Master Thesis

University of Cologne

Institute for Theoretical Physics

Explainable Artificial Intelligence and Deep Learning for Analysis and Forecasting of Complex Time Series: Applications to Electricity Prices

March 15, 2023

Julius Trebbien

Julius Trebbien
Explainable Artificial Intelligence and Deep Learning for Analysis and Forecasting of Complex Time Series: Applications to Electricity Prices

Referees:
Prof. Dr. Dirk Witthaut
Prof. Dr. Leonardo Rydin Gorjão

Supervisor:
Maurizio Titz

Institute for Theoretical Physics
Faculty of Mathematics and Natural Sciences
University of Cologne

Submission Date:
March 15, 2023

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Julius Trebbien

Remark

Parts of this thesis are accepted for publication in the journal Energy & AI (`https://www.sciencedirect.com/journal/energy-and-ai`). The manuscript is available at (`https://www.sciencedirect.com/science/article/pii/S2666546823000228`).

Abstract

A rapid energy transition from fossil fuel based generation to renewable energy sources is vital for the mitigation of climate change but requires complex market structures to manage the coordination of generation and demand. In particular, the German day-ahead market reacts to short-term forecasts one day prior to delivery and is driven by various external drivers. Its understanding and forecasting are essential for the energy transition as it allows renewable energy operators to make profits and promotes key technologies for a stable grid operation, such as battery storage.

In this work, we analyze the German day-ahead electricity market using eXplainable Artificial Intelligence (XAI) and forecast electricity prices using deep neural networks. We investigate the application of SHapley Additive exPlanations (SHAP) to study the driving factors of electricity prices. The dataset includes several power system features such as load or renewable forecasts but also fuel prices. Our analysis suggests that load, wind and solar generation are the central external features driving prices, as expected, wherein wind generation affects prices more than solar generation. Similarly, fuel prices also highly affect prices in a nontrivial manner. Moreover, large generation ramps are correlated with high prices due to the limited flexibility of nuclear and lignite plants. Based on the results from the XAI method, we establish Long Short-Term Memory (LSTM) networks to forecast electricity prices. We introduce a probabilistic forecast as output, increasing the applicability of the model. The LSTM model is able to outperform models from related works and enables additional applications using the predicted standard deviation.

Contents

1. Introduction

Mitigating climate change is one of the most important and difficult challenges facing humanity. A rapid and fundamental transformation of our energy system is necessary to limit global warming and its resulting impacts [Rog+15]. This includes decarbonizing electricity generation, heating, transportation and other sectors.

An essential part of this energy transition lies in the electricity grid. The reliable supply of electric power is vital for modern societies [VL10; Pra16] while the electrification of other sectors increases this dependency on the electricity grid even more. A stable operation of the electric power system requires that power generation and load are always balanced [WWS13]. Because of the ongoing energy transition, this coordination of generation and demand is becoming increasingly challenging. Generation from renewable sources such as wind and solar power is determined by the weather and therefore highly volatile [SP18].

Despite the challenges and the increasing complexity of the power grid, the amount of publicly available data has enabled researchers across various fields to study and optimise this critical infrastructure. Interdisciplinary work between fields such as statistical physics, nonlinear dynamics, network science, data science and machine learning provides further insights into the operation, stability and resilience of the power grid [Wit+22; Mac+22].

Electricity markets are critical for coordinating generation and demand prior to the actual delivery of electricity. To improve efficiency and reduce costs, European electricity markets were liberalised starting in the 1990s [JP05]. Before, generation, transmission and distribution were typically integrated into a single company holding a regional monopoly. Today, several markets exist in Europe which enable electricity trading on different time horizons [Eur19].

In particular, the day-ahead markets trade on a short-term basis and enable coupled trading across Europe. Market participants use forecasts to elaborate optimal trading strategies one day before actual delivery, while the fundamental framework of the market ensures that power generation matches load.

Accurate understanding and forecasting of electricity prices are critical for the energy transition. It enables renewable energy operators to make profits from the market by anticipating price movements and promotes smart applications for demand control [Tsc+22]. In general, the electricity price time series shows intricate statistical properties, including heavy tails and strong correlations [Han+22a]. Recently, European energy markets were heavily disturbed by the Russian invasion of Ukraine causing an intensified research interest (see, e.g. [OČ22; Zak+22; Böt+23]).

Overall, this thesis aims to contribute to a better understanding of the German day-ahead electricity market and the development of tools for the analysis and forecasting of electricity prices. It will provide insights into the current behaviour of the German day-ahead electricity market and use this knowledge to efficiently forecast electricity prices.

For the analysis of market behaviour, we use eXplainable Artificial Intelligence (XAI) techniques, enabling detailed explanations of driving factors of the markets. XAI is an emerging field that focuses on developing machine learning models that are able to provide understandable and transparent explanations for their predictions and decisions [Mol20]. The use of XAI in the analysis of electricity markets can help to increase the transparency and accountability of the markets and can also provide valuable insights into the factors that influence electricity prices [Mac+22]. We establish a Gradient Boosted Tree (GBT) model for the German electricity prices and apply SHapley Additive exPlanations (SHAP) to explain the model. We use an extended data set to identify driving factors which are commonly neglected in elementary studies [Lag+21]. The model substantially outperforms the more commonly employed merit order principle, revealing additional detail into the function of the market. For instance, the model quantifies the impact of fossil fuel prices and load ramps, as well as nonlinear interactions of different features. This takes us one step further in accurately understanding the drivers of electricity prices.

Based on the information gained using XAI, we establish a forecasting model using Long Short-Term Memory (LSTM) deep neural networks and analyze the added value of probabilistic outputs. The use of deep neural networks for price forecasting has the potential to improve the efficiency and accuracy of electricity trading and can also help to optimise the operation of the electricity grid [Lag+21; Tsc+22]. Accurate price forecasts can help market participants make informed trading decisions and can also support the operation of the grid by providing anticipatory information on the expected demand and supply of electricity. Using probability distributions as model output, the model is able to outperform a comparable model of Tschora et al. [Tsc+22] with its mean prediction. Additionally, the model allows to quantify the prediction uncertainty via the standard deviation. For instance, negative prices and high price peaks are predicted with a high standard deviation since their accurate prediction is usually hard.

The remaining thesis is structured as follows. In Chap. 2, we provide an introduction to the German electricity markets with focus on the day-ahead market and its approximations. We will give an introduction to the field of machine learning, eXplainable Artificial Intelligence (XAI) and Artificial Neural Networks (ANN). In Chap. 3, we provide the fundamentals of machine learning models. We then continue to explain more advanced machine learning techniques, focusing on Feedforward Neural Networks (FFNN) and Recurrent Neural Networks (RNN) in Chap. 4 and introduce methods of XAI in Chap. 5.

Methods and results of the XAI analysis of the German day-ahead electricity market are presented in Chap. 6. In Chap. 7, we discuss the methods and results of the proof of concept using LSTM networks for probabilistic electricity price forecasting. Conclusion and outlook are presented in Chap. 8.

2. Electricity Markets

The purpose of this chapter is to provide a brief introduction to the electricity markets in Germany with focus on the day-ahead market within the German bidding zone. We start with a broad overview of the overall market structure for electricity trading in Sec. 2.1. In Sec. 2.2, we discuss the structure of the day-ahead market in detail with particular focus on the merit order effect and the Single Day-Ahead Coupling (SDAC).

2.1. Overview

Electric power has been a crucial component in almost all aspects of our daily lives since its transmission and distribution became technically feasible. Today, the social and economic system is highly dependent on the reliable supply of electric power. Therefore, the power system is considered one of the most critical infrastructures in society [VL10].

As the aggregation of generation and demand for larger regions helps to further stabilize the reliability of electricity supply, large synchronous AC grids have emerged aggregating generation up to hundreds of GW. Each generator connected to the grid operates at approximately the same frequency, with the exception of tiny spatial fluctuations [Ryd+20]. For the purposes of this thesis, we will confine our work to the central European grid, which operates at a reference frequency of 50 Hz.

Due to the inability of the transmission grid to store electricity itself, generation and demand must be balanced at all times. Otherwise, frequency would deviate from its set value, which could eventually lead to a power system blackout. Various layers of control reserve are implemented to balance the generation but are limited to small imbalances.

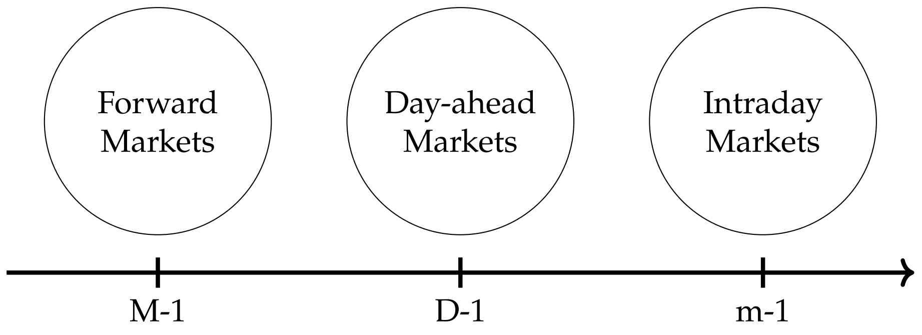

Figure 2.1.: An outline of the organizational structure of the German power market. Forward Markets begin trading financial contracts for electricity one month in advance of actual delivery. Day ahead-markets remain open until the day before delivery, while the intraday markets close just minutes before delivery.

Electricity markets are critical for coordinating generation and demand prior to the actual delivery of electricity. Since transmission capacities between regional areas are limited and create bottlenecks for grid stability, different bidding zones exist. These various bidding zones exist largely between countries, with some countries sharing a bidding zone or being divided into several inter-country bidding zones. Bidding zones are crucial to ensure that regional market conditions are reflected in the price while they also help to indicate constraints in transmission. European electricity markets were liberalized in the 1990s, which led to more efficient markets with lower costs due to competition between different suppliers [JP05].

The current transition from mainly fossil fuel-based power plants to more renewable generation is one of the main challenges in today's markets. Conventional power plants based on fossil fuels are connected to the grid via synchronous generators with intrinsic inertia, resulting in a stabilizing effect for the grid [Mil+18]. In contrast, renewable power plants like wind and solar generation are connected to the electricity grid via power electronic converters, which do not serve any inertia. Additionally, renewable power plants are determined by the weather and are thus highly volatile [SP18], making also the electricity market highly volatile [Han+22a]. This makes the efficient coordination of generation and demand increasingly challenging [Wit+22].

To address the evolving challenges of the energy transition, the structure of electricity markets has become more and more complex and is constantly changing. We will now concentrate solely on German markets, i.e. the bidding zone of Germany. Notably, Germany shares its bidding zone with Luxembourg and also shared it with Austria until 1st of October 2018 [EEX18]. In the following, we will only refer to these shared bidding zones as *German markets* for convenience. We will also focus only on the current market regulations and processes.

The German markets can be separated into regular trading markets from long-term future trading to short-term spot trading and reserve markets for different security measures. For an overview of the regular trading markets see Fig. 2.1. The first regular trading markets are the forward markets, where financial trading is done to mitigate the risks of changes in the spot prices. This takes place up to several years before the physical delivery of electricity.

The spot markets are the main market for physical trading of electricity and consist of the day-ahead and intraday markets. The day-ahead market is open until 12:00 the day before delivery, where an auction matches bids and offers and sets the corresponding electricity price as detailed below. Electricity is traded in hourly blocks for the 24 hours of the following day. Closing time and block size can vary on different exchanges but mostly follow the mentioned structure. The largest part of the day-ahead market is coupled via the Single Day-Ahead Coupling (SDAC), which extends trading over bidding zone borders with respect to transmission capacities. Section 2.2 explains the day-ahead market in greater detail.

Shortly after the publication of day-ahead market results, intraday market trading begins. The intraday market can be separated into auction-based and continuous trading. While the auction-based intraday market follows the structure of the day-ahead market with a daily auction at 15:00, the continuous intraday market opens at 15:00 and allows for continuous trading up until 0 minutes before delivery. Electricity is mainly traded in hour and quarter-hour blocks, while different exchanges offer alternative trading blocks [EPEb]. In the continuous market, bids and offers get matched instantaneously, allowing for quick responses against emerging mismatches due to weather changes or unplanned unavailabilities. The largest part of the continuous intraday market is coupled via the Single IntraDay Coupling (SIDC), which allows for instantaneous trades across bidding zone borders if transmission capacities are not reached. Once again, opening and closing times can vary on different exchanges but mainly follow the mentioned structure.

Since intraday markets open after day-ahead markets are closed and are smaller in volume, intraday prices are largely determined by the previous day-ahead auction. Most deviations are caused by short-term system changes, such as weather forecast errors. This leaves the day-ahead markets as the main driver for general electricity prices reacting directly to availability of generation and load.

Additionally, electricity is traded through non-public contracts between electricity producers and consumers via Over-The-Counter (OTC) agreements. Both parties close a direct contract that does not take place on any energy exchange and are only obliged to report volumes and delivery times. As a result, prices usually remain unknown to the public. Most contracts are arranged on a long-term basis and are used mainly by large market players. OTC trading is still one of the most popular forms of trading.

The reserve markets trade electricity reserves that are activated in the event of frequency deviations due to imbalances between generation and demand. In German markets, there are three different frequency control measures. They are activated at different times and therefore require different participants. In order to ensure sufficient capacity for the control measures, reserve capacity is traded separately for each control measure. Prices for control reserve are set for negative and positive reserve capacity separately. Each market follows a complex procedure, which we will not elaborate on in further detail.

2.2. Day-ahead Electricity Market

In this section, we will focus on the German day-ahead electricity market in more detail. Again, we emphasize the fact that Germany shares its bidding zone with Luxembourg and also shared it with Austria until 1st of October

2018 [EEX18]. In the following, we will refer to these shared bidding zones only as *German markets* for convenience.

While OTC and forward trading focus primarily on long-term price development, the day-ahead markets trade on a short-term basis. Market participants use forecasts to elaborate optimal trading strategies one day before actual delivery. The fundamental framework of the market ensures that power generation matches load. Forecasts include demand forecasts and therefore consumer behaviour of the next day, wind power plant generation forecasts and solar power generation forecasts. Since most forecasts are dependent on weather changes, precise weather forecasts are essential for day-ahead markets. Day-ahead markets are larger in volume than intraday markets, which mainly react to forecast errors or unplanned unavailabilities. Therefore, the electricity price resulting from day-ahead markets is the mainly used reference for general price development.

Day-ahead markets are operated by Nominated Electricity Market Operators (NEMOs). NEMOs are organizations designated by the competent authority to perform all tasks related to the coordination of production and demand in the respective markets [ALL22]. In Germany, the three designated NEMOs are EPEX Spot SE, Nord Pool EMCO AS and EXAA AG [Eur22]. Trading is possible via the different exchanges hosted by the NEMOs respectively. In the following, we will refer to the different exchanges by the name of the hosting NEMO.

Most European day-ahead markets are coupled via the pan European Single Day-Ahead Coupling (SDAC) which optimizes trading across bidding zone borders. The SDAC is explained in detail in Sec. 2.2.2. The auction for the SDAC is generally opened at 10:00 the day before delivery. Depending on the exchange this may differ due to different publishing schedules of the respective NEMOs and TSOs. All markets exchanges close trading at 12:00 for the SDAC without exception [Küh+21]. Electricity is traded for all 24 individual hours of the following day with special trading blocks depending on the operating exchange. While EPEX Spot and Nord Pool only offer trading for the SDAC [EPE22; Nor], EXAA additionally offers a regional market auction. The independent *Classic Auction* on EXAA closes at 10:15, 105 minutes before the closure of the SDAC. The auction offers trading of all 24 individual hours and also all 96 individual quarterly hours of the following day [EXA]. Notably, the individual EXAA auction is small in volume compared to the volume of the SDAC.

To understand the process of auction trading in the day-ahead market we will assume an uncoupled German market with only one operating exchange for now. We also assume the more popular hourly trading intervals. In this simple and uncoupled market, electricity producers and consumers can place orders at the exchange for all 24 hourly trading intervals of the following day. Orders are placed in the order book, where all orders are listed. Order books in electricity markets are usually anonymous and list both sides of a trade, called offers and bids. Offers are placed by electricity producers who offer a specific amount of electricity at one price for one of the trading intervals. The offered capacity from the producer is the volume of the offer. Oppositely, bids are placed by electricity consumers who bid for a specific amount of electricity at one price for one of the trading intervals. The requested capacity from the consumer is the volume of the bid.

Since the day-ahead market is auction based, bids and offers are collected inside the order book until market closure. Aggregating both offers and bids in volume creates the supply and demand curves (see Fig. 2.2). Supply and demand curves are usually only used after market closure with the full order book. The supply curve shows the available production capacity if a specific price is paid. Oppositely, the demand curve shows the capacity which would be bought for a specific price.

The intersection between supply and demand curve creates the Market Clearing Price (MCP), which is the resulting overall price of the day-ahead market. Every offer below the market price and every bid above the market price gets executed exactly at the MCP. Therefore, every electricity producer gets the same payment, while every consumer pays the same price, regardless of the initial price of the order. This market clearing procedure is called *pay-as-cleared* (in contrast to *pay-as-bid*). It ensures that power plants win the bid in the order of their bidding prices which is called the *merit order principle* [SRG08].

One peculiarity of the electricity market is the possibility of negative prices. This may occur in periods of high renewable electricity production due to

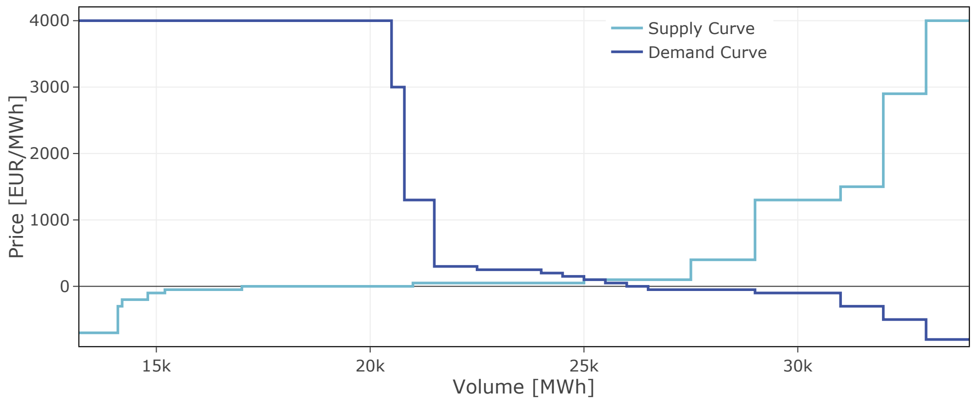

Figure 2.2.: Plot of the aggregated order curves for 1 hour in the day-ahead market. The supply curve increases in prices for higher volume and corresponds to the offers in the order book. The demand curve decreases with volume and corresponds to the bids in the order book. The Market Clearing Price (MCP) is the intersection of the supply and demand curves. Therefore, every offer below the MCP is executed, as well as every bid above it.

favorable weather conditions, while fossil fuel based power plants are unable to ramp down production. The inflexibility of generation can be attributed to power plants with limited ramping speeds, such as nuclear power plants, or to the high additional costs of ramping up generation again. Looking at the aggregated bidding curve, the supply curve moves to the right since more production volume is available, moving the intersection of both curves and therefore the MCP to lower and eventually negative prices.

2.2.1. Merit Order

In this section, we will briefly review one of the most common approximations of the day-ahead market based on the merit order principle. This base model will serve as a benchmark model for different parts of the thesis.

The electricity demand or load L is mostly inelastic in the short term [Lij07]. Demand side management aims to increase the short-term flexibil-

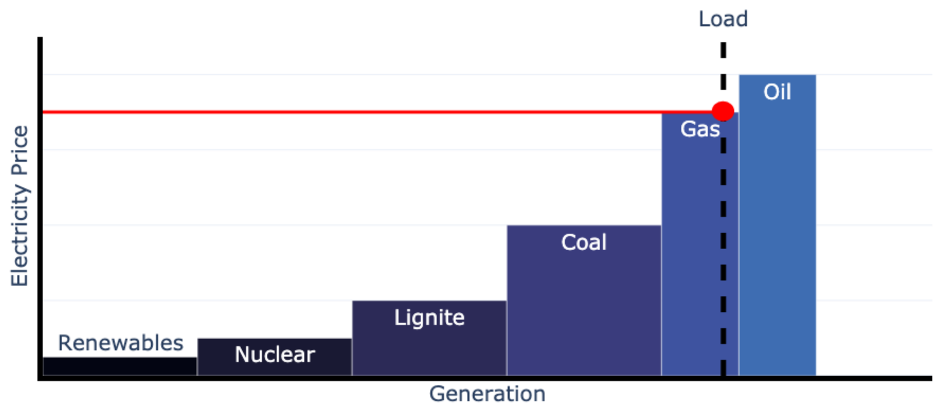

Figure 2.3.: Visualization of the merit order principle. Generation is sorted according to their estimated marginal cost. The load is approximated as independent of the price. The price is set by the power plant with the highest marginal cost needed to match the load. Since renewable generation has the lowest marginal cost, they are always included in the merit order principle. Therefore, power plants using fossil fuels, like coal, gas or oil power plants determine the price.

ity and elasticity of demand, such that it can be adapted to the availability of renewable power generation. However, the progress of demand side management in the German market is very limited as private consumers and small enterprises typically experience no price signals [Han+22b]. Furthermore, day-ahead trading relies only on load forecasts, not the actual load. Load forecasts in the ENTSO-E region are calculated on the historic load profile on similar days [ENTb] and generally assume "inelasticity of the load to price" [ENT23]. Hence, we assume that L depends only on the time t, but not on the price p for the time being.

In perfect competition, the demand is satisfied by generating units according to their marginal costs. All units with marginal costs below the market price p can realize positive contribution margins and are thus "on", all others are "off". Hence one can obtain an approximate view of the market outcome by sorting all generating units according to their estimated marginal costs. Figure 2.3 shows an example of this approximation. Renewable power plants, in particular wind and solar power, have high investment costs but almost vanishing variable costs. As marginal costs are dependent on the variable costs only, the marginal costs of these two renewable power sources are usually neglected. Furthermore, renewable plants are prioritized in the German market according to national regulations ("Erneuerbare Energien Gesetz"). Hence, we can assume that the total renewable generation G_{ren} is independent of the market price p. However, renewable generation depends on the weather and thus varies strongly with time t. If the availability of the dispatchable power plants varies little in time, we can assume that the generation G_{dis} depends only on the price p. In the German market, nuclear and lignite power plants have the smallest marginal costs and thus contribute first. Notably, the marginal costs of individual power plants are not known exactly and must be estimated.

We can now formulate the condition of market equilibrium. Supply and demand match if

$$L(t) = G_{\text{ren}}(t) + G_{\text{dis}}(p(t)). \tag{2.1}$$

Solving for the price yields

$$p(t) = G_{\text{dis}}^{-1}\left[L(t) - G_{\text{ren}}(t)\right], \tag{2.2}$$

where G_{dis}^{-1} denotes the inverse function. That is, the market price is a function of the difference of load and renewable generation, which is commonly referred to as the residual load.

2.2.2. SDAC

The Single Day-Ahead Coupling (SDAC) aims to create a single pan European cross zonal day-ahead market. Therefore, demand and supply orders in one bidding zone are no longer confined to the local territorial scope. The main benefit of market coupling is the increase in overall efficiency of the market by promoting effective competition, increasing liquidity and enabling more efficient utilization of generation resources across Europe [ENTe]. This also results in less volatile prices across all European bidding zones. Notably, it is not possible to simply aggregate the order books from all bidding zones as transmission capacities are limited.

SDAC was first implemented in February 2014 for the bidding zones in North-West Europe (NWE) and has been extended to include all European bidding zones with a few exceptions [EPEa]. One notable exception is the excluding of the Swiss day-ahead market, which results in Swiss prices being published before the closure of the SDAC markets. A full documentation of the extension of the SDAC can be found at [ENTe].

The SDAC makes use of an algorithm called *PCR EUPHEMIA*, which calculates electricity prices and allocates auction based cross-border capacities for all involved bidding zones. It calculates output for all bidding zones and all 24 individual hours of the following day simultaneously in under 17 minutes [N-S].

