
b) After transaction costs							
	TF(M)	LSTM(M)	LOG(M)	TF(E)	LSTM(E)	LOG(E)	B&H
Return	0.126	0.123	0.044	0.097	0.121	0.049	0.142
St.dev.	0.148	0.157	0.139	0.130	0.137	0.111	0.172
Skewness	-0.344	-0.715	-2.161	-1.196	-0.904	0.046	-0.836
Kurtosis	14.696	16.204	27.944	43.128	28.510	16.143	16.029
MDD	0.223	0.321	0.444	0.292	0.307	0.341	0.346
$VaR_{0.95}$	-0.014	-0.015	-0.013	-0.011	-0.012	-0.010	-0.016
$VaR_{0.99}$	-0.026	-0.028	-0.025	-0.021	-0.025	-0.022	-0.030
$ES_{0.95}$	-0.022	-0.024	-0.021	-0.018	-0.020	-0.017	-0.026
$ES_{0.99}$	-0.036	-0.039	-0.037	-0.033	-0.035	-0.029	-0.044
Sharpe ratio	0.851	0.781	0.319	0.745	0.877	0.446	0.827
# of transactions	2180	832	2940	2896	332	3361	-

LSTM models are the most volatile. This falls in line with the 126-day rolling accuracies in Figure 11, where the LSTM models had the most variation in their accuracies. In each model, while their performance metrics got worse, the volatilities are somewhat lower when technical indicators are utilized. Apart from the LOG(E) model, the returns of each portfolio are negatively skewed. The highest reported negative skewness of -2.182 is contrastingly in the LOG(M) model. All the values of excess kurtosis were extremely high.

The Sharpe ratio reports a measure of the generated return of the portfolio in regards to the risk it carries. As the models had lower standard deviations,

every model except for the LOG(M) is able to beat the buy-and-hold strategy which has a Sharpe ratio of 0.827. The TF(M) and TF(E) achieve the highest Sharpe ratios of 1.061 and 1.064, respectively. As the models that utilize technical indicators have lower standard deviations, the TF(E) is the best in terms of risk-return ratio. When the transaction costs are taken into account, only two models are able to attain better Sharpe ratios than the B&H strategy. The best model is the LSTM(E) model with a Sharpe ratio of 0.877 and the TF(M) model has a Sharpe ratio of 0.851.

The maximum drawdown (MDD) is the depreciation from the historical peak expressed in percentages. The B&H strategy, for example, lost at most 34.6% of its value from a previous peak. Only the logistic regression model LOG(M) endured more serious losses. From the neural networks, the Transformer models were able to avoid all declines of over 30%, and the LSTM models have declines of just slightly over 30%. The MDD of the TF(M) model was impressively small at 0.221 (0.223) before (after) transaction costs, and it defeated the second-best model of TF(E) by almost 8pp and 7pp, depending on whether transaction costs were taken into account. It suggests that the main Transformer model was able to recognize highly negative periods more precisely than other models.

The Value-at-Risk (VaR) and expected shortfall (ES) are risk measures that measure expected returns given a tail risk event. A VaR of 95% indicates that there is a probability of 0.05 to lose at least the value the measure indicates. For example, the buy-and-hold strategy loses at least -1.6% in a day with a 0.05 probability. Here, the historical VaR is reported which means that the return of the 5th percentile is used as the historical VaR measure. Every model portfolio seems to have a slightly lower VaR than the B&H strategy with both 0.05 and 0.01 probabilities. The logistic regressions mostly have the lowest tail risk according to the VaR measures, although the 99% VaR is the lowest for TF(E) model. The models with technical indicators also seem to have lower VaRs than their main version counterparts.

The VaR does not reveal anything of the loss beyond the 0.05 probability and thus, expected shortfall is often used to better describe the possible loss of the portfolio. The expected shortfall indicates the expected return given the worst α-th percentile. Here, an approximation using the average over 0.1% confidence intervals of the historical percentiles up until the αth percentile is used. This should be able to approximate the actual integral over the area to some degree. The results from the ES measures are almost identical to the ones of VaR in terms of portfolio performances, though now the LOG(E) has the lowest tail risks in both probabilities. The main Transformer model, on the other hand, has a lower

expected loss at the 0.01 probability than the logistic regression. Again, each portfolio has a lower expected loss than the buy-and-hold strategy.

The cumulative returns before transaction costs generated by the model portfolios are also visualized in Figure 12, where the upper (lower) plot shows the development of the main (extended) versions. The time series are displayed from the start of 2012 until the end of 2021. The upper plot shows almost no differences in the top-three strategies before 2018–2019, where the TF(M) and B&H fare slightly better than the LSTM(M) model. The LOG(M) model is clearly the worst of them all. It seems that the main difference between TF(M) and B&H is the start of 2020 when the COVID-19 pandemic hits. This is also representative of the lot smaller MDD that TF(M) had. This might be construed as a supportive result for time-varying return predictability, and especially highlights the possibly higher predictability around regime changes as suggested by, e.g., Pesaran & Timmermann (1995).

In the lower plot, there are some deviations already around 2015–2016 where TF(E) model achieves the best returns in the stock market sell-off. Unlike the TF(M) model with the main variables, the addition of technical indicators actually made TF(E) perform a lot worse during the COVID-19, and it also had another drop at the end of 2020, unlike other extended models. The LSTM(E) never recovered from the decline of 2016. The LOG(E) performed better than the LOG(M) model and the start of 2020 also saw a slighter drop compared to others.

The same visualizations of the cumulative returns after the transactions can be seen in Figure 13. Now the B&H strategy already gains an advantage almost right from the start. As the amount of transactions is large even after controlling for them with a modified trading strategy, the development of the TF(M) model just slightly overcomes the LSTM(M) model. The TF(M) seems to perform slightly worse from 2015 until 2018, and both show similar performance around the COVID-19 pandemic. The logistic regression just barely makes a profit over the ten-year period.

The cumulative returns of the extended models paint an even worse picture for the Transformer model, as it pretty clearly loses to the B&H strategy, as well as to the LSTM(E) model. This time the TF(E) and LSTM(E) show far more similar performance during 2015–2018. However, right after the start of 2018, the TF(E) starts to perform worse at the same time as TF(M) starts to perform better. Especially after the start of 2020, TF(E) model was not able to extract any useful information from the data to achieve better performance than the benchmarks of B&H and LSTM(E). As with the LOG(M) model, the LOG(E) model is far behind the other models.

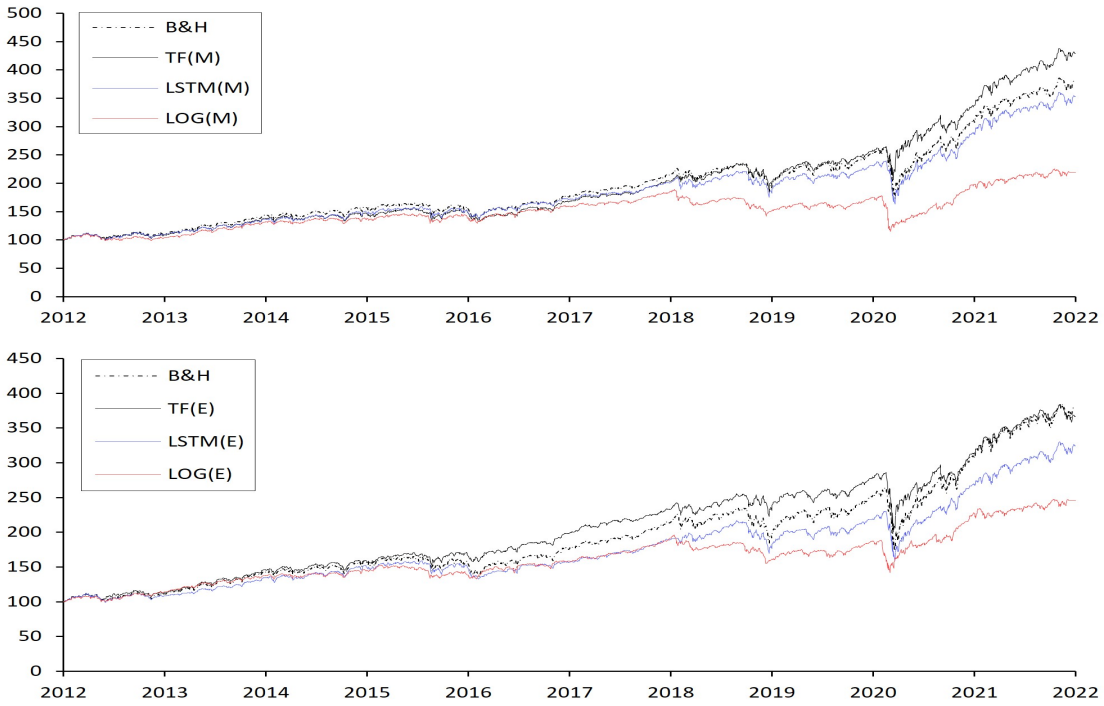

Figure 12: The cumulative returns of the models before transaction costs 2012–2021

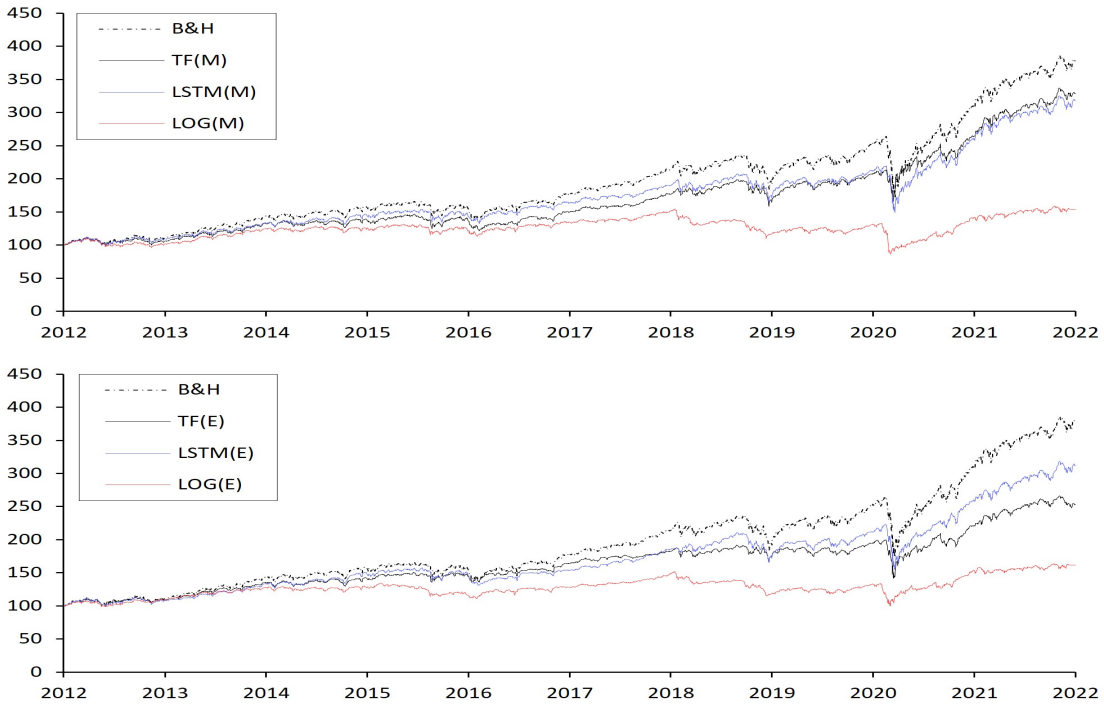

Figure 13: The cumulative returns of the models after transaction costs 2012–2021

The trading strategy portfolios are also examined individually for each of the four assets to illustrate any significant differences in their predictability. For simplicity, besides the B&H strategy of each respective index, only TF(M) and LSTM(M) are analyzed as they are the main versions of their respective models. The main logistic model is left out as its performance was subpar throughout the indices and does not contribute to any significant remarks. The trading portfolio statistics are displayed in Table 7. The performances of the models are only studied before transaction costs.

