Eberhard Karls University Tübingen
Natural Sciences and Mathematics
Wilhelm-Schickard Institute Department of Computer Science

Bachelor Thesis in Informatics

Deep Learning Methods for Stock Market Prediction: Benefits and Limitations

Stefan Thieringer

11.02.2022

Reviewer

Prof. Dr.-Ing. Setareh Maghsudi
Wilhelm-Schickard Institute Department of Computer Science
Eberhard Karls University Tübingen

Supervisor

Steven Bilaj
Wilhelm-Schickard Institute Department of Computer Science
Eberhard Karls University Tübingen

Thieringer, Stefan:
Deep Learning Methods for Stock Market Prediction: Benefits and Limitations
Bachelor thesis in informatics
Eberhard Karls University Tübingen
Thesis period: 11.10.2021-11.02.2022

Abstract

This bachelor thesis addresses the well-known stock-market-prediction-problem. The focus of this thesis is the prediction of the future price of an individual stock. As the title suggests, I am using deep learning methods to achieve a competitive result in the prediction values. The goal of this thesis is to highlight the benefits of using deep learning methods over other non-linear and linear forecasting methods.
To do so, the thesis includes a comprehensive introduction to the stock-market-prediction-problem in addition to established state-of-the-art forecasting methods within the stock-market-prediction-problem. As a result, the necessity for a more competitive model, like deep learning will become clear.
To understand the benefits and limitations of deep learning methods this thesis is giving a brief introduction to the working principle of deep neural networks and their most common models. I will also give a state-of-the-art overview of the forecasting methods using deep learning. At the end of this thesis, I am describing my stock-market-prediction approach, using an Long Short-Term Memory Model (LSTM-Model).
The model I have implemented shows, that deep neural networks are able to predict the trend of a stock, and even approximate prices. Although the model predicts the price mostly surprisingly well, the results can vary a lot. The importance of the information, that technical indicators do improve the forecasting result is big, since most of the technical indicators are easy and fast to calculate. Still, there is much room for improvement since the model is not always right.

Contents

List of Figures

List of Tables

List of Abbreviations

EMH	Efficient Market Hypothesis
MVP	Minimum Variance Portfolio
ECB	European Central Bank
P/E	Price-to-earnings ratio
P/B	Price-to-book ratio
P/CF	Price-to-cashflow ratio
P/S	Price-to-sales ratio
KNN	K-Nearest-Neighbours
SGD	Stochastic gradient descent
SVM	Support Vector Machine
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MAE	Mean Average Error
SGD	Stochastic Gradient Descent
EMD	Empirical Mode Decomposition
PCA	Principal Component Analysis
EV	Enterprise value
EBITDA	Earnings before Interest, Taxes, Depreciation and Amortization
EBIT	Earnings before Interest, Taxes
FCF	Free-cashflow before taxes

Chapter 1

Introduction

The stock market prediction problem attracted many investors and researchers, since the invention of the regulated market. The interest for the stock market is its sheer infinitely complex appearance resulting in a simple and vivid outcome, the profit or loss of a share or portfolio. To many investors, this becomes especially interesting when compared to traditional asset classes according to Thakkar [4](p.2135, chapter 1).
Unfortunately, the chance to gain more profit is accompanied by the risk to lose money due to bad investments. The seemingly random behavior of the stock market makes it difficult to distinguish a good investment from a bad investment. Therefore valuation methods have been invented, which can be classified into fundamental analysis and technical analysis according to Thakkar [4](p.2135, chapter 1). While the fundamental analysis relies heavily on metrics generated through the balance sheet the technical analysis uses mostly metrics calculated through the price itself. These valuation methods helped investors for a long time, trying to distinguish a good investment opportunity from a bad one. The downside of these valuation methods is that the evaluation of these metrics is still rather subjective.
The solution to the problem was found in Markowitz's paper [5], which provided a simple investment strategy only based on the expected return and the variance of a stock. It is known under the name portfolio selection, which was later accompanied by other approaches such as Walter [6], Peykani [7], Lefebvre [8] or Sefiane [9] and then summarized under the name portfolio theory.
However, it was clear that this theory and the portfolio selection algorithm is always a trade-off between profit and risk, due to definition. With the increasing popularity of computers, linear and non-linear forecasting methods became practicable. These forecasting methods are superior simply because their prediction relies on some more information like moving averages over the past days or other statistical variables calculated on the past days. Information is the key to a successful investment strategy.

Among the linear and nonlinear forecasting methods, the non-linear ones are dominating according to Thakkar [4](p. 2135, chapter 1), simply because of the inherent non-linear nature of the economy. In the field of nonlinear forecasting methods and machine learning, much progress has been made over the last years. From the application of Support Vector Machines (SVMs) over Particle Swarm Optimization realized by Thakkar [4] and Tree-based forecasting methods to Artificial Neural Networks (ANNs) implemented by Nabipour [10], the excellent performance of ANNs and particularly Deep Neural Networks (DNNs) always stood out. This competitive performance in addition to the high amount of parameters yielded by DNNs makes the use of them preferable.
The problem I am targeting within this thesis is to deal with the uncertainty of the market. As already mentioned there are different ways to evaluate shares, which produces lots of noise within the data. Additionally based on the technical analysis which is explained in section 2.5.2 it is assumed that the future price is in some way related to the previous prices which is claimed by Murphy [11](p. 24. chapter 1). Furthermore, it is commonly accepted that the market is behaving in a non-linear manner, which lays the foundation for DNNs.
The goal of this thesis is to implement a DNN and to test its ability to predict future trends. Additionally, this thesis shall demonstrate how big the difference is when you provide additional technical indicators to the DNN.
To do so I am preprocessing the data, which I downloaded with the yfinance library [12]. After this is done, I scale and split the data and then generate time series from the test and train set. Afterward, I am implementing the DNN with a LSTM-Model, which works great when using time-series data.
The methods used within this thesis are interchangeable within different scientific research fields, and thereby relevant not just for the stock price prediction. To understand the stock-related data and its complexity, it is advantageous to know some basic things about the inner structure of the stock market. This generates a very intuitive view of the price valuation process and becomes handy when selecting the features for price forecasting methods. I will therefore cover the underlying hypotheses as well as the already mentioned valuation methods. Additionally, I cover the mathematical framework, for the portfolio selection algorithm as well as the algorithm itself. The basic knowledge of the stock market prediction problem and the portfolio selection algorithm will help to understand why Deeps Learning Methods are so well suited for the stock market prediction problem. Afterward, I am introducing the working principle of an ANN and more specifically the Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), since I am using an RNN-variant within this thesis. After the theory, I am explaining how I downloaded and preprocessed the data. I do also explain the model used to evaluate the performance and share some of my experiences with the Keras deep learning API. In the end will summarize the results and give an outlook on further strategies.

Chapter 2

The Stock Market Prediction Problem

2.1 How Prices Are Determined

An important step when trying to predict prices on the stock market is to understand how prices are determined. In stock exchanges all over the world, the so-called most execution principle is applied.
To do so, according to Mayländer [3](p. 293, chapter 7.4.4) the stock exchange first gathers the ask-, and bid-orders in an order-book together, as you can see in table 2.1. The ask-limit in table 2.1 describes the lowest price at which a seller is willing to sell, and the bid price describes the highest price at which a buyer is willing to buy a certain stock. The corresponding volumes describe the number of shares the buyers or sellers are willing to buy or sell at the given limit.

bid volume	**bid limit**	**ask volume**	**ask limit**
150	cheapest	150	most expensive
350	2.68 €	250	2.74 €
200	2.70 €	100	2.72 €
50	2.72 €	300	2.68 €
100	2.74 €		

Table 2.1: Sample order book of an fictional share according to Mayländer [3](p. 293, chapter 7.4.4)

After the orders were gathered, the stock exchange determines the maximum turnover as you can see in table 2.2.

price	bid	ask	max turnover
2.68 €	850	450	450
2.70 €	500	450	450
2.72 €	300	550	300
2.74 €	250	800	250

Table 2.2: Summarized orders sorted by price according to Mayländer [3](p. 294, chapter 7.4.4)

According to Mayländer [3](p. 294, chapter 7.4.4) the price of this stock is determined gradually in the following order:

The determined price

1. has to reach the maximum turnover (given at price 2,68 € and 2,70 €)
2. has to include all unlimited orders
3. has to include all over the price limited bids and all under the price limited asks
4. has to minimize the gab between bid and ask volume and then partly carry out the limited bids and asks (given only at price 2,70 €)

According to these constraints, the determined price for this fictional stock would be 2.70 € with a flag, that additional demand remained. As you can see the price consist of the individual supply and demand of each market participant. As a consequence, each market participant is able to impact the price. This is according to Kissell [13](p. 99, chapter 4) due to the additional supply or demands each market participant is adding with its order.
The influence of each market participant and the reaction of the market to this additional supply or demand is considered within the Market Impact Models. These models try to describe how the impact of each order can be described. According to Kissell [13](p. 99, chapter 4) the market impact is categorized into temporary impact and permanent impact. A temporary impact is according to Kissell [13](p. 99, chapter 4) a change in price which is caused by the urgency to buy. The buyer or seller places an offer and adds additional supply and demand. This causes the price to rise or to fall.
A permanent impact is according to Kissell [13](p. 100, chapter 4) the information cost content. A rational investor is likely to buy an undervalued stock or to sell an overvalued stock. Thereby such investors add information due to their order. If other investors agree with the price change they will likely

adjust their prices to the new fair value, in their opinion. According to Kissell [13](p. 100, chapter 4) it is impossible to determine the exact market impact for the price of a stock because we cannot observe both price developments at the same time.
For example, we can never find out how the price would have developed if we had not bought the share. According to Kissell [13](p. 100, chapter 4) it is therefore referred to as the 'Heisenberg uncertainty principle of finance'. Still there are possibilities to approximate market impact, but I will not go deeper in this topic since the market impact is relatively low for small order sizes.

2.2 Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) is the dominating theory in the financial market when it comes to financial forecasting using statistical data. It tries to describe the financial market in a perfect mathematical-statistical way which suits great for the stock market prediction problem. The formal definition would exceed the extent of this thesis, therefore I am just highlighting the most important points.
According to Rapp [14](p. 6-7, chapter 2.2) the theory claims that the financial market is a kind of 'black box' and that the information leaking into the market leads to a balance of supply and demand and thereby to a fair price of each good. The Efficient Market Hypothesis claims that this is happening automatically since each market participant receives the same market-relevant information. From a modern perspective, this claim is very controversial and not a projection of reality. In the reality, market-relevant information can be bought from companies such as Bloomberg, Reuters, etc. to get an information advantage. Apart from that, the claim and all conclusions from it are summarized under the term 'information efficient'.
The hypothesis also claims that the returns are steady and normally distributed, which provided the justification and set the foundation for index funds. This is also referred to as the random-walk-characteristic and a test of whether this claim holds can be seen in the article from Roy [15].
According to Rapp [14](p. 6-7, chapter 2.2) many more books about this theory and the mathematically appealing appearance of the definition lead to the paradigm of calling this theory the modern capital market theory. Based on the modern capital market theory the portfolio theory has been integrated into the modern capital market theory and from there called modern portfolio theory, which is based on quantitative and statistical risk indicators.