EUPHEMIA takes inputs submitted by TSOs and NEMOs and maximizes social welfare. Social welfare is defined as the consumer and supplier surplus including congestion rent. Inputs are all order books including bids and offers from the respective NEMOs and networks capacities and constraints from the respective TSOs. Network constraints include restrictions on interconnectors. Possible restrictions are losses through lines, lines being subject to tariffs and ramping limits on lines [ALL20]. Price limits are also taken into account.

The resulting outputs for all participating bidding zones are the clearing prices, all matched trades, scheduled exchanges and net positions. Cross-border flows are allocated automatically with respect to all network constraints. This offers implicit auctions, where all involved traders only bid for electricity and do not need to reserve cross-border capacity beforehand.

The European market coupling via SDAC is subject to a strict schedule, which ensures that the smallest errors in the preparation of the coupling can be compensated. After market closure of all participating markets at 12:00, PCR EUPHEMIA starts its calculation. If errors occur in the inputs or the calculation, there exist clearly defined guidelines for possible solutions. For example the partial decoupling of bidding regions or a second round of auctions if extreme prices occur. A complete description of all processes can be found in [ALL20] and [EPE22].

3. Fundamentals of Machine Learning

This chapter serves as a brief introduction to the notation and basic concepts of machine learning. Notation follows the work of Goodfellow et al. [GBC16] and Hastie et al. [HTF09], which also provide more details about the concepts of machine learning. We will start with an overview of the basic methods of machine learning in Sec. 3.1. After that, we focus on tree-based machine learning models, starting with decision trees in Sec. 3.2 and building up to Gradient Boosted Trees (GBT) in Sec. 3.3.

3.1. Overview

Machine Learning (ML) is considered a subset of Artificial Intelligence (AI) and is concerned with algorithms that are able to learn from data. Learning in this context means the ability of the algorithm to improve its output by incorporating new information from received data. A learning task typically consists of input data and an unknown implicit or explicit mapping to an output. Input data is referred to as features and the output is called target. The output of the model is referred to as prediction.

Supervised learning describes learning tasks based on labeled input data. Each sample of data is associated with a label, and the model must learn the relationship between data and label. The labels are provided by an instructor or teacher who shows the model what mapping to learn, which is why this type of learning is called supervised learning. Unsupervised learning, on the other hand, operates without the guidance of an instructor or teacher. Data points are not labeled and the model needs to extract a meaningful structure from the input data on its own without being told what the output should look like. Neither term is formally defined, so the line between supervised and unsupervised learning is blurry. Additionally, reinforcement learning is a subset of machine learning where the algorithm interacts with its environment. An agent learns how to act in an environment through feedback loops between the learning system and the agent's experience. In this thesis, we focus on supervised learning.

Labels of datasets in supervised learning tasks can be categorized into two main subclasses. First, classification is the task of assigning different categories, called classes, to the input data. The labels of the data usually consist of at least two different integers representing each class. The model's task then is to predict the class of a sample from the input features. An example of a classification model would be the classification of handwritten numerical values. Second, regression is the task of assigning a numerical value to the input data. Each sample of the input data is labeled with a real number that needs to be predicted. An example of a regression task is the prediction of a regional temperature for the next day.

In the following, we will only focus on regression models. Therefore, all equations are written to illustrate regression tasks but can be easily adapted to classification tasks as well. In a supervised regression task, the goal of the algorithm is to approximate an unknown function $f(\mathbf{x}) = y$ based on the labeled data $(\mathbf{x}, y) \in D \subset \mathbb{R}^m \times \mathbb{R}$, where $\mathbf{x}$ is a vector summarizing the values of the input features and y is the target. The output of the model is defined as $\hat{f}(\mathbf{x}) = \hat{y}$. The learned function $\hat{f}$ is referred to as a hypothesis h chosen from the hypothesis space $\mathcal{H}$. The hypothesis space is dependent on the choice of the model. Note that we will only focus on single-output regression tasks for simplicity.

To measure the performance of the model, a loss function L, often also called error or cost function, is used. The loss function quantifies how close the predictions of the model are to the real values. A commonly used loss function is the squared loss defined as

$$L(y, \hat{y}) = \sum_{(\mathbf{x}, y) \in D} (y - \hat{f}(\mathbf{x}))^2. \tag{3.1}$$

With the loss function, we can define the best choice of the model depending on the input data as

$$\hat{f} = \operatorname*{arg\,min}_{h \in \mathcal{H}} \sum_{(\mathbf{x}, y) \in D} L(y, \hat{y}). \tag{3.2}$$

In real world scenarios, access to data is usually limited, resulting in finite datasets. However, if enough data is available, the model is still able to learn the underlying mapping of the data, provided that the subset of data is evenly distributed from the full dataset.

Finding the function $\hat{f}(\mathbf{x})$ that best approximates $f(\mathbf{x})$ for a finite subset of data is just an optimization problem. The more important task of machine learning is to find a function that not only performs well on seen data but also on unseen data. To approximate the generalization error of the model, meaning the performance on unseen data, the full dataset is split into two separate sets. The train set is used to train the model, i.e. perform the optimization in Eq. (3.2). The test set is not included in the optimization process and therefore serves as a proxy for the generalization error.

While the model learns by minimizing the training error, it also needs to have a low generalization error. More generally, the generalization gap should be low as well. The generalization gap is defined as the difference in the performance on the train and test set. If the model performs well on the train set but has a large generalization gap, the model overfits. Oppositely, if the generalization gap is small, but the performance on the train set is low, the model underfits. Overfitting and underfitting are one of the key challenges when it comes to the choice of the machine learning model. Overfitting arises due to over-complex models, while underfitting arises due to under-complex models. The complexity of the model is usually referred to as capacity. The capacity of the model controls the complexity of the resulting prediction function $\hat{f}(\mathbf{x})$ and therefore defines the hypothesis space $\mathcal{H}$.

A simple example is the approximation of data based on a quadratic function seen in Fig. 3.1*(top)*. A linear model lacks the complexity to approximate data based on quadratic functions. The prediction function is not able to model the data, which causes underfitting. Increasing the capacity of the model to incorporate polynomials up to the ninth order leads to excessive overfitting of the data. The model perfectly fits the train set but is unable to

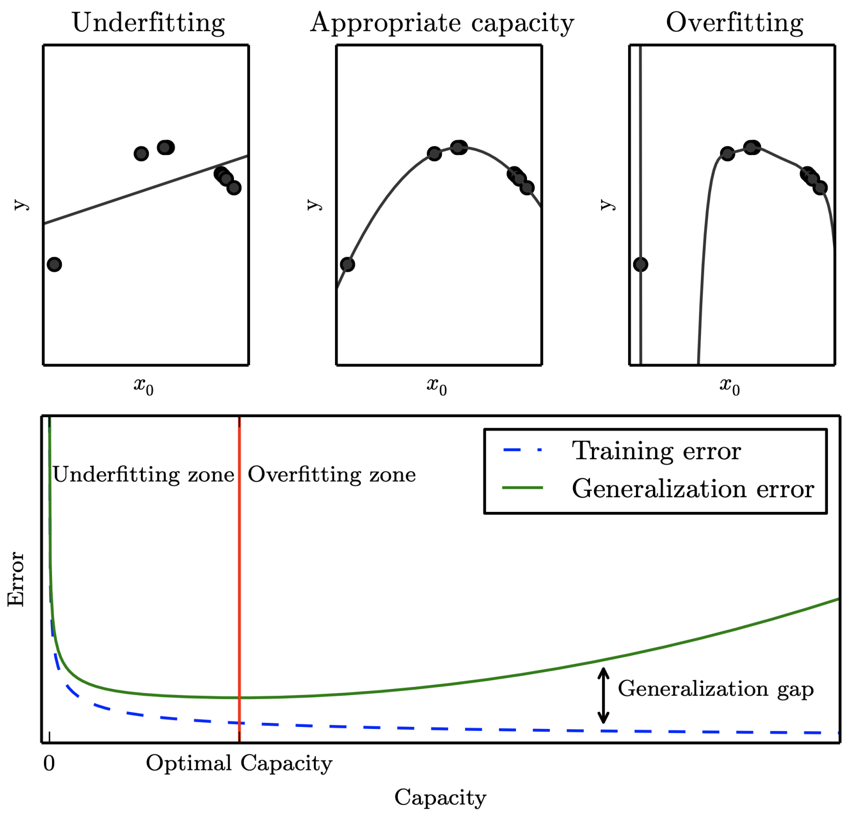

Figure 3.1.: Concept of underfitting and overfitting. *(top)* Different polynomials are fitted against several data points. The underlying function of the data is quadratic with added noise. Linear fit: The model lacks complexity and is unable to model the data resulting in underfitting. Quadratic fit: The model fits the data well and is able to generalize to unseen data. Ninth-order polynomial fit: The model fits all data points perfectly but is unable to generalize to new, unseen data. The model suffers from overfitting. *(bottom)* Typical relationship of model complexity and generalization error. Low complexity minimizes the gap between train and test error but lacks overall performance. High complexity models have good performance on the train set but are unable to adapt to new data. Therefore, the generalization error increases. The optimal model consists of a good trade-off between underfitting and overfitting.
(Figure reproduced from Ref. [GBC16].)

correctly predict unseen data from the test set. Choosing a moderate complexity allows the model to generate a good approximation of the data while keeping the generalization error low. The overall impact of the capacity can be seen in Fig. 3.1*(bottom)*.

Handling the problem of overfitting and underfitting can be challenging, depending on the learning task. In general, it is not obvious which class of machine learning models suits a problem best. Furthermore, each class of machine learning models can be optimized for particular learning tasks using hyperparameters. Hyperparameters define the complexity of the model and can be divided into parameters that change the initial complexity of the model and parameters that limit the complexity of the model during the training process. Depending on the choice of the model, two of the most common techniques to limit the complexity during training are regularization with respect to the model parameters and early stopping.

When using regularization, an additional term, called the regularization term, is added to the optimization problem. The regularization term punishes more complex models, resulting in simpler models if the performance of the models is not significantly reduced. The learning task can be written as

$$\hat{f} = \arg\min_{h \in \mathcal{H}} \sum_{(\mathbf{x},y) \in D} L(y, \hat{y}) + \lambda \Omega(h), \tag{3.3}$$

where $\Omega(h)$ is the regularization function that penalizes complex models. The parameter λ is the regularization parameter controlling the trade-off between minimizing the error or the model complexity.

Early stopping makes use of an additional unseen test set and is often used for iterative machine learning algorithms. A small part of the train set is used as a validation set, which serves as an unseen dataset during the training process. The model is trained on the rest of the train set and is evaluated on the validation set at each iteration of the algorithm. If the performance on the validation set does not increase for a predefined number of iterations, the algorithm terminates and the iteration step is selected that gives the maximum performance on the validation set.

Once a model has been successfully trained, it can be further validated using performance measures other than the loss function used during training. Here, we present the metrics used throughout the thesis to verify performance on the task of electricity price prediction and forecasting.

The Mean Absolute Error (MAE) serves as a good intuitive performance measure and is defined as the average distance between the prediction and the actual value given by

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \,. \tag{3.4}$$

Another common performance measure for price time series is the Symmetric Mean Absolute Percentage Error (SMAPE). It is based on the Mean Absolute Percentage Error (MAPE), given by

$$\mathrm{MAPE}(y, \hat{y}) = \frac{100}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{|y_i|}, \tag{3.5}$$

but ensures that prices close to 0 that are incorrectly predicted do not lead to unnecessary large errors. The SMAPE is defined as

$$\mathrm{SMAPE}(y, \hat{y}) = \frac{100}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{\frac{1}{2}\left(|y_i| + |\hat{y}_i|\right)}. \tag{3.6}$$

Finally, the R^2-score is another common error metric that provides easily interpretable results. It normalizes the quadratic error $(y_i - \hat{y}_i)^2$ of every sample by the quadratic deviation from the mean $(y_i - \bar{y})^2$. Roughly speaking, it quantifies the variance in the data explained by the model. Therefore, a score of 1 corresponds to a perfect prediction and a score of 0.5 would mean that 50% of the variance can be explained by the model. The R^2-score is defined as

$$\mathrm{R}^2(y, \hat{y}) = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{3.7}$$

where $\bar{y}$ is the mean of all labels. We also use the Negative Log-Likelihood (NLL) as a loss function. The NLL serves as a common performance measure when the predictions of the model are given in the form of probability distributions. The distance between prediction and true value is given in terms of the uncertainty of the prediction. Assuming that the model predicts a normal distribution with mean μ_i and standard deviation σ_i, the NLL is given by

$$\mathrm{NLL}(y, (\mu, \sigma)) = \frac{1}{N} \sum_{i=1}^{N} \frac{\log(2\pi\sigma_i^2)}{2} + \frac{(y_i - \mu_i)^2}{2\sigma_i^2}. \tag{3.8}$$

3.2. Decision Tree

Decision trees are a conceptually simple method for supervised machine learning. They provide robust models that are largely interpretable. Tree-based models, particularly ensemble approaches, provide decent performance on several learning tasks and remain among the most popular machine learning models [HTF09; Kag].

Using tree-based methods, one can solve both regression and classification problems. In both cases, the models partition the feature space into a set of hyperrectangles, assigning each hyperrectangle a constant value or the associated class. In the following, we will focus on regression.

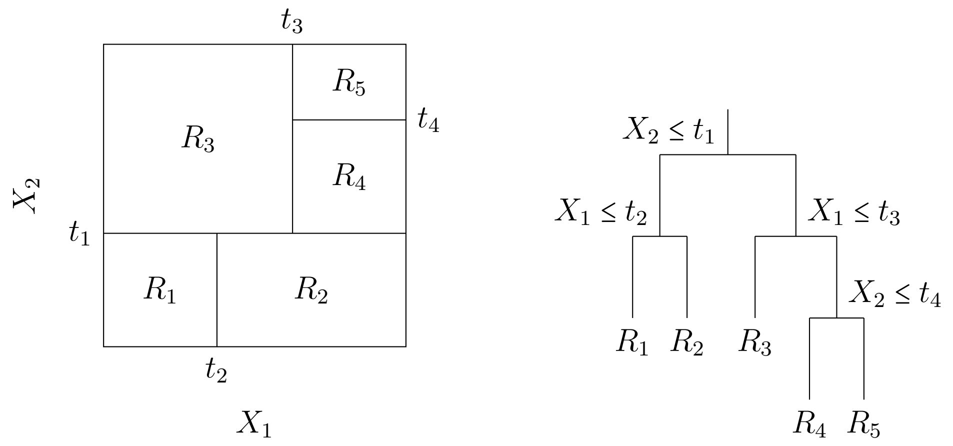

Figure 3.2.: Two different representations of the same decision tree. *(left)* The two-dimensional feature space is separated into rectangles. The splitting points of the feature space are denoted by t_i. The model assigns each input in the rectangle the value R_i. *(right)* Binary decision tree that splits the data at each node. Starting with the full dataset, data fulfilling the observation move down the left branch while the rest moves down the right branch. Terminal nodes are called leaves and give the final prediction.
(Figure inspired by Ref. [HTF09].)

In every step, the feature space is split into two regions and the variable and split point utilized for the division are chosen to achieve the best performance. In each region, the prediction is chosen as the mean of all target values in that region. All available regions are split recursively until some stopping rule applies. In Fig. 3.2*(left)*, a two-dimensional feature space is split into five regions $\{R_i\}_{i=1}^5$. The first split is done at t_1, splitting the feature space into two regions. The resulting regions are split again until there are five regions.

In Fig. 3.2*(right)*, the same process is shown as a binary decision tree. Starting at the top of the tree, the full dataset is split at each node. Data fulfilling the observation move down the left branch, while the rest moves down the right branch. The terminating nodes, called leaves, give the resulting prediction value. This representation of a decision tree is a key advantage because it remains interpretable even for higher-dimensional feature spaces. In comparison, hyperrectangles lose their interpretability for higher dimensional feature spaces.

The prediction function of the decision tree is given by

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{5} c_m I(\mathbf{x} \in R_m), \tag{3.9}$$

where c_m is a constant value assigned for each region and I is the indicator function defined as

$$I(\mathbf{x} \in R_m) = \begin{cases} 1 & \text{if } \mathbf{x} \in R_m, \\ 0 & \text{if } \mathbf{x} \notin R_m. \end{cases} \tag{3.10}$$

Unfortunately, finding the best partitions is computationally infeasible. Therefore, we use one of the most common algorithms, which constructs the tree top-down, starting from the root node. Notably, it is possible to further optimize a tree using algorithms such as pruning. We will skip optimization here as we are only interested in explaining the concept of decision trees.

Consider a dataset D with N samples $(\mathbf{x}_i, y_i)$. Starting from the top node, the data is split into two regions R_1 and R_2 for the feature variable j at the split point s. The best split is found by optimizing j and s with respect to the loss function $L(y, \hat{y})$ given by

$$\underset{j,\,s}{\arg\min}\left[\min_{c_1}\sum_{\mathbf{x}_i \in R_1(j,s)} L(y_i, c_1) + \min_{c_2}\sum_{\mathbf{x}_i \in R_2(j,s)} L(y_i, c_2)\right]. \tag{3.11}$$

Depending on the loss function, the values of c_i for the minimization task in Eq. (3.11) can be easily obtained. For the squared loss, the optimal constant prediction in each region is simply the mean of all target values in the respective set R_m. This process of splitting the data is repeated until a stopping criterion is met. The final prediction function after finding M regions is given by

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m). \tag{3.12}$$

Parameters such as the tree size are crucial to obtain a complexity of the tree suitable to the learning task. Hyperparameters like the maximum tree depth, the maximum number of leaves or the minimal number of data points at one leaf determine the overall size of the tree. They need to be chosen carefully to avoid overfitting or underfitting.

Decision trees offer decent performance for some tasks while they keep inherent interpretability and are computationally cheap. Nonetheless, they cannot compete with recent machine learning algorithms in terms of performance.

3.3. Gradient Tree Boosting

The goal of tree ensemble techniques like random forests and gradient tree boosting is to aggregate numerous decision trees into a single powerful learner. Random forests generate separate trees based on random subsets of the training data and input attributes. Their overall prediction is given by the average of all trees. Gradient Boosted Trees (GBTs), on the other hand, develop the model iteratively, seeking to correct for errors in the prior model with each consecutive tree. Their final prediction consists of the sum of all trees.

Since finding a tree that minimizes the total loss is computationally infeasible, GBTs use the concept of tree-boosting. Tree-boosting trains each new tree based on the negative gradient of the loss function.

The GBT model is initialized with a constant prediction that minimizes the loss function, i.e. the initial model is defined as

$$\hat{f}_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma). \tag{3.13}$$

For each new tree added to the model we compute the pseudo residuals for step m as

$$r_{im} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x})=\hat{f}_{m-1}(\mathbf{x})}. \tag{3.14}$$

Based on each of the targets r_{im}, we fit a regression tree giving the terminal regions $\{R_{jm}\}_{j=1}^{J_m}$, with the number of regions J_m. For each leaf R_{jm} we compute the output value γ_{jm} with

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{jm}} L(y_i, \hat{f}_{m-1}(\mathbf{x}_i) + \gamma). \tag{3.15}$$

Finally, the function $\hat{f}_m$ is given by

$$\hat{f}_m(\mathbf{x}) = \hat{f}_{m-1}(\mathbf{x}) + \eta \sum_{j=1}^{J_m} \gamma_{jm} I(\mathbf{x} \in R_{jm}), \tag{3.16}$$

where $I(\mathbf{x} \in R_{jm})$ is the indicator function (see Eq. (3.10)) and η is the learning rate. The learning rate controls how fast the loss is changing in the direction of the steepest descent of the gradient. The final output of the GBT for M fitted trees in the ensemble is given by

$$\hat{f}(\mathbf{x}) = \hat{f}_M(\mathbf{x}). \tag{3.17}$$

Optimizing the GBT model involves hyperparameter optimization. Some of the available hyperparameters are the number of fitted trees, which can be further controlled by early stopping, the size of the subset of training data used to grow each tree and the learning rate. Additionally, each tree can be optimized using the hyperparameters for a normal decision tree (see

Sec. 3.2).

The increased performance due to gradient tree-boosting comes at the expense of interpretability. While GBTs are able to fit complex structures in data, their complicated ensemble prediction makes simple interpretation impossible. Nevertheless, methods from the field of eXplainable Artificial Intelligence (XAI) can be used to compensate for this limitation (see Sec. 5).

4. Artificial Neural Networks

This chapter aims to give a brief introduction to the basic concepts of artificial neural networks. Notation follows the work of Aggarwall [Agg+18], Goodfellow et al. [GBC16] and Nielsen [Nie15], where a more detailed introduction can be found. Additionally, we follow the explanations and visualizations of Olah [Ola15] and the notations of Staudemeyer et al. [SM19] for recurrent neural networks. We will start with a short overview of neural networks in Sec. 4.1. In Sec. 4.2, we explain the mathematical background of neural networks by means of the perceptron and continue with more complex structures, in particular feedforward neural networks in Sec. 4.3. Finally, we will introduce the basic concepts of recurrent neural networks in Sec. 4.4 and focus on the special architecture of long-short term memory cells in Sec. 4.5.

4.1. Overview

Artificial neural networks or just neural networks are networks consisting of multiple neuron-like threshold switching units, short neurons. Each neuron takes some input $\mathbf{x} \in \mathbb{R}^n$ and produces a single output $\hat{y} \in \mathbb{R}$.

Different approaches for training neural networks, like supervised learning, unsupervised learning or reinforcement learning, result in distinct subclasses of neural networks. In this thesis, we will focus on supervised learning, where the neural network approximates some function $f(\mathbf{x}) = y$. The most common types can be classified into FeedForward Neural Networks (FFNN), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

4.2. Perceptron

The first prototype of a neural network, or more precise the first neuron, called a perceptron, was introduced by Rosenblatt in 1958 [Ros58], inspired by the work of McCulloch et al. in 1943 [MP43]. A single perceptron is the simplest neural network architecture. We use the perceptron to demonstrate the underlying concepts of complex neural networks.

A perceptron consists of multiple input links, a net input function, an activation function and a single output link (see Fig. 4.1). The input links connect the input value $\mathbf{x} \in \mathbb{R}^m$ to the net input function. The net input function of the perceptron is the weighted sum of the perceptron with weights $\mathbf{w} \in \mathbb{R}^m$ and a bias $b \in \mathbb{R}$. The resulting output

$$\mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{m} w_i x_i + b \tag{4.1}$$

of the net input function is passed into the activation function which then connects to the output link. The activation function of the perceptron is

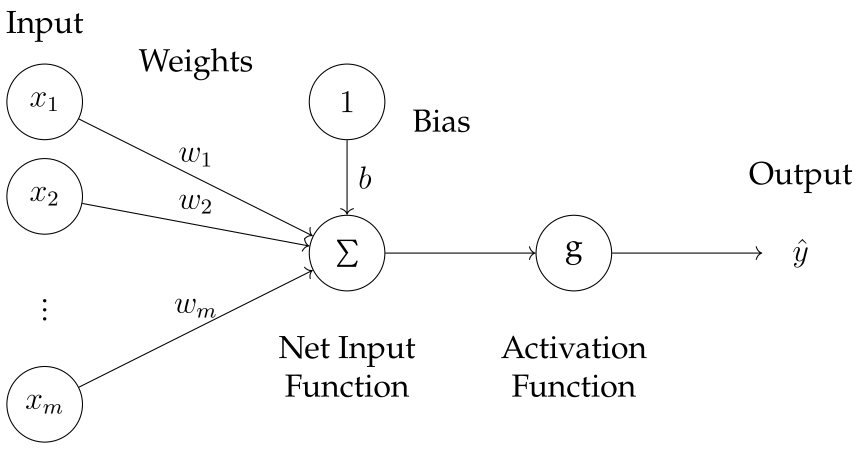

Figure 4.1.: Simple structure of a perceptron. Inputs are connected to the net input function via weights and bias. The net input function is passed through the activation function to produce the output.

defined as the sign function

$$\operatorname{sgn}(z) = \begin{cases} 1 & \text{if } z > 0, \\ -1 & \text{else }, \end{cases} \tag{4.2}$$

which maps all positive values to 1 and everything else to -1. For a given input x the final output, often called prediction, of the perceptron is therefore given by

$$\hat{y} = \operatorname{sgn}(\mathbf{w}^T\mathbf{x} + b). \tag{4.3}$$

The objective of the perceptron is to learn an unknown function $f(\mathbf{x}) = y$ using supervised learning, where $(\mathbf{x}, y) \in D$ are samples of a given dataset D. During the learning process, the weights $\mathbf{w}, b$ are adjusted such that the prediction $\hat{y}$ approximates the real output y,

$$\operatorname{sgn}(\mathbf{w}^T\mathbf{x} + b) = \hat{y} \approx y = f(\mathbf{x}). \tag{4.4}$$

For the learning process, we use the linear activation function $g(z) = z$ instead of the sign function for simpler updating rules. The bias b is also redefined as $w_0 := b, x_0 := 1$, so the output of the perceptron is given by

$$\hat{y} = \mathbf{w}^T\mathbf{x}, \tag{4.5}$$

with $\mathbf{w}, \mathbf{x} \in \mathbb{R}^{m+1}$. More precise, the correct designation of the modified perceptron is linear unit.