Table 7: Summary statistics of the trading strategy portfolios for individual assets

The table illustrates the summary statistics of the trading portfolios for each of the four indices before transaction costs for the out-of-sample period from 1st of January 2012 until 31st of December 2021. The annualized mean return, annualized standard deviation, maximum drawdown (MDD), historical Value-at-Risk (VaR) and Sharpe ratio are reported.

		Return	St.dev.	MDD	$\text{VaR}_{0.95}$	$\text{VaR}_{0.99}$	Sharpe ratio
S&P 500	TF(M)	0.155	0.145	0.177	-0.013	-0.026	1.067
	LSTM(M)	0.135	0.156	0.339	-0.014	-0.028	0.870
	B&H	0.138	0.163	0.339	-0.015	-0.030	0.843
DJIA	TF(M)	0.110	0.151	0.321	-0.013	-0.028	0.733
	LSTM(M)	0.105	0.162	0.347	-0.014	-0.028	0.647
	B&H	0.111	0.167	0.371	-0.015	-0.029	0.667
Nasdaq	TF(M)	0.199	0.162	0.230	-0.016	-0.030	1.224
	LSTM(M)	0.158	0.178	0.372	-0.017	-0.034	0.886
	B&H	0.189	0.186	0.301	-0.018	-0.035	1.016
Russell	TF(M)	0.148	0.195	0.300	-0.018	-0.033	0.758
	LSTM(M)	0.134	0.189	0.296	-0.019	-0.032	0.710
	B&H	0.115	0.209	0.434	-0.019	-0.034	0.549

Overall, in terms of annualized mean returns, the TF(M) was the top model in three out of the four indices. In the Dow Jones index, the B&H strategy was able to slightly defeat the TF(M) model. The TF(M) model also beat the benchmark LSTM(M) model on all four indices, and the LSTM(M) even lost to the B&H in three indices. The TF(M) also had the highest Sharpe ratios in each of the four indices while the LSTM(M) only beat the buy-and-hold strategy on two indices. The best Sharpe ratio obtained by the TF(M) was 1.224 in Nasdaq while the lowest one in Dow Jones was 0.733.

In S&P 500, the TF(M) had the best results in each of the metrics. Especially the maximum drawdown of 0.177 was extremely small compared to the benchmarks which both had a maximum drawdown of 0.339. While the LSTM(M) lost in

terms of return, the lower standard deviations achieved by the model improved its Sharpe ratio and it managed to beat the B&H strategy. Similar situations were also evident in the full combined portfolio. Similarly to S&P 500, the TF(M) had the top results in DJIA for almost every metric, although TF(M) had a slightly lower mean return than the buy-and-hold strategy. Unlike in the case of the SP 500, the TF(M) model now had a similar drawdown compared to the benchmarks. The DJIA also provided the lowest returns altogether.

On the opposite, the Nasdaq index provided the highest returns throughout. The TF(M) was again victorious in each of the metrics. The LSTM(M) model, on the other hand, was clearly the worst as it lost to the buy-and-hold strategy in terms of return, MDD and Sharpe ratio. Based on the result of the MDD, the LSTM(M) failed to recognize the COVID-19 pandemic in early 2020, while TF(M) seemed to be able to avoid some of the declines in the market.

Looking at the metrics for the B&H strategy in Russell, we can see that it had almost the lowest return while simultaneously being the most volatile and having the highest maximum drawdown of 0.434. Here, the performances of LSTM(M) and TF(M) were quite close to each other and both proved to be able to benefit from the higher volatility of the index. This is further proof that increased volatility might be beneficial in terms of economic significance, as discussed by Krauss et al. (2017), even though this behavior is not visible in the accuracies for the year, as seen in Appendix II. The reported average accuracy for the year was 49.8%, and both the AUC and MCC scores were negative for the whole year. Comparing the models, the LSTM(M) lost 1.4 percentage points in annualized returns to the TF(M), although it was able to beat the TF(M) in terms of standard deviation and MDD. In terms of Sharpe ratio, the TF(M) came slightly on top with a score of 0.758 while the LSTM(M) had a score of 0.710.

The corresponding time series graphs of trading portfolios' development on the four individual assets are illustrated in Appendix III. It can be observed that the TF(M) model was able to generate higher returns during the start of the COVID-19 pandemic in each of the four models than the benchmarks. The sudden declines were quickly reversed in both S&P 500 and Russell indices, while the performances in Nasdaq and DJIA were not as superior. The LSTM(M) only performed better than the buy-and-hold in Russell, and the performance was identical in the S&P 500. The performances in DJIA and Nasdaq were inferior to the B&H strategy.

4.3 Subperiod analysis of the COVID-19 pandemic

Based on the results presented earlier, the development of the models together with the overall market, i.e., the buy-and-hold strategy, seemed relevantly stable prior to the COVID-19 pandemic. The predictability of the stock market returns should also be more evident during financial turmoil as shown by, e.g., Krauss et al. (2017). Thus, motivated by this, a shorter subperiod analysis of the effects of the COVID-19 on the model performances is executed. Most papers that study the effects of COVID-19 crisis on stock markets focus on the very early days of the crisis and especially on the month of March (Al-Awadhi et al. 2020; Mazur et al. 2021). However, some studies expand the period until April and June (Baker et al. 2020; Chowdhury et al. 2022). Loosely following previous literature, the analyzed period around COVID-19 pandemic is set to range from the 1st of January 2020 until the 30th of June 2020, as it provides a slightly longer period to be analyzed. It also captures the most volatile periods indicated by the VIX index, which measures the expected volatilities of the S&P 500 index (CBOE 2021).

Table 8 reports the results of the trading portfolio strategies during the COVID-19 subperiod. The results during the period are very heterogenous as, e.g., the annualized mean returns between models vary from a negative return of -30.1% up to a positive return of 19.8%. The only two models to have a positive return during the period were the TF(M) and LSTM(M) models with respective returns of 19.8% and 0.9%. In addition, the LSTM(E) was also able to defeat the buy-and-hold strategy having returns of -3.6% and -4.2%, respectively. However, the additional technical indicators hurt the performance of the TF(E) model greatly, as it recorded an annualized return of -9.00%. The only model again, that benefited from the use of technical indicators was the logistic regression. The LOG(E) model recorded only a slightly negative return of -5.6% during the period whereas the LOG(M) had a return of -30.1%.

The standard deviations of the models were all below the B&H, with the LOG(E) having the lowest standard deviation. The skewness was somewhat negative for all models, except for the LOG(E) model. The excess kurtosis of the TF(M), LSTM(M) and LOG(E) are also close to B&H at just slightly positive values. However, the extended neural networks, as well as the LOG(M) model document kurtosis values well above the others. The maximum drawdowns were mostly the same as in Table 6, since the start of COVID-19 had the largest declines of the whole empirical study. However, the logistic regression models do not achieve the maximum drawdowns in this period.

Table 8: Summary statistics of the trading portfolios for January–June 2020

The table illustrates the summary statistics of all trading portfolios before transaction costs for the out-of-sample period from 1st of January 2020 until 30th of June 2020. The annualized mean return, annualized standard deviation, skewness, excess kurtosis, maximum drawdown (MDD), approximated Value-at-Risk (VaR), approximated expected shortfall (ES) and Sharpe ratio are reported. In addition, the number of transactions made in the out-of-sample period is reported for all the models.

	TF(M)	LSTM(M)	LOG(M)	TF(E)	LSTM(E)	LOG(E)	B&H
Return	0.198	0.009	-0.301	-0.090	-0.036	-0.056	-0.042
St.dev.	0.381	0.410	0.345	0.393	0.391	0.290	0.476
Skewness	-0.217	-0.546	-2.234	-0.801	-0.661	0.414	-0.573
Kurtosis	0.992	1.389	8.608	5.247	3.307	0.926	1.068
MDD	0.221	0.318	0.351	0.299	0.306	0.248	0.346
$\text{VaR}_{0.95}$	-0.038	-0.042	-0.037	-0.041	-0.040	-0.030	-0.049
$\text{VaR}_{0.99}$	-0.055	-0.060	-0.052	-0.058	-0.057	-0.043	-0.069
$\text{ES}_{0.95}$	-0.048	-0.052	-0.046	-0.051	-0.050	-0.037	-0.061
$\text{ES}_{0.99}$	-0.061	-0.067	-0.058	-0.065	-0.064	-0.048	-0.078
Sharpe ratio	0.520	0.021	-0.872	-0.228	-0.091	-0.194	-0.087

Now the reported VaR and expected shortfall are not the historical values as they were previously. The number of observations is too small and thus a more theoretical approach is used. The VaR is calculated as the mean return over the period summed up with the product of the standard deviation and the inverse of the cumulative distribution function of $N(0,1)$ at $1-\alpha$, where $\alpha = 0.95, 0.99$. The expected shortfall is again averaged over smaller increments to arrive at an approximation of the integral over the area. It can be observed that all the values are below the ones of the B&H strategy. As previously, the LOG(E) provides the lowest values on all four metrics. The TF(M) slightly surpasses the LSTM(M), however, the reverse is true for the extended versions.

In terms of the Sharpe ratio, the clear winner is the TF(M) with a Sharpe ratio of 0.520, while the second-best model, LSTM(M), only achieves a score of 0.021. The buy-and-hold strategy is able to beat all the other models with a Sharpe ratio of -0.087. The TF(E) had a poor performance over the period as the Sharpe ratio was only -0.228. Overall, there seems to be time-varying predictability in the stock market that has not been arbitraged away as discussed by Pesaran & Timmermann (1995) and Fischer & Krauss (2018).

Supported by these results, a small long-short experiment on the subperiod is also conducted. The experiment is backed up by the study of Krauss et al. (2017) who note the great performance of the long-short strategy during high volatility periods. The long-short strategy using the predictions of the TF(M) model yield

an annualized return of 39.5% over the half-year period and record a Sharpe ratio of 1.007. Furthermore, the maximum drawdown during the period is only 0.144 which is also well below the results of other models. To visualize the findings of the subperiod analysis, the time series of the trading strategy portfolios with an additional long-short portfolio (TF L-S) are illustrated in Figure 14.