2.3 Influence on the Pricing of a Stock

After seeing that the market impact model and the Efficient Market Hypothesis influences the price of a stock, we now consider other factors which can have influence on the price of a stock. Most of them fall, according to Mayländer [3](p. 299-300, chapter 7.4.7), in one of the following categories:

- *The Economical development* which is often represented through economic variables like the inflation rate, unemployment rate, GDP-growth rate, foreign trade surplus, the prosperity of the market to investment, currency price, interest rate, and incoming orders of the companies.
- *Psychological factors* like the psychological basic mood of the market participants and the political course, like tax policy, social policy, and import-export policy.
- *Business data of the underlying company* like earnings, dividends, market potential, public opinion of the management.
- *Capital market data* like the liquidity of the investors, interest rate inside the country in contrast to the investments abroad, taxation of the earnings.
- *Market-technical conditions* like support purchases of the stock to stabilize it, purchases to get the majority of shares, purchases to get technical course recovery.
- *Price impact on bonds* like credit demand of the economy, effective interest rate, and creditworthiness of the issuer.

The above-mentioned factors influence the stock price differently. Some of them cause a more extreme reaction than others. You can see the influence regularly when a government or court releases information about certain industry sectors. Such long-term strategies like sustainable energy, alternative drives, sustainable customer behavior, etc. can also have a high impact in the short-term price change and can additionally start a completely new trend.

Also very important are the decisions from the European Central Bank (ECB) which determine the interest rate and through that the amount of money within the stock market. As a rule of thumb, you say that the higher the interest rate the fewer people are willing to take the risk at the capital market because they get interest from their bank. As a consequence, they withdraw the money from the market which usually causes the prices to fall.

2.4 Financial Market Models

To understand the stock market prediction problem properly we first have to understand how the financial market is modeled nowadays. The model of the financial market is needed to build a framework for the trading decisions. So these models are rather theoretical. We use them to calculate the future payout of investment by determining its future value with a certain method. This is often referred to as discounting. Among the financial market models, there are two popular categories, the more basic one-period models and the more complex and more frequently used more-period models. In the following chapter 2.4.1 and 2.4.2 I will give a brief overview for both of them.

2.4.1 One Period Models

One-period models describe the basic approach to model the financial market. It is rather simple and does just describe the future payout of any portfolio within one time period.

According to Kremer [16](p. 4, chapter 1.1) the One Period Model gives its investor the choice to choose among his investment opportunities which are either risk- and profit-free investment or investing into financial instruments like shares, bonds, options, etc. When it comes to the investment opportunities, it's important for the model, that the future price is unknown and that the investment in a financial product is possibly either more profitable or yields a higher risk than the risk- and profit-free investment. Within a One Period Model, the existence of arbitrage opportunities is in accordance with Kremer [16](p. 23, chapter 1.5) usually forbidden. An arbitrage opportunity is a risk-free way to make a profit on the stock market using borrowed money.

The most important attributes which describe a One Period Model in accordance to Kremers [16](p. 4, chapter 1.1) are listed below:

- In a one-period model are two evaluation moments. The starttime $t = 0$ where the trading decisions are made and the endtime $t = 1$ where the profit or loss is evaluated.
- At time $t = 1$ exactly one of the states ω_i with $i = 1, ..., k$ and $\Omega = \{\omega_1, ..., \omega_k\}$ will occur. At the time $t = 0$ each state is known. Until $t = 1$ is not reached only the set of possible states is known, but not the state that is going to be realized.
- Within the model the investor considers N shares namely $S^1, ..., S^N$. For these shares exists a price process $S = \{S_t = (S_t^1, ..., S_t^N) \mid t = 0, 1\}$ which specifies the shares at each moment $t = 0$ or $t = 1$. The prices of the shares S_0^i with $i = 1, ..., N$ at the time $t = 0$ are real values. The

future prices of the shares S_1^i with $i = 1, ..., N$ at the time $t = 1$ are functions on Ω and defined as $S_1^i : \Omega \to \mathbb{R}$. $S_1^i(\omega)$ describes the price of the i-th share at the time $t = 1$ in state $\omega \in \Omega$. The investor knows S_0^i and the possible future prices $S_1^i(\omega)$, which can be realized. At time $t = 1$ the investor eventually knows which price of all possible prices $S_1^i(\omega)$ in dependence of the states ω_i were realized.

Portfolio Selection

The most common One Period Model is probably the portfolio theory from Harry M. Markowitz [5], who set the foundation of portfolio management. According to Modello [17](p. 103, chapter 2.1) the portfolio management tries to find the optimal risky portfolio for an investor with a certain risk aversion. An optimal risky portfolio is defined by the efficiency curve and the highest possible investor-specific indifference curve. The optimal portfolio of the investor is determined by the contact point between the efficiency curve and the highest possible indifference curve.

For the calculation of the efficiency curve, the expected return and the standard deviation as well as the covariance and the correlation between return and investment is needed. In accordance with Mondello [17](p. 105, chapter 2.2) the expected return $E(r)$ can be defined in two ways:

$$E(r) = \frac{1}{T} \sum_{t=1}^{T} r_t$$

whereby:
T = amount of periods,
r_t = profit of an investment within the period t

The expected return has proven to be a good approximator of the future return since it is defined as the arithmetic mean over time periods.

It follows the empiric variance $\sigma^2(r)$ and the standard deviation $\sigma(r)$ in accordance to Mondello [17](p. 106, chapter 2.2) which is needed to calculate the efficiency curve based on the expected return:

$$\sigma^2(r) = \frac{1}{T-1}\sum_{t=1}^{T}(r_t - E(r))^2$$

The standard deviation follows by:

$$\sigma(r) = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(r_t - E(r))^2}$$

The variance measures the average square deviation from the expected return, and thereby the risk of loss. The empiric variance is the a good estimator of the variance, because the empiric variance is unbiased.
The reason why the standard deviation is used over the variance is according to Mondello [17](p. 13, chapter 1.3.1) because the variance has not the same unit as the profit which is measured in percentage.
The standard deviation on the other hand is like the profit described in percentage which suits well for further calculations.

The weights w_i of two investments with $i = \{1, 2\}$ within the portfolio can be determined with [17](p. 115, chapter 2.3):

$$w_1 = \frac{\sigma_2}{\sigma_1 + \sigma_2}$$

and

$$w_2 = \frac{\sigma_1}{\sigma_1 + \sigma_2} = 1 - w_1$$

whereby:
σ_i = standard deviation of the investment i

The indifference curve measures the benefit for the investor of holding the portfolio as it is. It is calculated through the desired profit, the risk, and the risk aversion of the investor.
The expected return of a portfolio $E(r_p)$ consisting of two investments i with $i \in \{1, 2\}$ is according to Mondello [17](p. 108, chapter 2.3) calculated as followed:

$$E(r_p) = w_1 E(r_1) + w_2 E(r_2)$$

whereby:
w_i = amount of investment i within the portfolio,
$E(r_i)$ = expected return of the investment i

According to Mondello [17](p. 108-109, chapter 2.3) is the risk of the portfolio dependent on the weight of the investment, the individual risk and the covariance and correlation between the investments.
A positive covariance near 1 means that the profit of two investments is mostly following the same patterns for a certain period. A negative correlation near -1 means on the other hand that the profit of two investments follows opposing patterns. If the covariance is 0 or close to that it means that there is no correlation between the two investments. The covariance $Cov_{1,2}$ is dependent on the spread of the returns and according to Mondello [17](p. 108-109, chapter 2.3) defined as followed:

$$Cov_{1,2} = \frac{1}{T-1} \sum_{i=1}^{T} (r_{i,1} - E(r_1))(r_{i,2} - E(r_2))$$

whereby:
$r_{i,1}$ = return of investment 1 for scenario i,
$r_{i,2}$ = return of investment 2 for scenario i

Because the covariance is not standardized and therefore difficult to interpret, we calculate Pearson's correlation coefficient $\rho_{1,2}$, which is according to Mondello [17](p. 111, chapter 2.3) called the standardized covariance and defined as followed :

$$\rho_{1,2} = \frac{Cov_{1,2}}{\sigma_1 \sigma_2}$$

whereby:
σ_i = standard deviation of return from investment i

If we now look at the sum of the risk components of a portfolio, the variance of a portfolio σ_P^2 with two investments is calculated according to Modello [17](p. 112, chapter 2.3) like you can see below:

$$\sigma_p^2 = w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2Cov_{1,2}$$

with the standard deviation of the portfolio σ_P of:

$$\sigma_p = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2Cov_{1,2}}$$

By exchanging the correlation coefficient formula after the covariance, the standard deviation can be written as followed:

$$\sigma_p = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2\rho_{1,2}\sigma_1\sigma_2}$$

If there is a perfectly positive correlation (+1) which represents the highest portfolio risk, the standard deviation of the portfolio can be simplified in accordance with Mondello [17](p. 114, chapter 2.3) to:

$$\sigma_P = w_1\sigma_1 + w_2\sigma_2$$

If there is a perfectly negative correlation (−1), which represents the lowest portfolio risk, the standard deviation can be calculated as reported by Mondello [17](p. 114, chapter 2.3) with:

$$\sigma_P = |w_1\sigma_1 - w_2\sigma_2|$$

If you use the formula of the expected return and the standard deviation to calculate most of the portfolio possibilities you end up with a graph shown in figure 2.1 which is inspired by Mondello [17](figure 2.2).