To quantify the performance of the perceptron we use a loss function, also known as an error or cost function. The loss function maps the difference between the predicted values $\hat{y}$ and the real values y to a real number. For example, the squared error is defined as

$$L[\mathbf{w}] = \frac{1}{2} \sum_{(\mathbf{x},y)\in D} (y - \hat{y})^2 \tag{4.6}$$

$$= \frac{1}{2} \sum_{(\mathbf{x},y)\in D} (y - \mathbf{w}^T\mathbf{x})^2, \tag{4.7}$$

where the prefactor $\frac{1}{2}$ is added to simplify the first derivative of the error function.

During the supervised learning process, we use gradient descent to minimize the loss function in the space of the percepton's weights w. The gradient descent algorithm updates the weights of the perceptron in the direction of fastest error reduction until a predefined stopping condition is met. In each iteration, the weights are updated by

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}, \tag{4.8}$$

with

$$\Delta\mathbf{w} = -\eta\nabla L[\mathbf{w}], \tag{4.9}$$

$$\nabla L[\mathbf{w}] = \left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_m}\right]. \tag{4.10}$$

The parameter η regulates the degree to which the weights are altered and is known as the learning rate.

Using the squared error from Eq. (4.6), the resulting update function for the weights is defined by

$$w_i \leftarrow w_i + \eta \sum_{(\mathbf{x},y)\in D} (y - \mathbf{w}^T\mathbf{x})x_i. \tag{4.11}$$

More advanced and complex neural networks use different activation functions such as the sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{4.12}$$

the hyberbolic tangens

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4.13}$$

or the Rectified Linear Unit (ReLU)

$$g(z) = \max(0, z). \tag{4.14}$$

4.3. Feedforward Neural Networks

A FeedForward Neural Network (FFNN) is a type of artificial neural network in which information flows through the network in only one direction, from the input layer to the output layer. Connections back to previous layers are not possible unlike in Recurrent Neural Networks (RRNs) which have feedback connections (see Sec. 4.4). FFNNs are the most basic and widely used type of neural networks and serve as a building block for more complex architectures. In an FFNN, neurons are arranged in layers, with each layer having a specific number of neurons. A neuron is the basic processing unit in a neural network and is mainly based on the concepts of a simple perceptron (see Sec. 4.2). The neurons in each layer are connected to the neurons in the next layer via a set of weights. During training, the weights are optimized to reduce the discrepancy between the network's predictions and the observed output. FFNNs are not only loop-free but also fully connected, meaning that all the neurons in one layer are connected to all the neurons in the next layer (see Fig. 4.2). This structure forms a directed acyclic graph, where the edges represent the connections between neurons, and the direction of the edges indicates the flow of information through the network. The output of the FFNN is given by

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{x}) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(\mathbf{x}), \tag{4.15}$$

where each function $f^{(i)}$ represents the entire mapping between two layers. Each layer represents the output of the function with $\hat{\mathbf{y}}$ being the overall output of the network. In the literature, the input $\mathbf{x}$ is often also embedded into an extra layer called the input layer which connects all inputs to the first layer in the network. The last layer $f^{(L)}$ is called the output layer, while all layers between the input and output layer are called hidden layers, since their output is only used inside the neural network and is therefore not directly visible (see Fig. 4.2). The depth of an FFNN is defined by the number of layers L, while the width of layer l is defined by the number of neurons m_l in that layer.

The most commonly used optimization algorithm for training FeedFor-

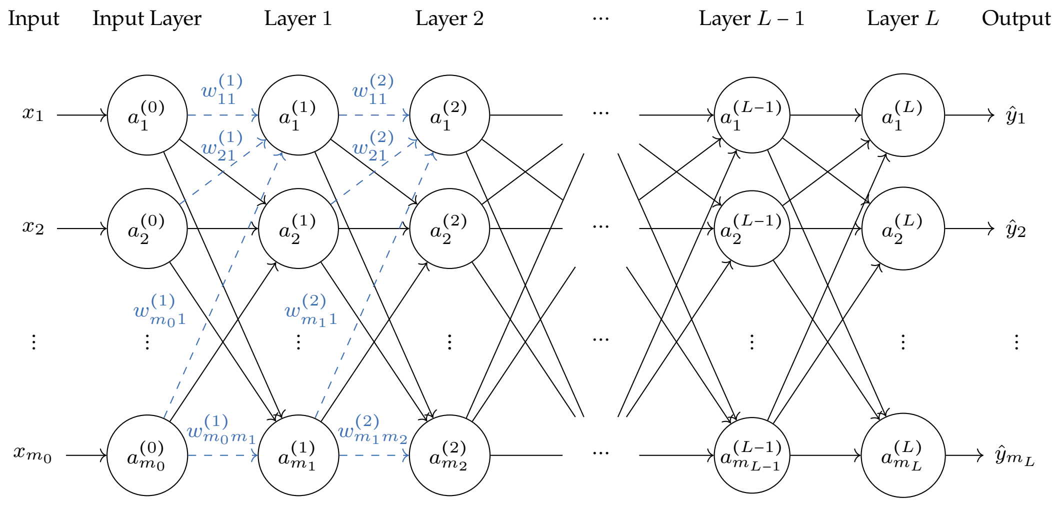

Figure 4.2.: Layout and notation of a generalized FeedForward Neural Network (FFNN) with L layers and width m_l for each layer l. Outputs of the neurons are denoted by $a_j^{(l)}$ with $j = \{1, \dots, m_l\}$ for each layer. Weights connecting one layer to the next are denoted by $w_{kj}^{(l)}$ connecting neuron k of layer $l-1$ to neuron j of layer l with $k = \{1, \dots, m_{l-1}\}$ and $j = \{1, \dots, m_l\}$. The bias $b_j^{(l)}$ is deprecated due to simplicity. Starting from the left, the input is embedded into an input layer mapping $\mathbf{x}$ onto $\mathbf{a}^{(0)}$. The input is then passed through each layer until it reaches the output layer L which then outputs the overall output of the network $\hat{\mathbf{y}}$.

ward Neural Networks (FFNN) is gradient descent, very similar to the training process of the perceptron. It uses forward propagation to compute the output of the network given an input, and backward propagation to compute the gradients of the loss function with respect to the model's parameters.

During the training process, the FFNN's parameters are updated iteratively in the direction of the negative gradients of the loss function, to minimize the error on the training data. This process is repeated until a stopping criterion is met. The magnitude of the step taken in the direction of the negative gradient is scaled by the learning rate η that controls the degree to which the weights are changed.

Using the notation of the general FFNN in Fig. 4.2, we demonstrate the forward propagation process in the following. Note that the bias b for each layer is deprecated for the sake of simplicity. Furthermore, the input $\mathbf{x}$ is embedded into an input layer with output $\mathbf{a}^{(0)}$ to simplify notation later on. Therefore, the output of any layer l in the network can be written as

$$a_j^{(l)} = g\left(\sum_k w_{kj}^{(l)} a_k^{(l-1)} + b_j^{(l)}\right),\ j = \{1, \ldots, m_l\}, \tag{4.16}$$

where g is the activation function. The input of the activation function is called the net input which is defined as

$$z_j^{(l)} := \sum_k w_{kj}^{(l)} a_k^{(l-1)} + b_j^{(l)}. \tag{4.17}$$

Following Eq.(4.16), the overall output of the network is given by

$$\hat{y}_j = a_j^{(L)} = g\left(\sum_k w_{kj}^{(L)} a_k^{(L-1)} + b_j^{(L)}\right), \tag{4.18}$$

which can be obtained by passing the input $\mathbf{x}$ to the network through each layer, starting by computing the output of the first layer.

In vector form, the output of any layer can be rewritten as

$$\mathbf{a}^{(l)} = g\left(\mathbf{W}^{(l)^T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{4.19}$$

where $\mathbf{W}^{(l)^T}$ is the transposed matrix of all weights mapping from layer $l-1$ to layer l and the function g is taken element-wise. The overall loss of the network is then computed on the output $\hat{\mathbf{y}}$ with the loss function $L(\hat{\mathbf{y}}, \mathbf{y})$.

After the forward propagation, the weights are updated using the gradient of the loss function with respect to the weights. The gradient corresponds to the change of loss when adapting a specific weight in the network with the update rules

$$w_{jk}^{(l)} \leftarrow w_{jk}^{(l)} - \eta \frac{\partial L}{\partial w_{jk}^{(l)}}, \tag{4.20}$$

$$b_j^{(l)} \leftarrow b_j^{(l)} - \eta \frac{\partial L}{\partial b_j^{(l)}}, \tag{4.21}$$

where η is the learning rate mentioned earlier that controls the degree to which the weights are changed. Formulated as vector operations for efficient computation this gives

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \nabla_{\mathbf{W}^{(l)}} L, \tag{4.22}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \nabla_{\mathbf{b}^{(l)}} L, \tag{4.23}$$

with $\mathbf{W}^{(l)}$ the matrix of all weights mapping from layer $l-1$ to layer l and $\mathbf{b}^{(l)}$ the bias of layer l.

The computation of the gradients in Eq. (4.20) and Eq. (4.21) is based on the backwards propagation algorithm or short, backpropagation. We use the backpropagation algorithm to efficiently compute the partial derivatives of the loss function. The main idea of the backpropagation algorithm is to define a local error to each neuron which is induced to the final output. This error can then be used to efficiently compute all partial derivatives in the network and therefore the gradient of the loss function.

Let $\delta_j^{(l)}$ be the local error of the network in the j^{th} neuron of the l^{th} layer defined by

$$\delta_j^{(l)} := \frac{\partial L}{\partial z_j^{(l)}}. \tag{4.24}$$

Focusing on the local error of the output layer $\delta_j^{(L)}$, using the chain rule of calculus, Eq. (4.24) simplifies to

$$\begin{aligned}
\delta_j^{(L)} &= \frac{\partial L}{\partial z_j^{(L)}} && (4.25)\\
&= \sum_k \frac{\partial L}{\partial a_k^{(L)}} \frac{\partial a_k^{(L)}}{\partial z_j^{(L)}} && (4.26)\\
&= \frac{\partial L}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} && (4.27)\\
&= \frac{\partial L}{\partial a_j^{(L)}} g'(z_j^{(L)}). && (4.28)
\end{aligned}$$

In vectorized form this gives

$$\boldsymbol{\delta}^{(L)} = \nabla_{\mathbf{a}^{(L)}} L \odot g'(\mathbf{z}^{(L)}), \tag{4.29}$$

for the overall error of the output layer. The operator $\odot$ indicates a point-wise multiplication between two vectors, called the Hadamard product.

Using the definition of $\delta_j^{(l)}$ from Eq. (4.34) to (4.35) and the definition of the net input from Eq. (4.35) to (4.36) with

$$z_k^{(l+1)} = \sum_j w_{jk}^{(l+1)} a_j^{(l)} + b_k^{(l+1)} \tag{4.30}$$

$$= \sum_j w_{jk}^{(l+1)} g(z_j^{(l)}) + b_k^{(l+1)} \tag{4.31}$$

$$\Rightarrow \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} = w_{jk}^{(l+1)} g'(z_j^{(l)}), \tag{4.32}$$

the local error for any layer is then given by

$$\delta_j^{(l)} = \frac{\partial L}{\partial z_j^{(l)}} \tag{4.33}$$

$$= \sum_k \frac{\partial L}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} \tag{4.34}$$

$$= \sum_k \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} \delta_k^{(l+1)} \tag{4.35}$$

$$= \sum_k w_{jk}^{(l+1)} \delta_k^{(l+1)} g'(z_k^{(l)}). \tag{4.36}$$

Finally, in a vectorized form this gives

$$\boldsymbol{\delta}^{(l)} = \mathbf{W}^{(l+1)} \boldsymbol{\delta}^{(l+1)} \odot g'(\mathbf{z}^{(\mathbf{l})}), \tag{4.37}$$

which can be used to compute the local error of each layer by starting from the output layer using Eq. (4.29) and recursively applying Eq. (4.37) to propagate the error back to the input layer. Equation (4.37) can be interpreted as first applying the weights backwards on the error of the previous layer, moving the error backwards through the network. Then moving the error backwards through the activation function.

After the local errors are computed for the whole network, they can be used to obtain the gradients needed to update the weights. For the bias, it follows that

$$\frac{\partial L}{\partial b_j^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} \tag{4.38}$$

$$= \frac{\partial L}{\partial z_j^{(l)}} \underbrace{\frac{\partial \left(\sum_k w_{kj}^{(l)} a_k^{(l-1)} + b_j^{(l)} \right)}{\partial b_j^{(l)}}}_{=1} \tag{4.39}$$

$$= \frac{\partial L}{\partial z_j^{(l)}} = \delta_j^{(l)}, \tag{4.40}$$

almost identically for the weights we get

$$\frac{\partial L}{\partial w_{kj}^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{kj}^{(l)}} \tag{4.41}$$

$$= \frac{\partial L}{\partial z_j^{(l)}} \underbrace{\frac{\partial \left(\sum_k w_{kj}^{(l)} a_k^{(l-1)} + b_j^{(l)} \right)}{\partial w_{kj}^{(l)}}}_{=a_j^{(l)}} \tag{4.42}$$

$$= \frac{\partial L}{\partial z_j^{(l)}} a_j^{(l)} \tag{4.43}$$

$$= \delta_j^{(l)} a_j^{(l)}. \tag{4.44}$$

Once again, expressed in vectorized form, this simplifies to

$$\nabla_{\mathbf{b}^{(l)}} L = \boldsymbol{\delta}^{(l)}, \tag{4.45}$$

$$\nabla_{\mathbf{W}^{(l)}} L = \boldsymbol{\delta}^{(l)} \mathbf{a}^{(l)T}. \tag{4.46}$$

Equation (4.45) and (4.46) show the efficiency of the backpropagation algorithm. The gradients are now only depending on two variables. The values for $\mathbf{a}^{(l)}$ are computed in the forward propagation in order to obtain the overall output of the network, while the local errors $\boldsymbol{\delta}^{(l)}$ can be computed recursively from the output to the input layer. In comparison, computing the gradients without backpropagation would need exponentially many computations of partial derivatives when starting from the input layer.

This process of updating the weights is applied until a stopping criterion is met. In practice, the weights are updated iteratively using only a specific subset of the entire data, commonly referred to as batch. The batch size is the predefined number of samples for all subsets. One possible stopping criterion is increasing the loss function on some unseen dataset called the validation set.

4.4. Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of artificial neural network in which connections between neurons can form loops, as opposed to an FFNN which is fully connected and loop-free. These loops or circular connections create dynamic systems with internal states at each step in time. The circular connections allow information to flow backward in the network but forward in time, propagating data from earlier events to current inputs (see Fig. 4.3). This results in a temporal dynamic behaviour of the network where the network is able to build a memory of time series events. RNNs can be fully or partly connected, while fully connected RNNs connect all time steps while also allowing self-feedback.

Connecting neurons via loops over time allows for much longer sequences to be processed than it would be practical with FFNN. Therefore, RNNs are perfectly suited for handling sequential data in speech- and handwriting recognition, language translation and also time-series analysis.

Figure 4.3 shows the overall information flow in an RNN, where the same

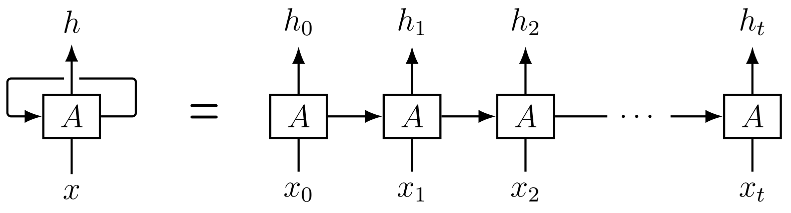

Figure 4.3.: General information flow in a Recurrent Neural Network (RNN). The RNN has a circular connection to itself or loop. Unfolding the RNN in time shows how each time step of the input is inserted.

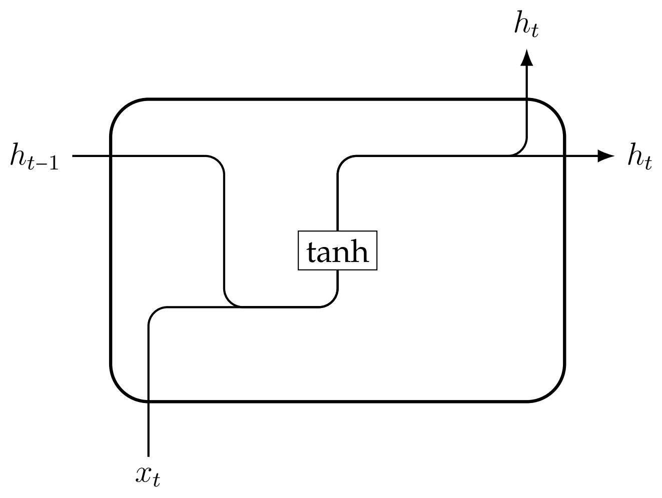

Figure 4.4.: Internal structur of an RNN. The input x_t of time step t gets concatenated with the hidden state h_{t-1} of the previous time step and passed into a tanh activation layer. The output is the hidden state h_t. The tanh layer contains all weights.

cell A is recursively used on each time step of the input x. The internal structure of the RNN cell is simple and consists of only one unit with the tanh activation function. In Fig. 4.4, the basic structure is shown, where h_{t-1} denotes the output of the same RNN cell in the previous time step, x_t the input in time step t and h_t the output of the cell. The output of the previous time step and the current input get concatenated before being passed into the tanh layer. Therefore, the information of previous input time steps is inherited in the state h, which is called the internal state of the RNN cell. In theory, it should enable the use of information from many arbitrary steps backward in time.

Because information must be propagated through recurrent connections between steps, optimizing the weights of an RNN requires a distinct strategy from that of an FFNN. Therefore, the standard backpropagation method cannot compute the necessary gradients to update the network's weights.

Nevertheless, there exist different algorithms to successfully train RNNs for temporal supervised tasks. The most common algorithm is BackPropagation Through Time (BPTT), which unfolds the network in time to construct an FFNN. Using this technique, it is possible to use normal backpropagation (see Sec. 4.3) to update the weights of an RNN. The detailed explanation and derivation of the BPTT algorithm is out of the scope of this thesis, and also does not add any additional understanding of RNNs. Understanding the backpropagation algorithm and the concept of unfolding an RNN into an FFNN is sufficient to comprehend the application of RNNs. A detailed explanation can be found in Goodfellow et al. [GBC16] or Staudemeyer et al. [SM19].

In practice, RNNs are unable to learn any long-term dependencies in sequential data due to the vanishing gradient problem [Hoc91; BSF94]. The vanishing gradient problem occurs when connections in the network from input to output are long, resulting in many small partial derivatives that are multiplied together using the chain rule of calculus. As RNNs get unfolded in time, they transform into deep FFNNs with long connections between input and output, exacerbating the vanishing gradients in the backpropagation algorithm.

Today, there are many approaches for different structures of RNNS that solve the vanishing gradient problem, two of the most famous ones being the Long Short-Term Memory (LSTM) cells and the Gated Recurrent Units (GRU). Both approaches work with some kind of internal long term memory, allowing information from deeper layers to flow into current layers. In this thesis, we will use LSTMs (see Sec. 4.5), because of their higher accuracy on long sequences and therefore superior performance on time series data [LB21].

4.5. Long Short-Term Memory (LSTM)

Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) or just LSTMs are a special kind of RNN introduced by Hochreiter et al. [HS96; HS97] as a solution to the vanishing gradient problem. They are able to handle long sequences of up to 1000 discrete time steps, making them widely used for language processing or time series analysis. Accordingly, LSTMs should be suitable for the field of electricity price forecasting.

Just like basic RNNs, LSTMs use connections to other time steps to pass

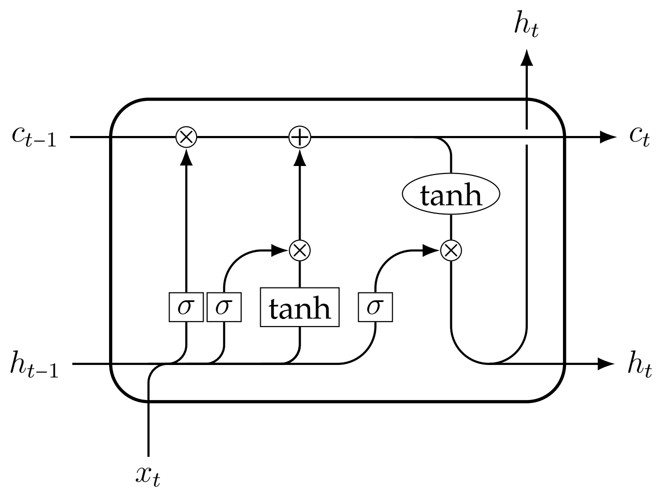

Figure 4.5.: Internal structure of an LSTM cell. The cell consists of several activation layers including sigmoid and tanh activation. The LSTM uses the cell state c to store information about previous time steps. The concatenation of the current input x_t and the previous hidden state $h_{t\text{-}1}$ is used to update the previous cell state $c_{t\text{-}1}$ to the new cell state c_t. The final output h_t of the cell is a modified version of the cell state. The activation layers contain all weights.
(Figure inspired by Ref. [Ola15].)

information through layers in the network. Instead of using only one simple neural network layer like the RNN, LSTMs use four layers interacting in a special way. The key idea of LSTMs is the introduction of a cell state which can be used as a sort of memory state, allowing information to flow for long time scales.

Figure 4.5 shows the detailed structure of an LSTM cell. Compared to the structure of RNNs (see Fig. 4.4), LSTMs use many internal connections with different activation functions. The main feature is the cell state c which passes the LSTM cell from left to right. The internal structure of the cell allows to update the cell state according to past outputs of the cell and its current inputs using so called gates.

The first gate is the forget gate on the left of the cell, which regulates how much of the information in the cell state is kept. Previous output $h_{t\text{-}1}$ and current input x_t are concatenated and passed through a sigmoid activation function, which maps to $[0,1]$. The resulting output is

$$f_t = \sigma\left(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right), \tag{4.47}$$

which is then multiplied with the cell state. Outputs f_t close to 0 will thus remove stored values while outputs f_t close to 1 will keep the values in the cell state. The second gate is the input gate which decides how much new information is added to the cell state. The input gate consists of two parts, the first part is a sigmoid activation of the previous hidden state and current input like in the forget gate. The sigmoid unit decides which values in the cell state should be updated. The second part consists of a tanh activation layer which creates possible new values for the cell state. Both parts are connected via multiplication, such that the sigmoid unit removes values from the possible update that should not be updated. The resulting output

$$i_t = \sigma\left(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right), \tag{4.48}$$

$$\tilde{\mathbf{c}}_t = \sigma\left(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right) \tag{4.49}$$

$$\Rightarrow \hat{\mathbf{c}}_t = \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{4.50}$$

then gets added onto the cell state after the forget gate was multiplied. This results in the final cell state

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \hat{\mathbf{c}}_t. \tag{4.51}$$

Finally, the output of the cell h_t is given by a filtered version of the cell state c_t. The cell state gets passed into a tanh layer acting point-wise on the cell state and is modified by a sigmoid layer. Again, the sigmoid layer acts on the concatenation of the previous hidden state h_{t-1} and the current input x_t. Finally, the output of the tanh layer and the sigmoid layer get point-wise multiplied. So to say, the sigmoid layer controls how much information of the output of the tanh layer is used. The final output of the cell is

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right) \tag{4.52}$$

$$\Rightarrow \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{4.53}$$

For the optimization of LSTMs, it is possible to use the same methods used for classic RNNs, i.e. weights are updated using the BPTT algorithm. Once again, this is out of the scope of this thesis and does not add any additional understanding of the behaviour of LSTMs. A detailed explanation of the optimization process of LSTMs can be found in Goodfellow et al. [GBC16] or Staudemeyer et al. [SM19].