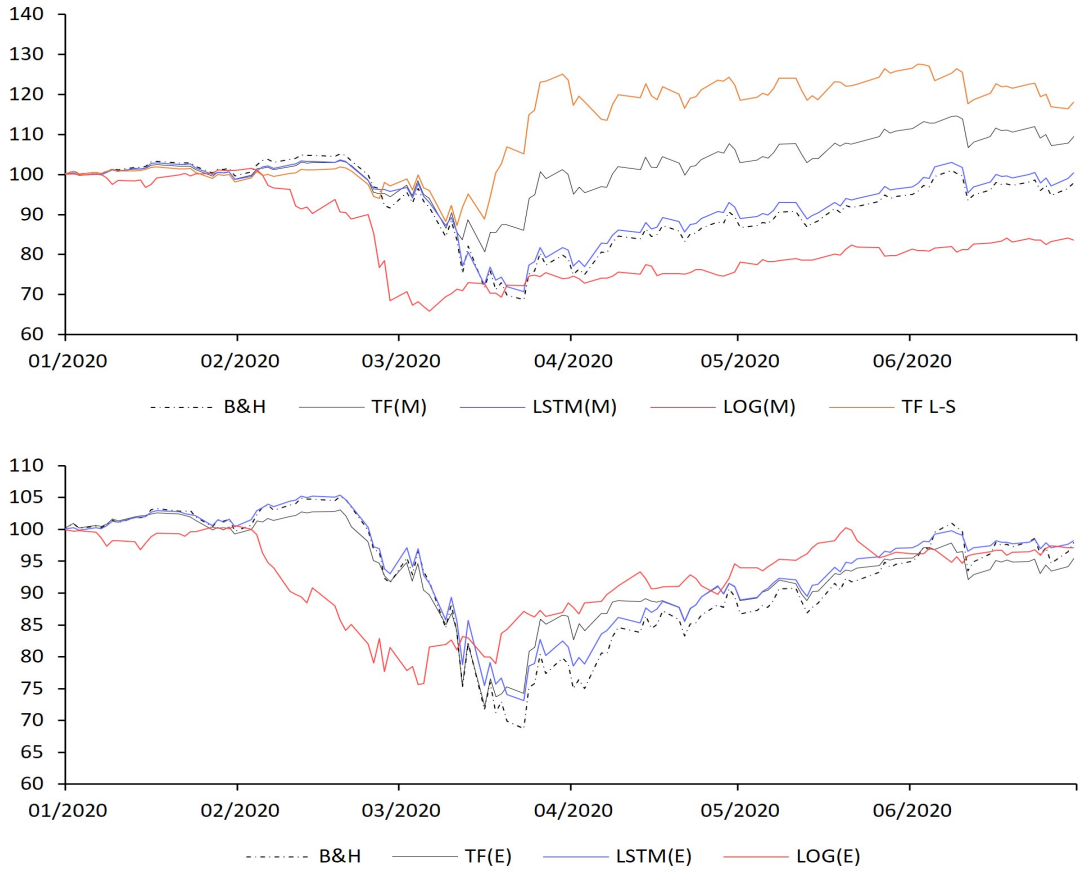

Figure 14: Development of the trading portfolios during COVID-19 pandemic

By observing the time series, it can be seen that the TF(M) model and the long-short strategy are both superb at recognizing the downturn in the market, and can significantly outperform all other models during turbulent markets. Even though the TF(M) also slightly dips at the beginning of the COVID-19 pandemic together with other models, it seems to be able to learn some predictable patterns from the data. However, the extended model that also utilizes the technical indicators seems to be unable to capitalize on the same signals. Both the LSTM models, and also the TF(E) model, seem to almost mimic the behavior of B&H. Though, for example, the LSTM(M) did have a marginally smaller decline in March and as an result it slightly beats the buy-and-hold strategy.

The logistic regression models seem to already decline before the pandemic, and bottom out when the COVID-19 pandemic is just beginning. Thus, the pandemic itself does not seem to bother the performance of the models. The LOG(E), which has benefitted from the technical indicators, actually seems to be the top-performing extended model for some time after mid-March. As the Transformer model was able to utilize the predictability at times of market distress, the findings strongly support the results of Pesaran & Timmermann (1995), Krauss et al. (2017) and Fischer & Krauss (2018).

4.4 Interpretability of the model

The interpretability of the Tranformer model is associated with the attention scores of the network, as discussed at the end of Section 2.2.3. While the information provided by the attention scores is debated, some heatmaps of attention scores are visualized and analyzed here. Moreover, an attempt to utilize the distance metric by Lim et al. (2021) without using volatility data is presented at the end of the section.

The Figure 15 shows six different attention maps attained from the S&P 500 index training period that precedes the validation and out-of-sample periods of 2011 and 2012. Augmented versions of the images can be seen from Appendix IV. Unlike in the other parts of the empirical study, here only one attention head is used for both TF(M) and TF(E) models, as it makes the visualization more interpretable. The attention maps in a), c) and d) are different realizations of the 252x252 attention map that is used to calculate the next day's forecast. The calculation of the attention scores visualized in the attention mapping is shown in Equation 7. The attention map in image b) is a close-up look at the most recent lagged observations of a). The e) and f) provide the average attention maps for the main and extended model, respectively. The formula for these is provided in Equation 8.

The first image a) in the upper left corner shows a visualization of the attention map for an individual arbitrary day. The first image shows that most of the more attended observations seem to focus on the more recent values. There are some white vertical and horizontal lines near the first observations that indicate higher attention scores for the areas. As their attention scores are larger, the model has considered them to be the most important variables for the next prediction.

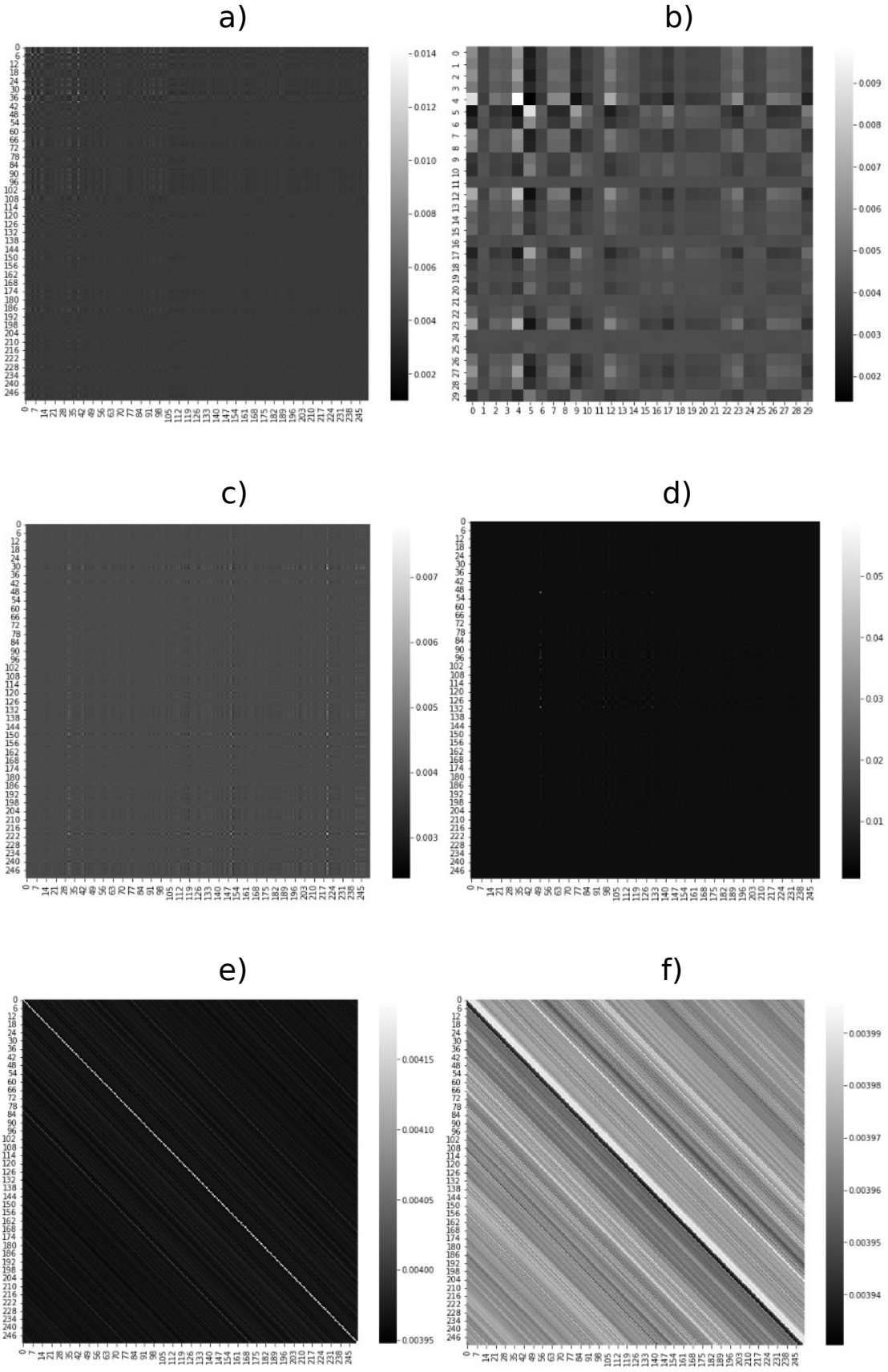

Figure 15: Visualizations of different attention score mappings

The image b) is a magnified version of the attention map in image a). Here only the 30 most recent lagged observations and their attention scores are shown, as now the relevance of each individual day can be demonstrated better. As the 0th observation indicates time t when forecasting for $t+1$, it can be observed that especially the $t-4$ and $t-5$ seem to have relatively high attention scores. Also, their mutual interaction has one of the lowest attention scores, indicating that they might have learned very different dynamics from one another. These values could possibly be used to indicate some kind of weekly seasonal effect. In addition, for example, the $t-12$ and $t-23$ also exhibit similar but not as strong relevance. These are values within a temporal distance of approximately half a month and one month, respectively.

Even though the example in a) and b), where the attention focuses on some of the nearest observations can be more common, different days will get entirely different attention mappings from the lagged observations. The image in c) shows the observations used for a prediction of some other individual day. Here we can see that the attention scores are more evenly distributed across the lagged observations. The overall attention scores shown on the right-side bar are also smaller, as the highest observations only get scores a bit over 0.007, while on a) the highest scores were close to 0.014. Uniformly distributed attention would indicate attention scores slightly under 0.004 throughout. Only a small number of days show slightly higher attention scores like $t-29$ and $t-31$, $t-150$ and $t-151$, and $t-220$. In this case, the values seem more random and do not necessarily have any further explainability.

Opposingly, in image d) the attention score is only focused on very few observations and the rest of the days share low and practically even attention scores. One thing to notice is that the value of the largest attention score gets a very high value of 0.05. The two most highlighted values in the attention map are $t-50$ and $t-131$. It seems that these values are somehow that important, but without knowing the specific days there is nothing more to analyze. The above examples highlight that while the importance of individual lagged observations can be visualized and analyzed, the attention scores change for each observation and thus do not necessarily show the whole picture. To address this issue, the average attention scores over a specific time span can also be visualized. In images e) and f) the average attention scores during the first training period are visualized.

Interestingly, although the individual days show vertical and horizontal lines, the averages over arbitrary periods show diagonal lines. The vertical and horizontal lines can be interpreted as the importance of single days and diagonal lines can be interpreted as the importance of connections within certain temporal distances.

As the key and query matrices learn different representations of the input, the importance of, e.g., $t-1$ on $t-3$ might differ from the importance of $t-3$ on $t-1$. The image e) shows the average attention scores obtained from the main model. It can be observed that the temporal distance of 0, i.e., the days themselves, have the most importance when predicting the results for the next day and other interaction terms are not as relevant. The white line in e) highlights the temporal distance of 0. The attention mapping also seems very symmetrical along the main diagonal.

The image in f) is even more interesting as it highlights the differences between the main model and the extended model. The extended model seems to take into account many of the interaction terms. Based on this picture alone, it could be argued that the extended model learns a fuller picture of the interactions of the independent variables. However, the addition of technical indicators possibly increases overfitting and thus the model does not generalize as well to unseen data where the relationships might differ significantly. It is more generally applicable to only attend to the values of the lagged observations themselves. In addition to the values of the lagged observations themselves, there seems to be high relevance of t with values in the range $[t-1,t-6]$. This can be seen as a thicker white line in the image. The interaction terms with previous observations seem much more important than with more recent observations. This is probably caused by the technical indicators, as they utilize only previous observations to construct their values. As the main model does not explicitly use interaction terms, it attends to the interaction term of $t-i$ on $t-j$ very similarly as to $t-j$ on $t-i$.