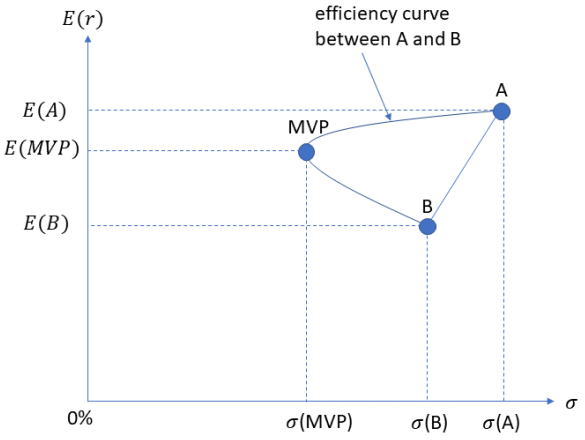

Figure 2.1: Efficiency curve of two investments A and B with the MVP of both investments

Additionally, you can calculate the Minimum Variance Portfolio (MVP) which has the lowest standard deviation within all possible combinations between A and B. According to Mondello [17](p. 118, chapter 2.3) the weight of A within w_A the MVP can be calculated with the following formula:

$$w_A = \frac{\sigma_B^2 - Cov_{A,B}}{\sigma_A^2 + \sigma_B^2 - 2Cov_{A,B}}$$

With that said, the portfolio curve changes its appearance depending on the correlation between A and B.
This introduction into the portfolio theory using the minimum variance approach shall demonstrate the issue of the portfolio theory. The whole investment decision and the portfolio constellation use the linear dependency between A and B. We have already seen that the financial market is very complex and does not behave linearly, which explains that the investors have changed over time to more complex methods to determine the most profitable investments. This is also the reason and justification why I am using deep learning methods within this thesis. DNNs yield many more parameters and are generally better suited for non-linear problems like this. Therefore much better results can be expected by applying DNNs to the stock-market-prediction-problem.

2.4.2 More Period Models

Because of the complexity of the financial market, the efficiency and expressiveness of the One Period Models lacks. A complex system has to be expressed by a complex model, otherwise, information or important aspects are lost which eventually worsens the financial prediction capability. Therefore the More Period Models have been invented with the binomial-trees as a famous representative which enhances the approach of the One Period Models. A More Period Model is characterized in accordance of Kremer [16](p. 45, chapter 2) through the attributes of a One Period Model plus the following attributes:

- There are $n+1$ training moments with $n = 0, ..., n$ and thereby n periods $[t-1, t]$ for $t = 1, ..., n$.
- It is assumed, that one state ω is realized at the End $t = n$ with $\Omega = \{\omega_1, ..., \omega_K\}$. All states are known at $t = 0$ but it is unknown which ω is about to be realized in $t = 1$.
- With increasing time the information on the state in $t = n$ increases. This increase is modelled by a filtration $\mathcal{F} = (\mathcal{F}_t)_{0 \leq t \leq n}$
- There are N financial instruments $S^1, ..., S^N$ whose prices are modelled as the on $\mathcal{F}$ adapted stochastic processes.

Generally, there are two new aspects when compared to a One Period Model. The filtration whose information content increases till $t = n$ and the price processes which are adapted to this filtration. So you could say that a More Period Model consists of several One Period Models. In contrast to a One Period Model, you gain information with increasing time and you can adjust your portfolio within $t = 0$ and $t = n$ according to the Filtration $\mathcal{F}$. This mathematical framework allows the investor to react more flexible within the $t = 1$ and $t = n$. That's why modern portfolios are modeled as More Period Models with a capable filtration.

2.5 Valuation Methods

2.5.1 Fundamental Analysis

Within the share valuation methods, the fundamental analysis concentrates on metrics that were generated through the balance sheet and the overall earnings situation of a company. According to Schmidlin [18](p. 111, chapter 7) is the major idea behind the fundamental analysis, that the absolute stock price itself does not give any information about the valuation of the underlying company. Therefore the fundamental analysis uses valuation metrics, which simplify the comparison of companies and the valuation level itself. Thereby this method is predominant within banks when a credit decision is made. Because of the popularity among banks and because it is based on uniform standards it is also widely used for long-term investing. Among the valuation metrics, which are also called multipliers, there are according to Schmidlin [18](p. 112, chapter 7) two different sets of valuation metrics. The equity multipliers compare the market capitalization with success factors that are entitled to the equity providers. Such success factors are for example the annual surplus or the free-cashflow (removable cash surplus). The most common equity multipliers are:

- P/E ratio (Price to earnings ratio)
- P/B ratio (Price to book ratio)
- P/CF ratio (Price to cashflow ratio)
- P/S ratio (Price to sales ratio)

According to Schmidlin [18](p. 128-142, chapter 7) the entity multipliers also include the borrowed capital and compare the whole enterprise value with success factors that refer to the whole capital. Common entity multipliers are:

- EV/EBITDA ratio (Enterprise value to Earnings before Interest, Taxes, Depreciation and Amortization)
- EV/EBIT ratio (Enterprise value to earnings before Interest, Taxes ratio)
- EV/FCF ratio (Enterprise value to free-cashflow before taxes)
- EV/Sales ratio (Enterprise value to sales)

2.5.2 Technical Analysis

The technical analysis uses a different approach. The underlying theory is the EMH introduced in chapter 2.2. It means that all the information present, whether fundamental, political, psychological, or others are already included within the price. So the price is a perfect representation of the real value of the company considered. According to Murphy [11](p. 22, chapter 1) further studies within the technical analysis don't make sense if this theory is not accepted. The so-called technicians claim, that the price is representing supply and demand. The basic fundamental theory is that if the demand exceeds the supply the prices should rise which is called bullish and if the supply exceeds the demand the prices should fall which is called bearish.

The technicians reverse this principle and claim: If the prices rise, for whatever reason, the demand exceeds the supply, so the fundamental must be bullish and vice versa. Murphy [11](p. 22, chapter 1) concludes that the technician is indirectly studying the fundamental information of the market through the price. The reasons for the market movement are not interesting for a technician, they just try to follow and recognize patterns within the price.

Additionally, it is supposed that prices move in trends. Murphy [11](p. 24. chapter 1) writes that a trend is more likely to continue than reverse. The whole process in trend following is to follow the trend until there is a sign that the trend is reversing.

The last claim is that history is repeating itself. Murphy [11](p. 24. chapter 1) justifies this with human psychology. The chart which has been evaluated over the past hundred years shows patterns. According to Murphy, [11](p. 24. chapter 1) these patterns show the bullish or bearish psychology of the marked. Because the patterns worked in the past, it is supposed that they will work in the present as well because the human psyche doesn't tend to change. These assumptions are the justification for technicians to search for patterns using different metrics. The metrics used are supposed to describe different attributes of the price like momentum, volume, volatility, and trend. Technicians use these metrics as the foundation of their further analysis and finally for their investment decision.

This is the reason why I am interested in these metrics. Although I don't fully agree with the claims made by Murphy [11], I assume that the market is a self-fulfilling prophecy, which means that the participants get what they want. If many investors use these metrics as a foundation for their investments, the price will mostly follow these patterns, whether they make sense or not.

2.6 State-of-the-art Methods

Classical linear forecasting methods are rarely seen in stock market prediction nowadays. Usually, these methods are applied in very specific markets like the energy market or the gas market. This is because most financial products behave nonlinear and are sometimes even far away from being rational.
According to Thadani [19](p. 198) who published a research article [19] in 2021 and applied linear forecasting methods to the commodity market, these methods perform better when the data has some sort of seasonality and cyclicity included.
Such seasonality is extremely predominant when it comes to options on raw materials like oil, gas, etc. Additionally, the performance of linear models is highly dependent on the incorporated features and the precise estimation of the parameters. His work is interesting because he is using classic methods to get success. Additionally, he is using the Simple Moving Average over the last 20 days which I am also using as an input feature for my LSTM Model in chapter 4.
Another interesting approach has been done by Ananthi [20] who implemented a K-Nearest-Neighbours (KNN) regression algorithm based on the candlestick representation of the historical stock data. His model is a classification task that generates buy and sell signals in different market conditions. This is also an interesting method and is maybe more practicable since many investors are interested in buy or sell signals instead of the future price.
A completely different approach was done by Awan et al. [21]. Within their models, they analyzed social media services like Twitter to include the results in the prediction. For the prediction itself, they compared Linear Regression, Random Forrest, Decision Tree, and Generalized Linear Regression models. Apart from the Decision Tree algorithm, every other model performed well. Using social media to boost the performance of any forecasting method is a good idea. In section 2.3 we have seen that psychological factors like the basic mood of the market can have a huge influence on future price development.
Additionally, these changes in price due to psychological factors have no connection to the simple historical data, which confuses every model. Closing this gap by providing a news feed can provide a forecasting model possibly relevant information about the psychological factors, and help it to recognize patterns.

Chapter 3

Deep Learning Methods

3.1 The Structure of an ANN

The foundation of an Artificial Neural Network (ANN) is determined by its perceptrons. The core of such an perceptron is in accordance with Deru [1](p. 69, chapter 3.1.1) represented by the linear function

$$\alpha = \sum_{i=1}^{n} w_i x_i + b$$

with n-dimensional weight vector $w = (w_1, ..., w_n)^T$, the n-dimensional input vector $x = (x_1, ..., x_n)^T$ and the bias b. The output of the ANN is determined as

$$y = \phi(\alpha)$$

where $\phi(x)$ is the placeholder for any activation function.
From this output function results the simplified illustration of an Deep learning network as shown in figure 3.1. The actual design of the ANN varies from task to task and there is not a design regulation that guarantees the efficiency or performance of the ANN. Usually, the design is based on the experiences of former ANNs which had comparable tasks. The ANNs are then adjusted to the new task and afterward reevaluated. A more modern but also more resource-hungry method is to determine the model configuration by a brute-force algorithm like the Keras-tuner [22] which is available for the Keras API.

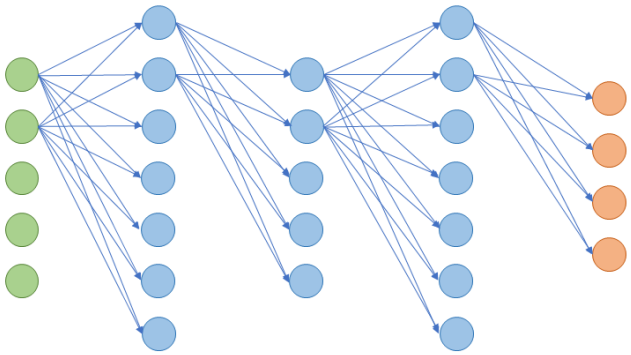

Figure 3.1: The general architecture of an deep learning network inspired by Deru [1](p. 88, figure 3.12)

In figure 3.1 you can see the **input layer** (green), which is the starting point of the ANN. Its task is to perceive the current input vector with its n features. As stated in Deru [1](p. 71-72, chapter 3.1.3), based on these features in combination with the weights, the ANN will later calculate its decision, within the hidden layers (blue). In the context of this thesis, such features are for example the Open, Close, High, or Low prices of a day.