5. Fundamentals of Explainable AI

5.1. XAI for Energy

Many machine learning approaches are based on black-box models, which limits scientific insights [Ros+20; AB18] and may induce security risks in critical sectors [Ahm+21; CKS19]. A promising alternative is provided by eXplainable Artificial Intelligence (XAI) methods, including inherently transparent models and post-hoc model explanations [Bar+20]. The field of XAI has gained a strong interest in energy systems analysis in recent years [Mac+22], in particular for applications to power system operation and stability. For instance, XAI was used in transient stability assessment [Che+19], the identification of risks for frequency stability [KSW21] or load and renewable generation forecasts [ML22]. Furthermore, XAI has been used to analyze factors that determine the success of large power system infrastructure projects [ATM21]. Applications of XAI methods for electricity markets are still in their infancy. A recent study by Tschora et al. [Tsc+22] mainly focused on the identification of relevant features in forecasting models.

5.2. SHapley Additive exPlanation (SHAP)

This section aims to give a short introduction to the concept of SHapley Additive exPlanation (SHAP). SHAP provides insights into black-box machine learning models. We explain the basic concept and mathematical foundation of SHAP values in Sec. 5.2.1 and SHAP interactions values in Sec. 5.2.2 but do not cover their implementation. We will also introduce several explanation tools used in this thesis based on the SHAP framework.

5.2.1. SHAP Values

SHapley Additive exPlanation (SHAP) values are a technique for providing post-hoc explanations of machine learning models with black-box character. The goal of SHAP values is to explain the prediction of input $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ by computing the contribution of each feature x_i. SHAP values are based on Shapley values, a concept from cooperative game theory [Sha+53]. The idea of SHAP values is to assign each feature a value that represents its contribution to the difference between the actual prediction and the prediction that would have been made in the absence of that feature.

We define $\mathcal{F}$ to be the set of all features with $|\mathcal{F}| = n$. The conditional expectation of the model's output given a subset of features $S \subseteq \mathcal{F}$ is given by

$$f_x(S) = E[f(X)|\text{do}(X_S = x_s)], \tag{5.1}$$

where S is the set of features conditioned on, X is a random variable representing the input features, x is the input sample and $\text{do}(X_S = x_s)$ is the causal do-notation [Pea00].

The SHAP value of a particular feature is calculated by averaging its marginal contribution

$$f_x(S \cup \{i\})) - f_x(S), \tag{5.2}$$

i.e. the difference after introducing this feature to a subset, over all possible subsets of features.

SHAP values are defined by satisfying the properties of local accuracy, consistency, and missingness simultaneously. For the statement of the three properties, we will primarily follow the work of Lundberg et al. [Lun+20].

Property 1 *(Local Accuracy)***.** The explanation of the model for a given input $\mathbf{x} \in \mathbb{R}^n$ should sum up to the original output $f(\mathbf{x})$ with

$$f(x) = \phi_0(f) + \sum_{i=1}^{n} \phi_i(f, x). \tag{5.3}$$

The values of $\phi_i(f, x)$ are called the SHAP values and $\phi_0(f) = E[f(x)]$.

Property 2 *(Consistency)***.** If a model changes such that the contribution of a feature increases or stays the same regardless of the other inputs, the attribution of the input should not decrease. To be precise, for any two models f and f', if

$$f'_x(S) - f'_x(S \setminus \{i\}) \leq f_x(s) - f_x(S \setminus \{i\}), \tag{5.4}$$

for all subsets of features $S \subseteq \mathcal{F}$, then

$$\phi_i(f', x) \leq \phi_i(f, x). \tag{5.5}$$

Property 3 *(Missingness)***.** All features that have no effect on the set function f_x should not have an assigned impact. That is, if

$$f_x(S \cup \{i\}) = f_x(S), \tag{5.6}$$

for all subsets of features $S \subseteq \mathcal{F}$, then

$$\phi_i(f, x) = 0. \tag{5.7}$$

Shapley has shown that the unique measure satisfying all three properties is given by

$$\phi_i(f, x) = \sum_{S \subseteq \mathcal{F} \setminus \{i\}} \frac{|S|!(|\mathcal{F}| - |S| - 1)!}{|\mathcal{F}|!} (f_x(S \cup \{i\})) - f_x(S)). \tag{5.8}$$

In general, the exact computation of SHAP values is NP-hard. The problem resides in the computation of the contribution function f_x and the exponential growth of the sum in Eq. (5.8) due to the number of feature subsets. Nevertheless, there exist several implementations for computing SHAP values efficiently in certain special cases. The TreeSHAP approach is a method for computing the SHAP values of tree-based models like random forests or GBTs (see Sec. 3.3) and was introduced in [Lun+20]. Using path dependence, the algorithm is able to compute SHAP values in low polynomial order runtime. The implementation of this algorithm is used throughout this thesis.

In the following, we will present some explanation tools based on SHAP values that will be used throughout the thesis. Since this thesis focuses solely

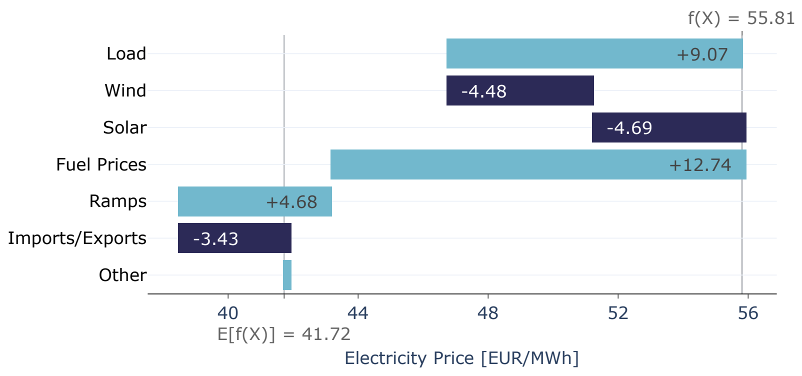

Figure 5.1.: Local explanation of the prediction of a machine learning model using SHAP values. The model prediction for an input is initialized with the mean of the overall function. Each input feature contributes positively or negatively, which adds up to the final prediction of the model.

on data from the German day-ahead electricity market, we use a simplified version of the data to demonstrate the explanation tools. A detailed introduction to the German electricity market can be found in Sec. 2, whereas a detailed description of the data can be found in Sec. 6.1.1.

Using Eq. (5.3) and (5.8), we can generate local explanations of the prediction of a model. Starting from the base value $\phi_0(f)$, each feature contributes positively or negatively to the final output of the model. A detailed example can be seen in Fig. 5.1, where the model is initialized at $\phi_0(f) = 41.72$ EUR/MWh and each feature contributes to a final prediction of 55.81 EUR/MWh.

Because the SHAP values are generated for each input sample individually, SHAP values are fundamentally a local explanation technique. But nonetheless, SHAP values are not confined to explaining individual samples. They may also be used to understand the global behaviour of the model by combining numerous local explanations.

One of the global measures based on SHAP values is the feature importance. For each feature j, the feature importance is defined as the normalized sum over all absolute values of the SHAP values. Using the overall sum of

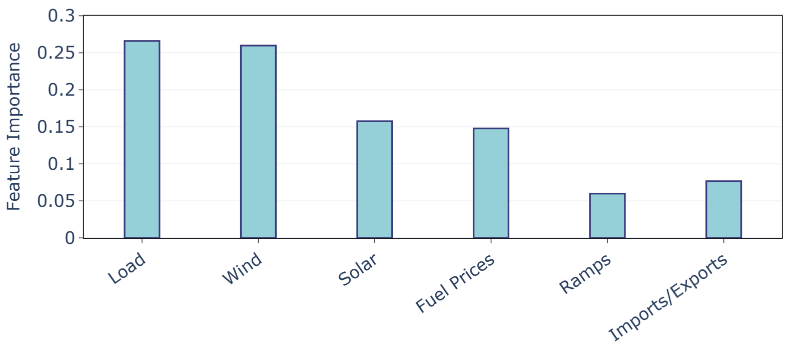

Figure 5.2.: Feature importance of a model using SHAP values. The figure illustrates the contribution of each feature to the output of a machine learning model, as calculated by SHAP values. Local SHAP values are aggregated and normalized by the sum of all SHAP values to obtain the relative importance of each feature. Values are expressed as a percentage ranging from 0 to 1, with a value of 1 indicating that the model's output is entirely dependent on that feature.

all SHAP values as normalization, we get

$$I_j = \frac{\sum_{m=1}^{M} \phi_j^{(m)}}{\sum_{j=1}^{|\mathcal{F}|} \sum_{m=1}^{M} \phi_j^{(m)}}, \tag{5.9}$$

where $\phi_j^{(m)}$ is the SHAP value of input m for the feature j and M the number of all inputs. Figure 5.2 shows an example of a feature importance plot. A feature importance of 0.25 of feature j corresponds to the feature having an impact of overall 25% on the prediction.

Another way of extracting global dependencies from the local explanations is to use SHAP dependency plots. In a SHAP dependency plot, the SHAP values of a feature k are plotted against its feature values. This provides more information than a traditional partial dependency plot could convey, as it shows interactions as scattering. The SHAP dependency plot is able to reveal global patterns of the feature in relation to the predictions of the explained model. Figure 5.3a shows a dependency plot of the generation ramp. We can see that the electricity price has an almost linear dependence

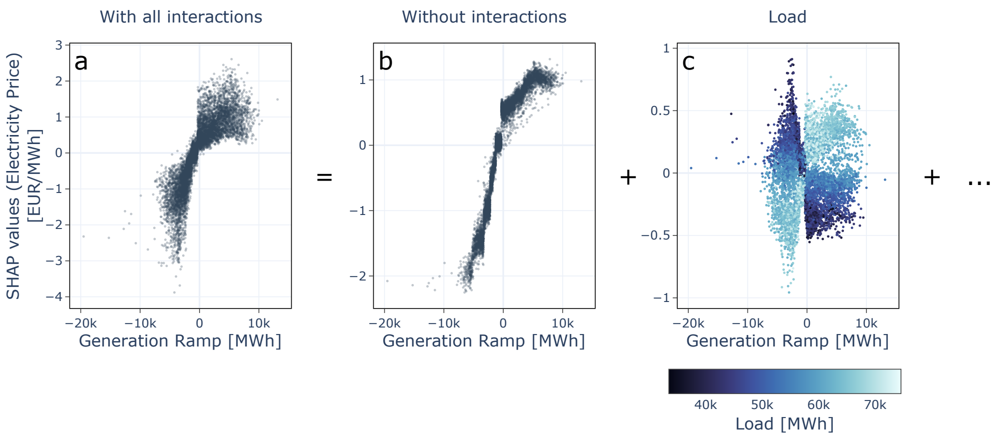

Figure 5.3.: Global explanations of a model using SHAP values and SHAP interaction values. (a) SHAP dependency plot of one feature for a model. The local SHAP values are plotted against the values of the feature. The model has an approximately linear dependency on the feature. Strong scattering at the extremes corresponds to feature interactions. (b) SHAP dependency plot without feature interactions from SHAP interaction values. SHAP values without feature interactions are plotted against the values of the feature. The linear dependency on the feature is even more obvious than in the normal SHAP dependency plot. (c) SHAP interaction plot of one feature with another feature. The SHAP values are plotted against the value of the feature but with respect to the interacting feature. The colour shows the value of the interacting feature. Interactions with the other feature lead to stronger (weaker) dependency on the feature for higher (lower) values of the interacting feature.

on the generation ramp, except for some strong scattering at higher values. The scattering results from the interactions between the features, which will be discussed in more detail in the following section.

5.2.2. SHAP Interaction Values

SHAP interaction values enable a richer type of local explanation than normal SHAP values. They are based on the Shapley interaction index from game theory and are able to capture local interaction effects. Instead of allocating credit to only one feature, SHAP interaction values are able to assign credit to pairs of features. Therefore, SHAP interaction values are given by a matrix of feature attributions, unlike normal SHAP values given by a vector. The diagonal of the matrix corresponds to the effect of the features without any feature interaction. The off-diagonal shows the interaction between every combination of two features. They show similar properties to SHAP values but allow a separate consideration of interaction effects.

For two features $i \neq j$, SHAP interaction values are defined as

$$\phi_{ij} = \sum_{S \subseteq \mathcal{F} \setminus \{i,j\}} \frac{|S|!(|\mathcal{F}| - |S| - 2)!}{2(|\mathcal{F}| - 1)!} \delta_{ij}(f, x, S) \tag{5.10}$$

with

$$\delta_{ij}(f, x, S) = f_x(S \cup \{i, j\}) - f_x(S \cup \{i\}) - f_x(S \cup \{j\}) - f_x(S). \tag{5.11}$$

The overall output of the model is given by

$$f(x) = \sum_{i=0}^{|\mathcal{F}|} \sum_{j=0}^{|\mathcal{F}|} \phi_{ij}(f, x), \tag{5.12}$$

with $\phi_{00} = E[f(x)]$.

The SHAP interaction values can be interpreted as the difference between the SHAP values of feature i when feature j is either present or absent. Therefore, it is possible to implement the computation of SHAP interaction values with the same algorithms used for normal SHAP values for tree-based models, resulting in low polynomial order runtime. The detailed implementation can be found in [Lun+20].

SHAP interaction values can be used to generate SHAP interaction plots, which provide further insight into the interactions between different features. The process of decomposing the dependency plot into feature interactions can be seen in Fig. 5.3, which shows the impact of generation ramps on the electricity price. Figure 5.3b shows the diagonal SHAP interaction value for generation ramps, where it is visible that the scattering present in the dependency plot was reduced. Figure 5.3c shows the most important interaction with another feature, which is the interaction of the generation ramp with the load. The colour indicates the value of the interacting value, in this case, the load. We can see that the effect of the generation ramp on the electricity price is increased at high load and reduced at low load. Adding up the plot without any interaction and all feature interactions results in the normal SHAP dependency plot in Fig. 5.3a.

Combining all explanation tools based on the SHAP framework, we gain significant insights into the contributions of the input features. The feature importance gives an overall impact of the feature for the predictions of the model. Using the SHAP dependency plot, important features can be further analyzed to gain global insights about the dependency on the feature. Additionally, the resulting scattering can be further analyzed using SHAP interaction plots.

6. Understanding Day-ahead Electricity Prices

In this chapter, we aim to explain the dependencies and correlations within the energy markets using a data based approach. We establish a machine learning model for the electricity prices on the German day-ahead spot market. For the model, we use gradient tree-boosting methods (see Sec. 3.3) that outperform our benchmark model. We apply SHapley Additive exPlanations (SHAP) (see Sec. 5.2) to explain the model and provide insights into which factors determine the market price. We use an extended data set to identify driving factors which are commonly neglected in elementary studies or machine learning models [Lag+21].

The chapter is structured as follows. In Sec. 6.1.1, we discuss how we obtained and processed data and discuss how the machine learning model is trained and interpreted in Sec. 6.1.2. We then continue in Sec. 6.2 to analyze the results of the machine learning model, in particular demonstrating how load, wind and solar generation but also fuel prices critically influences electricity prices. The results of this chapter were previously published in Energy & AI [Tre+23].

6.1. Methods

We develop an explainable machine learning model to understand German day-ahead electricity prices beyond the merit order effect introduced in Sec. 2.2.1. Our focus lies on predicting electricity prices given all other feature values at that point in time. That is, we transparently model and analyze the electricity market and the feature-price-interactions, which would not be fully possible in a forecasting setting.

6.1.1. Data

As our prediction target, we use the hourly day-ahead electricity prices for Germany, which we collect from the ENTSO-E transparency platform [ENTb]. Since European day-ahead market prices are coupled via the SDAC explained in Sec. 2.2, we get only one price for all exchanges.

It is important to note that Germany shares its bidding zone with Luxembourg and also shared it with Austria until the 1st of October 2018 [EEX18]. Throughout the thesis, *German electricity prices* denote the price in the bidding zone of the given time period. Prices before and after the change of the bidding zones are joined together to create one continuous time series (see Fig. 6.1a).

As inputs for our prediction model, we use power system features and fuel prices. Power system features are collected from the ENTSO-E transparency platform [ENTb] and fuel prices are collected from ARIVA.DE AG [ARI22]. A full list of the features can be seen in Fig. 6.2.

Power system features include day-ahead forecasts of load, solar generation, wind generation, the day-ahead total generation and imports and exports. The features are aggregated for the four control areas of 50Hertz Transmission, Amprion, TenneT and TransnetBW. Wind generation is aggregated from wind on- and offshore generation. Total generation corresponds to the total scheduled generation in the day-ahead market. Import and export is aggregated from the cross-border flows between Germany and the neighbouring bidding zones, where a positive (negative) value corresponds to more energy imports (exports). We also supplement the power system features with ramps for each feature, which are calculated using the formula $\text{ramp}(t) = f(t) - f(t-1)$ where $f(t)$ denotes the feature at a point t in time.

Fuel prices include oil prices and natural gas prices. Because both features have a daily time resolution we create a linear interpolation to get an hourly time resolution matching the time sampling of the model. Coal prices vary only very little during the considered time span. We note that we exclude CO_2 prices, although they affect electricity prices in the long-term. During

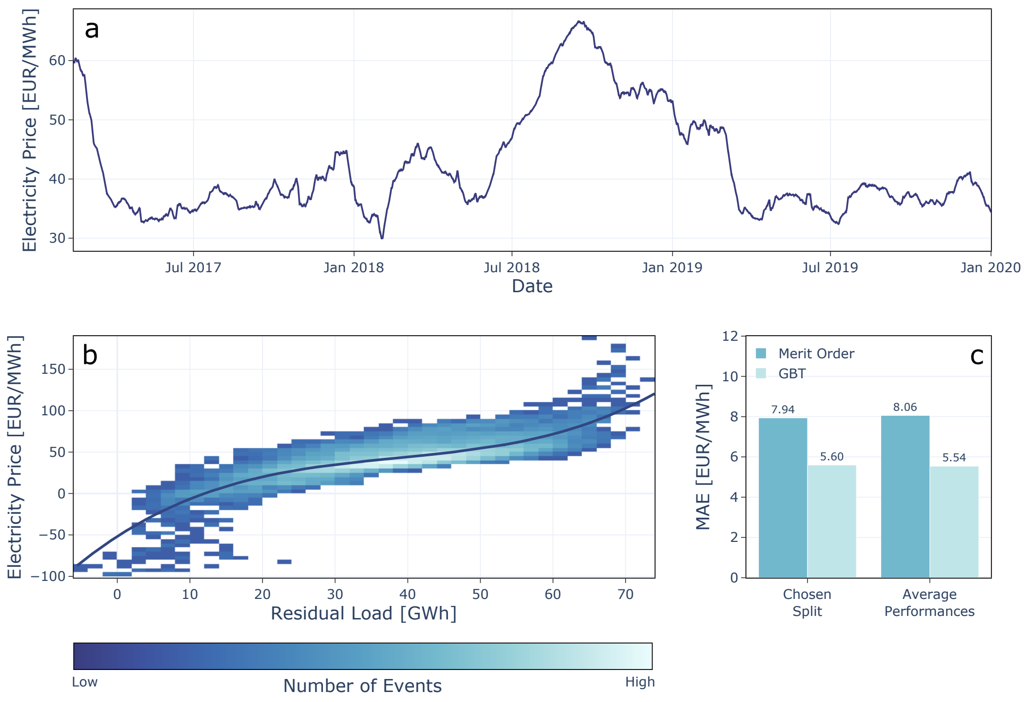

Figure 6.1.: Explainable Machine Learning for day-ahead electricity prices. (a) Electricity price time series from the German day-ahead EPEX spot market from January 2017 to 2020. (b) In a single-feature benchmark model based on the merit order principle, prices are a function of the residual load, i.e. the difference of load and non-dispatchable renewable generation. The colormap shows a 2D histogram of the raw data, the line is a 3rd order polynomial fit. (c) Performance of the benchmark model and a gradient boosted tree (GBT) model, measured by the mean absolute error (MAE) on the test set. The GBT model outperforms the single-feature benchmark model and reveals more information about the electricity market.

the considered time period, CO_2 prices have increased almost monotonically allowing the ML model to memorize the train set leading to immediate overfitting. In particular, we tested models including the CO_2 price and found a high generalization error.

We use 3 years of data from the years 2017, 2018 and 2019 in order to get enough data for training and evaluation. Hourly data points with missing values for any feature are dropped to prevent fitting corrupted data.

6.1.2. Model

As a benchmark model we use a model based on the merit order principle discussed in Sec. 2.2.1. Figure 6.1b shows the price in the German day-ahead market as a function of the residual load. Data has been collected for 3 years and is displayed as a 2D histogram. We observe that the assumption (2.2) provides a reasonable approximation of the actual market behavior — the price generally increases with the residual load. We fit a third-order polynomial to the data, which had the best performance among all polynomials up to the ninth order on the test set. The third-order polynomial fit will serve as a benchmark model in the following. We find that the data scatters quite strongly around this fit, as various effects are not taken into account in this approximate treatment.

To model the German electricity price, we use Gradient Boosted Trees (GBTs) on our input data consisting of power system and fuel price features. GBTs offer complex non-linear models which we need in order to get more precise predictions of the electricity price than the benchmark model based on a common approximation of the merit order principle [CG16]. We use the LightGBM framework for our implementation in order to achieve a fast model training [Ke+17].

While single decision or regression trees are interpretable by reporting their decision path, ensemble methods, such as GBTs, trade a higher performance for a harder-to-interpret model. Still, using methods such as SHAP enables us to get a detailed explanation of the GBTs which we explain in detail in Sec. 5.2.

For the training process, we split our data into a training (48%), validation (32%) and test (20%) set. The reason for the unusual size of the validation set is that overfitting represents a serious issue for the given dataset. By doubling the validation set and splitting the set into four separate validation sets, we were able to reduce overfitting to an acceptable level. Additionally, reducing the train set did not cause any performance loss. The four validation sets are used for evaluation of the performance after each training epoch, where the training is stopped if the performance of one of the validation sets is not improving for a predefined number of epochs. Since we focus on explaining our model instead of forecasting electricity prices as mentioned above, we shuffle our data before splitting. We use a weekly shuffle, where we only shuffle the dataset by weeks instead of hours before splitting. This gives us a more general model because we reduce the amount of similar data points in the training, validation and test set. We use the L^2 loss for the training process and the corresponding MAE score for evaluating the performance of the models.

We use a random search to find the best hyperparameters where we evaluate the performance of the fully trained models on the unseen test set. We choose the model with best performance on the test set. For specific analysis tasks, we need to ensure the consistency of our models. We achieve this by analyzing the 10 best models of our random search for 10 different weekly random splits.

6.2. Results

6.2.1. Model Performance

The developed machine learning model is capable of predicting day-ahead electricity prices with an average performance of MAE = 5.54 (Fig. 6.1c). Therefore, the model explains the price with an error of 5.54 EUR/MWh on average. The performance is substantially better than for the benchmark model based on a common approximation of the merit order principle reaching only MAE = 8.06. Our model outperforms the benchmark model by 2.52 EUR/MWh in absolute values, which is a relative increase in performance of 31.3%. We also evaluated both models with the SMAPE and R^2-score for further validation of the results (Tab. 6.1). In summary, all three metrics are within a reasonable range for predicting electricity prices. Our model outperforms the benchmark model for all three metrics. We conclude that the machine learning model captures several market effects which are neglected in the single-feature benchmark model described in Sec. 2.2.1. We will now discuss these effects in detail, interpreting the machine learning

Table 6.1.: Summary of the performance measures on the merit order based benchmark and GBT model. The GBT model outperforms the merit order model for every metric up to a relative increase of 31.1% for the MAE.