Figure 16 showcases the results obtained from the distance metric $\text{dist}(t)$ shown in Equation 9. It is visualized together with the volatility index VIX, as it provides a better visualization counterpart than the S&P 500 index in terms of regime changes. The attention scores used for the regime calculations are from the first training data of the empirical analysis. The significant regimes were chosen to represent instances where the 10-day moving average of the distance scores was above the third quartile, i.e., the 75th percentile. After the regimes were calculated, the areas were shifted five steps backwards in time to represent the center of the 10-day moving average. The usage of moving averages smooths down the areas of significant regimes and makes the graph more interpretable.

The deviations from the average attention scores in the graph can be found near periods of higher volatility. Especially during the financial crisis of 2008–2009, the distance metric captures the volatility regime shifts almost instantaneously with the VIX index. The metric does also highlight periods in the start of 2008 when the VIX index is slightly peaking. It seems, however, that the metric is not as persistent as the periods of high volatility represented by the VIX index. The

model shows no further significant regimes after the start of 2009, even though the VIX stays at elevated levels for more than half a year after that. Thus, the distance metric seems to attend to more sudden increases in volatility like at the start of the financial crisis. However, the distance metric did not attend to the first two small but sudden peaks in 2007.

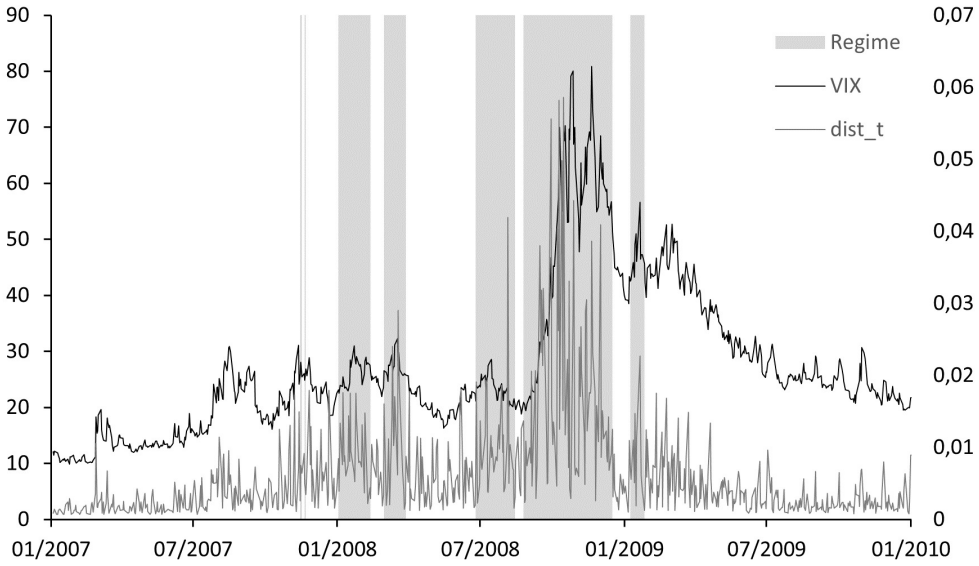

Figure 16: The distance metric versus the VIX index in 2007–2009

5 CONCLUSION

The goal of this thesis was to find out whether a Transformer-based deep learning model is viable for forecasting stock market returns. The research aimed to evaluate the feasibility of the model with comparisons against benchmarks through performance metrics and a trading strategy. The empirical analysis focused on predicting the directional return of four US indices between 2012–2021. Altogether, five research questions focusing on the empirical analysis were considered in this thesis. Here, they are gone through one by one in order to conclude the main findings of the study.

Is the Transformer-based deep learning model a viable tool for financial time series prediction?

Given the earlier research, it seemed that the Transformer model could prove extremely useful at predicting financial time series. However, the results of the studies have not been very robust and leave room for data mining and pure chance to play a part. This study tried to focus more on the robustness of the analysis, as the models were trained three times for each instance to avoid random chance from affecting the results considerably. Also, multiple assets and a rather long time period were used as it should provide stronger evidence of the true potential of the model.

Based on the empirical results of this thesis, the Transformer-based deep learning model seems to be a valid tool to be used in financial time series forecasting. The reported classification accuracies showed levels of over 50% throughout the indices, and overall the model's performance was fairly stable. Similarly, the AUC and MCC metrics appeared positive in every single instance, which is further proof of decent and stable performance. Furthermore, the Pesaran-Timmermann tests showed that the Transformer model had some significant directional predictability. The training time of the model was also quite low considering the complex network and the number of trainable parameters. As such, the model could possibly be also used for applications with high-frequency data.

Does the model achieve higher performance compared to the benchmark models?

In previous literature, the Transformer model has been able to beat all benchmark models, including LSTMs, in terms of accuracy, as well as in terms of other metrics. The empirical results here partly state otherwise, as the main and extended LSTM models seemed to post higher accuracies throughout the indices.

However, when the imbalance of the data is considered, at least the main Transformer model is capable of achieving higher performance than the benchmark models. The Transformer is superior to the benchmarks in terms of AUC and MCC. In addition, no other model had significant directional accuracy according to the Pesaran-Timmermann test. Thus, it can be concluded that the Transformer was able to achieve higher performance, even though the classification accuracies were somewhat lower.

Do the model predictions contribute to any considerable financial gains over other methods?

Previous machine learning literature has shown that trading strategies based on the predicted probabilities have usually beaten the benchmark models and the buy-and-hold strategy, even after transaction costs. The empirical results here indicate that the main Transformer model does generate financial gains exceeding the benchmarks, at least before transaction costs are considered. However, when transaction costs are taken into account, the benchmark of a simple buy-and-hold strategy is superior for the full period, even though the transactions are already reduced by the trading strategy. It has to be noted, however, that risk-adjusted returns indicated by the Sharpe ratio are higher for the Transformer model than for the overall market.

It is also noticeable, especially from the Russell index, and from the time series graphs over the out-of-sample period, that the Transformer model does exhibit more profitable features in times of higher volatility. Thus, an additional analysis was conducted for the early months of the COVID-19 pandemic as the markets experienced unusually high volatility in the period. These results show that the Transformer model can be extremely profitable in times of market distress.

Does the inclusion of technical indicators help the model performance?

The academic literature on the effectiveness of technical indicators has been rather mixed. It was hypothesized, however, that the inclusion of technical indicators would benefit the model greatly as the input transformation matrices would gain a finer representation of the model inputs. However, it seems that the inclusion of technical indicators reduces the performance of both the Transformer and the LSTM model in most situations.

Based on the visualizations of the average attention scores in the results, it may be that the technical indicators provided the model with too much noise, as the information presented by them was too elaborate to be generalized to new data.

As such, the model does learn the data that it is fitted to extremely well, but fails to show similar performance out-of-sample. Possibly, the model configuration was wrong and better regularization methods should have been used. However, similar results from LSTM might point to the fact that they provided too noisy information to be applicable.

Can the black box decisions of the model be interpreted to some degree?

A few previous studies have shown some possible visualizations of the Transformer model for time series. Thus, the hypothesis here was that some interpretability through visualizations might be offered by the Transformer model. Using the methods presented in the previous studies, as well as a more original idea of using the average attention scores over an arbitrary period, some interpretability can be shown. Nonetheless, the actual takeaways from the explanations might not be satisfying as it is hard to provide any fundamental reasons behind the model decisions. However, the distance metric did seem to provide some insight into the model behavior. The model seems to give higher attention to certain observations during times of higher volatility, which can also partly explain the excellent trading performance in early 2020.

In summary, the Transformer model seemed to work as a predictive model. Even though the results were not outstanding, it still defeated the benchmarks more often than not. It seems that any daily return predictability, at least in the recent years, is very hard to find and oftentimes, decent results might be a result of chance. However, the favorable trading portfolio development during the COVID-19 pandemic was evident in all stock indices and it provides some evidence of the Transformer models' predictive power.

As the Transformer model is still a recent invention and it has so few implementations for financial time series, future research should be conducted. One could experiment with different attention mechanisms that have been developed to, e.g., reduce the memory attributes. In fact, almost every single layer could, in theory, be replaced and experimented on. Besides the changes to model configurations, weekly or monthly observations could be employed as they might possess more directional predictability. The model could also be experimented with other data sets than US stock indices where there might be less efficiency and more volatility to be exploited. Furthermore, some fundamental or macroeconomic explanatory variables could be included to see whether they would improve the model performance.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016) Tensorflow: A System for Large-Scale Machine Learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283.

Abu-Mostafa, Y. S., Magdon-Ismail, M. – Lin, H.-T. (2012) *Learning From Data: A Short Course*. AMLBook.com, New York.

Agresti, A. (2015) *Foundations of linear and generalized linear models*. Wiley: Hoboken, NJ.

Al-Awadhi, A. M., Alsaifi, K., Al-Awadhi, A. – Alhammadi, S. (2020) Death and contagious infectious diseases: Impact of the COVID-19 virus on stock market returns. *Journal of Behavioral and Experimental Finance*, Vol. 27, 100326–100326.

Amemiya, T. (1981) Qualitative Response Models: A Survey. *Journal of Economic Literature*, Vol. 19 (4), 1483–1536.

Andreou, E., Pittis, N. – Spanos, A. (2001) On Modelling Speculative Prices: The Empirical Literature. *Journal of Economic Surveys*, Vol. 15 (2), 187–220.

Bahdanau, D., Cho, K. – Bengio, Y. (2014) Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Baker, S. R., Bloom, N., Davis, S. J., Kost, K., Sammon, M. – Viratyosin, T. (2020) The Unprecedented Stock Market Reaction to COVID-19. *The Review of Asset Pricing Studies*, Vol. 10 (4), 742–758.

Ballings, M., Van den Poel, D., Hespeels, N. – Gryp, R. (2015) Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, Vol. 42 (20), 7046–7056.

Bao, W., Yue, J. – Rao, Y. (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, Vol. 12 (7), e0180944–e0180944.

Bekiros, S. D. – Georgoutsos, D. A. (2007) Evaluating direction-of-change forecasting: Neurofuzzy models vs. neural networks. *Mathematical and Computer Modelling*, Vol. 46 (1-2), 38–46.

Bengio, Y. (2012) Practical Recommendations for Gradient-Based Training of Deep Architectures. In *Neural networks: Tricks of the trade*, 437–478, Springer.

Bodie, Z., Kane, A. – Marcus, A. (2003) *Essentials of Investments*. 5th edition. McGraw-Hill Irwin, Boston.

Boughorbel, S., Jarray, F. – El-Anbari, M. (2017) Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PloS one*, Vol. 12 (6), e0177678–e0177678.

CBOE (2021) Cboe Volatility Index. White paper. <https://cdn.cboe.com/resources/vix/vixwhite.pdf>, retrieved 11.5.2022.

Chefer, H., Gur, S. – Wolf, L. (2021) Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 782–791.

Chevapatrakul, T. (2013) Return sign forecasts based on conditional risk: Evidence from the UK stock market index. *Journal of Banking & Finance*, Vol. 37 (7), 2342–2353.

Chicco, D. – Jurman, G. (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, Vol. 21 (1), 1–13.

Cho, K., Van Merriënboer, B., Bahdanau, D. – Bengio, Y. (2014) On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv preprint arXiv:1409.1259*.

Chollet, F. (2017) *Deep Learning with Python*. Manning Publications: New York, NY.

Chollet, F. (2021) *Deep Learning with Python*. 2nd edition. Manning Publications: New York, NY.

Chollet, F. et al. (2015) Keras. <https://github.com/fchollet/keras>, retrieved 8.5.2022.

Chowdhury, E. K., Dhar, B. K. – Stasi, A. (2022) Volatility of the US stock market and business strategy during COVID-19. *Business Strategy & Development*, 1–11.

Christoffersen, P. F. – Diebold, F. X. (2006) Financial Asset Returns, Direction-of-Change Forecasting, and Volatility Dynamics. *Management Science*, Vol. 52 (8), 1273–1287.