The **hidden layers** consist of the already introduced perceptrons. These perceptrons are organized in several layers which are also connected. The first and the last hidden layer is connected to the input-, and output layer. The way the connections are organized is individual and can be adjusted if necessary. A very popular deep learning network is the multilayer perceptron (MLP), which essentially means that each perceptron is connected to each perceptron of the previous and following layer. According to Deru [1](p. 71-72, chapter 3.1.3) is the goal of the hidden layers is the further processing of the input vector to find hidden patterns within the data set.

The **output layer** gets the result of the last hidden layer and projects it after the use of the activation function in form of the result of the ANN. This result is a real number that represents either a class or a probability for a certain class when the ANN is within a classification task. As stated in Deru [1](p. 71-72, chapter 3.1.3) another possibility is that the ANN outputs a real value which is the case when a ANN is applied on a regression task like the stock marked prediction problem. According to Schmidhuber [23] an ANN is considered deep when it consists of many layers. Furthermore, RNNs are considered as

the deepest of all neural networks, because they can process memory of an arbitrary amount of input patterns.

3.1.1 Activation Functions

The activation function is the part of the perceptron which is responsible for the decision making. Depending on the type of activation function, the output of the perceptron changes. The activation function is one of the many adjusting screws to boost the performance of the ANN. The activation function is also the part which adds nonlinearity to the ANN when using a nonlinear activation function according to Feng [24]. The following activation functions are mentioned by Deru [1](p. 70, chapter 3.1.2) and showed in figure 3.2.

- Identity function: $y = x$
- Heaviside function: $y = \begin{cases} 0 & x \leq 0 \\ 1 & x \geq 0 \end{cases}$
- Sigmoid function: $y = \frac{1}{1+e^{-x}}$
- Tanh function: $y = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$
- ReLU function: $y = max(0, x)$
- SoftSign function: $y = \frac{x}{1+|x|}$

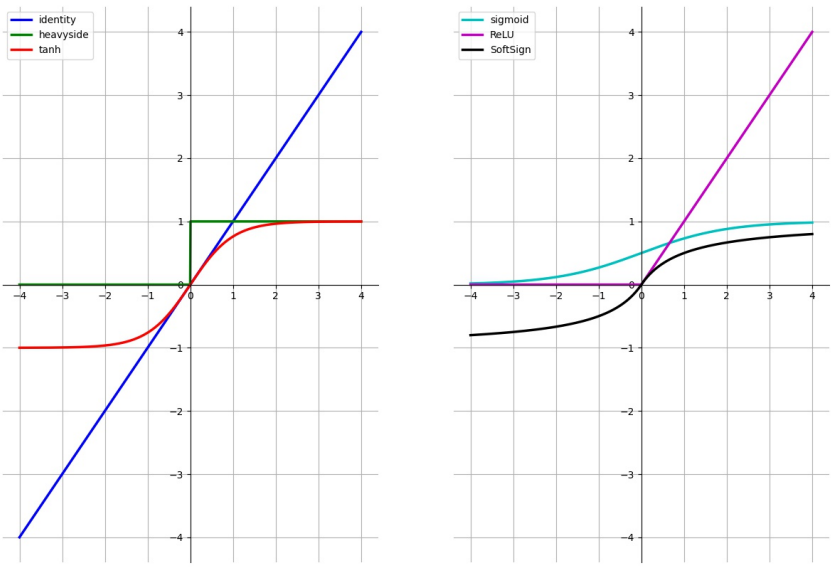

Figure 3.2: Activation functions

3.2 Weight Adjustment

The learning process in an ANN is done by updating the weights of the different layers in each iteration. The goal of the learning ability of an ANN is to minimize the error between the expected result and the calculated result. This information is calculated by the so-called loss function and afterward backpropagated to the layers where the adjustment of the weights takes place. There are different loss functions most commonly applied to different tasks, which are generally divided into classification and regression tasks. Deru [1](p. 76, chapter 3.2.2) mentions in his book the most used loss functions, for the different tasks, shown in table 3.1.

Classification	**Regression**
Categorial Crossentropy	MAE
Log Loss	MSE
Relative Entropy	
Exponential Loss	

Table 3.1: Most common loss functions for classification and regression problems

For the purpose of the stock market prediction, which is regression problem, the Mean Average Error (MAE) and the Mean Squared Error (MSE) are the most suitable ones. The MSE loss function is mentioned by Deru [1](p. 77, chapter 3.2.2) and defined as followed:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

it follows the MAE with

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

whereby:
n = amount of training samples,
y_i = real value,
$\hat{y}_i$ = predicted value

The advantage of the MSE over the MAE is visible. The MSE weight bad results of the DNN by the power of two, which results in faster convergence against the optimal weight configuration.
The information of the loss function is then processed by the optimizer function. Famous optimizer functions are the Stochastic Gradient Descent (SGD), Adam, or Adagrad algorithm.

3.3 Network Architectures

3.3.1 Feedforward Neural Networks

Convolutional Neural Networks

According to Deru [1](p. 91, chapter 3.5.1) a Convolutional Neural network (CNN) is a feedforward neural network that works well for classification tasks like the picture- or video recognition. It has been developed by Yann LeCun [25] in the mid 90th and is inspired by the information processing of the visual cortex. In this brain area, the neurons are placed in a manner, that they can solve complex problems by dividing them into smaller pieces. A CNN uses therefore a special structure, which is shown in figure 3.3:

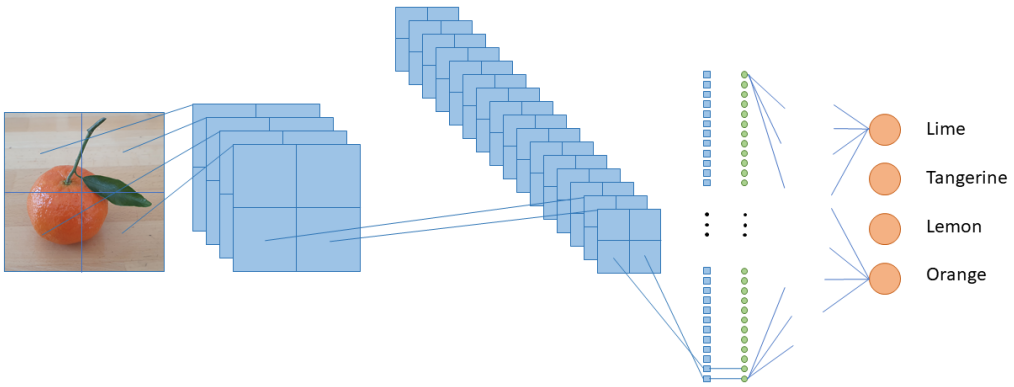

Figure 3.3: The general principle of an CNN with arbitrary number and types of layers inspired by Deru [1](p. 94, figure 3.19)

The components of a CNN differ significantly from those of an MLP. Therefore I am explaining the different layers of the CNN inspired by Deru [1](p. 92-94, chapter 3.5.1) beginning with the input picture:

The input layer is responsible for perceiving the current input picture which is usually described in pixels or other representations of the picture. This matrix with the information of the picture is then divided in smaller chunks, by a pooling layer. Through the division, the pooling layer reduces the dimension of each partial picture outgoing from the original picture. The convolutional layer is responsible for applying a filter on the partial picture and generating the feature map for each partial picture. After that, an activation function is

applied to each feature map.
Another pooling layer reduces the dimension of the feature maps, in order to simplify the problem further. On these feature maps a second convolutional layer applies an filter to generate a new feature map with the most striking features. These features are then flattened in order to process the most striking features by several fully connected layers. According to Deru [1](p. 92-94, chapter 3.5.1) the output layer then classifies the input picture, based on the result of the fully connected layers.

3.3.2 Recurrent Neural Networks

The so-called Recurrent Neural Networks (RNN) form, according to Deru [1](p. 96, chapter 3.5.2), another very important network architecture within the Deep Learning networks. According to Sherstinsky [26] RNNs are designed to process sequential data or data with a time component like language processing or stock market data. To identify relations within the data in terms of time, the RNNs have built-in memory. This memory is realized through a special structure of the layer which I will explain briefly.
When using a feed-forward ANN the preassumption is, that the datapoints of the dataset are independent of each other. When using an RNN this is not the case anymore. Therefore RNNs have direct or indirect feedback loops to the same neuron or other neurons of other layers. A direct feedback loop means, that the output of a neuron is also an input of the same neuron. An indirect feedback loop means, that the output of the neuron is connected to the input of another neuron of the previous layer. According to Deru [1](p. 96, chapter 3.5.2) there is also the special case of a bidirectional RNN where the output of the neuron is connected to the input of another neuron of any other layer. Thus a dependency of the different moments within the sequence is achieved, which essentially works as short-term memory. This memory can be used to predict different values as shown in figure 3.4. In figure 3.4 the green nodes represent the input layer. The orange nodes represent the output layer, and the blue nodes represent any RNN-cell.

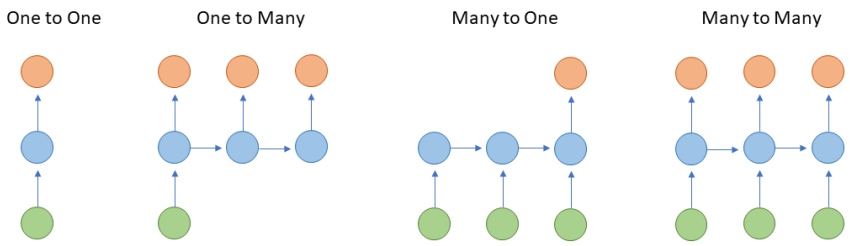

Figure 3.4: The types of RNNs inspired by Roman [2]

The Simple RNN

A simple RNN is the most basic version, with short-term memory. In figure 3.5, you can see the structure of a RNN. As you can see, the previous RNN influences the prediction of the next RNN, which is definitely an advantage over regular feed-forward DNNs. Unfortunately, the RNN only uses the information of the previous layer, which results in a relatively bad classification result. This is especially true in the case of data dependencies reaching back over several data steps. In this situation, the LSTM Network is a better choice which I explain in the next section.