	MAE	SMAPE	R^2-score
Merit Order	8.06	23.25	0.66
GBT	5.53	17.82	0.8

model with the SHAP framework.

6.2.2. Features Affecting the Electricity Prices

The developed machine learning model takes into account a variety of different features beyond the residual load. The SHAP values provide a consistent measure of the feature importance and thus reveal which factors have the strongest influences on the prices. Cumulative feature importance are shown in Fig. 6.2.

As expected, the main driver of the electricity prices is given by the residual load. More precisely, the three residual load features (load, wind and solar generation) are also the most important features in the machine learning model. The dependency of these features will be discussed in more detail in Sec. 6.2.3.

Fuel prices rank at position 4 and 6, with oil prices being more important than gas prices. This dependency is not surprising as fuel prices directly affect the variable costs of the respective power plants. However, the precise interpretation of this finding is less clear and will be further discussed in Sec. 6.2.6.

Prices are obviously related to cross-border trading. The import-export balance is the fifth most important feature and will be discussed in detail in Sec. 6.2.4. The total generation and its ramp rank at position 7 and 8. The generation ramp is particularly interesting as it reveals the influence of previous time steps, see Sec. 6.2.5 for details.

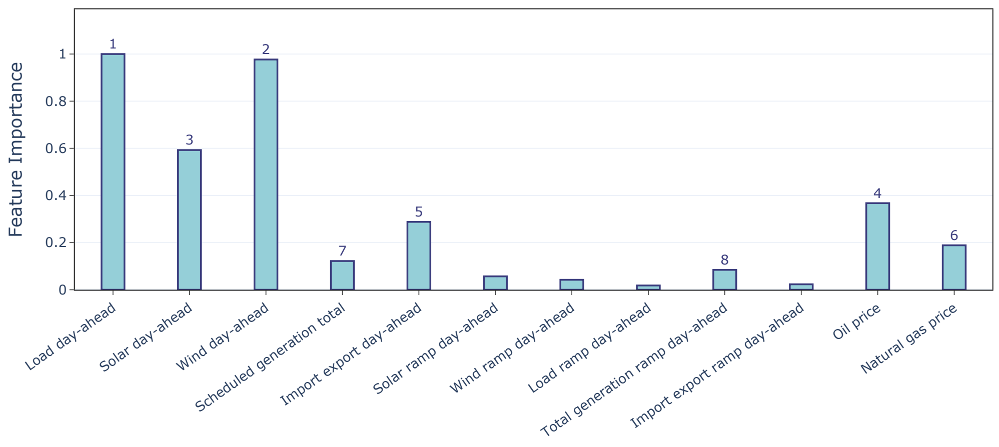

Figure 6.2.: Feature importance in the GBT model for the day-ahead electricity prices. Feature importances are computed from SHapley Additive exPlanations and normalized to one (see text for details). Features contributing to the residual load are most important as expected, but fuel prices also rank high.

6.2.3. The Role of Wind, Solar and Load

The residual load features, i.e. load, solar generation and wind generation, are the most important features for the machine learning model (Fig. 6.2), in agreement with the benchmark model based on the merit order principle explained in Sec. 2.2.1. We take a more detailed look at the contribution of the residual load features by analyzing the corresponding partial dependency plots and interaction plots.

The benchmark model assumes that the price depends only on the residual load, hence the three features enter in an equal way up to a sign. Analyzing the respective partial dependency plots in Fig. 6.3a-c, we observe a similar dependency as expected, but also some subtle differences. For the detailed analysis of the differences and the observed scattering we simplify the comparison of the three features by multiplying the renewable generations by -1.

The dependency on load, wind and solar generation is approximately linear (Fig. 6.3a-c), hence we use a linear fit for a quantitative analysis. We

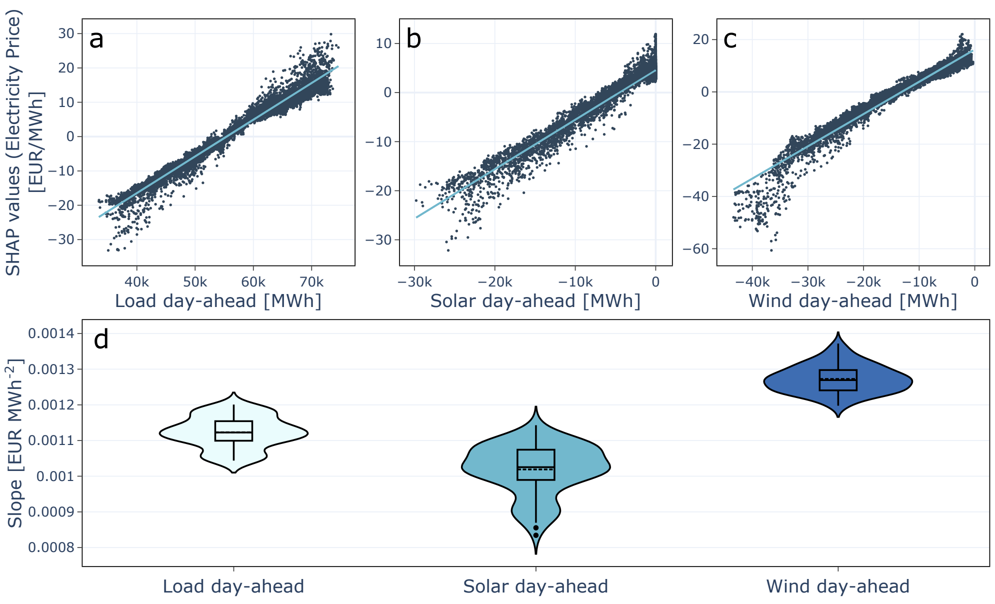

Figure 6.3.: Impact of the residual load features in the GBT model. (a)-(c) Dependency plots for the residual load features load, solar generation and wind generation (measured by mean absolute SHAP values). The light blue line is a linear fit. (d) Slope of the linear fits in the dependency plots. Violin plots shows the results for the ten best models for ten different data splits. The residual load features have slightly different linear relation to the electricity price, which is not captured by the benchmark model based on the merit order principle.

create linear fits for the partial dependency plots for the 10 best models after random search of 10 different random weekly shuffled splits. Figure 6.3d shows the slope of the linear fits on the dependency plots for the 100 different models as violin plots. The different models seem to be consistent with their dependencies since the violin plots show a clear distribution around the mean value of the slope. Only the violin plot for solar generation shows some outliers.

The benchmark model based on the merit order principle assumes an equal contribution of all residual load features, but the machine learning model reveals some subtle differences. The slopes of the dependency on load and solar are rather similar, with solar being slightly smaller. In contrast, the slope of the dependency on wind is notably larger. The smaller influence of solar generation may be due to the fact that solar generation is more distributed in the German power grid, with a large fraction installed directly at the consumers. This could lead to solar generation acting as a negative load in the power grid, which would naturally cause load and solar generation to have a similar impact on the electricity price. In contrast, the generation of wind is more concentrated in the north of Germany, especially in the case of off-shore wind generation. This could lead to wind generation acting more like other power plants in the energy system, making the presence or absence of wind generation more important for the electricity price.

A further reason for the different role of wind and solar may lie in their respective market rules. While small scale PV installations typically rely on fixed feed-in tariffs, wind turbines are incentivized to sell their power according to the wholesale electricity market prices ('Direktvermarktung', see [Ger14]).

The small scattering visible in the dependency plots can be explained by feature interactions. Neglecting all interactions in the dependency plot in Fig. 6.4a-c second column, scattering becomes smaller and the dependency of the residual load features is even clearer. The dependency on load and solar generation is approximately linear, while the dependency on wind shows a clear non-linearity for large in-feeds above 30 GW. In this case, flexible power plants such as natural gas or oil plants have already left the market. Then, market equilibrium requires either an increase in exports, an increase of the load or a reduction of generation from mostly inflexible power plants such as lignite or nuclear. It appears plausible that these three mechanisms are comparatively inelastic such that the price decreases rapidly.

Looking at three of the most important interacting features, we can attribute most of the scattering present in the normal partial dependency plots.

Load has its strongest interactions with the renewable generation wind and solar. Both interactions are similar and enhance the dependency on load. This is reasonable since these features combined serve as a good approximation for the residual load, which is again already a good predictor

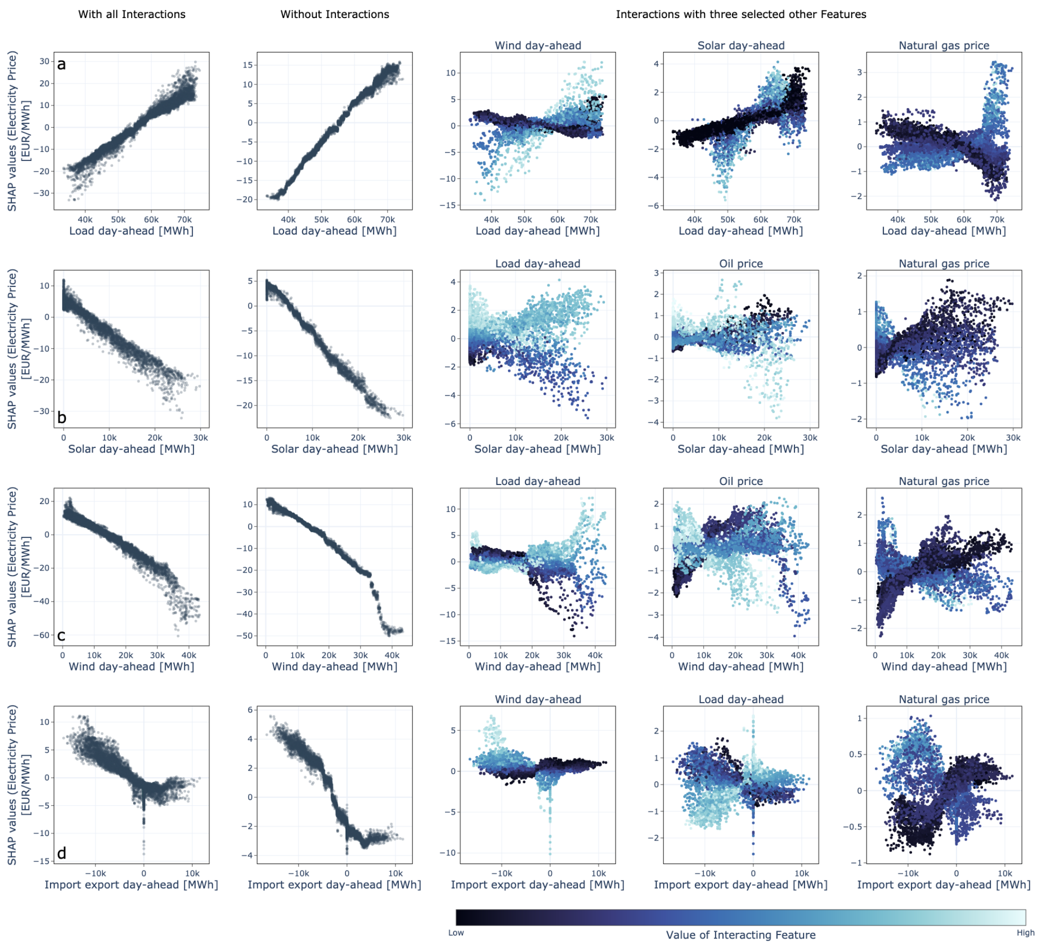

Figure 6.4.: Dependencies and interactions for (a) the load, (b) solar generation, (c) wind generation and (d) cumulative import or export, respectively. The first column shows the full SHAP dependency plot, the second column the SHAP dependency plot without any interactions. The third to fifth column show the SHAP interactions of three selected features, where the color indicates the value of the interacting feature. Fuel prices play an important role when interacting with other features in the GBT model. Further details are given in the text.

for the electricity price. We can also see a strong interaction with the gas price, especially for high gas prices and high load. This is because more gas power plants are active for higher load, which then leads to higher electricity prices if gas prices are also high.

Wind and solar generation have a similar interaction structure, in particular there is a strong interaction with the load but also with fuel prices. For the load, interactions are particularly strong for high renewable generation. If the load is small, we recover the situation discussed above for the case of high wind generation, where market equilibrium requires the adaption of comparatively inelastic participants and thus entails strong prices signals which may even include negative prices. This effect is largely compensated if the load is also high, leading to a strong increase of the price.

We also see strong interaction with the fuel prices, both amplifying the dependency for low wind or solar generation and reducing it for high wind or solar generation. These interactions originate from the fact that more fuel-dependent power plants are active if renewable generation is low and therefore the electricity price is more dependent on fuel prices in this case.

Summarizing, the residual load features have an overall strong interaction with fuel prices, mostly due to more dispatchable generation being active for specific values of the residual load features. Fuel prices not being included in the single-feature benchmark model could explain its lower performance compared to the GBT model.

6.2.4. Prices and Cross-border Trading

Electricity prices and trades are intimately related. In the machine learning model, the import-export balance ranks at fifth place in terms of the feature importance (Fig. 6.2). Different mechanisms can contribute to this dependency. On the one hand, high prices foster imports from other countries. On the other hand, imports provide a further source of electricity and should thus lead to lower prices. These two interactions can be interpreted as opposite causal relations, where either the price or the import-export balance is acting as the driver.

However, we stress that a causal interpretation is not that straightforward. The prices in different bidding zones and the cross-border trades are not determined sequentially, but simultaneously via the EUPHEMIA algorithm [Küh+21], see also Sec. 2.2.

The SHAP dependency plot reveals a negative correlation of the import balance and the day-ahead price (Fig. 6.4d). That is, imports are typically related to lower prices, while exports are related to higher prices. To understand this correlation, we consider one specific market situation. Assume that there is a high wind power generation in Germany. Typically, there are many low price offers in the German bidding zone, leading to a low market clearing price. However, if here is a strong demand in a neighboring country, additional offers will be accepted for exports, leading to an increase of the market clearing price. Hence, it appears as if the exports drive the market price, but in fact both are driven by a common cause: the total supply and demand in the two neighboring countries combined.

The strongest feature interactions are found for wind power generation and load. The interaction is opposite, which is comprehensible because the two feature enter the residual load with opposite sign. We find that an increase of the residual load reduces the observed dependency, while a decrease of the residual load increases it.

Since SHAP values reveal only correlations of features and targets, it is difficult to reach a comprehensive causal interpretation. Still, based on our results, we formulate the following hypothesis. A high demand from a neighboring country generally leads to exports and to an increase of prices. But if the domestic demand (the residual load) is also high, there are no cheap offers left in the order book that would allow for exports. Hence, the dependency of exports and prices diminishes.

Notably, there is large scatter to lower prices in the case of a vanishing import export balance. This might be due to a temporary reduction in the transmission capacity preventing exports, or corrupted data (see [HMB18] for a discussion of the data quality of the ENTSO-E transparency platform).

Finally, we also see an interesting interaction of the import-export balance with gas prices. Lower gas prices tend to reduce the overall contribution of import-export while higher prices amplify this contribution. This could indicate that at low gas prices, local gas generation reduces the dependency of a country to exchange power with neighboring countries and hence the price is influenced less by its imports and exports. In the opposite case of high gas prices, countries will be more willing to exchange energy and the effect of cross-border flows on prices increases.

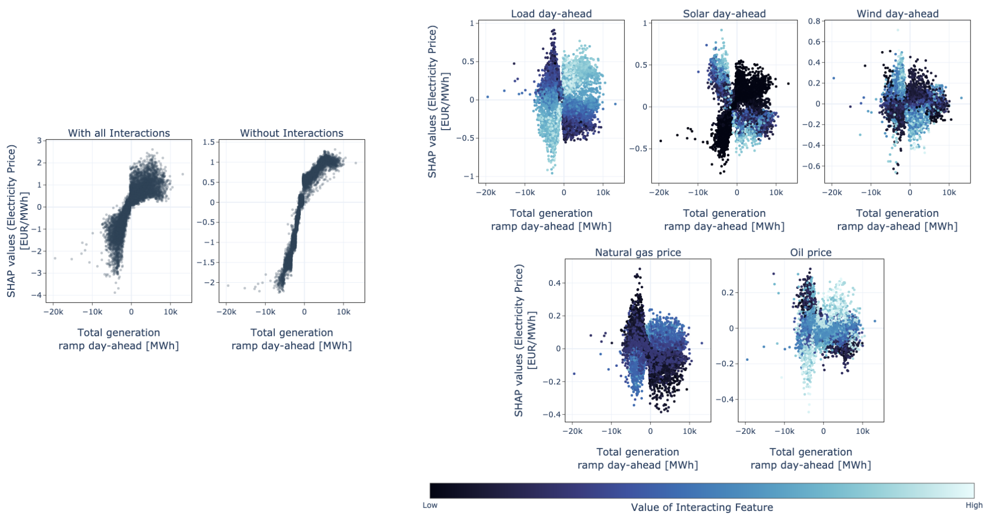

Figure 6.5.: Effect of generation ramps and limited generation flexibility on electricity prices. The first column shows the full SHAP dependency plot of the total generation ramp, the second column the SHAP dependency plot without any interactions. Further panels show the SHAP feature interactions for five selected interacting features. The color indicates the value of the interacting feature. Ramps affect/alter the electricity price prediction by up to 10%, depending on factors as load and renewable generation, but also on fuel prices.

6.2.5. Impact of Ramps

The machine learning model reveals a weak dependency of the price and the power generation ramps (Fig. 6.2). Hence, the market outcome in a certain hour is affected by previous hours. The partial dependency plot (Fig. 6.5 left) indicates a positive correlation of the price and the total generation ramp. Hence, prices tend to be higher if generation is ramped up and lower if generation is ramped down.

This finding can be attributed to a limited flexibility of conventional power plants, in particular nuclear and lignite plants [VD15]. First, technical limits exist for the ramping speed and the minimum generation in partial load. Second, ramping and cycling induce additional costs, for instance due to a wear and tear of the power plant. Hence, there is an incentive to limit generation ramps which can affect the bidding on the market. In case of a decreasing total generation, operators may bid at a lower price to remain in the market and avoid ramping downwards. Similarly, in case of an increasing total generation, operators may bid at a higher price. As a consequence, the price increases with the total generation ramp.

The role of generation ramps depends on several other factors. The SHAP dependency plot is strongly scattered which can be attributed to the presence of feature interaction. A high value of the load amplifies the impact of generation ramps. If the load is high, more conventional generation is needed in general, such that ramping limits and costs are more important. Vice versa, high values of wind and solar generation mitigate the impact of generation ramps as less conventional plants are needed. Furthermore, high oil and gas prices amplify the impact of generation ramps, too. Oil and gas power plants typically have a higher flexibility than nuclear or coal power plants. Higher fuel price penalize these plants, such that nuclear or coal power plants may have to contribute more strongly to the ramping process.

6.2.6. Impact of Fuel Prices: Correlation or Causality?

Looking at the feature importance in Fig. 6.2 we note that the machine learning model is critically dependent on fuel prices. The oil price is the 4th most import feature, i.e. the most important feature after the residual load features, while the gas price is the 6th most important. We analyze the dependency on oil and gas prices in detail.

The dependency plots for the fuel prices in Fig. 6.6a-b are strongly non-linear, with an almost step-like behaviour. The dependency of the electricity price on the oil price is approximately constant below a threshold of 69 [USD/bbl]. Above this threshold, the dependency increases until it saturates. While the change in dependency is almost linear for oil prices, the dependency for gas prices shows a step-like behaviour, with a threshold at 2.75 [USD/mmBTu], albeit with a stronger scattering. A causal interpretation is comprehensible as an increase in fuel prices leads to an increase in the operational costs of the respective power plants and thus to offers at higher prices.

For further analysis, we focus on the time series of the electricity price

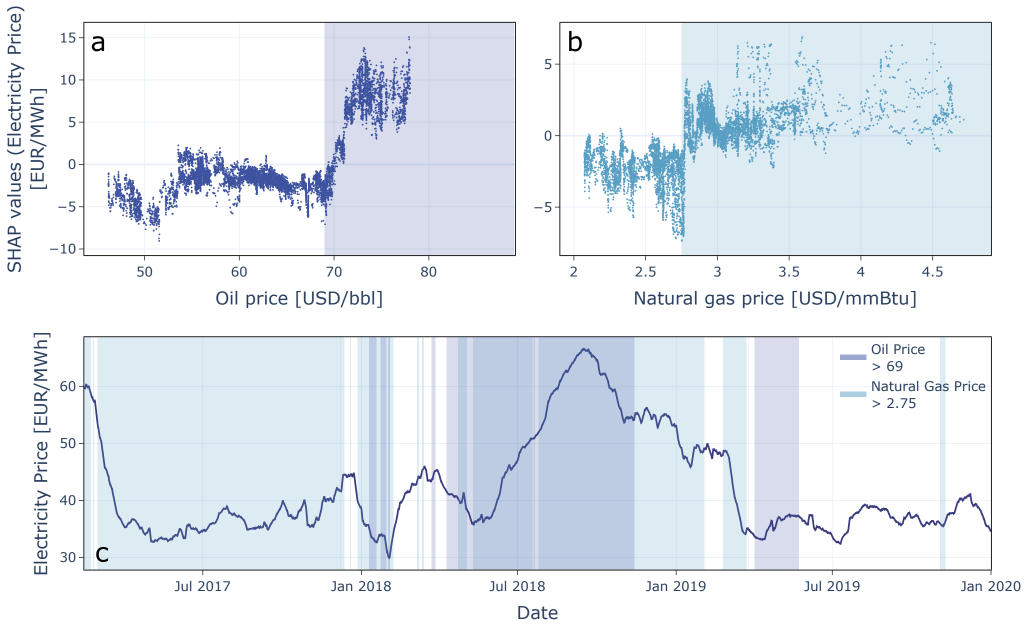

Figure 6.6.: Impact of oil and gas prices on the electricity price predictions. (a)-(b) SHAP dependency plot of the oil price (gas price), where the colored area marks a change for the dependency in the GBT model. (c) Electricity price time series from the German day-ahead EPEX spot market from January 2017 to 2020. Dark (light) blue areas mark the time periods with oil (gas) prices above the dependency threshold. We observe a clear change in the dependency on fuel prices in the model, but it cannot be asserted whether this is a causal relation.

in Fig. 6.6c, highlighting the time periods with oil and gas prices above the dependency threshold respectively. High oil prices seem to be correlated with high electricity prices, especially for the maximum of electricity prices at the end of 2018. In contrast, in the middle of 2019, electricity prices stay low while oil prices are above the threshold. For gas, the dependency is even less clear. Lower prices seem to be a proxy for low electricity prices in 2019. Meanwhile, high gas prices do not display a clear correlation with overall electricity prices in the time series. We further note that a delayed relation could also be possible, if power plants purchase fuel well before the usage.

In general, it is difficult to pinpoint the relation of fuel prices to the electricity price beyond statistical correlations. Although we find a strong change in the dependency on the fuel prices in the machine learning model, it is still possible that the machine learning model is using the fuel prices to remember specific time periods where electricity prices are higher or lower than expected from the other features. Nevertheless, we find reasonably strong interactions of the residual load features with fuel prices, as discussed in Sec. 6.2.3. This points to a causal interaction, but a confounding effect is also possible. Overall, fuel prices are correlated with electricity prices which could be one of the reasons the machine learning model outperforms the benchmark model.

7. Electricity Price Forecasting

In this chapter, we move from the ex-post analysis of electricity prices to the extrapolation and forecasting. We aim to achieve maximum performance on the task of forecasting day-ahead electricity prices. We establish deep neural networks using LSTMs that are able to achieve state-of-the-art performance. Furthermore, we investigate the level of model complexity necessary for LSTMs to capture important system dynamics.