Christoffersen, P. F., Diebold, F. X., Mariano, R., Tay, A. – Tse, Y. (2007) Direction-of-Change Forecasts Based on Conditional Variance, Skewness and Kurtosis Dynamics: International Evidence. *Journal of Financial Forecasting*, Vol. 1, 1–22.

Claesen, M. – De Moor, B. (2015) Hyperparameter Search in Machine Learning. *arXiv preprint arXiv:1502.02127*.

Courtault, J.-M., Kabanov, Y., Bru, B., Crépel, P., Lebon, I. – Le Marchand, A. (2000) Louis Bachelier on the centenary of Théorie de la spéculation. *Mathematical Finance*, Vol. 10 (3), 339–353.

Cowles, A. (1933) Can Stock Market Forecasters Forecast? *Econometrica*, Vol. 1 (3), 309–324.

De Bondt, W. F. – Thaler, R. (1985) Does the Stock Market Overreact? *The Journal of Finance*, Vol. 40 (3), 793–805.

Diebold, F. X. – Mariano, R. S. (2002) Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, Vol. 20 (1), 134–144.

Ding, Q., Wu, S., Sun, H., Guo, J. – Guo, J. (2020) Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. In *29th International Joint Conferences on Artificial Intelligence (IJCAI)*, 4640–4646.

Dong, L., Xu, S. – Xu, B. (2018) Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5884–5888.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. (2020) An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*.

Fama, E. F. (1970) Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, Vol. 25 (2), 383–417.

Fama, E. F. (1991) Efficient Capital Markets: II. *The Journal of Finance*, Vol. 46 (5), 1575–1617.

Fama, E. F. (1998) Market efficiency, long-term returns, and behavioral finance. *Journal of Financial Economics*, Vol. 49 (3), 283–306.

Fischer, T. – Krauss, C. (2018) Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, Vol. 270 (2), 654–669.

Géron, A. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems.* 2nd ed. O'Reilly: Sebastapol, CA.

Gers, F., Schmidhuber, J. – Cummins, F. (2000) Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, Vol. 12 (10), 2451–2471.

Glorot, X. – Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–456.

Goodfellow, I., Bengio, Y. – Courville, A. (2016) *Deep learning.* MIT Press, London.

Graves, A. (2012) Supervised Sequence Labelling with Recurrent Neural Networks. *Studies in Computational Intelligence*, Vol. 385, 1–131.

Grudnitski, G. – Osburn, L. (1993) Forecasting S&P and gold futures prices: An application of neural networks. *Journal of Futures Markets*, Vol. 13 (6), 631–643.

Gu, A. Y. (2003) The declining January effect: evidences from the U.S. equity markets. *The Quarterly Review of Economics and Finance*, Vol. 43 (2), 395–404.

Halimu, C., Kasem, A. – Newaz, S. S. (2019) Empirical comparison of area under ROC curve (AUC) and Mathew correlation coefficient (MCC) for evaluating machine learning algorithms on imbalanced datasets for binary classification. In *Proceedings of the 3rd international conference on machine learning and soft computing*, 1–6.

Haugen, R. A. – Jorion, P. (1996) The January Effect: Still There After All These Years. *Financial Analysts Journal*, Vol. 52 (1), 27–31.

Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation*. Prentice Hall: Upper Saddle River, NJ.

Henrique, B., Sobreiro, V. – Kimura, H. (2019) Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, Vol. 124, 226–251.

Hochreiter, S., Bengio, Y., Frasconi, P. – Schmidhuber, J. (2001) Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press.

Hochreiter, S. – Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, Vol. 9 (8), 1735–1780.

Horowitz, J. L. – Savin, N. (2001) Binary response models: Logits, probits and semiparametrics. *Journal of Economic Perspectives*, Vol. 15 (4), 43–56.

Hsu, P.-H., Hsu, Y.-C. – Kuan, C.-M. (2010) Testing the predictive ability of technical analysis using a new stepwise test without data snooping bias. *Journal of Empirical Finance*, Vol. 17 (3), 471–484.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M. – Eck, D. (2018) Music Transformer. *arXiv preprint arXiv:1809.04281*.

Hyndman, R. J. – Athanasopolous, G. (2018) *Forecasting: Principles and Practice*. 2nd edition. OTexts, Melbourne, <otexts.com/fpp2>, retrieved 16.04.2022.

Jain, S. – Wallace, B. C. (2019) Attention is not Explanation. *arXiv preprint arXiv:1902.10186*.

Jegadeesh, N. – Titman, S. (1993) Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance*, Vol. 48 (1), 65–91.

Jiang, X., Pan, S., Jiang, J. – Long, G. (2018) Cross-Domain Deep Learning Approach For Multiple Financial Market Prediction. In *2018 International joint conference on neural networks (IJCNN)*, 1–8.

Johnson, K. (2021) pandas-ta. <https://https://github.com/twopirllc/pandas-ta>, retrieved 8.5.2022.

Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P. – Brubaker, M. (2019) Time2Vec: Learning a Vector Representation of Time. *arXiv preprint arXiv:1907.05321*.

Kendall, M. G. – Hill, A. B. (1953) The Analysis of Economic Time-Series – Part I: Prices. *Journal of the Royal Statistical Society*, Vol. 116 (1), 11–34.

Khandelwal, U., He, H., Qi, P. – Jurafsky, D. (2018) Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. *arXiv preprint arXiv:1805.04623*.

Knüpfer, S. – Puttonen, V. (2018) *Moderni rahoitus*. 10. uud. p. Alma Talent, Helsinki.

Krauss, C., Do, X. A. – Huck, N. (2017) Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, Vol. 259 (2), 689–702.

Leitch, G. – Tanner, J. E. (1991) Economic Forecast Evaluation: Profits Versus the Conventional Error Measures. *The American Economic Review*, Vol. 81 (3), 580–590.

Leung, M. T., Daouk, H. – Chen, A.-S. (2000) Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of Forecasting*, Vol. 16 (2), 173–190.

Li, H., Xu, Z., Taylor, G., Studer, C. – Goldstein, T. (2018) Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, 6391–6401.

Lim, B., Arik, S. O., Loeff, N. – Pfister, T. (2021) Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, Vol. 37 (4), 1748–1764.

Lin, M., Qiang, C. – Yan, S. (2013) Network in network. *arXiv preprint arXiv:1312.4400*.

Lo, A. W., Mamaysky, H. – Wang, J. (2000) Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, Vol. 55 (4), 1705–1765.

Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E. – Winkler, R. (1982) The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, Vol. 1 (2), 111–153.

Makridakis, S., Spiliotis, E. – Assimakopoulos, V. (2018a) The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, Vol. 34 (4), 802–808.

Makridakis, S., Spiliotis, E. – Assimakopoulos, V. (2018b) Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one*, Vol. 13 (3), e0194889.

Makridakis, S., Spiliotis, E. – Assimakopoulos, V. (2021a) The M5 accuracy competition: Results, findings and conclusions. *International Journal of Forecasting*.

Makridakis, S., Spiliotis, E., Assimakopoulos, V., Chen, V., Gaba, A., Tsetlin, I. – Winkler, R. (2021b) The M5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*.

Malkiel, B. G. (2003) The Efficient Market Hypothesis and Its Critics. *Journal of Economic Perspectives*, Vol. 17 (1), 59–82.

Marsland, S. (2015) *Machine Learning: An Algorithmic Perspective*. 2nd edition. CRC Press: Boca Raton, FL.

Mazur, M., Dang, M. – Vega, M. (2021) COVID-19 and the March 2020 Stock Market Crash. Evidence from SP1500. *Finance Research Letters*, Vol. 38, 101690–101690.

McLean, R. D. – Pontiff, J. (2016) Does Academic Research Destroy Stock Return Predictability? *The Journal of Finance*, Vol. 71 (1), 5–32.

Nagel, S. (2021) *Machine Learning in Asset Pricing*. Princeton University Press: Princeton, NJ.

Neely, C. J., Rapach, D. E., Tu, J. – Zhou, G. (2014) Forecasting the Equity Risk Premium: The Role of Technical Indicators. *Management Science*, Vol. 60 (7), 1772–1791.

Nelson, D. M., Pereira, A. C. – De Oliveira, R. A. (2017) Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)*, 1419–1426.

Nevasalmi, L. (2020) Forecasting multinomial stock returns using machine learning methods. *The Journal of Finance and Data Science*, Vol. 6, 86–106.

Nyberg, H. (2011) Forecasting the direction of the US stock market with dynamic binary probit models. *International Journal of Forecasting*, Vol. 27 (2), 561–578.

Nyberg, H. – Pönkä, H. (2016) International sign predictability of stock returns: The role of the United States. *Economic Modelling*, Vol. 58, 323–338.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L. et al. (2019) Keras Tuner. <https://github.com/keras-team/keras-tuner>, retrieved 10.4.2022.

Ozbayoglu, M., Gudelek, U. – Sezer, O. (2020) Deep learning for financial applications : A survey. *Applied Soft Computing*, Vol. 93, 106384.

Park, C.-H. – Irwin, S. H. (2007) What Do We Know About the Profitability of Technical Analysis? *Journal of Economic Surveys*, Vol. 21 (4), 786–826.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Å., Shazeer, N., Ku, A. – Tran, D. (2018) Image Transformer. *arXiv preprint arXiv:1802.05751*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, Vol. 12, 2825–2830.

Pesaran, M. H. – Timmermann, A. (1992) A Simple Nonparametric Test of Predictive Performance. *Journal of Business & Economic Statistics*, Vol. 10 (4), 461–465.

Pesaran, M. H. – Timmermann, A. (1995) Predictability of Stock Returns: Robustness and Economic Significance. *The Journal of Finance*, Vol. 50 (4), 1201–1228.

Qi, M. – Maddala, G. (1999) Economic factors and the stock market: a new perspective. *Journal of Forecasting*, Vol. 18 (3), 151–166.

Qiu, M. – Song, Y. (2016) Predicting the Direction of Stock Market Index Movement Using an Optimized Artificial Neural Network Model. *PloS one*, Vol. 11 (5), e0155133.

Ramachandran, P., Zoph, b. – Le, Q. V. (2017) Searching for Activation Functions. *arXiv preprint arXiv:1710.05941*.

Ramos-Pérez, E., Alonso-González, P. J. – Núñez-Velázquez, J. J. (2021) Multi-Transformer: A New Neural Network-Based Architecture for Forecasting S&P Volatility. *Mathematics*, Vol. 9 (15), 1794.

Rapach, D. – Zhou, G. (2013) Forecasting stock returns. In *Handbook of Economic Forecasting*, Vol. 2, 328–383, Elsevier.

Rothman, D. (2021) *Transformers for Natural Language Processing*. Packt Publishing, Birmingham.

Shiller, R. J. (2000) *Irrational Exuberance*. Princeton University Press: Princeton, NJ.

Shmueli, G. (2010) To Explain or to Predict? *Statistical Science*, Vol. 25 (3), 289–310.

Siami-Namini, S., Tavakoli, N. – Namin, A. S. (2018) A comparison of ARIMA and LSTM in forecasting time series. In *17th IEEE international conference on machine learning and applications (ICMLA)*, 1394–1401, IEEE.

Skabar, A. (2013) Direction-of-Change Financial Time Series Forecasting using a Similarity-Based Classification Model. *Journal of Forecasting*, Vol. 32 (5), 409–422.

Tay, F. E. – Cao, L. (2001) Application of support vector machines in financial time series forecasting. *Omega*, Vol. 29 (4), 309–317.

Tsay, R. S. (2010) *Analysis of financial time series*. 3rd edition. Wiley: Hoboken, NJ.