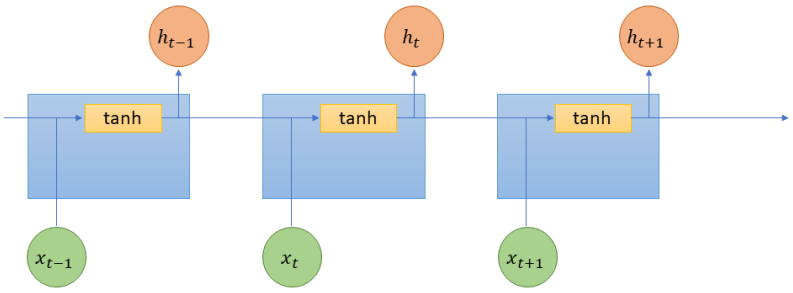

Figure 3.5: The general principle of an RNN is inspired by Roman [2]

In figure 3.5 you can see the inputs x_t and the outputs h_t in this sample Many-to-Many RNN. According to Mittal [27] the function can be described as:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

whereby:
h = single hidden vector,
W_{hh} = weight at the previous hidden state,
W_{xh} = weight at the current input state,
tanh = the activation function which implements non-linearity

Additionally the output y_t is in accordance with Mittal [27] defined as:

$$y_t = W_{hy} h_t$$

whereby:
h_t = the output state,
W_{hy} = weight at the output state

Long Short-Term Memory Networks

A special variant of an RNN is the Long Short-Term Memory (LSTM), which also contains memory cells and was first introduced by Hochreiter [28] in 1997. In contrast to the Simple RNN, the LSTM can forward older information independent from the output of the previous layer. This means that the LSTM suits well, when short-term and long-term dependencies occur in the data. Figure 3.6 is inspired by Mittal [27] and shows the significantly more complex LSTM-cell.

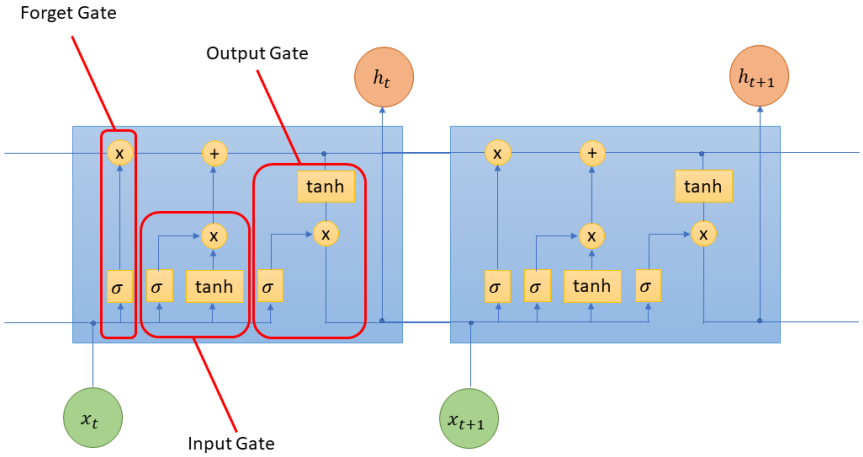

Figure 3.6: The general principle of an LSTM-cell considering two historical data points with the Input, Forget, and Output Gate

As you can see in figure 3.6 there are different tasks highlighted for different gates inside the structure of the LSTM-cell. According to Mittal [27] there is the Forget Gate which decides which details will be thrown away, by looking at the previous state h_{t-1} and content input x_t. It is mathematically described as followed:

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

whereby:
f_t = output of the sigmoid function within the Forget Gate,
h_{t-1} = the previous state,
x_t = the content input,
W_f = the Forget Gate weights ,
b_f = the Forget Gate bias

The Input Gate decides which information is used to update the memory. According to Mittal [27] the Input Gate can be written as:

$$i_t = \sigma(W_i(h_{t-1}, x_t) + b_i)$$
$$C_t = \tanh(W_C(h_{t-1}, x_t) + b_C)$$

whereby:
i_t = output of the sigmoid function within the Input Gate,
C_t = output of the tanh function within the Forget Gate,
W_i = the weights for the sigmoid function,
W_C = the weights for the tanh function,
b_i = the bias for the sigmoid function,
b_i = the bias for the tanh function

The Output Gate decides based on the Input and the memory which information is used. According to Mittal [27] the Output Gate can be written as:

$$o_t = \sigma(W_o(h_{t-1}, x_t) + b_o)$$
$$h_t = o_t \tanh(C_t)$$

whereby:
o_t = output of the sigmoid function within the Output Gate,
h_t = output of the tanh function within the Output Gate,
W_o = the weights for the sigmoid function,
b_o = the bias for the sigmoid function,

3.4 Attributes of DNNs

Nonlinear forecasting methods do have particular advantages over linear models, especially in such complex problems. But despite being very suitable for complex tasks they have also several drawbacks. The biggest benefits and limitations of DNNs are listed below.

3.4.1 Benefits

DNNs can solve complex problems

Probably the biggest advantage of DNNs is their capability of solving complex problems. This is mainly due to their nonlinear behavior but also due to the capability of observing structures and dependencies within the data. Additionally this is feasible because of the possibility of optimizing a DNN for a certain task. We have seen, that there are many different layers and functions that have influence on the forecast result. Almost all can be seen as modules that can work together. So there is an infinite amount of DNN designs which can be tried, to achieve the best prediction result.

DNNs are learning over time

One of the big benefits of Deep Learning is the fact that these models are capable of learning over time. Like other machine learning methods DNNs can observe complex structures and dependencies and save their parameters. What sets DNNs apart from other machine learning methods is the build-in capability of adjusting their basis for decision-making automatically.
Technically you can also code other forecasting models with self-adjusting parameters, but this adds complexity. With Deep Learning models the self-adjusting capability is build-in the structure of the model and doesn't add any further complexity.

DNNs can use huge amounts of data

A big advantage of Deep Learning Models is that these models are designed to process a huge multidimensional input vector. This input vector can potentially be unlimited big. The only real limitation is your computer hardware. So we can expect that we will be able to solve much more complex tasks in the future than we are currently able, simply due to the improvements of the hardware.

DNNs are theoretically unlimited big

Another big advantage is that Deep Learning methods can be scaled and adjusted in size which allows a high amount of parameters. In comparison to other models, these parameters are relatively cheap and reach a similar level of complexity.
Unfortunately, we are right now not able to enhance existing models over time to reach a better accuracy. Such functionality would boost the performance of the models and would help improve the efficiency while developing a model.

DNNs became relatively easy to implement

An important advantage is also the support of libraries within popular programming languages. Integrated with Tensorflow or PyTorch you can choose different ANN libraries within python and other programming languages. These libraries simplify the building process of the model, especially when compared to the beginnings of ANN programming, when the models manually have been developed with arrays. Due to the simplicity of the model-developing, the focus is set on the preprocessing of the data, which still can be pretty challenging, especially when you try to optimize the shape of each input.

DNNs are relatively easy to understand

Compared to other machine learning methods, The working principle of an ANN is relatively easy to understand. When using other machine learning methods for forecasting you have to know a huge amount of mathematics to understand the model and then adjust it to the problem you are facing. With these machine learning methods, it is known how to apply them and it is known and traceable why they are calculating a specific result.
With Deep Learning methods on the other hand it is different. We do know the structure of each neural net and we do have a formal definition, but we do not have a proper understanding of the behaviour of the net due to its highly complex structure. This is the reason why ANNs in general are also referred to as 'black-box models'.

3.4.2 Limitations

DNNs require large datasets

A big disadvantage of DNNs is that they require immense amounts of data in order to highlight their advantage over other machine learning methods. In some situations, it may occur that there is not enough data gathered to run a Deep Learning net efficiently. This is why DNNs are not suitable in all situation, and why other machine learning methods are also widely used.

DNNs are calculation intensive

Due to the complexity of a DNN the required computing power is growing with the size of the DNN. So if a complex task requires a huge and complex DNN, there is also much calculation needed and therefore strong hardware components. Nowadays the resource hunger of DNN is counteracted by GPU acceleration. That means that the many simple computation tasks within the DNN are processed by the graphics card which is designed to calculate many simple operations parallel. Because of the relatively cheap price of GPUs considered the amounts of data they can process, DNNs are became affordable in many industries and even private persons.

DNNs have a limitated explainability

Another disadvantage is the limited explainability of DNNs. As mentioned above, the structure and the working method of a DNN is widely known. Unfortunately, there is right now no way to express the basis for decision-making of the DNN apart from its weights. There are some ways to receive hints, due to reverse-engineering methods, but there is no clear scientific description for the output of the neural networks.

DNNs are low adaptable

Another disadvantage is the low adaptability of Deep Learning networks. After a DNN is trained, it is highly specialized in this task. All weights are set to perform best with the provided data shape and the desired output. Adding or dropping features or using it for a different task means essentially that you have to train the whole model from the beginning, and determine the best set of hyperparameters.

3.5 State-of-the-Art Methods

Deep learning methods are currently experiencing a renaissance. This is mostly due to the great performance improvements of chips over the last years, but also because of the flexibility and ease of use of Deep Learning models. Among the current deep Learning approaches some particularly stand out, like the one made by Jia [29] who applied an Empirical Mode Decomposition (EMD) and the Principal Component Analysis (PCA) before she processed the data by a DNN. According to Maćkiewicz [30] is the PCA a method for dimension reduction of the observation space. The EMD is in accordance with Ge [31] a method for simplifying time series to simple and intrinsic oscillations, which then can be used for further processing. This enhanced approach in preprocessing the data is very interesting in the context of this thesis because it helps to reduce the training and prediction time of the DNN by sorting out the redundant data. When I trained my DNN-Model in Chapter 4 I did also encountered long training and forecasting times depending on the input size and window size. Reducing the features could help solve this problem. Additionally, Jia [29](p. 3) uses an unusual way to determine price and volume. She claims that the so-called Yin-Yang volatility is a novel approach to defining volatility.
The most exciting approach has been done by Nti and his colleagues [32]. They have used six different information sources and processed them through CNN. The data sources they have used were stock data, google trend index, macroeconomic variables, web news, forum discussions, and tweets. Besides the CNN they also applied a random search algorithm and a feature selection algorithm to optimize the initial hyperparameters. Finally, the data was fed in a multilayer LSTM and then fine-tuned. The combination of these six information sources resulted as stated in Nti [32](p. 19) in an accuracy of 95.78% which is pretty impressive.
Another approach has been done by Lin and her colleague [33]. They also preprocessed the data with the EMD followed by the prediction which was done by an LSTM-model. What makes their article interesting is, that they try to boost the overall performance through the selection of the stock. They therefore only used stocks that are listed in the Taiwan Corporate Governance 100 Index. Additionally, they sorted the stocks according to their daily trading amount and liquidity test standards. Afterward, they ranked these stocks according to their after-tax net profit. Finally, they used an LSTM-Model to predict future prices which is a great idea, because there may be different patterns in the stock price when trading stocks from different industry sectors or with different corporate social responsibilities. This could also result in a better performance of the forecasting model.