In contrast to other works such as Lago et al. [LDD18; Lag+21] or Tschora et al. [Tsc+22], we attempt to optimize the model for the German market only. This enables us to demonstrate the capabilities of LSTMs for real-world scenarios, which would also always be optimized for specific markets. We use an extended dataset based on the results of Chap. 6 but also add additional features based on system knowledge.

The chapter is structured as follows. In Sec. 7.1.1, we discuss how we obtained and processed the dataset in detail. Furthermore, the models and the associated learning processes are discussed in Sec. 7.1.2. We then proceed in Sec. 7.2 to analyze the results of the machine learning model. In the first part, we focus on the model complexity and present the best performing model. The best model is further analyzed in the second part. We discuss its overall performance during different time periods and investigate the added value of a probabilistic forecast.

7.1. Methods

7.1.1. Data

For the task of forecasting electricity prices in the day-ahead market we try to stay as close to a real world scenario as possible. This means, that we try to only include features that would be available right before the market closure at 12:00. For the exact timings in the day-ahead market see Sec. 2.2. The features used for this study are primarily based on the information we gained from the XAI model in Sec. 6.2. Furthermore, we include features based on system knowledge about meaningful data for the day-ahead market.

Since we are only interested in the prediction of the day-ahead electricity price in the German market, the data is also optimized for the forecasting of the German electricity prices. The target we use is the electricity price in the day-ahead market of the German bidding zone given by the SDAC (see Sec.2.2.2) for each hour. Notably, Germany shares its bidding zone with Luxembourg and also shared it with Austria until 1st of October 2018 [EEX18]. Throughout the thesis, German electricity prices denote the price in the bidding zone of the given time period. Prices before and after the change of the bidding zones are joined together to create one continuous time series (see Fig. 7.2). Furthermore, we will refer to different countries using the ISO 3166-1 alpha-2 codes assigned by ISO [ISOa] which can be found at [ISOb]. A full list of all country codes used throughout this thesis can be found in Appendix A. Shared bidding zones will be denoted by both country codes, e.g., DE-LU denotes the German-Luxembourg bidding zone.

The collected data can be categorized into different classes of features, presented in detail in Fig. 7.1. First we collected power system features for DE-LU the and the neighbouring bidding zones, which are detailed below. Additionally we collected fuel prices for different assets. In the following we will highlight the data preprocessing in detail. Data is used from 01.01.2016 until 01.01.2023.

For all power system features we use data available at the ENTSO-E transparency platform [ENTb]. Data is retrieved via the restful API provided by ENTSO-E [ENTf] using the entsoe-py open-source implementation for python available at GitHub [PB].

Electricity prices are collected from DE-AT-LU and DE-LU for their respective time period and concatenated into one single price time series. These prices will usually be referred to as the *day-ahead prices*. The full price time series can be seen in Fig. 7.2.

For DE-LU and all neighbouring bidding zones AT, BE, CH, CZ, DK1,

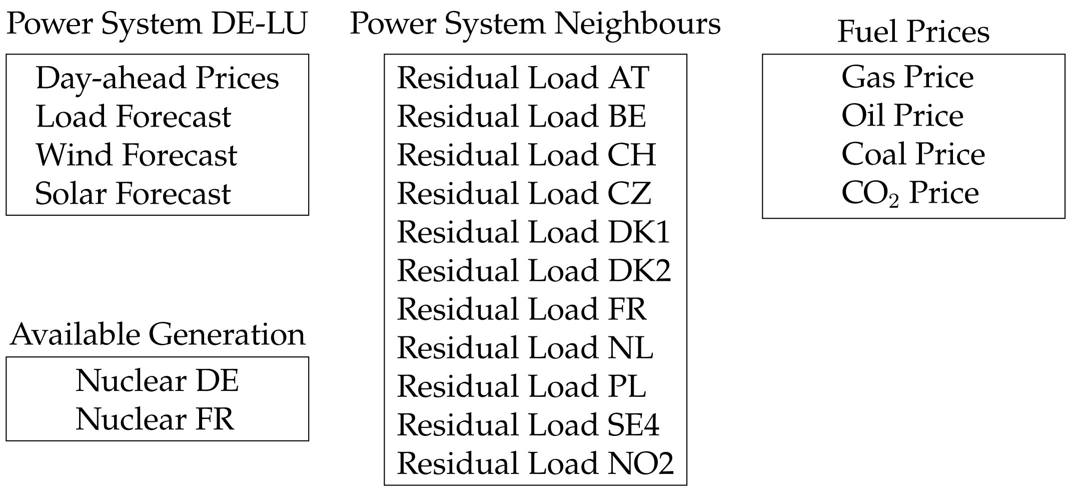

Figure 7.1.: Overview of the dataset used for electricity price forecasting. The data can be categorized into the subclasses of German power system features, neighbouring power system features and fuel prices. Day-ahead prices for DE-LU are used as a target.

DK2, FR, NL, PL, SE4 and NO2, we collected day-ahead forecasts for load, wind generation and solar generation. Note that the bidding zones DK2, SE4 and NO2 are not direct neighbours of DE-LU, but are connected via High Voltage Direct Current (HVDC) lines. Also only neighbouring bidding zones of DE-LU are included, not of the former bidding zone DE-AT-LU.

Load forecasts are published two hours before the closure time of the corresponding market, which is 10:00 for the SDAC. The load forecast is calculated based on historic load profiles including weather changes and social factors [ENTa]. Unfortunately, forecasts for wind and solar generation are published at 18:00 the day before delivery [ENTc]. This makes the data of wind and solar forecast not suitable for the study in the sense of real scenario forecasting. Nevertheless, we use this data for two reasons. First, there is no available alternative and all comparable studies use the same data sources [Lag+21; Tsc+22; LB21]. Changing our data source would therefore result in the loss of comparability. Second, using forecasts published after market closure should not increase forecasting performance but at most deteriorate it. This is because the market behaves according to the information known at market closure time. Since market participants do not know what forecasts at

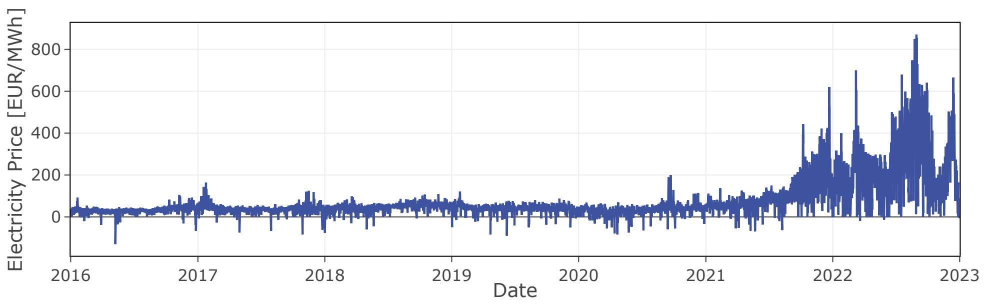

Figure 7.2.: Electricity price time series from the German day-ahead market from January 2016 to 2023. We observe a strong increase of electricity prices at the end of 2021, with a simultaneous increase of the volatility due to the unfolding European energy crisis. Prices continue to stay high during 2022.

18:00 will look like, these forecasts differ from the information known to the model with respect to a real scenario, but not for the better. Wind forecasts are available separately for on-shore and off-shore wind. We aggregate both types of wind into one feature for wind forecasts.

In order to use the same target and features for the whole time period of this study we create a virtual bidding zone for DE-LU before the 1st October 2018. Since load, wind generation and solar generation forecasts are also available for AT separately in this period, due to AT being a control area (CTA), we can subtract data of AT from DE-AT-LU. The resulting virtual bidding zone of DE-LU from 01.01.2016 until 30.09.2018 is concatenated with the real bidding zone DE-LU. The resulting data can be seen in Fig. 7.3, where outliers of the data were removed.

For SE4, the data of solar generation forecasts are missing before 2022. Since there also exist no data about the installed capacity of solar generation on the ENTSO-E transparency platform, we decided to remove SE4 from the dataset entirely.

PL suffers the same problem, having no data of solar generation forecasts before 10.04.2020. But since the ENTSO-E transparency platform offers additional data about the installed solar generation capacity of PL, we were

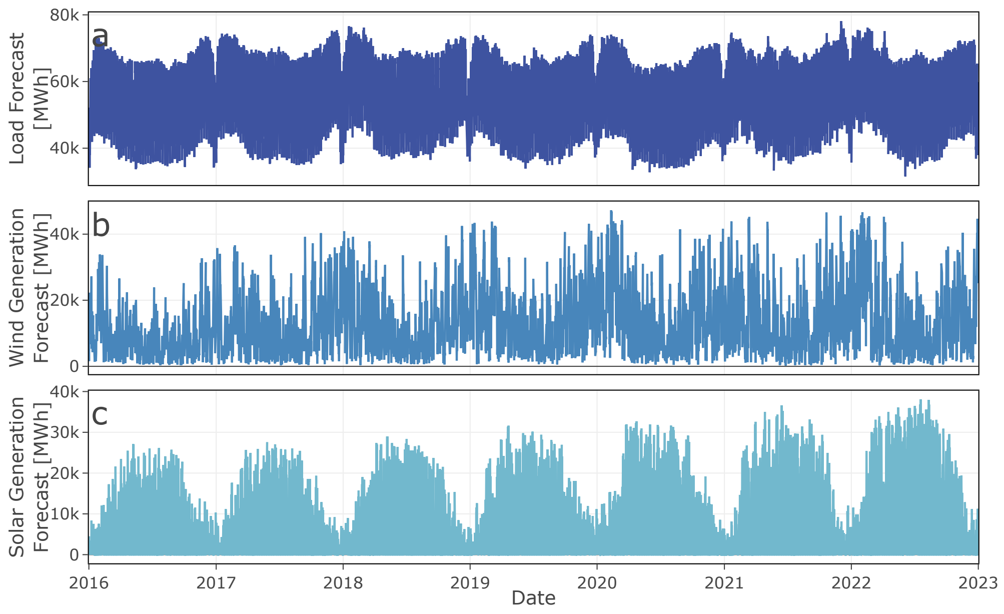

Figure 7.3.: Power system features of the German bidding zone from 2016 until 2023. (a) Time series of the load forecast of DE-LU. (b) Time series of wind generation forecasts of DE-LU. The feature is aggregated from forecasts of on-shore and off-shore wind generation. (c) Time series of the solar generation forecast of DE-LU.

able to model the missing data points. Using the solar generation forecasts from neighbouring bidding zones as input features, we trained a simple FFNN to predict the capacity factor of the solar generation in PL. The model was trained on data after 10.04.2020 and yielded a R^2-score of 0.87. We used the model on the features before 10.04.2020 and rescaled the resulting data with the installed capacity of PL. The simulated and available data can be seen in Fig. 7.4.

For the neighbouring bidding zones, load, wind generation and solar generation forecasts are combined into a single feature per bidding zone, the residual load,

$$\text{Residual Load} = \text{Load} - (\text{Wind} + \text{Solar}). \tag{7.1}$$

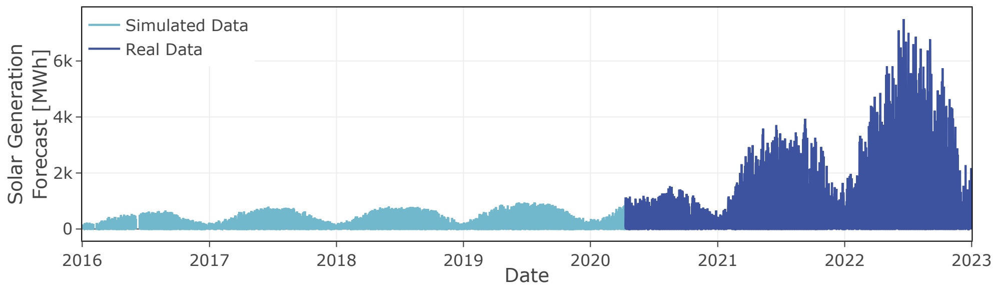

Figure 7.4.: Solar generation forecast of the Polish bidding zone. The real data (dark blue) was only available from 10.04.2020. Data before was simulated using the yearly capacity of solar generation of PL and the solar forecasts of neighbouring bidding zones.

The final dataset includes only the residual load of all neighbouring bidding zones.

We remove outliers that are obviously erroneous datapoints for all features.

As an additional feature, we use a lagged price time series of the day-ahead prices of DE-LU. The prices are shifted by 24 hours which guarantees the data to stay valid for realistic price forecasting. More sophisticated shifting of prices is also possible with respect to the market timings, but are beyond the scope of this thesis and will be discussed as outlook.

Based on [Rin19], we comprehend the power system data with available nuclear capacity for each respective trading hour, other generation is expected to be of minor importance. We restrict the data to DE-LU and FR, as France has by far the largest installed nuclear capacity in Europe. For both bidding zones, we create data on the installed capacity and subtract the planned unavailability of nuclear power plants afterwards. Planned unavailability relates to the data published on the ENTSO-E transpency platform for the planned and unplanned unavailabilities of power plants [ENTg].

For both DE-LU and FR, we get the data for installed nuclear capacity from ENTSO-E [ENTd]. For DE-LU, we additionally back up the data using *Marktstammdatenregister* [Marc]. Afterwards, we get data from [ENTg] for

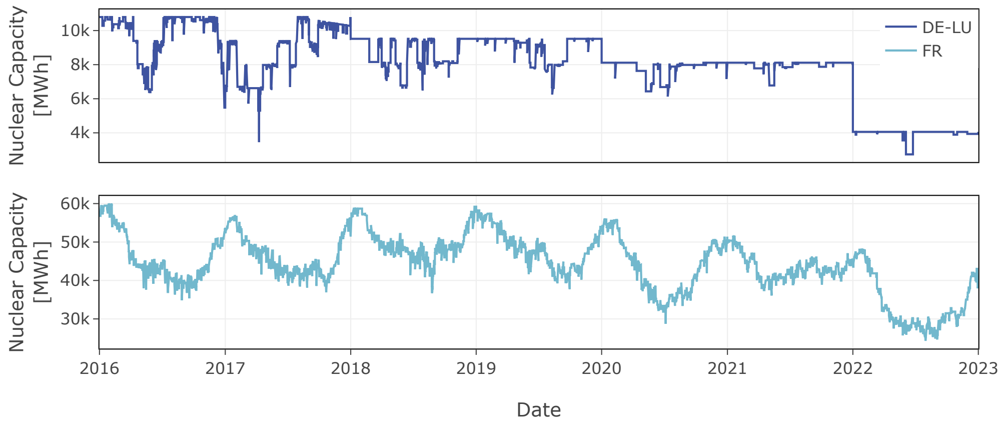

Figure 7.5.: Available nuclear generation capacity for each hour from January 2016 until 2023. For both bidding zones DE-LU and FR the available capacity is computed from the installed capacity minus the planned maintenance.

planned unavailabilities and subtract from the installed capacity. The final data can be seen in Fig. 7.5. Notably, the data of planned unavailability seems to have a low quality. Nevertheless, we use the data since no reliable alternatives exists.

In addition to power system features, we include several fuel prices in the dataset. Fuel prices are collected from different sources which will be addressed in the following. Notably, fuels are traded on markets similar to the stock market, where trading is continuous during trading hours. As opposed to the electricity market, fuel markets are only open during the day on weekdays, resulting in opening and closing prices for the markets. To ensure a realistic forecasting approach, we only take the opening price for each trading day and shift them by exactly 24 hours. Afterwards, we forward fill all hours of the respective day, resulting in constant fuel prices for each day. More sophisticated techniques for shifting and interpolation of the fuel prices are possible, but are not the focus of this thesis and will only be addressed in the outlook.

Gas prices are taken from two separate data sources, since the Dutch

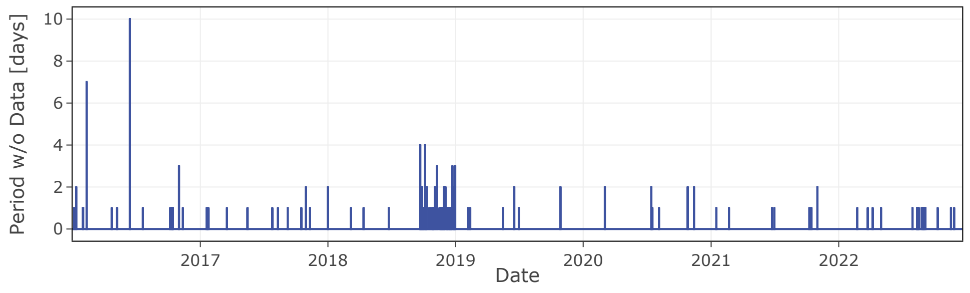

Figure 7.6.: Periods of missing data in days for the full dataset. We observe several days without data at the end of 2018.

TTF gas price was only available from 23.10.17. Dutch TTF gas prices were taken from Investing.com [Inv]. Data before the 23.10.2017 was taken from MarketWatch [Marb] and was scaled to match the price level of the rest of the price time series. Oil prices are taken from FRED [Fed] and coal prices from MarketWatch [Mara] for the whole time period. We also include the price of carbon emission certificates in the dataset. Data was taken from EEX [EEX] for the whole time period.

The final dataset had to be cleaned with respect to missing data points. Especially the data from the ENTSO-E transparency platform misses many hours of data for several features, see [HMB18] for a discussion of the data quality. Since sophisticated approaches to recover missing data points are beyond the scope of this thesis, we discard any hour with at least one missing feature. This results in several periods without any data, presented in detail in Fig. 7.6. Nevertheless, we gathered enough valuable data suitable for neural networks even with many missing hours.

7.1.2. Model

To forecast the day-ahead electricity price in Germany we use long short-term memory (LSTM) networks. From the dataset, we use data with a fixed sequence length of 96 hours as input and the electricity price of a single hour as target. Even though most related studies try to forecast all 24 hours of the next day simultaneously (e.g. [Lag+21; Tsc+22]), we decided to use single point forecasts to enable a larger dataset. Additionally, we use a probabilistic forecast to increase usability of the model in real world scenarios, explained in detail below.

We address two different tasks for the LSTM networks. First, we explore the impact of model complexity on the performance. Aftwerwards we try to achieve best performance using LSTMs to validate the concept of LSTMs for electricity price forecasting. Therefore we use different structures to change the complexity of the models. We change the two hyperparameters depth and width in the study of the complexity. For the best model, we additionally change the early stopping parameter.

Depth changes the number of LSTM cells that are stacked in each step of time. We use LSTM networks with a depth of one layer up to a depth of five layers. The width changes the size of the hidden layer inside the LSTM cells. For networks with multiple layers we use the same width for all layers. All networks use two FFNN layers to map the output of the LSTM layers to the final output of the model. They map from the width to a hidden layer with size 32 and then to the output of the model.

As output of the networks we use two values to create a probabilistic forecast assuming a normal distribution with the probability density function

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{7.2}$$

for the sake of simplicity. The first output value is used as the mean μ of the distribution and the second value as the standard deviation σ. We impose a lower bound for the standard deviation, $\sigma \geq 0.01$, to prevent the neural network to predict negative values.

In LSTM networks, we use a dropout layer at each hidden state. Dropout layers are used as a regularization parameter in deep neural networks. They prevent overfitting by setting values of the hidden states to zero with a certain probability [Sri+14]. We set the probability of the dropout layers to 0.2 in all LSTM networks.

Mean and volatility of the data is changing over time which can decrease model performance after a certain time. Additionaly, new regulations or ex-

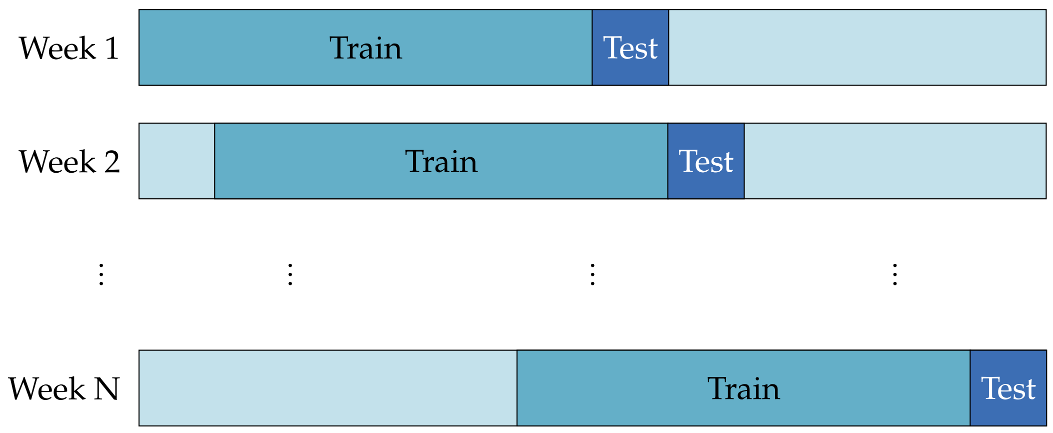

Figure 7.7.: Schema of the recalibration process used for all models. The overall testing period is covered by iteratively shifting the train and test set by one week. We train a new model for each train and test split.

ternal factors may alter feature dependency. Therefore, retraining the model after each prediction would be optimal. Since recalibration is computationally expensive, we restrict our models to be retrained only after one week instead of one day or even one hour. Retraining refers to the training process of a completely new model that is trained on the same amount of data, but shifted for one week. In analogy to a real-world forecasting scenario, the model would be used for one week until it gets updated on recent data. See Fig. 7.7 for a schematic representation of the recalibration process.

For each week in the evaluation period, we train a new model based on a train, validation and test set. The test set corresponds to the week we evaluate the final performance on. Notably, we removed every week with less than 120 hours from the evaluation process. We use 17000 hours before the test set as the train and validation set, which corresponds to approximately two years of data. Depending on the density of missing values, this creates train and validation sets with different time ranges. Since most values are highly non-stationary, it is essential to include most recent data points in the training set. We decided to shift the validation set, which is usually at the end of the training set, to older data. The validation set uses 1000 hours and ranges from hour 15500 until hour 16500. This leaves 500 hours of the most recent data points for the training set. Figure 7.8 shows a visualization of

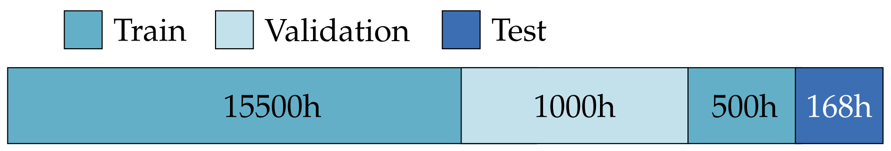

Figure 7.8.: Train, validation and test split of the dataset. The test set consists of one week corresponding to 168 hours. The initial train set includes all 17000 hours before the test set. The validation set consists of 1000 hours and is not taken from the end of the train set but shifted by 500 hours. Therefore the most recent data is included in the train set.

the data split.

To ensure a normalization of the data with no look-ahead bias, we normalize all data points according to the train set. All data points are divided by the maximum feature values in the train and validation set. Additionally, we scale all values by a factor of 0.8 to ensure that most of the values of the test set lay between -1 and 1. The target is normalized in the same way and gets rescaled before performance evaluation.

For all models we adjust the maximum number of epochs such that no model stops training before the early stopping condition is triggered. For models that reached the maximum number of epochs allowed, we repeated the process with a larger number of epochs.

As usual for probabilistic predictions, the neural network is trained in a maximum likelihood fashion. That is, the network should maximize the likelihood of the observations x_i,

$$\prod_i f(x_i|\mu_i, \sigma_i), \tag{7.3}$$

where the product is taken over all time steps i in the training set. In practice, one rather minimizes the Negative Log-Likelihood (NLL) (see Eq. (3.8)), as the likelihood is prone to numerical instability. We use the Adam optimizer for the training process, which computes individual adaptive learning rates for different parameters and is more efficient than the standard stochastic gradient descent [KB14]. For the model and the optimization we use the PyTorch implementation [Pas+19].

7.2. Results

7.2.1. Complexity of LSTM Networks

The performance of machine learning models strongly depends on the initial choice of complexity. Different regularization parameters try to control the trade-off between underfitting and overfitting but cannot rectify large mismatches of complexity. Therefore we analyze LSTM models for different complexities, changing the depth and the width of the layers.