Tsay, R. S. – Chen, R. (2019) *Nonlinear time series analysis*. Wiley: Hoboken, NJ.

Vasilev, I. (2019) *Advanced Deep Learning with Python: Design and Implement Advanced Next-Generation AI Solutions Using TensorFlow and Pytorch*. Packt Publishing, Birmingham.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. – Polosukhin, I. (2017) Attention Is All You Need. In *31st Conference on Neural Information Processing Systems (NIPS)*.

Welch, I. – Goyal, A. (2008) A Comprehensive Look at the Empirical Performance of Equity Premium Prediction. *The Review of Financial Studies*, Vol. 21 (4), 1455–1508.

White, H. (1988) Economic prediction using neural networks: The case of IBM daily stock returns. In *ICNN*, 451–458.

Wiegreffe, S. – Pinter, Y. (2019) Attention is not not Explanation. *arXiv preprint arXiv:1908.04626*.

Wu, N., Green, B., Ben, X. – O'Banion, S. (2020) Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *arXiv preprint arXiv:2001.08317*.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L. – Liu, T. (2020) On Layer Normalization in the Transformer Architecture. In *Proceedings of the 37th International Conference on Machine Learning*, 10524–10533.

Yoo, J., Soun, Y., Park, Y.-c. – Kang, U. (2021) Accurate Multivariate Stock Movement Prediction via Data-Axis Transformer with Multi-Level Contexts. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2037–2045.

Zhang, A., Lipton, Z. C., Li, M. – Smola, A. J. (2021) *Dive into Deep Learning*. Draft version.

Zhang, Q., Qin, C., Zhang, Y., Bao, F., Zhang, C. – Liu, P. (2022) Transformer-based attention network for stock movement prediction. *Expert Systems with Applications*, Vol. 202.

Zhong, X. – Enke, D. (2019) Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, Vol. 5 (1), 1–20.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H. – Zhang, W. (2021) Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of AAAI*.

APPENDIX I

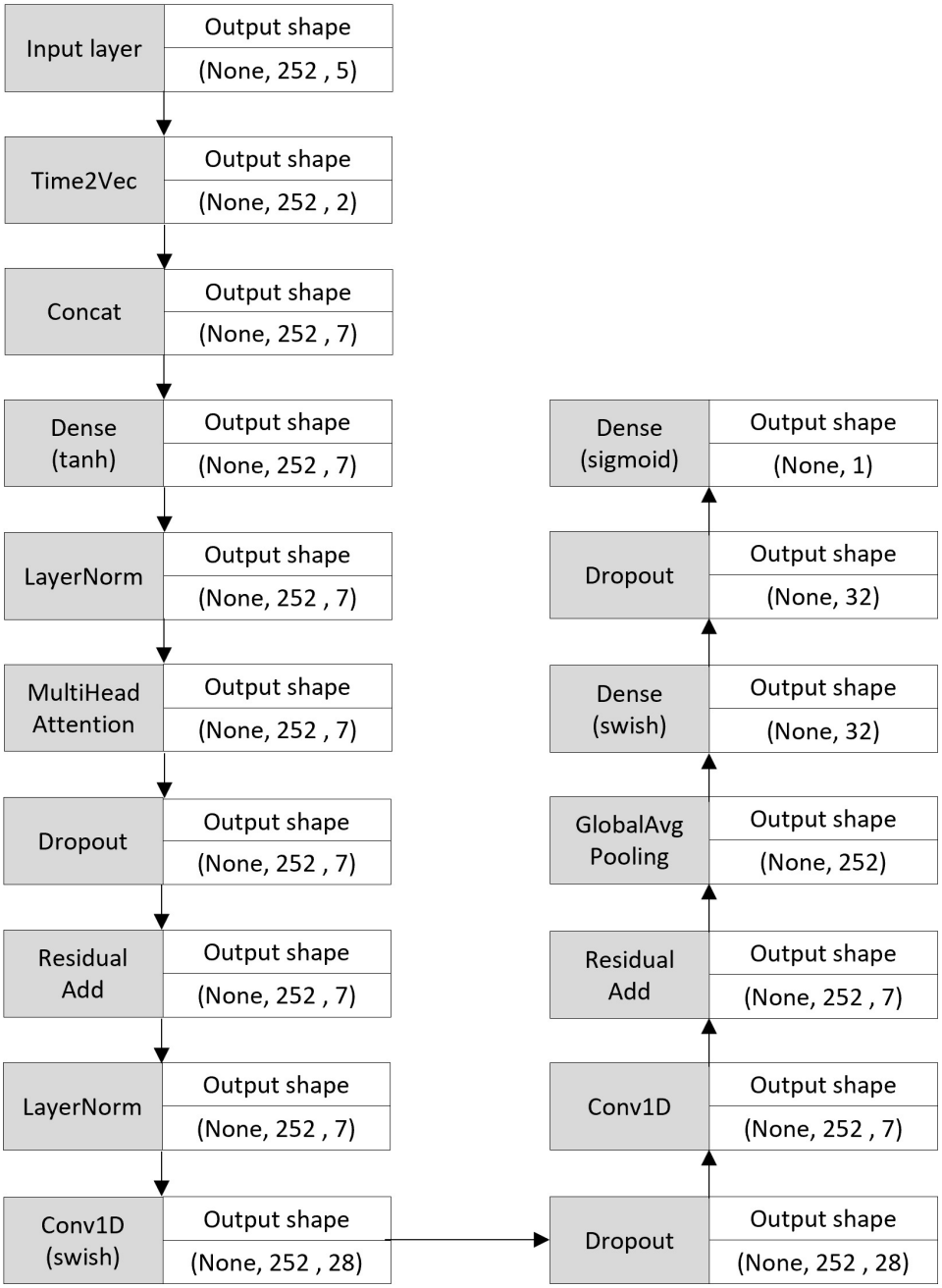

Figure 17: The full model architecture of the TF(M) model

APPENDIX II

Table 9: Full results of TF(M) model for the S&P 500

		2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	Avg.
Seed #3	Acc.	0.548	0.584	0.567	0.524	0.528	0.494	0.496	0.530	0.482	0.518	0.527
	AUC	0.533	0.519	0.534	0.535	0.518	0.474	0.491	0.508	0.450	0.489	0.505
	MCC	0.078	0.066	0.076	0.077	0.039	-0.053	-0.018	0.016	-0.110	-0.023	0.015
Seed #6	Acc.	0.544	0.592	0.556	0.468	0.556	0.554	0.524	0.474	0.522	0.585	0.537
	AUC	0.529	0.523	0.497	0.484	0.554	0.513	0.521	0.463	0.506	0.555	0.515
	MCC	0.069	0.092	-0.012	-0.038	0.108	0.032	0.042	-0.073	0.013	0.122	0.035
Seed #9	Acc.	0.524	0.528	0.516	0.480	0.524	0.582	0.552	0.510	0.553	0.522	0.529
	AUC	0.511	0.481	0.488	0.490	0.513	0.563	0.544	0.496	0.524	0.482	0.509
	MCC	0.026	-0.045	-0.025	-0.022	0.030	0.130	0.093	-0.008	0.052	-0.044	0.019
Avg.	Acc.	0.539	0.568	0.546	0.491	0.536	0.543	0.524	0.505	0.519	0.542	0.531
	AUC	0.524	0.508	0.506	0.503	0.528	0.517	0.519	0.489	0.494	0.509	0.510
	MCC	0.058	0.038	0.013	0.006	0.059	0.036	0.039	-0.021	-0.015	0.018	0.023

Table 10: Full results of TF(M) model for the Dow Jones Industrial Average

		2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	Avg.
Seed #3	Acc.	0.476	0.548	0.520	0.524	0.556	0.554	0.484	0.510	0.490	0.522	0.518
	AUC	0.479	0.508	0.502	0.528	0.542	0.540	0.478	0.491	0.466	0.496	0.503
	MCC	-0.046	0.019	0.004	0.057	0.088	0.081	-0.045	-0.019	-0.078	-0.010	0.005
Seed #6	Acc.	0.496	0.464	0.492	0.480	0.524	0.614	0.488	0.470	0.530	0.542	0.510
	AUC	0.499	0.454	0.464	0.488	0.522	0.573	0.484	0.458	0.518	0.514	0.497
	MCC	-0.003	-0.092	-0.082	-0.026	0.044	0.176	-0.033	-0.085	0.036	0.032	-0.003
Seed #9	Acc.	0.464	0.568	0.512	0.488	0.524	0.562	0.548	0.566	0.518	0.506	0.526
	AUC	0.467	0.523	0.500	0.495	0.510	0.545	0.534	0.542	0.495	0.475	0.509
	MCC	-0.072	0.057	0.000	-0.010	0.020	0.091	0.076	0.087	-0.011	-0.059	0.018
Avg.	Acc.	0.479	0.527	0.508	0.497	0.534	0.576	0.507	0.515	0.513	0.523	0.518
	AUC	0.482	0.495	0.489	0.504	0.525	0.553	0.499	0.497	0.493	0.495	0.503
	MCC	-0.040	-0.006	-0.026	0.007	0.051	0.116	0.000	-0.005	-0.018	-0.012	0.007

Table 11: Full results of TF(M) model for the Nasdaq Composite

		2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	Avg.
Seed #3	Acc.	0.524	0.544	0.536	0.508	0.516	0.546	0.528	0.494	0.545	0.605	0.535
	AUC	0.519	0.491	0.510	0.502	0.498	0.498	0.519	0.461	0.499	0.577	0.508
	MCC	0.040	-0.020	0.022	0.004	-0.004	-0.004	0.040	-0.084	-0.002	0.173	0.016
Seed #6	Acc.	0.520	0.556	0.516	0.504	0.575	0.550	0.524	0.542	0.522	0.573	0.538
	AUC	0.512	0.518	0.477	0.498	0.572	0.484	0.512	0.522	0.487	0.548	0.513
	MCC	0.028	0.040	-0.054	-0.004	0.144	-0.037	0.026	0.045	-0.026	0.105	0.027
Seed #9	Acc.	0.500	0.576	0.524	0.476	0.508	0.538	0.580	0.550	0.573	0.522	0.535
	AUC	0.493	0.514	0.495	0.469	0.488	0.492	0.569	0.523	0.502	0.484	0.503
	MCC	-0.014	0.037	-0.010	-0.070	-0.027	-0.018	0.147	0.049	0.004	-0.039	0.006
Avg.	Acc.	0.515	0.559	0.525	0.496	0.533	0.544	0.544	0.529	0.547	0.567	0.536
	AUC	0.508	0.508	0.494	0.490	0.519	0.491	0.534	0.502	0.496	0.536	0.508
	MCC	0.018	0.019	-0.014	-0.024	0.038	-0.020	0.071	0.003	-0.008	0.080	0.016

Table 12: Full results of TF(M) model for the Russell 2000

		2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	Avg.
Seed #3	Acc.	0.492	0.468	0.476	0.548	0.548	0.506	0.564	0.550	0.478	0.498	0.513
	AUC	0.493	0.446	0.478	0.550	0.532	0.492	0.560	0.538	0.456	0.491	0.504
	MCC	-0.013	-0.111	-0.043	0.100	0.067	-0.016	0.122	0.078	-0.095	-0.019	0.007
Seed #6	Acc.	0.516	0.536	0.472	0.508	0.544	0.530	0.524	0.494	0.506	0.542	0.517
	AUC	0.518	0.511	0.456	0.514	0.544	0.511	0.516	0.481	0.503	0.536	0.509
	MCC	0.037	0.024	-0.092	0.030	0.087	0.024	0.034	-0.040	0.005	0.076	0.019
Seed #9	Acc.	0.520	0.580	0.500	0.464	0.528	0.510	0.508	0.542	0.510	0.522	0.518
	AUC	0.522	0.567	0.505	0.468	0.513	0.497	0.505	0.531	0.485	0.512	0.510
	MCC	0.044	0.135	0.009	-0.066	0.026	-0.007	0.010	0.063	-0.033	0.029	0.021
Avg.	Acc.	0.509	0.528	0.483	0.507	0.540	0.515	0.532	0.529	0.498	0.521	0.516
	AUC	0.511	0.508	0.480	0.511	0.529	0.500	0.527	0.517	0.481	0.513	0.508
	MCC	0.023	0.016	-0.042	0.021	0.060	0.000	0.055	0.034	-0.041	0.029	0.016