Chapter 4

Implementation of Deep Learning Methods

4.1 LSTM-Model for Stock Market Prediction

4.1.1 Data Source

The foundation of the LSTM-Model is set by the yahoo finance API which provides a very easy and fast way to download historic stock market data from almost any stock. For all models, I am using historical stock data from the Daimler AG to compare the performance difference between using technical indicators and using no technical indicators. To be precise, I am using the period from the 1st January 2011 till the 1st January 2021, so exactly ten years of Daimler stock market data.

4.1.2 Data Preprocessing

Unfortunately, the data from yahoo finance is not always entirely complete when it comes to the days on which no trading took place. Such events can be national public holidays or technical errors on the stock exchange itself. Usually, that would be no problem since the trading takes place on the next possible day, which we can indeed predict. However when it comes to the exact calculation of each day within a week and a year the incompleteness of data points is an issue. I believe that the exact position of a day within a week and a year is relevant information for the DNN since the stock market follows cyclic patterns to a certain extent. The standard shape of the date in the data frame is Day, Month, Year + Timezone which is insufficient for a DNN. We would rather have a date representation ranging from zero to one which is connected cyclically, like the sinus and co-sinus curve. This idea is actually from the TensorFlow website, which provides a full introduction into the usage

of an LSTM with TensorFlow [34].

In the code, I expanded each week to five days, no matter how many days trading took place. Afterward, it was no problem to set a new index with ascending integers, from which I calculated the sinus and co-sinus values. These four values replaced the 'Date' column and the index column with the integers. Finally, the basic input values are shown in figure 4.1.

	Open	High	Low	Close	Volume	Dividends	Stock Splits	Week sin	Week cos	Year sin	Year cos
0	30.936776	31.629662	30.845606	31.429089	3097604	0.0	0	0.000000	1.000000	0.000000	1.000000
1	31.453398	31.818074	31.009708	31.599270	3030350	0.0	0	0.951057	0.309017	0.023979	0.999712
2	31.368309	32.833095	31.198127	32.577820	6930387	0.0	0	0.587785	-0.809017	0.047945	0.998850
3	32.517046	33.690091	32.413721	33.185619	5424381	0.0	0	-0.587785	-0.809017	0.071883	0.997413
4	33.130914	33.921050	32.960732	33.143070	4977177	0.0	0	-0.951057	0.309017	0.095779	0.995403

Figure 4.1: The basic data representation

The second data set is using the same parameters like the basic data set. Additionally, the second data set includes 52 different technical indicators from different categories such as momentum, volume, volatility, and trend.

Data Normalization

After the preprocessing was done, the data had to be normalized. Normalization is absolutely necessary for the network to run correctly. You can use not-normalized data, but your predictions will be very bad. The normalization process scales the data between zero and one without losing information or destroying the relative distances between the data points. Within Python three types of normalization are common. There is the MinMaxScaler which lets you scale the data between zero and one according to the minimum and maximum value within each column. Another good solution is the RobustScaler which lets you scale the data between zero and one according to lower and upper quantile within a column. These quantiles can be set manually, according to the intensity of the scattering of the data. Finally, there is the StandardScaler which scales the data in a way that the mean of each column is zero and the standard deviation of each column is one. The StandardScaler is the one I am using in this model for the Daimler stock. The reason for that is simply because I got the best results from it. This can differ from stock to stock and from the length of the training set.

The StandardScaler from the sklean library calculates the score z of a data point x according to sklearn [35] as followed:

$$z = \frac{x - u}{s}$$

whereby:
x = the data point
u = the mean of all data points
s = the standard deviation of all data points

Timeseries Generation and Splitting

In this model, I am using the TimeseriesGenerator which is a function that generates time-series. The output of the TimeseriesGenerator is a sequence of arrays consisting of historical data points together with the defined targets. In addition to that, I am using the training set in the correct order together with one or more LSTM-layers. So the LSTM-layers should memorize the relevant data from the past.

4.1.3 Model Design

For the DNN model, I am using the Keras API [36] which is integrated with TensorFlow [34]. This API has the advantage, that the development of the model is very fast and convenient especially when compared to PyTorch. In addition to that, TensorFlow is often used and applied in the context of time-series analysis, which is why there is much information available. This is especially handy when you get started in the field of deep learning. I used Google Colab for this thesis which provided a working environment with all the suitable versions of the libraries.

Layers

For the LSTM-models, I am using one, two and a three layer LSTM with 1000 outputs each. The input on the first layer is adjusted to the shape of the TimeseriesGenerator sequences. The first layer returns sequences, so the following layers (if they exist) are also receiving sequences. The last layer returns no sequences since the Dense layer at the end of the model can't interpret sequences.

Loss Function

In the LSTM I have tried two different loss functions which are defined in section 3.2. In my experience, the MSE produced slightly better results than the MAE. Therefore I used the MSE which is defined in chapter 3.2 by Deru [1](p. 77, chapter 3.2.2) as followed:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

n = amount of training samples,
y_i = real value,
$\hat{y}_i$ = predicted value

Optimizer Function

I tested all model configurations with both, the Adam and SGD optimizer. For all models the learning rate was set to $5e-5$. The difference in performance when changing the optimizer is seen in section 4.1.4.

4.1.4 LSTM-Model Evaluation

In this section I compare the performance difference between using technical indicators and no technical indicators with the Adam optimizer and the SGD optimizer.
All models use one, two or three LSTM layers and have according to the amount of layers an equally large window size. So a one layer LSTM considers only the previous day. A two layer LSTM considers two previous day for the prediction, and so on.
All curves are shifted in a way that they predict the price of the next day. So each point on the predicted curves was known one day beforehand. The close-meshed grid should help to orientate and inspect the results.
In these all shown figures I am using the MSE as the loss function.
In the first four figures I am using the Adam optimizer and after that the SGD optimizer. All models no matter whether they use the Adam or SGD optimizer use the same learning rate of $5e-5$.
I train all models for 20 epochs, which means that the whole data set is processed 20 times before the test set is tried.
All parameters I have changed from model to model are described in the headline of the figure. The precise model specification can be analyzed in the attached code, where I have added all runs as a Jupyter Notebook.
In the following figures you can see the real price in dark blue as a function depended on time and other variables which we don't know. The light blue price is the prediction only using the basic input values which is shown in figure 4.1. We use the light blue prediction as the reference value, because the LSTM-Model only uses the market data itself, so there is no information advantage provided. The purple line is the prediction result when we provide the LSTM-model additionally with 52 technical indicators. As you can see all models detect the overall trend.
As you can see in figure 4.2, 4.3 and 4.4 where I used the Adam optimizer the long-term prediction improves when adding more LSTM layers and therefore considering more historical days for the prediction. That is made clear by a smooth prediction line, which still detects the important trends. This effect is particularly visible for the prediction using technical indicators. In figure 4.5, 4.6 and 4.7 you can see a closeup of the predictions and the real price development. Within these pictures, you can see that the prediction result using technical indicators reacts faster to changing market conditions in contrast to the prediction without using technical indicators.
When switching to the SGD optimizer which can be seen in figure 4.8, 4.9 and 4.10 the prediction result is similar to the result of the Adam optimizer. Interestingly is the fact that the overall prediction results of the SGD become worse when using three layers seen in figure 4.10 and figure 4.13. Apart from that, the prediction results of the SGD are better than the results of the

Adam optimizer. You can see this, especially in the closeup in figures 4.11 and 4.12, where the response to a trend change is almost instantaneously. In the figures 4.9 and 4.12, two days of historical data were considered using two LSTM layers. This model configuration works best in my opinion because the overall trend using technical indicators is detected very precisely. Additionally, extreme changes in the trend are detected early enough to react to the new market condition.
Generally, there is no overall trend seen in how to improve the prediction result. Adding more historical days can improve the result but is not always the right step. As we have seen changing the optimizer function had a big influence on the classification results. Changing other hyperparameters would also be worth trying out since they can also influence the prediction result.