First evaluations were made on the entire testing period from 2019 until 2023. Because performance for the years was correlated, we decided to test only on one year of test data to reduce computation time. The price time series can be separated into two regimes of low prices until 2021 and high prices starting in late 2021 (see Fig 7.2). Since the year 2021 is the only year that includes both regimes, we chose 2021 as the year of evaluation.

For every set of hyperparameters, the models were evaluated using the Mean Absolute Error (MAE) and the Negative Log-Likelihood (NLL) (see Eq. (3.4) and Eq. (3.8) in Sec. 3.1). The MAE enables comparison to models from other works that do not use probabilistic forecasting and also enables easier interpretation than the NLL. Nevertheless, since the NLL was used during training, we include the NLL for interpretation of underfitting and overfitting during training. Notably, since our models were trained using NLL, performance measures based on the mean prediction give lower results than possible when solely optimizing the mean.

Using the recalibration method presented in Sec. 7.1.2, we fit a new model for each week of test data. We fit 50 different models for 2021, after removing weeks with too many missing values. Therefore we analyze the complexity with respect to the mean performance of all 50 models trained for 2021.

To ensure a fair comparison between different model complexities, we analyze the impact of the early stopping parameter first. Early stopping is the main regularization parameter of deep neural networks. It tries to makes sure that the training process stops when the generalization error starts to increase. In theory, LSTM models with a higher complexity may need more epochs to adapt weights into optimal regions, while models with

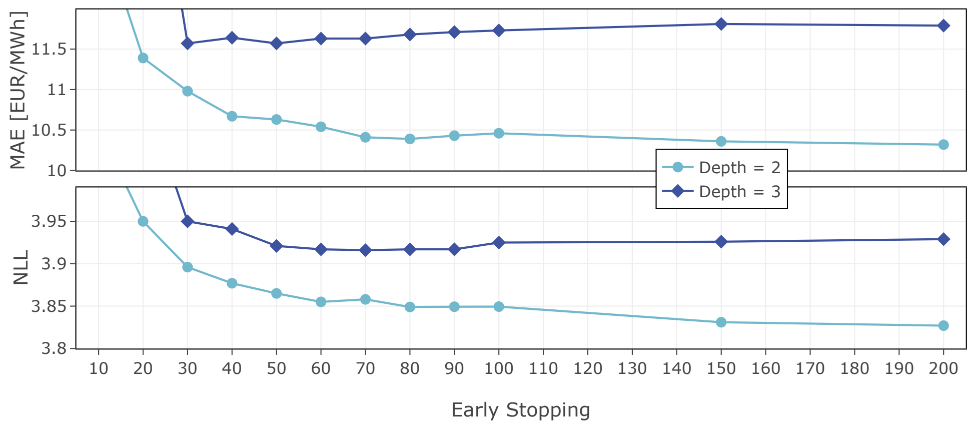

Figure 7.9.: Impact of early stopping on the model performance. The figure shows the mean absolute error (MAE) and negative log-likelihood (NLL) for two LSTM models with depth 2 and 3 for different values of the early stopping parameter. The width of both models is fixed at 32. We observe that performance increases for more epochs but starts to decrease again at a certain threshold. This is probably due to overfitting.

lower complexity are expected to start to overfit after less epochs.

We evaluated the performance of two LSTM models with fixed complexity, e.g. fixed depth and width, using early stopping from 10 to 200 (see Fig. 7.9). Both models have a width of 32 for each layer and a depth of 2 and 3 respectively. Performance for both models starts to converge after a value of 50 for early stopping. A higher early stopping parameter results in a small increase in performance but does not seem to result in strong overfitting. We conclude that the regularization of the models is stable and stops the training process before strong overfitting can take place.

For further analysis of the complexity, we establish models for the early stopping parameters 25, 50 and 100. This should ensure a fair comparison of all LSTM models.

The analysis of the depth of LSTM models was conducted using depths ranging from 1 to 5, results are shown in Fig. 7.10. The width for all models and every LSTM layer is fixed at 32.

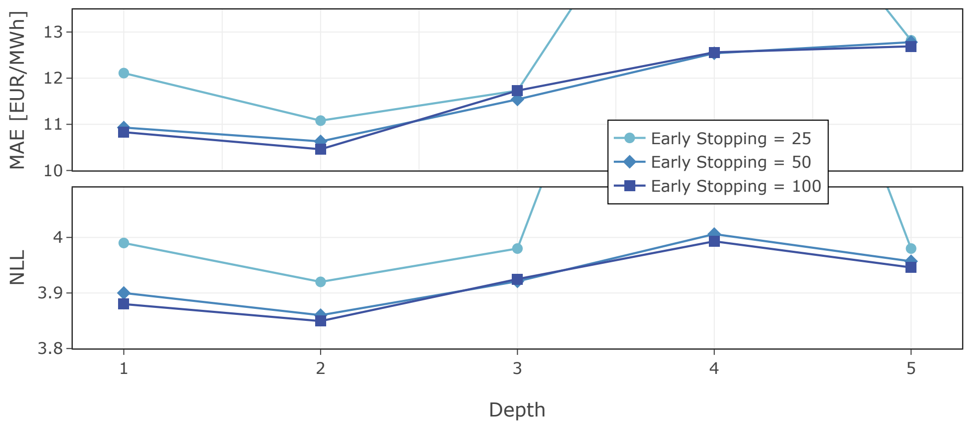

Figure 7.10.: Impact of depth on the LSTM model performance for different values of early stopping. The error metrics used are the mean absolute error (MAE) and the negative log-likelihood (NLL). Every layer has the same width fixed at 32. We observe that higher complexity does not result in better predictions. The best model for both early stopping parameters consists of 2 layers.

We see that MAE and NLL show the same dependency. Differences between MAE and NLL can be explained by the predicted standard deviation of the model. Even though the NLL decreases, the MAE can increase if the model only optimizes the standard deviation while deteriorating the mean prediction.

LSTM models with two layers achieved the best performance for all early stopping parameters. Models with only one layer seem to lack the complexity to model market behaviours. This is likely due to underfitting as a single layer cannot enable sufficiently rich feature extraction necessary for electricity price forecasting.

In contrast, LSTM models with more layers seem to have problems to use the complexity for generalization. For the MAE, the performance decreases monotonically with greater depth of the model. Using the NLL, models with 5 layers seem to increase in performance again. This effect is probably due to bad initialization of the weights for models with a depth of 4, but needs further investigation. However, it does not seem that models with more

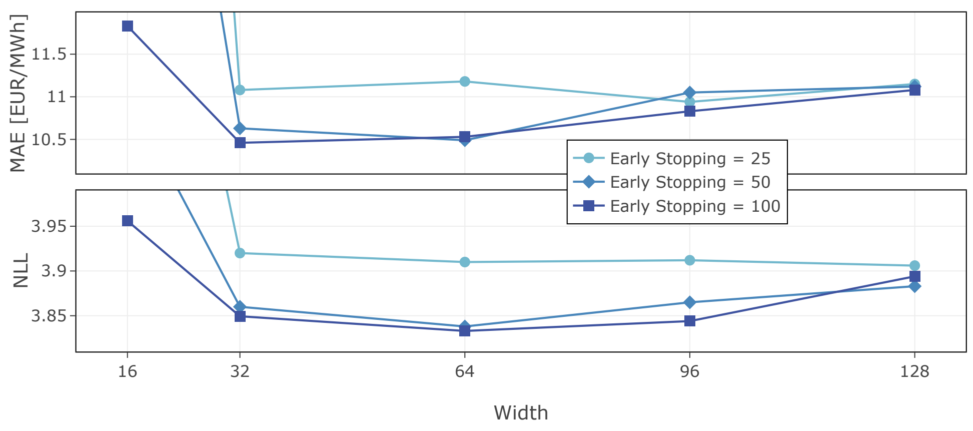

Figure 7.11.: Impact of width on the LSTM model performance for different values of early stopping. The error metrics used are the mean absolute error (MAE) and the negative log-likelihood (NLL). Every model has the same depth fixed at 2. We observe that the best model depends on the performance measure used. While a width of 32 achieves the best MAE, a width of 64 achieves the best NLL. Models with not enough complexity seem to have difficulty predicting a suitable standard deviation.

than two layers are able to achieve comparable performance than models with only two layers.

To investigate the impact of network width on complexity, we conducted a series of experiments using LSTM models with a fixed depth of 2 and varying widths from 16 up to 128. Results are presented in Fig. 7.11.

The optimal model width differed depending on the performance measure used. Specifically, a width of 32 produced the best results for the MAE while a width of 64 was optimal for the NLL. We conclude that models with higher complexity are better suited for probabilistic predictions while models with smaller width focus more on mean prediction. This can be explained by the fact that models with lower complexity are unable to separate the prediction of mean and standard deviation due to limited amount of neurons per layer. This forces them to prioritize mean prediction which results in better performance in terms of MAE but worse performance in terms of NLL.

Overall, models with width below 32 perform poorly. Only the model for largest early stopping achieves a performance in range of other widths. The models with lower early stopping had several weeks with no convergence at all. Training was stopped after only a few epochs since the model was not able to leave the initial local minimum. We conclude that this results from underfitting due to a lack of parameters in each layer. Assuming that each neuron is able to capture only one feature characteristic, a hidden size of 16 is not enough to efficiently analyze all 20 features. Adding dropout to the hidden states amplifies this lack of complexity even more.

In theory, models with a higher width should be able to achieve at least the same performance then less wide models by using less of the available neurons. In practice, the search space used for optimization becomes too large and the model converges to local minima which usually leads to overfitting. Our result reproduced this behaviour where more complex models were not able to achieve comparable performance. Especially a width of 128 lead to overfitting for early stopping of 100.

Comparing the analysis of depth and width, extracting more features with wider models seems more important than enabling richer features using deeper models. For the electricity price features this means, that a lot of feature characteristics are prominent, but dependency on these is rather simple. This is consistent with the feature analysis in Sec. 6.2, where features showed several interactions but dependencies were simple. Most dependency plots showed approximately linear or step-wise behaviour.

7.2.2. Model Performance

We now evaluate the final model performance based on the insights presented above. For a detailed analysis, the best LSTM model was fitted for 4 years, ranging from the 01.01.2019 until 31.12.2022. This allows to validate that the complexity of the model is not only suitable for 2021 but for the general electricity market. The best model in terms of MAE had a depth of 2, a width of 32 and an early stopping parameter of 200.

Figure 7.12 shows the overall performance of the model for 4 years. The real time series of the day-ahead electricity price can be seen in Fig. 7.12a

Table 7.1.: Performance of the LSTM model for several years. For probabibilistic forecasting, the negative log-likelihood (NLL) and accuracy scores for the first and second standard deviation (ACC1, ACC2) are used. For point prediction, the model is evaluated by the mean absolute error (MAE) and the symmetric mean absolute percentage error (SMAPE) using its mean prediction.

	NLL	MAE	SMAPE	ACC 1	ACC 2
2019	2.94	3.73	15.12	0.67	0.93
2020	2.97	3.93	20.71	0.72	0.94
2021	3.83	10.32	15.41	0.76	0.96
2022	5.01	29.85	18.21	0.71	0.94
19-23	3.69	11.92	17.42	0.71	0.94

Table 7.2.: Comparison of the performance of the presented LSTM model and the model of Tschora et al. [Tsc+22] evaluated on 2020 until 2022. The LSTM model outperforms the compared model for both MAE and SMAPE.

	MAE	SMAPE	NLL
This work	7.08	18.09	3.39
Tschora et al.	7.66	18.79	-

plotted in dark blue. The light blue values correspond to the mean predictions of the LSTM model. We observe that the model is able to adapt to the overall pattern and trend of the price time series but seems to miss several extreme events like price peaks or negative prices. Especially for high price peaks, the model is unable to scale the prediction into higher price regions. This may be caused by the rare occurring of these events, which results in an unbalanced dataset according to extreme events. Therefore, the model focuses on more general price scenarios rather than extremes.

In Fig. 7.12b, the MAE and NLL are evaluated for each day and plotted over time to enable a time dependent analysis of the performance. In general, we observe that both performance measures are highly correlated. Compared to the MAE, the NLL seems more stable over the years, which is reasonable due to the fact that the NLL is the loss function used during training. Furthermore, we observe that both measures increase at the end of 2021 simultaneous to the increase of electricity prices. The performance

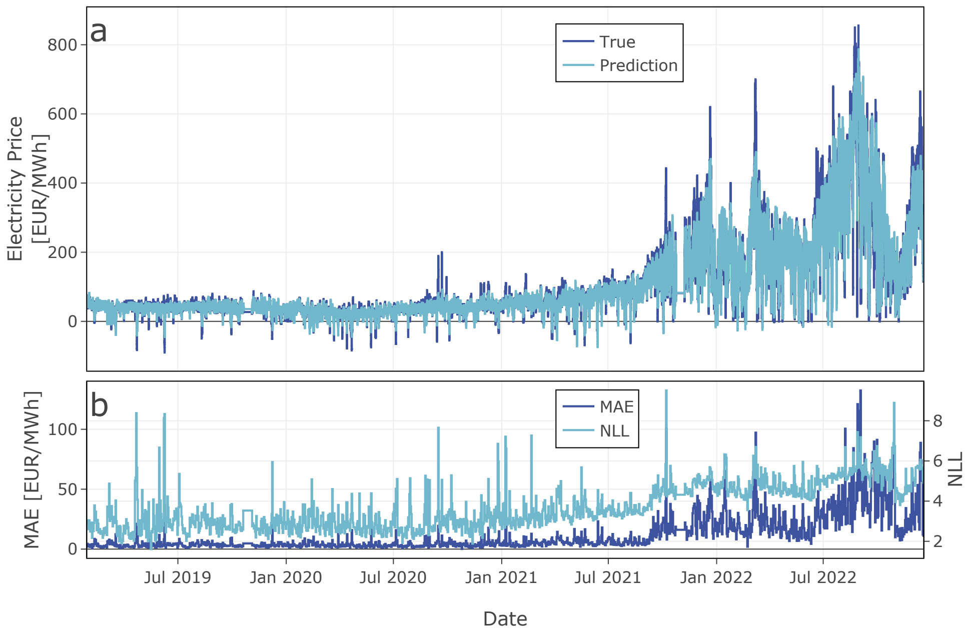

Figure 7.12.: Overall performance of the best LSTM model for price forecasting. (a) Time series of the true day-ahead electricity price (dark blue) and the model's prediction (light blue) from 01.01.2019 to 31.12.2022. (b) Time dependent error measures of the model's prediction. Both measures are evaluated for each day and plotted against the time. The light blue line shows the negative log-likelihood (NLL). The dark blue line corresponds to the mean absolute error (MAE). The model is able to model general price trends but misses some extreme values. Additionally, we observe an increase of both error measures at the end of 2021, corresponding to the European energy crisis.

does not recover to previous levels until the end of the dataset. We suspect that forecasting electricity prices has become more difficult in general due to the European energy crisis and rising inflation.

To improve comparability of the LSTM model to future works, the yearly and overall performance of the model is presented in Tab. 7.1. We included the negative log-likelihood (NLL) used as a loss function and accuracy scores for the first and second standard deviation (ACC1, ACC2) for probabilistic forecasting performance. For pointwise predictions, we also included the mean absolute error (MAE) and the symmetric mean absolute percentage error (SMAPE). We note that the SMAPE does not seem reasonable for electricity prices because of their high amount of almost zero values but is included for completeness.

To confirm the good performance of the LSTM model, we use the model presented by Tschora et al. [Tsc+22] as a benchmark. The presented model is trained on similar features and also uses neural networks. Furthermore, Tschora et al. include prices from CH published before German market closure in the feature set. We did not include this feature, since we wanted the model to predict electricity prices only from their driving features not from a correlated feature. The benchmark was evaluated for 2020 and 2021 at once.

In Tab. 7.2, we present a comparison of the LSTM model to the model of Tschora et al. The LSTM model outperforms the benchmark for both performance measures and achieves an increase in MAE performance by 7.6%. Notably, our model achieved a lower MAE than the model of Tschora et al. despite the fact that we do not use the Swiss price and focus on probabilistic instead of point predictions.

A more detailed analysis of the model prediction is presented in Fig. 7.13. Each panel shows exactly one week of the electricity price time series with the model prediction and its predicted standard deviation. All three weeks are chosen to demonstrate the prediction of rather difficult patterns in the time series. In general, the model follows the daily pattern closely and catches price movements throughout the week. Most values stay within the first standard deviation and almost all values within the second standard deviation. The accuracy scores can be seen in the bottom left corner of each panel.

Figure 7.13a shows a scenario with several following days achieving negative prices. The model is able to predict all four price dips and increases the standard deviation during the dips. This is a strong indicator, that the model utilizes the standard deviation for difficult forecasting scenarios. In Fig. 7.13b, we observe that the model increases the standard deviation not only for negative prices but also for price peaks. Additionally, we see that the

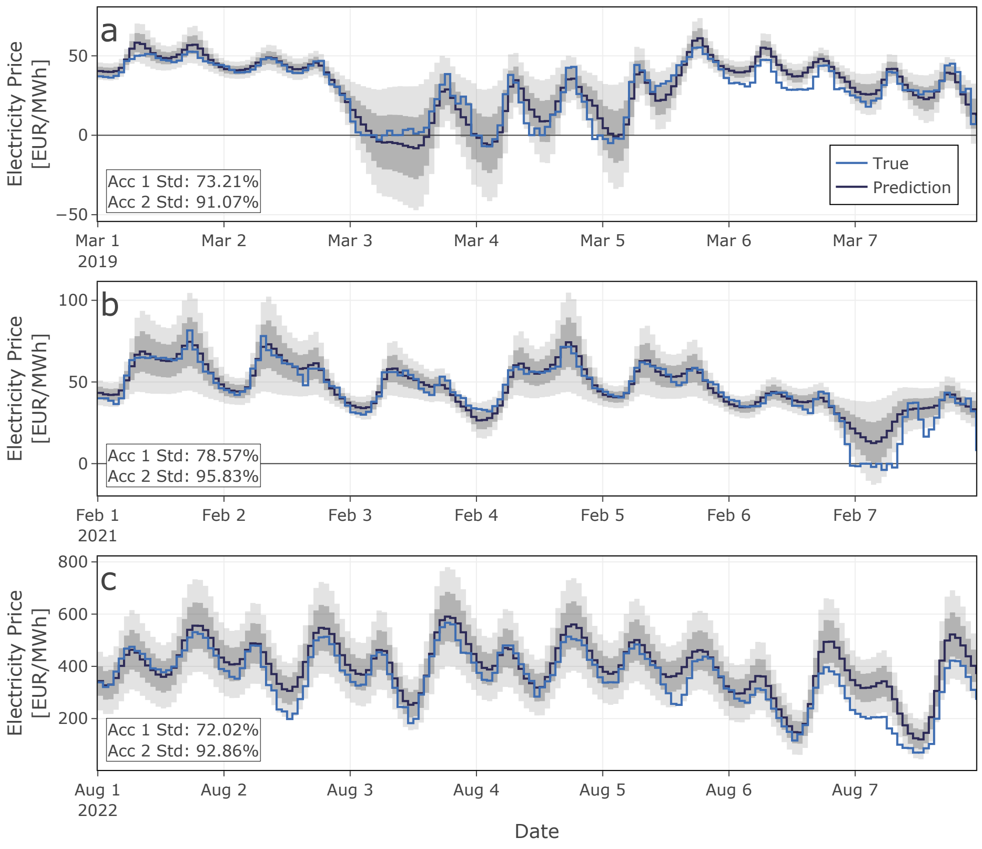

Figure 7.13.: Electricity prices and predicted values of the model for multiple weeks. (a)-(c) Time series of the true day-ahead electricity price (blue) and the model's prediction (dark blue). The grey areas around the prediction corresponds to model's uncertainty. Dark grey shows one standard deviation and light grey two standard deviations. The accuracy of the model for the time period is shown at the bottom left corner of each plots with respect to the standard deviation. We obverse that the model has no problem in forecasting the overall pattern of the electricity price time series. Even small fluctuations are forecasted by the model. Most of the true values lie in the first confidence interval and almost all values in the second.

mean prediction misses the negative prices on the 7th of February but the model was able to increase the standard deviation accordingly. Figure 7.13c presents a week of generally high prices after the energy crises. The model adapts to the high price regimes but overestimates prices over almost one day at the end of the week.

We continue the analysis of the model by focusing on the forecasted standard deviation. For this, we only use the prediction of the model for 2021 to simplify analysis. Prices during 2021 increased at the end of the year, changing also the market behaviour due to the European energy crisis. As expected, the standard deviation increases as well after the start of the energy crisis (see Fig. 7.14a).

Earlier results in Fig. 7.13 showed that the standard deviation not only increases for higher prices but also for prices close to zero or negative prices. The week from 1st to 9th of February can be seen in Fig. 7.14b with the corresponding value of the predicted standard deviation presented in Fig. 7.14c. Figure 7.14d confirms the dependency on the price for both negative price and high prices.

The standard deviation of the model follows a piece-wise linear function in dependence of the predicted price centered around 50 EUR/MWh (see Fig. 7.14d). Nevertheless, scattering within the curve suggest that the model is not only setting the standard deviation according to the predicted price but according to the predictability of the input sample. The same effect can be seen in the section of the price time series presented in Fig. 7.14b+c. The predicted price at the 4th of February and the 6th of February are within the same range, while the standard deviation for the second prediction is substantially higher. The model is able to adapt the standard deviation if the probability for low prices is higher than usual.

Using the predicted standard deviation for electricity price forecasting enables further strategies compared to normal point predictions. Forecasts can be seen with respect to the uncertainty of the model and evaluated with an accuracy score. The accuracy score of the model (see Tab. 7.1) is stable throughout all years which makes the model uncertainty reliable during all price regimes. Overall, around 75% of the true values lay within the first standard deviation of the model. Furthermore, almost all true values lay withing the second standard deviation of the model, achieving a accuracy score of almost 95%.

Overall, the LSTM model is able to outperform the DNN model of Tschora et al. [Tsc+22] while enabling the use of probabilistic forecasting. LSTM models are well suited to electricity price forecasting because they can deal with time-dependent patterns in features. They also mitigate the need for an exhaustive search for optimal lagged feature values, as they are able to self-select important information from the full time series.

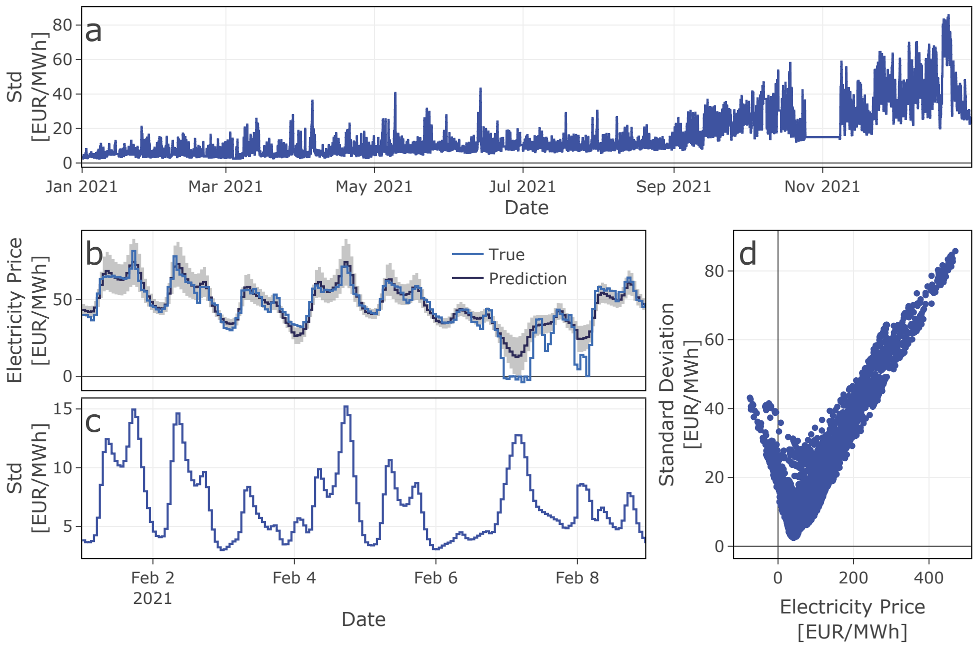

Figure 7.14.: The use of probabilistic forecast for electricity prices with focus on the predicted standard deviation. (a) Predicted standard deviation over time from 01.01.2021 until 01.01.2022. (b) Time series of the electricity price and the prediction of the model from 01.02.2021 to 09.02.2021. (c) Predicted standard deviation over the same time period as in b. (d) Predicted standard deviation against the predicted electricity price. The standard deviation depends on the predicted electricity price. Nevertheless, it enables additional information about the uncertainty of the model, especially for negative prices.

8. Conclusion

In this thesis, we analyzed the German day-ahead electricity market. We studied the impact of different features on electricity prices and developed a forecasting model to accurately predict future prices. We employed two different machine learning techniques to achieve these objectives. We used XAI methods for explaining market behaviour and LSTM models for forecasting electricity prices.

In the first part of this thesis, we have developed a machine learning model based on gradient boosted trees and demonstrated how it accurately estimates electricity prices, outperforming a single-feature benchmark model based on a common approximation of the merit order principle. Using SHAP to interpret our black-box model, we obtained deeper insights into the characteristics of the day-ahead market. SHAP values quantify how the price depends on the input features and thus reveal drivers beyond the benchmark model. Our analysis confirmed that high load leads to high prices, while large shares of wind or solar generation reduces prices. Beyond these expected dependencies, the model quantified the role of fuel prices, imports and exports, as well as load and generation ramps.

We saw that the SHAP analysis of our model is limited when it comes to a deeper causal interpretation of feature impacts. Nevertheless, the SHAP values provide detailed insights into the working of the model by revealing how and which features are mostly used and by quantifying dependencies and interactions. Only by including domain knowledge of market mechanisms and power systems we can hypothesize to causal relations. Estimating causal models directly from data is in principle possible, e.g. using Causal SHAP [Hes+20] or causal representation learning [Sch+21] but requires more explicit assumptions about the underlying causal structure than we wanted to employ in this first exploratory study.

Concluding the market analysis using XAI, we have demonstrated the usefulness of XAI models to analyze electricity price dynamics in the German market. These insights may contribute to the improvement of mechanistic models of electricity markets as well as data-driven forecasting models by identifying the relevant features to be included [Cra+22]. For instance, our results suggest slightly different roles of wind and solar power, while they enter the residual load equally. Furthermore, the SHAP analysis quantifies the role of generation ramps, which are subject to strong feature interactions.

There remain many open questions and starting points for further research. It would be interesting to investigate how XAI price models differ between electricity markets in different countries. Similarly, XAI models may also be used to compare markets at different times to quantify changes. For instance, the phase-out of nuclear power in Germany or changes of the regulatory framework should impact the dependencies of prices and features. Furthermore, XAI methods may also be used to analyze the impact of exceptional events such as the energy crisis after the Russian invasion of Ukraine once sufficient data is available.

In the second part of the thesis, we developed a proof of concept using long short-term memory (LSTM) models to forecast electricity prices accurately. We analyzed different model complexities by varying the depth and width of the LSTM networks. We obtained a detailed overview of how different levels of complexity of the LSTM models affect the performance of forecasting electricity prices.

We found that LSTM models do not need high complexity to successfully forecast electricity prices. Models with higher complexity seem unable to adapt weights for the limited training data available. In particular, models with depth of 3 or higher showed no comparable performance even for higher early stopping. Furthermore, early stopping can improve performance but does not need to be optimized carefully. Regularization through dropout and early stopping seems to prevent overfitting very efficiently.

The best LSTM model was able to outperform comparable models on similar data. The model achieved excellent point predictions of the electricity prices while enabling the use of a predicted standard deviation. On small time scales, the model is able to follow the price time series very closely and is able to stay within its standard deviation most of the time. Overall, we saw that electricity price forecasting has become increasingly difficult since the start of the European energy crisis.

Our analysis of the predicted standard deviation showed that probabilistic forecasting can give more useful forecasts. Even though the uncertainty is directly correlated with the overall electricity price, we gain valuable information from it. Usually, prices close to zero and below are hard to predict but lay within the standard deviation of the model. Probabilistic forecasting is especially valuable in the energy industry, where accurate but also certain forecasts can help to make decisions regarding allocation, pricing strategies and energy trading. Using uncertainty allows for more robust forecasting and prevents possible wrong decision based on imprecise predictions.

Concluding the second part, we showed that LSTM models are able to outperform comparable models when forecasting electricity prices. The main advantage of LSTM models is their ability to self-select important information from the full time series. This facilitates the exhaustive search for optimal lagged feature values where each feature combination needs to be tested in advance. Furthermore, the prediction of a standard deviation enables additional use cases for the forecasting models.

There are many unanswered questions and directions for future research on LSTM models and probabilistic forecasting. Starting with the data, we could deploy additional methods to improve the quality of the dataset. The feature set could be improved by analyzing the importance of current features or introducing new features based on system knowledge, e.g. generation and transmission capacities, storage capacities or economic features like inflation rates or stock market development. Existing features could also be improved by deploying more sophisticated methods for interpolation and shifting. Fuel prices for example are shifted by one day but could be shifted to the exact times when information is available for the day-ahead market. The same goes for lagging prices. Furthermore, the data could be normalized using more advanced techniques, e.g. methods that also normalize the volatility.

For the LSTM models, several hyperparameters were not optimized in this study, which could further improve their performance. Additional research for an optimal sequence length could reduce computation time significantly while keeping the same performance or improving performance. Using shorter recalibration intervals could also lead to better performance. Optimizing dropout, learning rate or other parameters for the learning process would probably result in better LSTM models when combined with optimal values for depth, width and early stopping. Overall, hyperparameters could also be optimized individually for each recalibration period instead of using the same parameters for the entire period.

Furthermore, the output of the LSTM models could be improved by investigating the added value of the simultaneous forecasting of all 24 hours. The prediction of the standard deviation could be improved by allowing for more complex probability distribution.

In summary, LSTM models could be used as powerful forecasting methods for electricity prices but should be analyzed in more detail to ensure optimal utilization of the available resources.

Overall, our research provides valuable insights into the factors that drive electricity prices and presents a methodology for accurate price forecasting. The combination of XAI and LSTM models provides a comprehensive framework for analyzing and predicting electricity prices. By utilizing this methodology, energy companies and policymakers can make informed decisions that improve energy market efficiency and benefit consumers.

A. Country Codes

Code	Country
AT	Bidding zone of Austria
BE	Bidding zone of Belgium
CH	Bidding zone of Swiss
CZ	Bidding zone of Czech Republic
DE-AT-LU	Shared bidding zone of Germany, Austria and Luxembourg
DE-LU	Shared bidding zone of Germany and Luxembourg
DK	Bidding zone of Denmark
FR	Bidding zone of France
NL	Bidding zone of the Netherlands
PL	Bidding zone of Poland
NO	Bidding zone of Norway
SE	Bidding zone of Sweden

Bibliography

[AB18] A. Adadi and M. Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2018.2870052`.

[Agg+18] C. C. Aggarwal et al. "Neural networks and deep learning". In: *Springer* 10.978 (2018), p. 3.

[Ahm+21] T. Ahmad et al. "Artificial intelligence in sustainable energy industry: Status Quo, challenges and opportunities". In: *Journal of Cleaner Production* 289 (Mar. 2021), p. 125834. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2021.125834`.

[ALL20] ALL NEMO COMMITTEE. *Euphemia Public Describtion*. Tech. rep. `https://www.nemo-committee.eu/assets/files/euphemia-public-description.pdf`. Oct. 2020.

[ALL22] ALL NEMO COMMITTEE. *About the All NEMO Committee*. 2022. URL: `https://www.nemo-committee.eu/nemo_committee` (visited on 02/08/2023).

[ARI22] ARIVA.DE AG. *ARIVA.DE*. `https://www.ariva.de/`. 2022. (Visited on 04/26/2022).

[ATM21] G. Alova, P. A. Trotter, and A. Money. "A machine-learning approach to predicting Africa's electricity mix based on planned power plants and their chances of success". In: *Nature Energy* 6.2 (Feb. 2021), pp. 158–166. ISSN: 2058-7546. DOI: `10.1038/s41560-020-00755-9`.

[Bar+20] A. Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (June 2020), pp. 82–115. ISSN: 1566-2535. DOI: `10.1016/j.inffus.2019.12.012`.

[Böt+23] P. C. Böttcher et al. "Initial analysis of the impact of the Ukrainian power grid synchronization with Continental Europe". In: *Energy Advances* 2.1 (2023), pp. 91–97.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[CG16] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. Association for Computing Machinery, Aug. 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: `10.1145/2939672.2939785`.

[Che+19] M. Chen et al. "XGBoost-Based Algorithm Interpretation and Application on Post-Fault Transient Stability Status Prediction of Power System". In: *IEEE Access* 7 (2019), pp. 13149–13158. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2019.2893448`.

[CKS19] J. L. Cremer, I. Konstantelos, and G. Strbac. "From Optimization-Based Machine Learning to Interpretable Security Rules for Operation". In: *IEEE Transactions on Power Systems* 34.5 (Sept. 2019), pp. 3826–3836. ISSN: 1558-0679. DOI: `10.1109/TPWRS.2019.2911598`.

[Cra+22] E. Cramer et al. "Multivariate Probabilistic Forecasting of Intraday Electricity Prices using Normalizing Flows". In: *arXiv preprint arXiv:2205.13826* (2022).

[EEX] EEX. *EEX EUA Primary Auction Spot*. URL: `https://www.eex.com/de/marktdaten/umweltprodukte/eex-eua-primary-auction-spot-download` (visited on 02/23/2023).

[EEX18] EEX. *Vorbereitung der Trennung der Stromgebotszone zwischen Deutschland und Österreich.* https://www.eex.com/fileadmin/EEX/Downloads/Products/Power/Phelix-DE/20180904-customer-information---bidding-zone-split-data.pdf. 2018.

[ENTa] ENTSO-E. *Day-ahead forecast of the total load per market time unit.* URL: https://transparency.entsoe.eu/content/static_content/Static%20content/knowledge%20base/data-views/load-domain/Data-view%20Total%20Load%20-%20Day%20Ahead%20-%20Actual.html (visited on 02/08/2023).

[ENTb] ENTSO-E. *ENTSO-E Transparency Platform.* https://transparency.entsoe.eu/. (Visited on 02/23/2023).

[ENTc] ENTSO-E. *Generation Forecasts for Wind and Solar.* URL: https://transparency.entsoe.eu/content/static_content/Static%20content/knowledge%20base/data-views/generation/Data-view%20Generation%20Forecasts%20-%20Day%20Ahead%20for%20Wind%20and%20Solar.html (visited on 02/08/2023).

[ENTd] ENTSO-E. *Installed Capacity per Production Type.* URL: https://transparency.entsoe.eu/content/static_content/Static%20content/knowledge%20base/data-views/generation/Data-view%20Installed%20Capacity%20per%20Production%20Type.html (visited on 02/23/2023).

[ENTe] ENTSO-E. *Single Day-ahead Coupling (SDAC).* URL: https://www.entsoe.eu/network_codes/cacm/implementation/sdac/ (visited on 02/09/2023).

[ENTf] ENTSO-E. *Transparency Platform RESTful API - user guide.* URL: https://transparency.entsoe.eu/content/static_content/Static%5C%20content/web%5C%20api/Guide.html (visited on 02/21/2023).

[ENTg] ENTSO-E. *Unavailability of Production and Generation Units.* URL: https://transparency.entsoe.eu/content/static_content/Static%20content/knowledge%20base/data-views/outage-domain/Data-view%20Unavailability%20of%20Pr oduction%20and%20Generation%20Units.html (visited on 02/23/2023).

[ENT23] ENTSO-E. *Enhanced Load Forecasting*. https://www.entsoe.eu/Technopedia/techsheets/enhanced-load-forecasting. 2023. (Visited on 02/13/2023).

[EPEa] EPEX SPOT. *European Market Coupling | EPEX SPOT*. URL: https://www.epexspot.com/en/marketcoupling (visited on 02/10/2023).

[EPEb] EPEX SPOT. *Market Data | EPEX SPOT - Intraday Volume DE-LU*. URL: https://www.epexspot.com/en/market-data?market_area=DE&trading_date=&delivery_date=2023-03-03&underlying_year=&modality=Continuous&sub_modality=&technology=&product=15&data_mode=table&period=&production_period= (visited on 03/04/2023).

[EPE22] EPEX SPOT. *Single Day-Ahead Coupling (SDAC)*. Tech. rep. June 2022. URL: https://www.epexspot.com/en/downloads.

[Eur19] European Power Exchange (EPEX SPOT). *Annual Report 2019*. https://www.epexspot.com/sites/default/files/sites/catalogue/catalogue (accessed 28.10.2022). 2019.

[Eur22] European Union Agency for the Cooperation of Energy Regulators (ACER). *List of currently designated NEMOs*. https://www.acer.europa.eu/electricity/market-rules/capacity-allocation-and-congestion-management/implementation/designation-of-nemos. 2022.

[EXA] EXAA. *Trading with EXAA*. URL: https://www.exaa.at/en/energytrading/handel-mit-exaa/ (visited on 02/07/2023).

[Fed] Federal Reserve Bank of St. Louis. *Federal Reserve Economic Data | FRED | St. Louis Fed*. URL: https://fred.stlouisfed.org/ (visited on 02/23/2023).

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[Ger14] German Bundestag. *Gesetz für den Ausbau erneuerbarer Energien (Erneuerbare Energien-Gesetz – EEG 2021)*. `http://www.gesetze-im-internet.de/eeg_2014/EEG_2021.pdf`. 2014.

[Han+22a] C. Han et al. "Complexity and persistence of price time series of the european electricity spot market". In: *PRX Energy* 1.1 (2022), p. 013002.

[Han+22b] L. Hanny et al. "On the progress in flexibility and grid charges in light of the energy transition: The case of Germany". In: *Energy Policy* 165 (2022), p. 112882.

[Hes+20] T. Heskes et al. "Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4778–4789.

[HMB18] L. Hirth, J. Mühlenpfordt, and M. Bulkeley. "The ENTSO-E Transparency Platform–A review of Europe's most ambitious electricity data platform". In: *Applied Energy* 225 (2018), pp. 1054–1067.

[Hoc91] S. Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". In: *Diploma, Technische Universität München* 91.1 (1991).

[HS96] S. Hochreiter and J. Schmidhuber. "LSTM can solve hard long time lag problems". In: *Advances in neural information processing systems* 9 (1996).

[HS97] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[Inv] Investing.com. *Dutch TTF Natural Gas Futures Price*. URL: `https://www.investing.com/commodities/dutch-ttf-gas-c1-futures` (visited on 02/23/2023).

[ISOa] ISO. *ISO - Glossary for ISO 3166*. URL: https://www.iso.org/glossary-for-iso-3166.html (visited on 02/21/2023).

[ISOb] ISO. *Online Browsing Platform (OBP)*. URL: https://www.iso.org/obp/ui/#search (visited on 02/21/2023).

[JP05] T. Jamasb and M. Pollitt. "Electricity market reform in the European Union: review of progress toward liberalization & integration". In: *The Energy Journal* 26.Special Issue (2005).

[Kag] Kaggle. *State of Data Science and Machine Learning 2022*. URL: https://www.kaggle.com/kaggle-survey-2022 (visited on 03/14/2023).

[KB14] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[Ke+17] G. Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in Neural Information Processing Systems* 30 (2017).

[KSW21] J. Kruse, B. Schäfer, and D. Witthaut. "Revealing drivers and risks for power grid frequency stability with explainable AI". In: *Patterns* 2.11 (Nov. 2021), p. 100365. ISSN: 2666-3899. DOI: 10.1016/j.patter.2021.100365.

[Küh+21] J. Kühling et al. *8th Energy Sector Report of the Monopolies Commission*. Tech. rep. https://www.monopolkommission.de/images/PDF/SG/chapter-3-competition-between-electricity-exchanges.pdf. Monopolkommission, 2021. Chap. Competition between electricity exchanges in Germany.

[Lag+21] J. Lago et al. "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark". In: *Applied Energy* 293 (2021), p. 116983.

[LB21] W. Li and D. M. Becker. "Day-ahead electricity price prediction applying hybrid models of LSTM-based deep learning methods and feature selection algorithms under consideration of market coupling". In: *Energy* 237 (2021), p. 121543.

[LDD18] J. Lago, F. De Ridder, and B. De Schutter. "Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms". In: *Applied Energy* 221 (2018), pp. 386–405.

[Lij07] M. G. Lijesen. "The real-time price elasticity of electricity". In: *Energy economics* 29.2 (2007), pp. 249–258.

[Lun+20] S. M. Lundberg et al. "From local explanations to global understanding with explainable AI for trees". In: *Nature machine intelligence* 2.1 (2020), pp. 56–67.

[Mac+22] R. Machlev et al. "Explainable Artificial Intelligence (XAI) techniques for energy and power systems: Review, challenges and opportunities". In: *Energy and AI* 9 (Aug. 2022), p. 100169. ISSN: 2666-5468. DOI: `10.1016/j.egyai.2022.100169`.

[Mara] MarketWatch. *MTFC00 | Coal (API2) CIF ARA (ARGUS-McCloskey) Continuous Contract Overview*. URL: `https://www.marketwatch.com/investing/future/mtfc00` (visited on 02/23/2023).

[Marb] MarketWatch. *NG00 | Natural Gas Continuous Contract Overview*. URL: `https://www.marketwatch.com/investing/future/ng00` (visited on 02/23/2023).

[Marc] Marktstammdatenregister. *Aktuelle Einheitenübersicht | MaStR*. URL: `https://www.marktstammdatenregister.de/MaStR/Einheit/Einheiten/OeffentlicheEinheitenuebersicht` (visited on 02/23/2023).

[Mil+18] F. Milano et al. "Foundations and challenges of low-inertia systems". In: *2018 power systems computation conference (PSCC)*. IEEE. 2018, pp. 1–25.

[ML22] G. Mitrentsis and H. Lens. "An interpretable probabilistic model for short-term solar power forecasting using natural gradient boosting". In: *Applied Energy* 309 (Mar. 2022), p. 118473. ISSN: 0306-2619. DOI: `10.1016/j.apenergy.2021.118473`.

[Mol20] C. Molnar. *Interpretable machine learning*. 2020.

[MP43] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[N-S] N-SIDE. *N-SIDE makes pan-European, single-day-ahead coupling possible with EUPHEMIA*. URL: `https://energy.n-side.com/resources/case-studies/euphemia` (visited on 02/09/2023).

[Nie15] M. A. Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA, 2015.

[Nor] Nord Pool. *Day-ahead market*. URL: `https://www.nordpoolgroup.com/en/the-power-market/Day-ahead-market/` (visited on 02/08/2023).

[OČ22] J. Osička and F. Černoch. “European energy politics after Ukraine: The road ahead”. In: *Energy Research & Social Science* 91 (2022), p. 102757.

[Ola15] C. Olah. “Understanding lstm networks”. In: (2015).

[Pas+19] A. Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[PB] J. Pecinovsky and F. Boerman. *entsoe-py*. URL: `https://github.com/EnergieID/entsoe-py` (visited on 02/23/2023).

[Pea00] J. Pearl. *Causality*. Cambridge university press, 2000.

[Pra16] A. Praktiknjo. “The value of lost load for sectoral load shedding measures: The German case with 51 sectors”. In: *Energies* 9.2 (2016), p. 116.

[Rin19] S. Rinne. “Radioinactive: Do nuclear power plant outages in France affect the German electricity prices?” In: *Energy Economics*. Eighth Atlantic Workshop on Energy and Environmental Economics 84 (Oct. 2019), p. 104593. ISSN: 0140-9883. DOI: `10.1016/j.eneco.2019.104593`. URL: `https://www.sciencedirect.com/science/article/pii/S0140988319303883`.

[Rog+15] J. Rogelj et al. "Energy system transformations for limiting end-of-century warming to below 1.5 C". In: *Nature Climate Change* 5.6 (2015), pp. 519–527.

[Ros+20] R. Roscher et al. "Explainable machine learning for scientific insights and discoveries". In: *IEEE Access* 8 (2020), pp. 42200–42216.

[Ros58] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[Ryd+20] L. Rydin Gorjão et al. "Open database analysis of scaling and spatio-temporal properties of power grid frequencies". In: *Nature communications* 11.1 (2020), p. 6362.

[Sch+21] B. Schölkopf et al. "Toward causal representation learning". In: *Proceedings of the IEEE* 109.5 (2021), pp. 612–634.

[Sha+53] L. S. Shapley et al. "A value for n-person games". In: (1953).

[SM19] R. C. Staudemeyer and E. R. Morris. "Understanding LSTM–a tutorial into long short-term memory recurrent neural networks". In: *arXiv preprint arXiv:1909.09586* (2019).

[SP18] I. Staffell and S. Pfenninger. "The increasing impact of weather on electricity supply and demand". In: *Energy* 145 (2018), pp. 65–78.

[SRG08] F. Sensfuß, M. Ragwitz, and M. Genoese. "The merit-order effect: A detailed analysis of the price effect of renewable electricity generation on spot market prices in Germany". In: *Energy policy* 36.8 (2008), pp. 3086–3094.

[Sri+14] N. Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[Tre+23] J. Trebbien et al. "Understanding electricity prices beyond the merit order principle using explainable AI". In: *Energy and AI* (2023), p. 100250.

[Tsc+22] L. Tschora et al. "Electricity price forecasting on the day-ahead market using machine learning". In: *Applied Energy* 313 (2022), p. 118752.

[VD15] K. Van den Bergh and E. Delarue. "Cycling of conventional power plants: Technical limits and actual costs". In: *Energy Conversion and Management* 97 (2015), pp. 70–77.

[VL10] E. Van der Vleuten and V. Lagendijk. "Transnational infrastructure vulnerability: The historical shaping of the 2006 European "Blackout"". In: *Energy Policy* 38.4 (2010), pp. 2042–2052.

[Wit+22] D. Witthaut et al. "Collective nonlinear dynamics and self-organization in decentralized power grids". In: *Reviews of modern physics* 94.1 (2022), p. 015005.

[WWS13] A. J. Wood, B. F. Wollenberg, and G. B. Sheblé. *Power Generation, Operation, and Control*. 3rd. John Wiley & Sons, Hoboken, New Jersey, 2013. ISBN: 978-0-471-79055-6.

[Zak+22] B. Zakeri et al. "Energy Transitions in Europe–Role of Natural Gas in Electricity Prices". In: *Available at SSRN* (2022). URL: `http://dx.doi.org/10.2139/ssrn.4170906`.

Acknowledgments

Many thanks to Dirk for the opportunity to write my thesis in your working group. The last year was definitely the best of the whole degree. Thank you for the exceptional supervision and the many hours you sacrificed for me. Your family-like and personal guidance of your working group is outstanding and creates a relaxed yet very productive atmosphere.
Many thanks to Leo for the productive and supportive collaboration. Thanks also for agreeing to do the second correction without hesitation.
Special thanks to Benjamin, your supervision has helped me a lot in my scientific development. Thank you for the many hours of discussion and advice. Your writing coaching will probably shape me forever.
Many thanks to Maurizio for the friendly support and supervision during my thesis. Above all, thank you for your patience with me in the last weeks before submission.
Thank you Sebastian for the constant exchange during the master thesis and the years before. I hope to continue our productive work and friendship.
Also, thanks to Hannes for his help during the last weeks before submission. Thanks for the discussions and the many proofreading sessions.
Thanks to Philipp for his willingness to help with topics of all kinds. I would also like to thank the entire working group for the great time, I felt very comfortable right from the start.
Zum Schluss bedanke ich mich noch bei meiner Familie, besonderer Dank gilt meinen Eltern für ihre bedingunglose Unterstützung während meines gesamten Studiums. Ihr hattet stets ein offenes Ohr für mich und habt mir in den schwierigsten Phasen meines Lebens stets zur Seite gestanden.
Danke Fiona, dass du auch in den stressigen Phasen immer für mich da warst. Schön, dass es dich gibt.