APPENDIX III

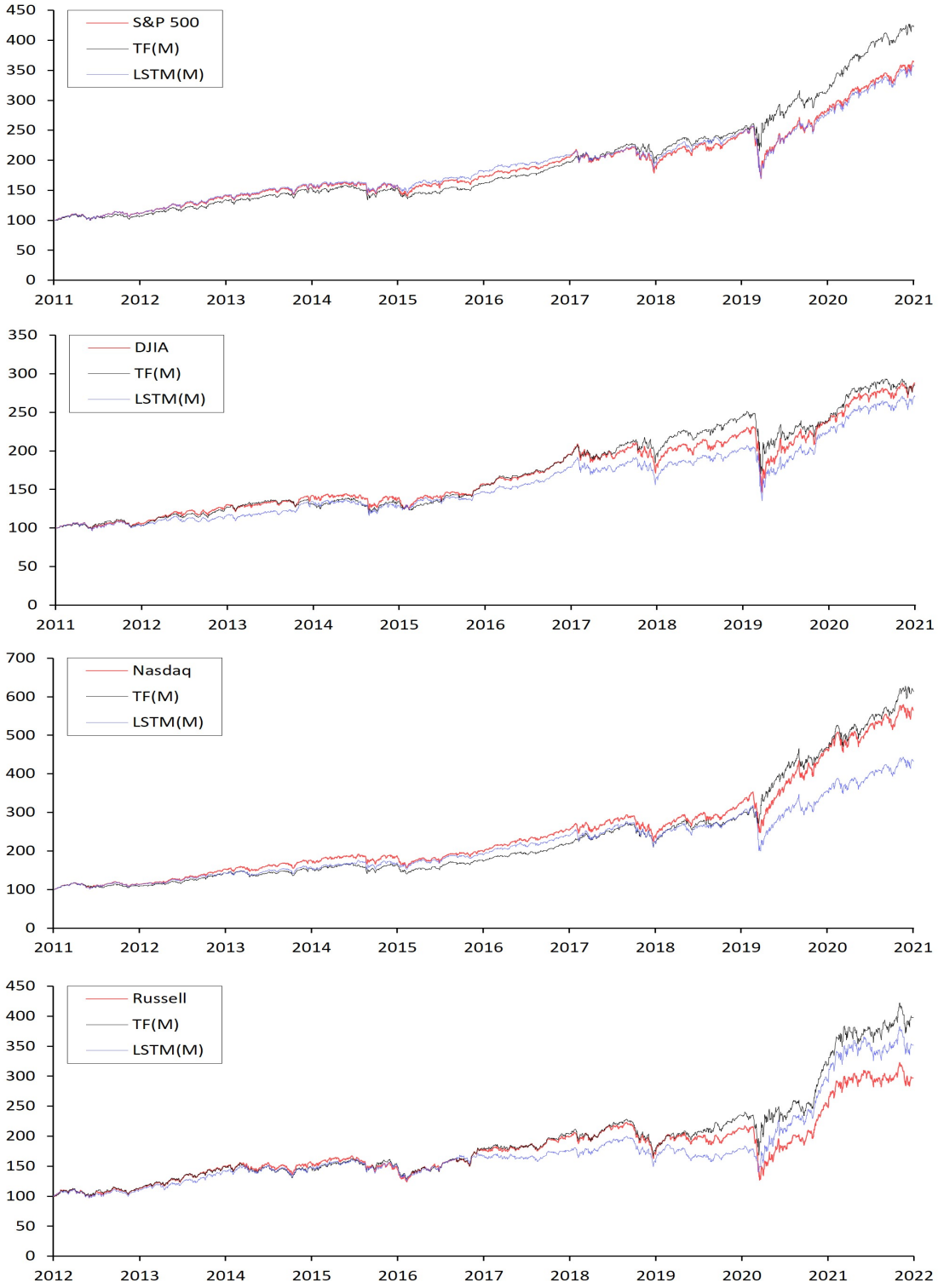

Figure 18: The development of the individual trading strategy portfolios 2012–2021