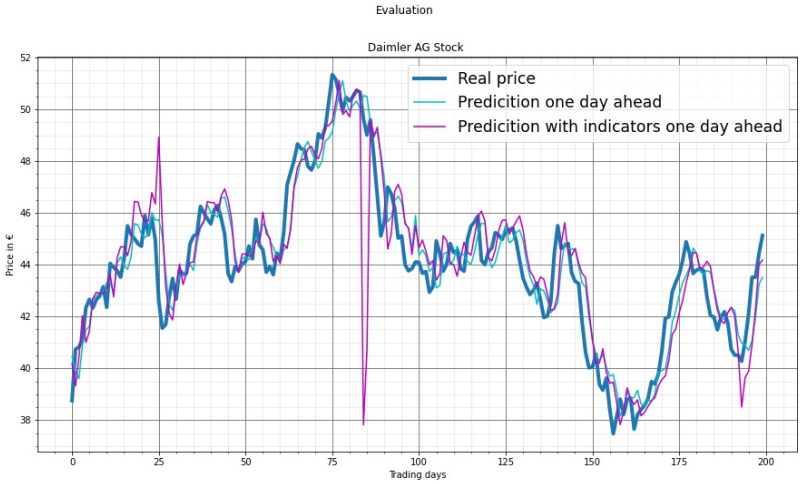

Figure 4.2: Adam optimizer used in a one layer LSTM with 1000 outputs

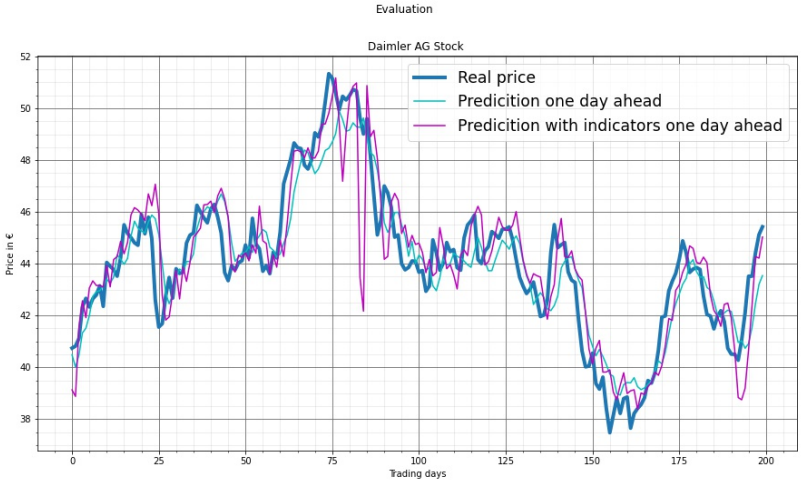

Figure 4.3: Adam optimizer used in a two layer LSTM with 1000 outputs each

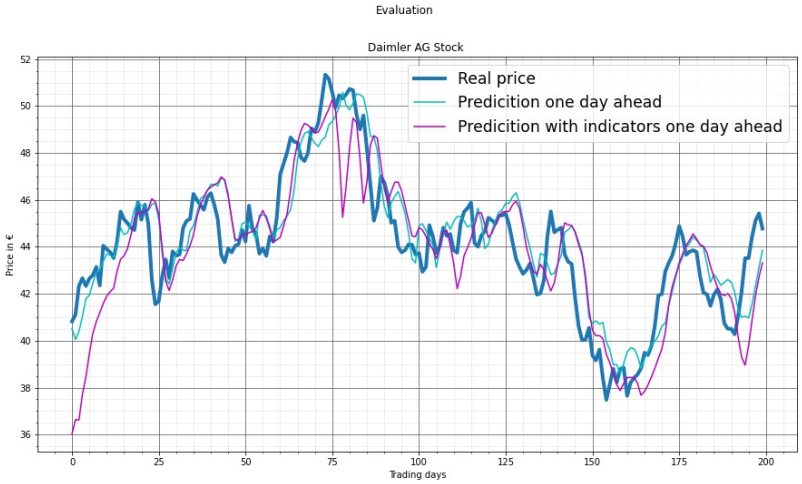

Figure 4.4: Adam optimizer used in a three layer LSTM with 1000 outputs each

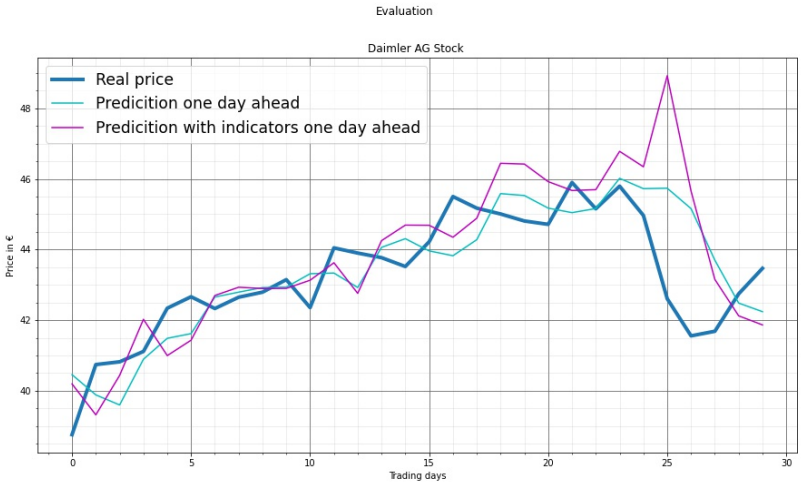

Figure 4.5: Adam optimizer used in a one layer LSTM with 1000 outputs

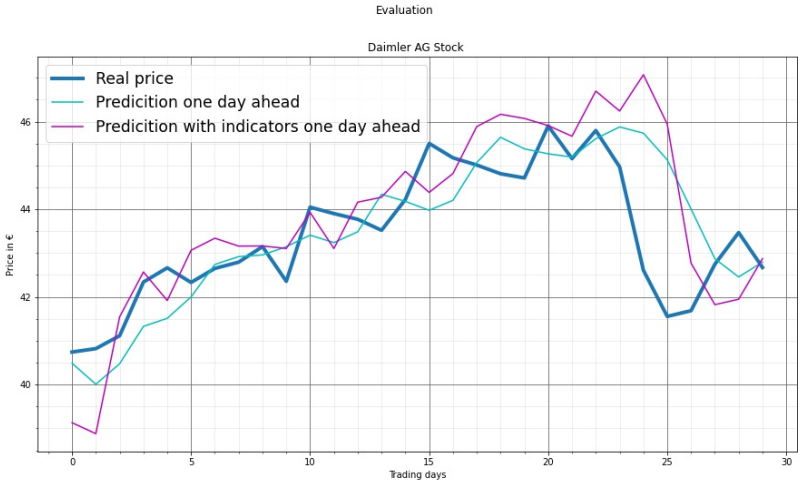

Figure 4.6: Adam optimizer used in a two layer LSTM with 1000 outputs each

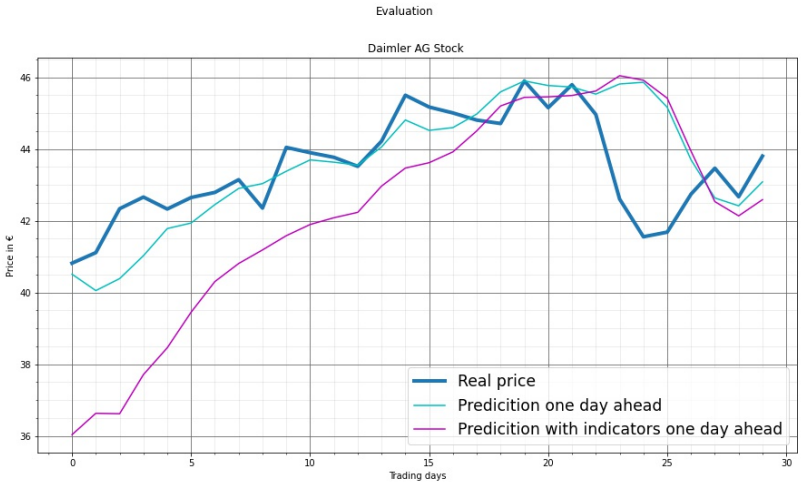

Figure 4.7: Adam optimizer used in a three layer LSTM with 1000 outputs each

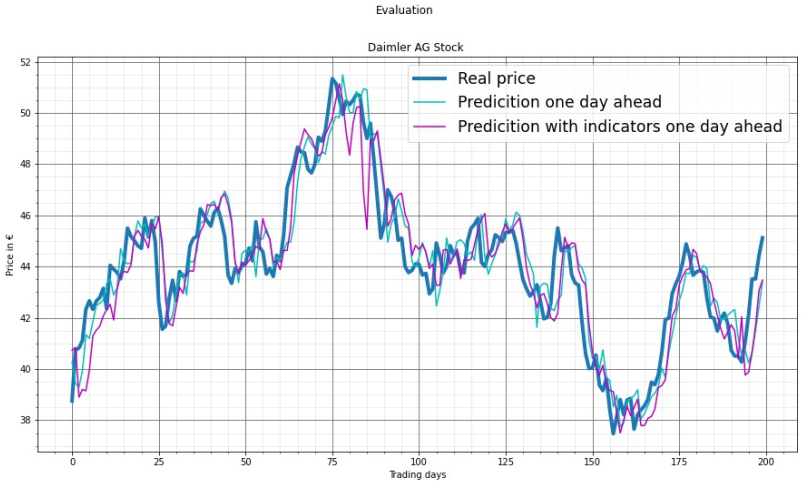

Figure 4.8: SGD optimizer used in a one layer LSTM with 1000 outputs

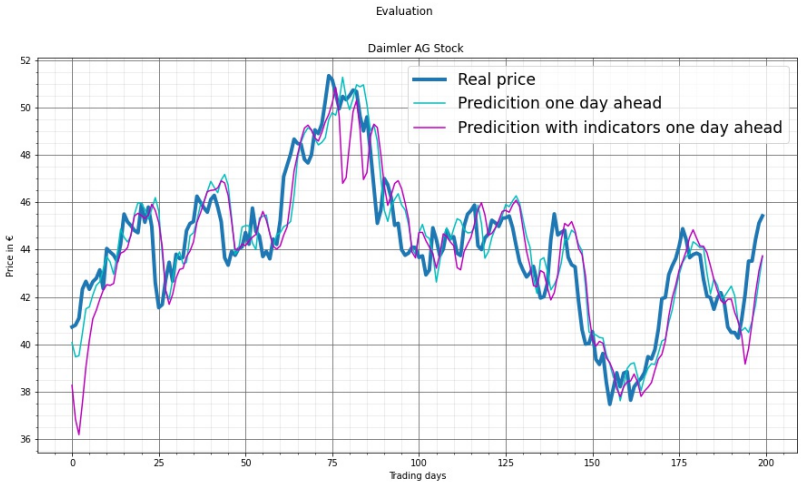

Figure 4.9: SGD optimizer used in a two layer LSTM with 1000 outputs each

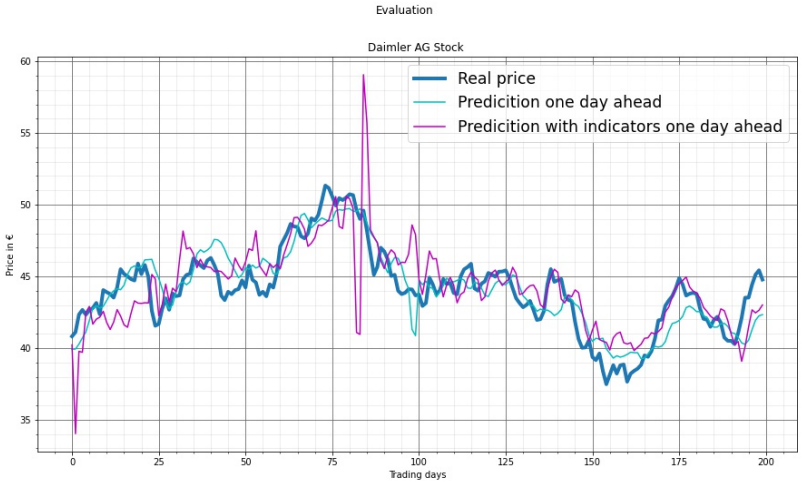

Figure 4.10: SGD optimizer used in a three layer LSTM with 1000 outputs each

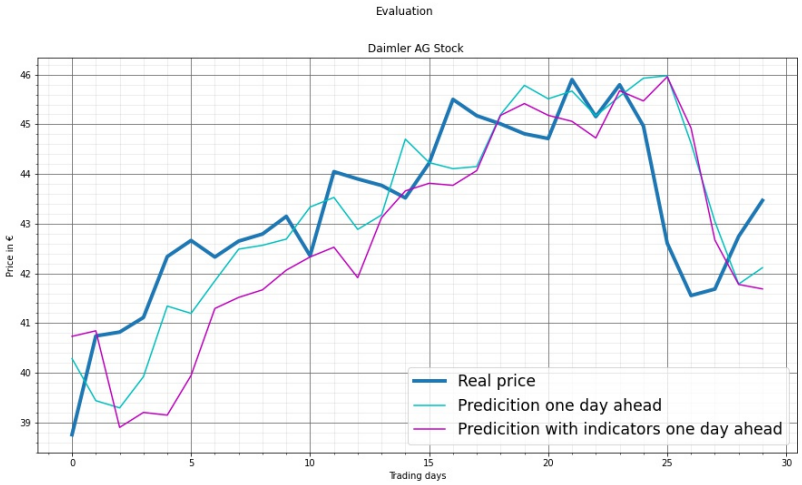

Figure 4.11: SGD optimizer used in a one layer LSTM with 1000 outputs

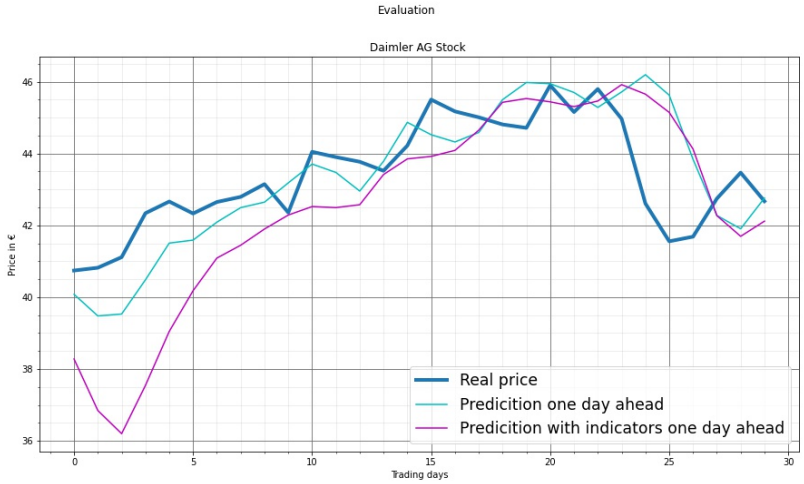

Figure 4.12: SGD optimizer used in a two layer LSTM with 1000 outputs each

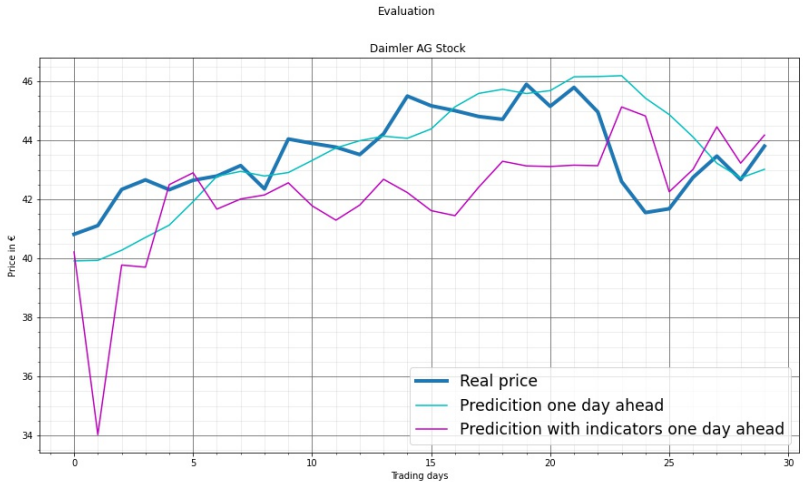

Figure 4.13: SGD optimizer used in a three layer LSTM with 1000 outputs each

Chapter 5

Discussion and Outlook

As we have seen in chapter 4, the LSTM-model I implemented using historical ticker data and technical indicators can predict the future trend up to a certain extent. Within this thesis, we have also seen that the stock market prediction problem is a very challenging one.
We have seen, that DNNs are very suitable for this task because they can deal with noisy data very well. Additionally, we have seen, that adjusting the model parameters to improve the performance is easy and fast. This is great for investors and scientists since both of them favor fast results. We have also seen, that it improves the performance of the DNN when you provide additional technical indicators. The DNN provided with technical data will predict the duration and the intensity of the trend much more precisely than a DNN not provided with technical data. Unfortunately, the technical data does not help to predict the trend earlier, which would have been a very desirable result for the investment and scientific community.
When considering the LSTM with technical Data, the good performance of the model is not ensured in every situation. There are some false predictions and there is no indication which trend change must be taken seriously. Therefore I imagine several differently trained LSTM Models together could give a hint which trend is relevant.
Another point worth mentioning is the difference in the performance of the same model configuration when applied on a different stock. This could be because of a different scattering of the historical or because other technical indicators are usually used to trade. However, when trying to predict a certain stock, I recommend training the model just with that certain stock.
To boost the performance a bigger, more complex model and better-optimized model could improve the performance. Keep in mind that my calculation power was limited. With more time and calculation power you could brute-force each model design with the Keras-tuner [22] and thereby boost the performance.
In terms of training time, it could be advantageous to implement a non-linear feature selection algorithm beforehand. Such an algorithm would sort out

the features with little information thereby boosting the training time and prediction time. Maybe this would also lead to better performance because you would eliminate contradictory technical indicators.

Bibliography

[1] M. Deru and A. Ndiaye, *Deep Learning mit TensorFlow, Keras und TensorFlow.js, 1. Edition.* Rheinwerkalle 4, 53227 Bonn: Rheinwerk Computing, 2019.

[2] V. Roman, "Recurrent Neural Networks Understand the intuition behind RNN!," tech. rep., Medium, https://towardsdatascience.com/recurrent-neural-networks-56e1ad215339, 3 May 2020.

[3] R. Mayländer, W. Krapf, and J. Biro, *Bankkaufleute Bankbetriebslehre nach Lernfeldern, 7. Auflage.* Postfach 3320, 38023 Braunschweig: Winklers, 2014.

[4] A. Thakkar and K. Chaudhari, "A comprehensive survey on portfolio optimization, stock price and trend prediction using particle swarm optimization," *Archives of Computational Methods in Engineering*, pp. 2034–2164, 2020.