APPENDIX IV

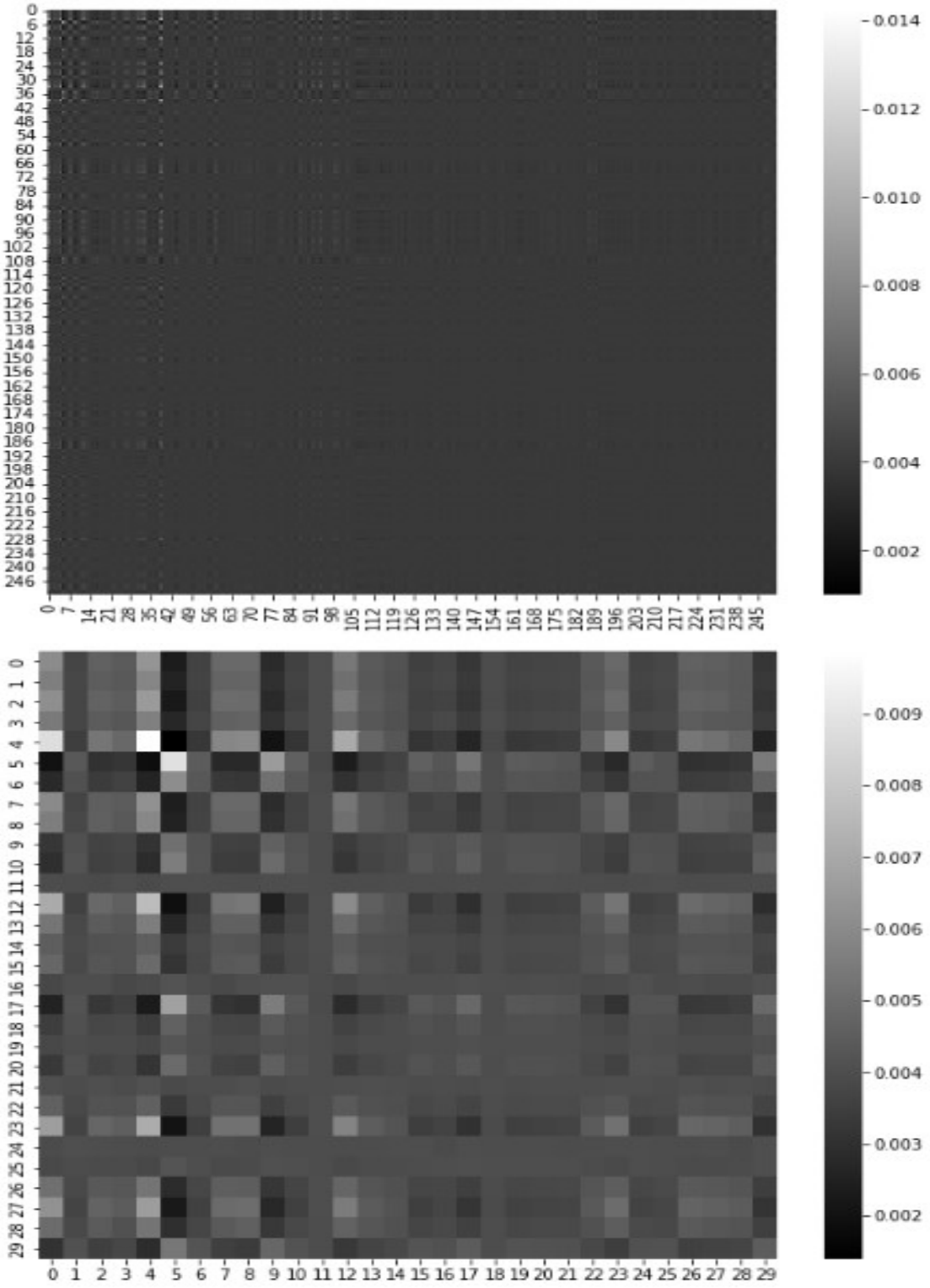

Figure 19: The augmented versions of images a) and b) from Figure 15

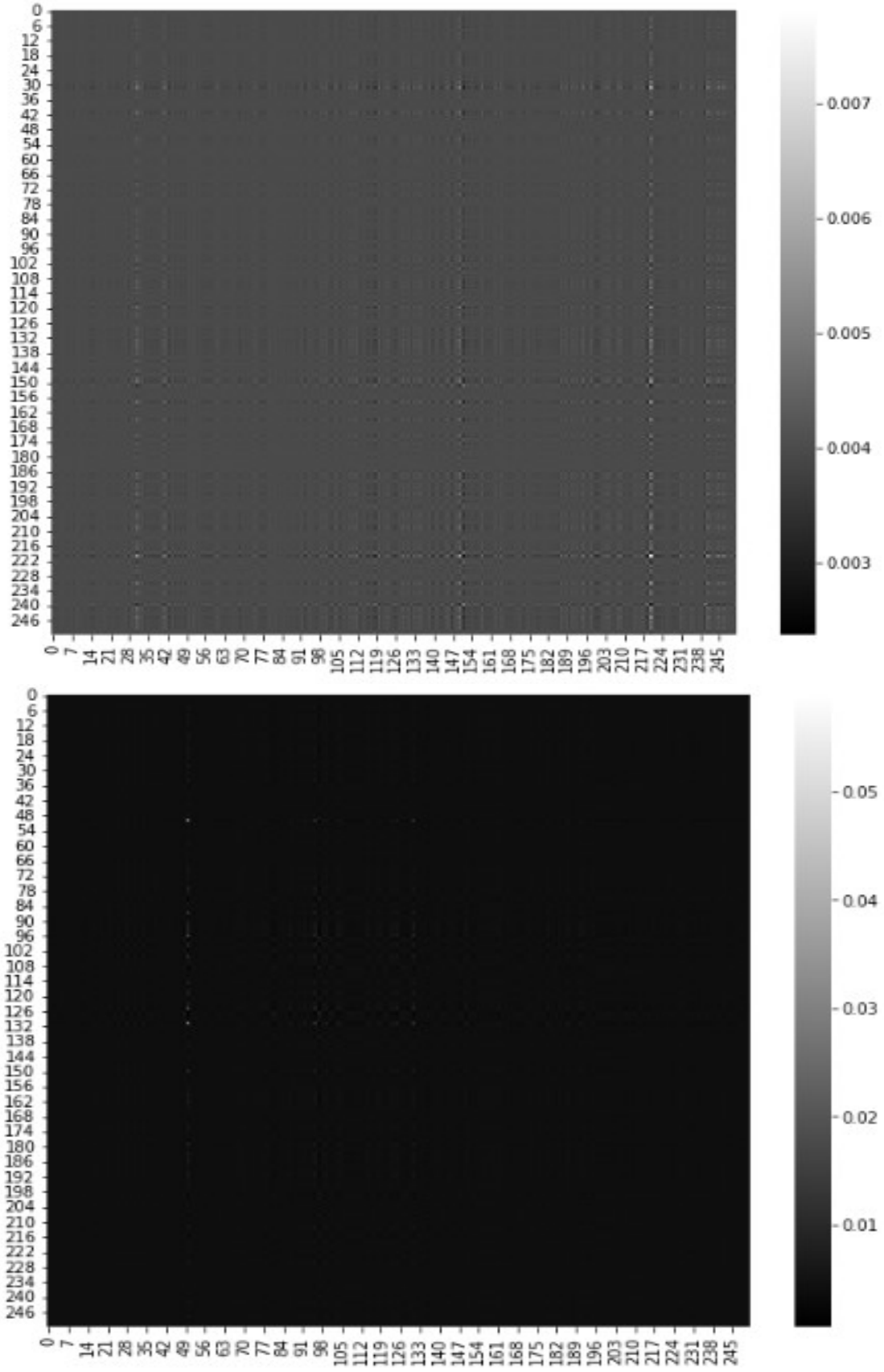

Figure 20: The augmented versions of images c) and d) from Figure 15

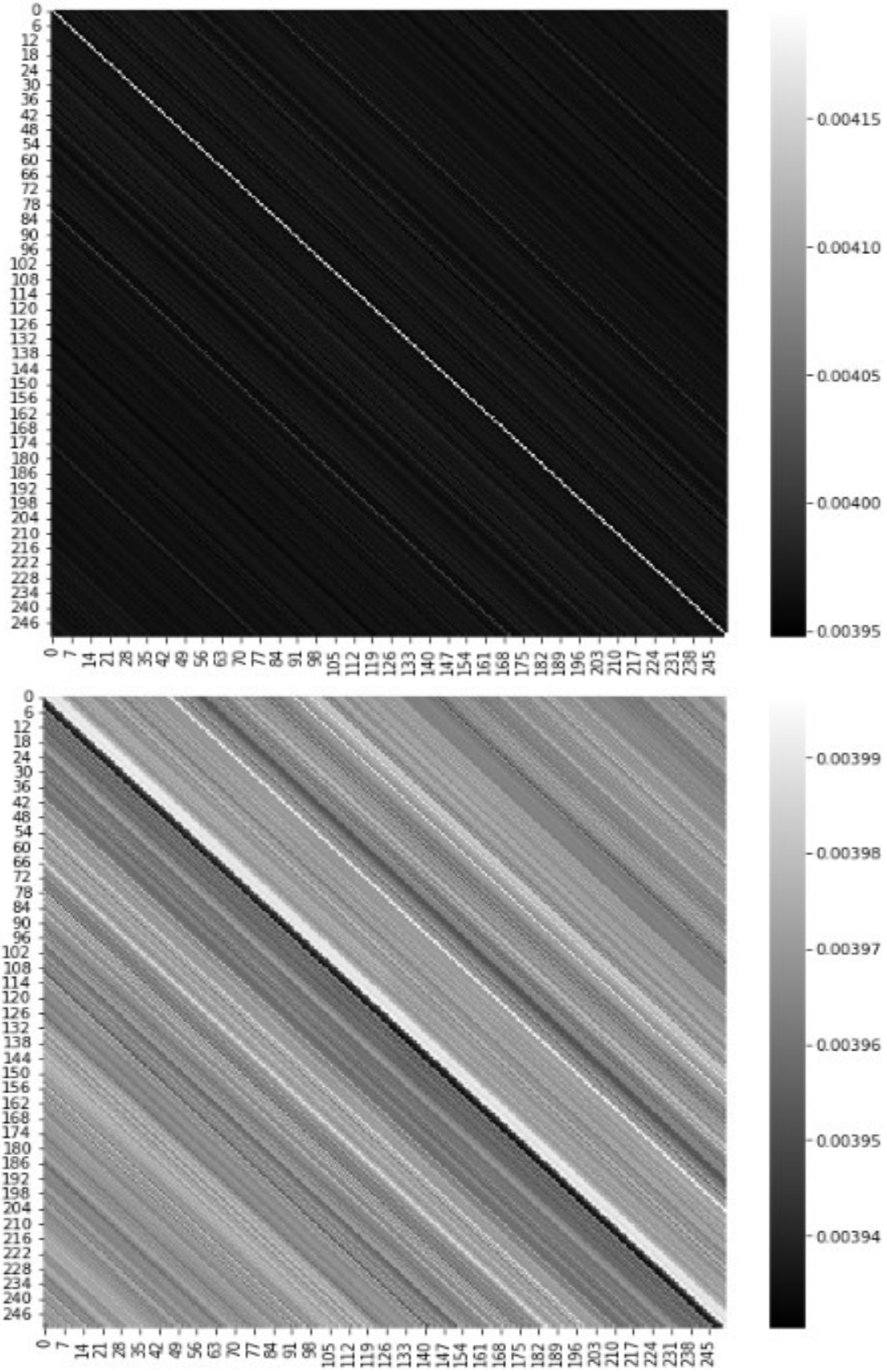

Figure 21: The augmented versions of images e) and f) from Figure 15

APPENDIX V

The Python code displays a simplified and pruned version of the code used in the empirical analysis, and hence it is for illustrative purposes only:

```
### NETWORK SETTINGS ###
seed = 3
tech_ind = 0    # 0 = main model, 1 = extended model
seq_len = 252
input_shape = x_train.shape[1:]


# Train the model for the out-of-sample period
for testyear in range(2012,2022):

  ### DATA PROCESSING ###
  data = pd.read_excel(io.BytesIO(uploaded_data['LRPG_data.xlsx'])) # load data

  if tech_ind == 1:
      data['SMA10'] = ta.sma(data["Close"], length=10)        # Simple Moving
    Average of 10 days
      data['SMA50'] = ta.sma(data["Close"], length=50)        # Simple Moving
    Average of 50 days
      data['MOM'] = ta.mom(data["Close"], length=9)           # Momentum
      data['STO_K'] = ta.stoch(high = data["High"], low = data["Low"], close =
    data["Close"]).iloc[:,0] # Stochastic %K
      data['STO_D'] = ta.stoch(high = data["High"], low = data["Low"], close =
    data["Close"]).iloc[:,1] # Stochastic %D
      data['RSI'] = ta.rsi(data["Close"])                     # Relative Strength
    Index
      data['MACD'] = ta.macd(data["Close"]).iloc[:,0]         # Moving Average
    Convergence Divergence
      data['CCI'] = ta.cci(data["High"], data["Low"], data["Close"]) # Commodity
    Channel Index
      data['PPO'] = ta.ppo(data["Close"]).iloc[:,0]           # Percent Price
    Oscillator
      data['BB_low'] = ta.bbands(data["Close"]).iloc[:,0]     # Lower Bollinger
    Band
      data['BB_upp'] = ta.bbands(data["Close"]).iloc[:,2]     # Upper Bollinger
    Band
      data['RVI'] = ta.rvi(high = data["High"], low = data["Low"], close = data["
    Close"]) # Relative Volatility Index
    else:
      pass

  # Create binary response variable
  df = data.pct_change()
  df.loc[df["Close"]>0,"Sign"] = 1
  df.loc[df["Close"]<=0,"Sign"] = 0
  df["Sign"] = df["Sign"].shift(-1)

  # Train-val-test split
  start = datetime(testyear - 4, 1, 1)
  split1 = datetime(testyear - 1, 1, 1)
  split2 = datetime(testyear , 1, 1)
  end = datetime(testyear + 1, 1, 1)
```

```
41
42  df_train = df[(df.index.to_pydatetime() + timedelta(days=365) >= start) & (df.
      index.to_pydatetime() < split1)]
43  df_val = df[(df.index.to_pydatetime() + timedelta(days=365) >= split1) & (df.
      index.to_pydatetime() < split2)]
44  df_test = df[(df.index.to_pydatetime() + timedelta(days=365) >= split2) & (df.
      index.to_pydatetime() < end)]
45
46  # Create x's and y's
47  x_train , y_train = [],[]
48    for i in range(seq_len, df_train.shape[0]):
49        x_train.append(df_train.iloc[i-seq_len+1:i+1,:-1])
50        y_train.append(df_train.iloc[:, -1][i]) # Using -1 since we want to
      predict 'Sign'
51  x_train, y_train = np.array(x_train), np.array(y_train)
52
53  x_val , y_val = [],[]
54  for i in range(seq_len, df_val.shape[0]):
55      x_val.append(df_val.iloc[i-seq_len+1:i+1,:-1])
56      y_val.append(df_val.iloc[:, -1][i])
57  x_val, y_val = np.array(x_val), np.array(y_val)
58
59  x_test , y_test = [],[]
60  for i in range(seq_len, df_test.shape[0]):
61      x_test.append(df_test.iloc[i-seq_len+1:i+1,:-1])
62      y_test.append(df_test.iloc[:, -1][i])
63  x_test, y_test = np.array(x_test), np.array(y_test)
64
65  # Value rescaling
66  scaler = StandardScaler()
67  x_train = scaler.fit_transform(x_train.reshape(-1, x_train.shape[-1])).reshape(
      x_train.shape)
68  x_val = scaler.transform(x_val.reshape(-1, x_val.shape[-1])).reshape(x_val.shape
      )
69  x_test = scaler.transform(x_test.reshape(-1, x_test.shape[-1])).reshape(x_test.
      shape)
70
71
72  ### MODEL ARCHITECTURE USING KERAS FUNCTIONAL API ###
73  def build_tf_model():
74
75    inputs = keras.Input(shape=input_shape)
76    pos_enc = Time2Vec(seq_len)(inputs)
77    pos_enc = layers.Concatenate(axis=-1)([inputs, pos_enc])
78    x = layers.Dense(pos_enc.shape[-1], activation='tanh')(pos_enc)
79
80    for _ in range(1):
81      x = layers.LayerNormalization(epsilon=1e-6)(x)
82      x, weights = layers.MultiHeadAttention(key_dim=504, value_dim = pos_enc.
      shape[-1], num_heads=2, attention_axes = 1)(x, x, return_attention_scores =
      True)
83      x = layers.Dropout(0.4)(x)
84      res = x + pos_enc
85
86      x = layers.LayerNormalization(epsilon=1e-6)(res)
87      x = layers.Conv1D(filters=pos_enc.shape[-1]*4, kernel_size=1, activation="
      swish")(x)
```

```
      x = layers.Dropout(0.4)(x)
      x = layers.Conv1D(filters=pos_enc.shape[-1], kernel_size=1)(x)
      x = res + x

    x = layers.GlobalAveragePooling1D(data_format="channels_first")(x)
    x = layers.Dense(32, activation="swish")(x)
    x = layers.Dropout(0.4)(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)

    tfmodel = keras.Model(inputs,outputs)

    tfmodel.compile(
    loss="binary_crossentropy",
    optimizer=keras.optimizers.Adam(beta_2=0.98, epsilon=1e-09),
    metrics=["binary_accuracy", "AUC"])

    return tfmodel

# Initiate model with random seed
random.seed(seed)
np.random.seed(seed)
tf.random.set_seed(seed)
model = build_tf_model()

callbacks = [
  keras.callbacks.ModelCheckpoint(
      "best_model.h5", save_best_only=True, monitor='val_loss', mode='min'
  ),
  keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=1),
  LRscheduler,
]

# Model fitting
start = time.time()
history = model.fit(
    x_train,
    y_train,
    validation_data=(x_val, y_val),
    epochs=200,
    batch_size=32,
    callbacks=callbacks,
    verbose=0,
)
train_time = time.time() - start

# Load the best model & get the predictions
best_model = keras.models.load_model("best_model.h5",
                                     custom_objects={'Time2Vec': Time2Vec})
y_pred = best_model.predict(x_test)
y_pred[y_pred>0.5]=int(1)
y_pred[y_pred<=0.5]=int(0)

# Save performance metrics of each year to a list
acc = np.append(acc, accuracy_score(y_test,y_pred))
auc = np.append(auc, roc_auc_score(y_test,y_pred))
mcc = np.append(mcc, matthews_corrcoef(y_test,y_pred))
```