[5] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[6] W. Briec, K. Kerstens, and J.-B. Lesourd, "Single-period markowitz portfolio selection, performance gauging, and duality: A variation on the luenberger shortage function," *Journal of Optimization Theory and Applications*, vol. 120, pp. 1–27, 01 2004.

[7] P. Peykani, E. Mohammadi, A. Jabbarzadeh, M. Rostamy-Malkhalifeh, and M. Pishvaee, "A novel two-phase robust portfolio selection and optimization approach under uncertainty: A case study of tehran stock exchange," *PLOS ONE*, vol. 15, p. e0239810, 10 2020.

[8] W. Lefebvre, G. Loeper, and H. Pham, "Mean-variance portfolio selection with tracking error penalization," *Mathematics*, vol. 8, p. 1915, 11 2020.

[9] S. Sefiane and M. Benbouziane, "Portfolio selection using genetic algorithm," *Journal of Applied Finance Banking*, vol. 2, pp. 143–154, 01 2012.

[10] M. Nabipour, P. Nayyeri, H. Jabani, and A. Mosavi, "Deep learning for stock market prediction," *Entropy*, vol. 22, pp. 1–25, 2020.

[11] J. J. Murphy, *Technische Analyse der Finanzmärkte, Grundlagen, Strategien, Methoden, Anwendungen.* Nymphenburger Straße 86, 80636 München: FinanzBuch Verlag, 2014.

[12] R. Aroussi, "yfinance," *https://pypi.org/project/yfinance/*, 2022.

[13] R. L. Kissell, *Algorithmic Trading Methods: Applications Using Advanced Statistics, Optimization, and Machine Learning Techniques, 2. Edition.* 125 London Wall, London EC2Y 5AS, United Kingdom: Academic Press, 04.09.2020.

[14] H.-W. Rapp and A. Cortés, *Cognitive Finance Neue Sicht auf Wirtschaft und Finanzmärkte.* Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany: Springer Gabler, 2017.

[15] S. Roy, "Testing random walk and market efficiency: A cross-stock market analysis," *Foreign Trade Review*, vol. 53, p. 001573251879718, 10 2018.

[16] J. Kremer, *Preise in Finanzmärkten, Replikation un verallgemeinerte Diskontierung.* Heidelberger Platz 3, 14197 Berlin: Springer Gabler, 2017.

[17] E. Mondello, *Portfoliomanagement Theorie und Anwendungsbeispiele, 2. Auflage.* Heidelberger Platz 3, 14197 Berlin: Springer Gabler, 2015.

[18] N. Schmidlin, *Unternehmensbewertung Kennzahlenanalyse Praxisnahe Einführung mit zahlreichen Fallbeispielen börsennotierter Unternehmen, 2. Auflage.* Wilhelmstr. 9, 80801 München: Vahlen, 2015.

[19] P. Thadani, "Financial forecasting using stochastic models: reference from multi-commodity exchange of india," *Data Science in Finance and Economics*, vol. 1, pp. 196–214, 10 2021.

[20] M. Ananthi and K. Vijayakumar, "Stock market analysis using candlestick regression and market trend prediction (ckrm)," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, 05 2021.

[21] M. J. Awan, M. S. M. Rahim, H. Nobanee, A. Munawar, A. Yasin, and A. M. Zain, "Social media and stock market prediction: A big data approach," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 2569–2583, 2021.

[22] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.*, "Kerastuner." https://github.com/keras-team/keras-tuner, 2019.

[23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[24] J. Feng and S. Lu, "Performance analysis of various activation functions in artificial neural networks," *Journal of Physics: Conference Series*, vol. 1237, p. 022030, 06 2019.

[25] Y. Bengio and Y. Lecun, "Convolutional networks for images, speech, and time-series," 11 1997.

[26] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 03 2020.

[27] A. Mittal, "Understanding RNN and LSTM," tech. rep., Medium, https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e, 12 October 2019.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[29] H. Jia, "Deep learning algorithm-based financial prediction models," *Complexity*, vol. 2021, pp. 1–9, 03 2021.

[30] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[31] H. Ge, G. Chen, H. Yu, H. Chen, and F. An, "Theoretical analysis of empirical mode decomposition," *Symmetry*, vol. 10, no. 11, 2018.

[32] I. K. Nti, A. F. Adekoya, and B. A. Weyori, "A novel multi-source information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction," *Journal of Big Data*, vol. 8, 01 2021.

[33] S.-L. Lin and H.-W. Huang, "Improving deep learning for forecasting accuracy in financial data," *Discrete Dynamics in Nature and Society*, vol. 2020, pp. 1–12, 03 2020.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous systems." https://www.tensorflow.org/, 2015.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift