Unterschrift BetreuerIn

DIPLOMARBEIT

Household Energy Consumption Forecasting using Recurrent Neural Networks

ausgeführt am Institut für
Angewandte Physik
der Technischen Universität Wien

unter der Anleitung von

Ao.Univ.Prof.Dr. Martin Gröschl

in Kooperation mit dem
Austrian Institute of Technology
Electric Energy Systems

Dr. Mark Stefan

durch

Thomas Brunnhofer

Wien,
Dezember 12, 2019

Unterschrift StudentIn

Contents

List of Figures

List of Tables

Abstract

The energy sector is transforming rapidly. The use of distributed renewable energy sources is fostering the development of more decentralized energy grids. Optimally using those distributed energy sources in combination with energy storage systems is the challenge of the future. Advanced metering infrastructure is monitoring and recording the grid state and the energy consumption of its participants. Making use of those energy recordings for optimizing the functionality of decentralized grids is a major objective in the energy sector.
This thesis investigates the usage of machine learning techniques –especially recurrent neural networks– for the task of energy consumption forecasting by utilizing smart meter recordings of individual households.
Good short term forecasts of energy consumption are a vital part in the overall optimization of decentralized grids –in the form of smart grids and local energy communities–. The task, however, is challenging due to the volatility of a single household's energy consumption.
By comparing various neural network architectures with benchmarks and studying the effect of data granularity and exogenous features, this thesis is part of the growing research in the realm of short term energy consumption forecasting.
Recurrent neural networks were found to be slightly advantageous; as is the approach for using dedicated models for single households. Using the exogenous features did not result in better forecasts.

Zusammenfassung

Der Energiesektor unterliegt einem Wandel. Die optimale Nutzung von dezentralen und erneuerbaren Energiequellen ist eine der großen Herausforderungen für das Energienetz der Zukunft. Die Entwicklung hin zu effizienten dezentralen Smart-Grids benötigt die optimale Nutzung von dezentraler Energieerzeugung gekoppelt mit Energiespeichern. Smart Meter –intelligente Stromzähler– überwachen den Netzstatus und werden für das Aufzeichnen der Energieverbrauchsdaten eingesetzt. Analysieren dieser Energieverbrauchsdaten kann beim optimalen Betrieb von Smart-Grids unterstützen.
In dieser Arbeit werden Machine Learning Techniken –insbesondere Recurrent Neural Networks– verwendet, um mithilfe von Energieverbrauchszeitreihen, möglichst genaue Vorhersagen des Energieverbrauchs von Einzelhaushalten zu bekommen.
Gute Vorhersagen des Energieverbrauchs sind ein wichtiger Bestandteil für die Optimierung und Ausführung von dezentralen Verbundnetzen. Präzise Vorhersagen für den Energieverbrauch von Einzelhaushalten ist eine große Herausforderung, besonders wegen der hohen Volatilität der Verbrauchszeitreihen.
Diese Arbeit vergleicht unterschiedliche Neuronale Netze, und untersucht den Effekt von Zeitauflösung sowie zusätzlicher Information, auf die Qualität der Verbrauchsvorhersagen.
Recurrent Neural Networks waren die besten Modelle hinsichtlich der Verbrauchsprognosen. Weiters scheint der Ansatz –für einzelnen Haushalte spezifische Modelle zu trainieren– vorteilhaft. Keine Vorteile wurden gefunden für das Nutzen zusätzlicher Information.

Acknowledgment

I want to thank my supervisor Dr. Mark Stefan as well as Dr. Friederich Kupzog from the Austrian Institute of Technology, who gave me the opportunity to write my diploma thesis in a vibrant and supportive working environment. Especially, thanks to Dr. Mark Stefan, who was always available for feedback and guidance. His helpful tips on approaching the writing process of a diploma thesis helped me immensely for not getting lost during the –sometimes overwhelming– writing process. I can only wish for other students to have a generous and experienced mentor as I did with Dr. Mark Stefan.
I am very grateful to Dip. Ing. Bernadette Fina for being incredibly generous with her time in answering my questions about structuring my thesis. I benefited enormously from her tips, templates and encouraging words –especially when re-writing the Related Work chapter the third time–. I can only aspire to one day write elegant scientific papers as she does.
Special thanks to all the encouraging colleagues, I was fortunate to work with, during my time at the AIT. Some of them already helped me a lot during our time together at Vienna University of Technology and then again at the AIT.

Next, I want to thank my supervisor at the University Prof.Dr. Martin Gröschl for his patience and guidance during the last few months. His helpful attitude always gave me the impression that I am capable of writing a decent thesis. Looking back at my time at the university I can say, it's those kinds of Professors who make studying at the University a rewarding experience.

Special thanks to my girlfriend Julia, who was as enthusiastic as me, when I got the opportunity to do my diploma thesis at the AIT. The difference is, that she stayed enthusiastic throughout the months and was an enormous support in times when I started doubting the quality of my work. She kept encouraging me, giving me confidence, and even put the prospect of a motivating reward in front of me.

Finally, I want to thank my family, father, mother, and sister for consistently encouraging me throughout the last few months. Believing in my abilities and providing helpful tips and perspectives for the process of finishing the thesis.
Thank you all.

Chapter 1

Introduction

The energy sector is transforming. The penetration of renewable energies, not only on the high grid-level but also on the low level of the individual consumer is increasing. Through the installation of photovoltaic modules, using the radiation of the sun to generate electric energy on private properties, households will increasingly produce there own share of electricity, using its energy output in times of demand and taking part in the energy market in times of no self-consumption by selling its generated energy to the grid. The term *prosumers* characterizes the idea of individual customers not only buying from the grid for their energy needs, but taking an active role in the energy market by generating and providing.

Electric vehicles are becoming widely adopted and pose another challenge to the grid of the future. The electric power needed for electric vehicles is high and especially speed charging can jeopardize the stability of the grid, especially at times of peak demand. Restricting the high power loading curves is a common practice to circumvent stresses to the grid. This restriction of loading behaviour, however, can result in a dissatisfaction of the customers, leading to lower adoption of electric transport technologies, which might play a major role in the global movement to reduce greenhouse gas emissions and move the transportation sector in a more sustainable direction.

Heat pumps and solar thermal installations are further parts in the overall energy mix, which will play an increasing role in the coming years. To deal with these developments in the grid, innovations, especially in the field of energy storage, are of importance. Battery Storage Systems are developed to store energy in times of over-generation and provide the stored energy in times where the generation can not balance the demand. Energy storage might be especially useful when combined with photovoltaic modules, where there are relative predictable day and night cycles, that might allow for optimal charging and discharging schedules of the combined system. All of this development and innovation is driven by the imperative objection to drastically reduce greenhouse gas emissions and move the global economy and lifestyles of billions of people towards a more sustainable way of operation.

An important part of the overall evolution of the energy system is the forming of more decentralized grids. The growing penetration of renewable energy sources on every level of the grid, not only onto the High Voltage Grid –through wind farms, hydropower plants or solar power plants–, but also smaller generations at the household level in form of photovoltaic or geothermal systems, are increasing the need for a more decentralized energy system management. The evolution of the present grid is, therefore, trending towards smaller scale grids, which can be seen as enclosed grids, with the idea of virtual power plants orchestrating the generation from distributed energy sources on the low voltage level. A major challenge of moving to a more decentralized way of operation is the efficient use of the numerous distributed energy sources.

This development is going hand in hand with the increase of sensory input that is derived from the grid itself. The spread of *Advanced Metering Infrastructure* is taking place in grids around the world, with the goal of gathering information of grid parameters on every level, for making informed decisions about grid stability, grid operations, and future grid modifications. *Smart Meters* are part of this advanced metering infrastructure. The major objection of a *Smart Meter* is to monitor (record and save) the energy consumption of the circuit it is deployed on. In Austria, smart meters are deployed on about 80% of households, with the goal of getting the percentage of households connected to a smart meter device up to 95% by the year of 2022 according to e-control Austria [1]. Furthermore, European energy policies are targeting the roll-out of smart metering devices all over Europe, with the goal of achieving 80% coverage by the year of 2020 [2]. This increase in measurements of electric energy demand on every grid level yields huge amounts of recorded data. The goal is to use these measurements to drive informed decisions and take actions that advance the grid into a more sustainable and efficient mode of operation.
Machine Learning can be used to get insights from this huge amount of recorded smart meter and sensor readings. Varying machine learning approaches can be deployed, to analyse the recordings at different voltage levels and with various objectives in mind. Found insights can then help with the optimal scheduling and incorporation of emerging technologies, playing a vital role in transforming the grid. Following operations are especially valuable for an efficient grid:

- Load Scheduling/Demand Response Algorithms
- Managing Charge/Discharge cycles for Energy Storage Systems
- Flexibility Management (Home Energy Management Systems/vehicle-to-grid technologies)
- Energy Consumption Forecasts
- Anomaly Detection in the Grid

Almost all of these mentioned control schemes, in terms of demand response or grid management, require short-term electricity demand forecasts to aid in informed decision making.
This thesis investigates the suitability of a special kind of machine learning methodology, namely *Recurrent Neural Networks* for a day ahead energy consumption forecast of individual households. This work, therefore, partakes in the research of analysing different machine learning approaches for the task of energy consumption forecasting on the low voltage level. Accurate forecasts of energy demand play a crucial part in the goal of making the energy system more stable, more reliable and more efficient.

This thesis is structured as follows: Chapter 2 describes the motivation of why forecasting is important and the challenges with this task. Forecasts play an important role in the overall optimization of energy communities. Why they are useful for utilities, end-consumers, and the environment is mentioned in more detail. Chapter 3 gives an overview of the existing work and the state of the art in the field of forecasting energy consumption. The aspects in which this thesis is expanding the current research field are explored, and the main questions this thesis sets out to answer are motivated. In chapter 4 the theory behind the forecasting methodology of neural networks used in this work is described. Chapter 5 explains the experimental setup for the task of forecasting short term energy consumption of individual households. The data of the household recordings, as well as the data processing steps, are discussed. The feature engineering steps, as well as how the data is processed into input and target pairs is explained. After the theoretical background in chapter 4 and explaining the use of those methods on the task of energy consumption forecasting in chapter 5; chapter 6 presents the results on the different experiments taken. In chapter 7, important points to consider when working on the task of forecasting energy consumption profiles on the low level are discussed. Furthermore, interesting suggestions for future work are pointed out, before a short ending statement is concluding this final chapter.

Chapter 2

Problem Definition and Motivation

This thesis investigates the question if and how a particular machine learning method can help in optimizing one part of the operational state of the energy grid. In particular, it looks at the challenging problem of forecasting the energy consumption profiles of individual households, using recurrent neural networks. Due to the volatile and dynamic nature of consumption profiles on the low level of individual households, reasonable accurate forecasts pose a big challenge. This thesis builds open the research on low-level forecasting and investigates the suitability of recurrent neural networks for a particular household stock in Vienna with 117 households. Not only the historic energy consumption but also additional features are regarded to aid in getting an accurate day-ahead energy consumption forecast. The recurrent neural network approach –in particular an LSTM (Long Short Term Memory) network– gets compared to a feedforward neural network. The LSTM network is once provided with additional information and gets compared to its counterpart with only the historic energy consumption as input, to evaluate the importance of the additional information. The neural network methodologies are compared to benchmarks, to see if the computational expensive methods are actually justifiable. Further, the question is investigated, if an LSTM network is best used for all households –that is, using it as a "General" model to forecast any household in the household stock–; or if every household should be modeled by its own distinct LSTM network –that is, using several "Individual" models–. The theory and experimental set up in chapters 4 and 5 respectively are explaining this in more detail. This chapter expands on the challenges faced by the current power system and its ongoing development. A focus is then given on why accurate forecasts matter and what can be gained for the various energy system participants.

2.1 Motivation

Electricity consumption on the residential level makes up a bigger share in the overall energy consumption than ever before. According to [3] the residential sector is going to make up about 30% of the electric energy consumed by the year 2030. Because of this increasing share of energy used in the residential sector, the optimal energy strategy for this sector is becoming ever more important. Furthermore, the growing penetration of renewable energy sources and the transition from the centralized

controlled grid to a more decentralized paradigm of smart grids and local energy community grid comes with the need of optimizing the low voltage level down to the individual household.
A smart grid is an electricity network that can integrate the actions of all players connected to it – generators, consumers and those that do both – in order to efficiently deliver sustainable, economic and secure electricity supplies [4]. Smart girds and local energy communities are, therefore, part of the next-generation power system, enabling grid operators to manage electricity demand in a sustainable, reliable and economical manner.
Due to the rapid growth in data generation, data acquisition and data storage in power systems, the optimal use of this information will be a paramount challenge in the coming years. Because of the roll-out of smart meters and the build-up of advanced metering infrastructure throughout every level in the energy grid, huge amounts of data are recorded and stored. *Machine Learning* techniques are increasingly used to analyze and extract useful insights from this sensory data, with the overall goal of making informed decisions to aid in achieving an ideal operational state of the grid.
One domain for which machine learning techniques are used in energy systems is forecasting. Accurate forecasts of demand and generation of electrical energy are important for the optimization of network operation, demand responses, home energy management systems, battery energy storage systems and planning of future infrastructure. However, energy consumption forecasting is a challenging problem, especially on the level of individual households. The highly volatile and dynamic nature of the energy consumption profiles on this level makes it difficult. Forecasts on higher voltage levels like substation level or region level achieve good results. These forecasts on the high voltage level have been the major focus in the past. However, due to the ongoing transformation of the energy grid, accurate forecasts on the low level are becoming more important. Forecasting methodologies, that are accurate at the high voltage level, are often not suited for the low voltage level.

2.1.1 Challenges for the Grid of the Future

Renewable Energy Sources: The increasing penetration of renewable energy sources –coming from the usage of wind, solar, hydro or geothermal power– into the grid causes problems for the save and stable operational state of the energy grid. Especially the incorporation of the non-steady sources like wind or solar challenge the balance between generation and consumption at every time in the grid. Nevertheless, the increasing usage of these sources of energy is the main objective in the development of the global energy system. The challenge of matching generation with consumption is prevalent on every grid level. Therefore, combining efficient energy storage technologies with energy production coming from solar or wind is a promising avenue to mitigate the mismatch in production and consumption.

Households as prosumers: The transformation of the centralized grid towards a more decentralized way of operation with the integration of numerous players, especially the active role of individual households as prosumers, is causing challenges to the stability and of the grid. The term prosumers describe the ability of households to play the role of a consumer as well as a producer in the energy system.

Energy system authorities are enabling its costumers with the ability to partake in the energy market. Households, therefore, are not only consuming energy from the grid but increasingly play their role in providing decentralized energy to the grid.

HEMS: Home Energy Management Systems (HEMS) are scheduling the optimal energy usage, regarding time and amount of energy used in the household by controllable appliances. Certain appliances can get scheduled in a way to optimize the objective function of the Home Energy Management System. Common objectives are to minimize the overall cost of energy usage per household, maximize customer comfort or maximizing self-consumption. Home Energy Management Systems can also be used to communicate with the grid and are therefore playing a role in demand response programs of utilities. This is useful in times of grid strain when generation and demand are unbalanced. At such times, utilities can take measures to influence the demand of households, by targeting the Home Energy Management System of various households.

BEMS: Energy storage technologies are becoming a vital part of the energy system. Especially the incorporation of renewable energy sources drives the need to store access energy in times of overproduction and provide the stored energy in times, where the energy production coming from renewable sources is not available or can not match the demand. The dominant energy storage technologies are based on thermal energy storage, batteries or fuel cells. In the case of individual households, battery storage technologies are likely to play the dominant role in the future. A Battery Energy Management Systems (BEMS) schedules the optimal charging and discharging of available resources. The ideal size of storage capacity and the optimal system deployment –whether used individually for every household or used as community storage– need to be optimized as well.

Demand Response Programs: Demand Response Programs are actions performed by utilities and network operators to secure stable and safe operation of the energy grid. Actions commonly performed to ensure the optimal state of the grid are (a) Peak Shifting, (b) Load Cutting.

Each of the above-mentioned technologies and procedures can greatly benefit in its operational efficiency from accurate forecasts, either on the generation or on the consumption side. The combination and optimization of all of these various factors, merging into an overall efficient operational state of the grid, is the challenge of the coming years.

2.1.2 Smart Meters

Information about energy consumption and energy demand is coming from recordings of smart meter devices. With the roll-out of smart meters and the penetration of advanced metering infrastructure, the availability of high-quality smart metering data is increasing. With varying time resolutions, the electric energy consumption can be recorded at various voltage levels.
Smart meters enable two-way communication. Not only can smart meters save and transmit energy data from the electric circuits it is deployed on, but it can also

operate on commands, and therefore plays a role in demand response programs, especially if the smart meter is placed on the circuit of individually controllable appliances like dishwasher, washing machine, or other controllable devices.

2.1.3 Why forecasting

Accurate forecasts can help to utilize the aforementioned technologies and programs optimally. This benefits both participators in the energy grid, the customers and the network operators. The third beneficiary is the environment, because the transition to a more efficient grid, results in less greenhouse gas emissions. The goal of the future grid is to use as many distributed energy sources as possible and optimize the scheduling of generation and consumption as well as charging and discharging of all of its eminent participants while satisfying the safety and stability requirements of the grid.

Forecasting: A part of Grid Optimization The value of accurate forecasts comes with the integration of forecasts with the aforementioned optimization programs and emerging technologies. Newly developed forecasting methods might yield more accurate forecasts in the range of small percentage points. Acting on those improvements is the challenge. Precise optimization algorithms, that not only incorporate forecasts of energy consumption and energy generation, but additionally consider weather forecasts, customer preferences, and schedule the optimal energy policy according to the current and predicted state of the grid, is the value of accurate forecasts. Combining forecasts of energy generation and energy consumption with home energy management systems (HEMS) and battery energy management system (BEMS) is crucial for the optimization of smart grids and local energy communities.

The integration and combination of those different technologies is no trivial task. Many of the technological innovations rely on or might be greatly improved by incorporating accurate forecasts. Therefore, developing reliable and precise forecasting methods of energy generation and energy consumption plays an important role in moving the grid into a more sustainable direction.
There are three main stakeholders, that benefit in ever better forecasts:

- Utilities: Some of the potential benefits for utilities are:
  - Reduction of grid strains, do to peak and overall demand
  - Reduced transmission and distribution losses
  - Better balancing capabilities of demand and supply
  - Reduction of operational costs
- Costumers: The awareness of energy consumption, energy usage, and CO2 footprint, leads customers to regard energy-efficient technology as highly preferable. Furthermore, the transition from traditional consumers to prosumers allows customers to play an active part in the energy market. Some of the potential benefits for costumers are:
  - Economic savings

  - Reduction of the environmental footprint
  - Market participation
  - Peer to Peer trading

- Environment: With more accurate forecasts of electric energy consumption on every level, –nation level, region level, substation level, transformer level, household level–, the effect of better forecasts will lead to a more efficient energy system which in turn will be more environmentally friendly. More accurate forecasts of energy generation coming from renewable sources, and balancing this generation with the actual (or predicted) energy consumption, can reduce the dependency on energy generation coming from fossil fuel power plants, resulting in less CO2 emissions.

2.2 P2PQ Project

In this section, a short introduction to the research project of *Peer-to-Peer im Quartier (P2PQ)* [1] is given. This thesis covers a small part of this research project and focuses on forecasting day-ahead energy demand of individual households with recurrent neural networks.
The European Commission presented an initial version of the framework *Clean Energy for All Europeans* on the 30th of November 2016 aiming to help energy markets include more renewable energy sources, empower consumers and better manage energy flows across the European Union. [5, 6] In March 2019, the Commission announced the finalization of negotiations facilitating the clean energy transition and make the electricity market fit for the future. The main goal is to bring regulatory certainty through the introduction of national energy and climate plans.

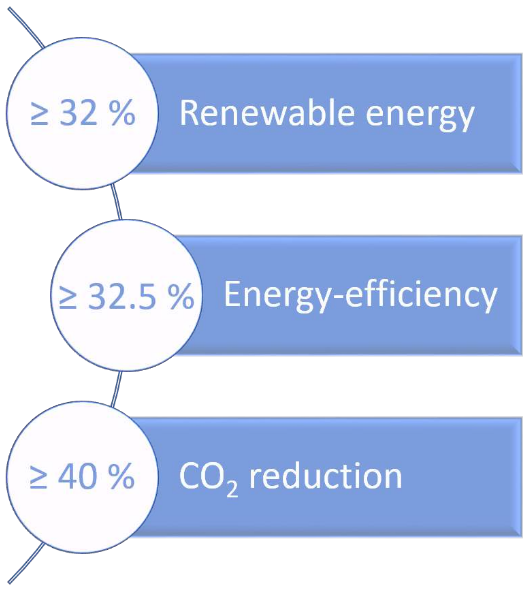

Figure 2.1: Clean Energy Package – Goals for 2030.

[1]The Peer-to-Peer im Quartier project is funded by the FFG, the Austrian Research Promotion Agency.

Figure 2.1 shows some of the goals for 2030 addressing the following issues:

- Increasing the share of renewable energy sources (RES) to at least 32 percent.
- Increasing energy efficiency by at least 32.5 percent.
- Reduction of greenhouse gas emissions by at least 40 percent.
- Empowerment of consumers.

Within the Austrian research project, *Peer-to-peer im Quartier (P2PQ)* a Blockchain-based platform is developed providing the optimization of self-consumption of photovoltaic (PV) systems and the utilization of community storage as well as the option to participate in peer-to-peer energy trading within a local energy community (LEC). In general, the project addresses the previously mentioned goals of the European Commission, in the following way:

Increasing the share of renewable energy sources: To be able to reach the goal of a share of 32 % renewable energy sources, it is necessary to optimize the utilization of flexible production and consumption. *P2PQ* addresses these issues by introducing new management concepts on the level of local energy communities.

Increasing the energy efficiency: The main goal regarding energy efficiency is the optimal utilization of the available energy as well as the reduction of consumption. *P2PQ* implements an optimization in the self-consumption of households on one hand and the optimization of an energy community in Vienna on the other hand. Within this community, photovoltaic power plants are already available, and a community battery storage will be implemented. By optimizing the utilization of the storage –based on consumption, generation, user-preferences, and tariffs– the self-consumption of the entire quarter will be increased. As a result, the energy transfer on the grid connection point can be reduced leading to lower energy costs for all members of the community.

Due to the previously mentioned measures, the energy consumption stemming from the outer grid can be reduced for every single customer as well as for the entire energy community. As a result, a higher rate of the consumed energy is provided by renewable energy sources within the quarter and thus, the energy transfer to and from the grid connection point is lowered, resulting in lower greenhouse gas emissions due to the lower need of energy from the grid which is generated by conventional power plants (e.g., gas power plants with high greenhouse gas emissions).

Empowerment of consumers: The so-far passive consumers of conventional grids are becoming active (prosumers) due to their generation (e.g., by the installation of PV power plants on private houses or apartment buildings). The European Commission aims to strengthen their right of generation, energy storage, consumption, and selling of surplus energy. As a result, prosumers should be permitted to participate as active members of energy communities or as part of a virtual power plant (VPP) in controlling the generation and consumption.

In general, the research project *Peer-to-peer im Quartier* aims to develop an innovative infrastructure for energy communities –in particular, within a community in the second district of Vienna, Austria– with the following goals:

- Increasing the self-consumption of households.
- Increasing the self-consumption of energy communities and thus, reducing the energy transfer on the grid connection point.
- Optimization of community battery storage within energy communities.
- Optimization of charging behavior on the customer level and the community level.

The optimization of the self-consumption, the utilization of renewable energies and the community storage are based on forecasts –on the generation and the consumption side–. Since private customers have a volatile energy consumption behavior, demand forecasting is quite challenging and the quality of many existing approaches is rather low. In particular, accurate forecasts are necessary to provide realistic input for self-consumption optimization, optimal utilization of battery storage, and peer-to-peer energy trading with very low amounts of energy.

Chapter 3

Related Work

3.1 Energy Consumption Forecasting

Before introducing studies concerned with the topic of energy consumption forecasting on the low voltage level, this section gives a short overview of the different areas in the forecasting domain, leading to a better understanding of the broad field and the diverse applications that are being developed.

3.1.1 Short-term, Medium-term, Long-term

The first difference in forecasting studies regards the time horizon of the target prediction. These horizons refer to the length of time into the future the forecast is made. The final forecast can either be a single prediction point –for instance the amount of energy consumed during the whole forecasting horizon–, or a consumption profile, –predicting the whole duration of the forecasting horizon with a predefined timely resolution–. The four main categories regarding the forecasting horizon are:

- Very-short-term: Minutes to Hours
- Short-term: Hours to Days
- Medium-term: Days to Weeks and Months
- Long-term: Month to Years

3.1.2 High-Level, Medium-Level, Low-Level

The next important distinction in the domain of forecasting is regarding the level at which the energy consumption or demand is considered. The level is referring to the voltage level of the grid. Mainly three different voltage levels are mentioned in the literature referring to various aggregation levels [7]:

- High-voltage level: Energy consumption in the range of several megawatt-hours is considered. Aggregation level at this stage can be seen as region or country level.

- Medium-voltage level: Often referring to the substation level or aggregated feeder level. Amounts of electric energy considered at this level range between hundreds of kilowatt-hours to megawatt-hours. Aggregation level can be seen as city or district level.

- Low-voltage level: This level is referring to the level of individual households. Here a low energy consumption baseline is common, characterized by a volatile consumption pattern. The aggregation of several individual households also falls into this category. The very-low voltage level is sometimes used to refer to the level of individual appliances.

In addition to the pure distinction of the voltage level, in some studies, there is also the distinction between different kinds of consumers. That is the differentiation of households, small business operations, and industrial buildings, due to the obvious differences in energy consumption patterns regarding those consumers.

3.1.3 MISO and MIMO

Most of the studies in this domain of forecasting energy demand try to leverage certain information about the systems state to get accurate forecasts. Historic energy consumption, weather information, temporal information, demographic information might be considered and used to get a reasonably accurate forecast. This approach then refers to Multiple-Input. Regarding the output of the forecasting model, many studies are focusing on predicting a single value, which is only the energy consumption value for a certain time-interval at time-step t into the future. This approach is often referred to as Single-Output. However, if the task is to forecast energy consumption over a certain period, the output will be a vector. The number of elements to be predicted depends on the time resolution for which the forecast is performed. For instance, forecasting two days with a time resolution of one hour, the forecasting vector consists of 48 elements. Every element corresponding to one particular hour in the forecasting horizon. This approach then refers to Multiple-Output.

- MISO: Multiple-Input-Single-Output
  In this case, the input might either be a vector, matrix or tensor. Usually, in machine learning terminology every input is referred to as a tensor. The Single-Output is referring to the fact that the output is a scalar or single value.
- MIMO: Multiple-Input-Multiple-Output
  Again the input is a tensor. Mostly a matrix (2d-tensor) or a 3d-tensor. In the case of the Multiple-Output, the output is commonly a vector.

3.1.4 Further Differences

- Time-resolution: Various time resolutions are considered in forecasting studies. Hourly time-intervals are often the target time resolution. However, more emphasis and research in recent years is in forecasting smaller time resolutions to get a finer granularity of the forecasts. 30-minute, 15-minute and 5-minute intervals are found in the literature. The smaller the considered time resolution, the more volatile and erratic the energy consumption profiles become.
- Error- and Performance Metrics: A crucial point in every forecasting study is the evaluation of the final approach. Often pointwise error metrics are used, for describing the forecasting performance. However, especially in the case of the low voltage level, this might not be the ideal solution, and therefore alternative metrics are being developed in the literature.

- Information as Input: Different studies use varying methodologies and varying data inputs to those methods. The most common information considered is (a) historic energy consumption, (b) temporal encodings and (c) weather information. Some studies also incorporated demographic information or historic consumption profiles of individual appliances.
- Size of data sets: The differing data sets considered in the regarded studies vary from single households over a short time period of several weeks or months, to bigger household stocks encompassing hundreds of households over a period of several years.

The variability in the studies leads to difficulties when trying to make valid comparisons between different studies and their proposed methodologies. For this reason, the use of standard data sets might be recommended, on which every study should test their approach. However, it is not obvious, that the "best" forecasting method on the standard data set, is then actually the "best" for the distinct data set, on which a certain study is conducted [8].

3.1.5 Formal Definition of Forecasting

After elaborating on some of the main points characterizing the task of forecasting energy consumption, the following paragraph gives a short introduction to some of the formal naming conventions in the forecasting domain:

The target of a forecasting problem is referred to as y. In the case of a Single-Output problem, this value is a scalar. In the case of a Multi-Output problem $\mathbf{y}$ is referring to a vector. The length of the vector $\mathbf{y}$ is determined by (a) the time horizon of the prediction and (b) by the target time resolution. The corresponding prediction to the true target value y is referred to as $\hat{y}$. In the context of Multi-Output, the target vector $\mathbf{y}$ and the predicted vector $\hat{\mathbf{y}}$ are often indexed by t referring to the time step of the vector, resulting in the convention $\mathbf{y_t}$ for the true target vector and $\hat{\mathbf{y_t}}$ for the corresponding prediction. Where $t >= 0$ and $t < n$ with n being the length of the vector. In the case of a 24-hour forecast with a half-hourly resolution, the target vector $\mathbf{y_t}$ is of length 48 and one element of this vector $\mathbf{y_t}$ at time step t is holding the amount of energy consumed during a particular half-hour; as does the element of the prediction vector $\hat{\mathbf{y_t}}$ respectively. In the case of Multi-Output, the predicted amounts of energy demand can either be (a) forward-looking or (b) backward-looking. These two distinctions describe if the value $\hat{y}_t$ at time step t of $\hat{\mathbf{y}}$ refers to the time interval from $t-1$ to t –backward-looking–, or to the time interval t to $t+1$ –forward-looking–.

This short introduction of some of the main differences in the forecasting domain is providing a rough understanding of the diversity in energy consumption forecasting. The field is advancing fast, and efforts are undertaken to distill the best forecasting approaches for different use cases. However, the "best" approach for one task is not automatically the "ideal" candidate when considering a different voltage level, different time resolution or different time horizon. This makes it challenging and the active research in this discipline is trying to distill insights, of when and where what kind of method is probably the optimal solution. It is crucial to evaluate the performance or "goodness" of different approaches based on a certain metric –or several metrics–, however, there is no consensus on what the "best" metrics are, based on which to compare the different methods, especially for the low voltage level. Additionally, the ideal methodology is not only dependent on the above-mentioned characteristics describing the problem task, but also on the considered household stock itself –the kind of buildings included in the analysis–. Although the problem is in the same domain, in regard to voltage level, and can be described in terms of a MIMO problem; there is high variance in the considered buildings and households itself –stemming from the lifestyle differences of the residents–, which makes a reasonable comparison challenging.

3.2 Related Work

Numerous projects and studies are done on the task of forecasting energy consumption over different time horizons and different voltage levels with varying time resolutions. It is important to point out the difference between forecasting on a higher aggregation level, –the high or medium voltage level– versus dealing with the task of forecasting individual households. Therefore, this related work section first introduces some studies that focus on a higher aggregation level, which include interesting concepts. After this, studies dealing with forecasting on the low voltage level are considered.

3.2.1 Physical-, Statistical- and Machine Learning- Models

Numerous forecasting methodologies are investigated in the literature on various forecasting problems. A rough distinction can be made between the following modeling approaches:

- Physical models: Forecasting approaches falling into this category aim at predicting future energy consumption based on physical principles and their thermal dynamics. These models describe or monitor certain parameters that might indicate future energy usage. Parameters of interest are air conditioning system, HVAC, natural ventilation, passive solar heating, photovoltaic systems, financial considerations, occupant's lifestyle, the building's thermal properties (insulation), etc.
- Data-driven models:
  - Statistical Models: Statistical models tailored to time series forecasting like ARMA (autoregressive moving average) or ARIMA –a modification of the ARMA model–, are linear models and are valuable for time series where trends and seasonality are present.
  - Machine Learning Models: Decision Trees, Random Forests, Gradient Boosted Trees, Support Vector Machines, and Neural Network Models are some of the common machine learning models used to perform forecasts. These models are rather data intensive. Especially neural network methods require reasonable amounts of data samples to produces reliable forecasts.

The above list is not conclusive but is mentioning some of the most common approaches found in the literature on forecasting energy consumption. A clear distinction between those modeling approaches is not obvious in many cases, and hybrid approaches are often adopted. Nevertheless, machine learning approaches are seeing a growing adaptation in recent years, due to the increase in data availability as well as the flexibility they provide –due to there capability to model non-linear data– compared to more classical forecasting approaches –that are often dependent on predefined feature representations–.

3.2.2 High voltage level

The predictions and forecasts of aggregated energy consumption i.e. on higher voltage level deliver promising results [9, 10]. At the high aggregation level, different model approaches like statistical models, physical/functional models or machine learning models can provide reasonably accurate predictions on short-term, long-term, on single, as well as multi-output forecasting tasks.
Many of the studies dealing with forecasting on a high level, found that the incorporation of weather data (especially ambient temperature and precise weather forecasts) does help with getting more accurate predictions [11]. Furthermore, it is found, that temporal information regarding day type (workday, holiday, weekend) does have predictive power –helping in getting more accurate forecasts–.
Review studies for the field of forecasting energy demand looked at many forecasting methodologies and found, that a wide variety of modeling techniques do perform well on the higher level [12, 13]. Statistical time series models like ARMA and ARIMA are commonly used and achieve satisfying results [14]. These linear models work well in the presence of trends or seasonality in the time series. Studies investigating the use of statistical modeling techniques are frequently trying to get the energy consumption data into the correct shape. For statistical models this often means differencing, de-trending and de-seasoning the energy consumption time series, to make the time series stationary. If stationarity can be achieved, these linear modeling approaches showcase good results on a variety of tasks. All of this de-trending, differencing and de-seasoning does make sense on the high-voltage level. However, for individual household consumption profiles, the demand curves become highly non-linear, and statistical models are mostly replaced by more flexible machine learning approaches.
It can be said, that on the high voltage level, many different forecasting methods are reaching good performances, resulting in reliable predictions of energy consumption. Machine learning techniques seem to be adapted more often than statistical models in recent years, although not all studies find a superior performance when using ML-techniques compared to statistical models or even to rather simple benchmarks. Nevertheless, particular Artificial Neural Networks are adopted by many studies and are often one of the top-performing models.

3.2.3 Low voltage level

Support Vector Machines (SVM)[15], Random Forests (RF)[16], K-Nearest Neighbors (KNN)[17], and Artificial Neural Networks (ANN)[?, 18] have been regarded for the task of short-term low voltage level forecasting. ANN and its advanced deep

learning architectures, especially Recurrent Neural Networks (RNN), are a popular research topic in recent years and show advantageous performances compared to more traditional forecasting approaches according to [19, 20, 12, 11, 21, 22].
A lot of difficulties emerge when the considered aggregation level drops from the high voltage level to the low voltage level. In particular, many modeling approaches performing well at the high voltage level are not able to deliver satisfying forecasts at the level of individual households. This is due to the high variability in residential energy consumption. The lifestyle and daily routines of residents are the primary impacts on the electricity demand [3]. Not only do the energy consumption patterns vary drastically from household to household, due to differences in lifestyles, but even the same household shows high variance on a day to day basis. This diversity in the energy demand profiles of individual households makes forecasting on the low voltage level very challenging.
Due to the volatility and non-linearity in residential energy consumption profiles, standard statistical forecasting models like Autoregressive (AR), Moving Average (MA) or Autoregressive Integrated Moving Average (ARIMA) are less effective in this domain [23, 24, 25, 26]. These modeling approaches work well if a few conditions can be met, they include (a) Stationarity of the profile and (b) strong autocorrelation of the profile itself. These assumptions can rarely be satisfied for individual households. The lower the aggregation level, the more volatile and erratic the energy consumption profiles become, thus more flexible methodologies are applied in the low voltage domain, often yielding superior performances.
Another difference between high level and low level is regarding the evaluation of the forecasts itself. Error metrics that provided useful evaluations on a high level are not suitable for the low level. In particular the *Mean Absolute Percentage Error* or *MAPE* (see equation 3.1) is a widely used error metric in the realm of high voltage level forecasting.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{3.1}$$

However, this error metric is not beneficial for the low voltage level, due to the often very low baseline consumption, resulting in very large values.
The most common error metric used for evaluating the "goodness" of a forecast on the level of single households is the *Mean Absolute Error* or *MAE* (see equation 3.2), describing the average absolute difference between predictions $\hat{\mathbf{y}}$ and true consumption $\mathbf{y}$ at every given time step t for the total length of the forecast n.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{3.2}$$

However, the *MAE* is a pointwise error metric. In [27] Haben et al. describe the problems arising when evaluating forecasts based on a pointwise error metric on the household level. The authors, therefore, propose an adjusted error metric that mitigates some of the shortcomings of often used error metrics. The difficulties coming with a pointwise error metric, especially the effect of the "double-penalty" error are discussed in section 7.2.1.
Hybrid approaches are investigated in [10, 9]. Hybrid means combining various forecasting models, and to merge the different forecasts to get a more "robust" final

prediction. In the corresponding studies the author's combined Decision Trees (DT) and Support Vector Machines (SVM), where the final forecast did perform better on the total analyzed data set, then any single models on its own.

In [3] the authors investigate the question about aggregation level and its impact on forecasting performance. By applying various machine learning algorithms for the task of forecasting 24 hours ahead, they found that regardless of the applied approach, all models did perform better, the higher the aggregation level. This was also found by other studies, when comparing region level energy demand, with the demand of a single household. The region demand follows a more steady and therefore predictable pattern. In the study they further found that although forecasting individual households came with high errors; aggregating the individual forecasts to a certain level (in the study they aggregated over a number of households, first 10 then 20 then 30 and so on up to 100 households); that the aggregation of the individual "bad" forecasts lead to better results, then when forecasting the aggregation level directly. The input to the models did not seem useful when regarding single household predictions, but when aggregating all the forecasts, it seems as though the information might carry predictive power. Unfortunately, the study did not compare the single household forecasts once with the features and once only looking at historical load, which would point out the importance of the incorporated information.

In [8] the authors find that data resolution plays an important role in the forecasting performance. With decreasing data resolution, –broadening the regarded time intervals–, the forecasting performance of a machine learning model tends to get better. This can be explained by the fact, that the increase in the time interval results in an out-smoothing effect, resulting in less volatile consumption profiles. The more erratic, the more difficult it is for models to distill the functional relationship between input and target.

Many studies investigate the importance of various features on the forecasting performance. Review studies found, that although weather information and temporal information do help in getting more accurate forecasts on the high voltage level; these features do loose their predictive power when looking at the level of individual households. For individual households, the most relevant factor of energy consumption is the occupant's lifestyle, which is highly volatile and erratic. For this reason, temporal or weather features seem to lose their predictive power in comparison to the dominant influence of occupants' lifestyle.

Most studies include benchmarks for comparison. To evaluate the "goodness" of the proposed methodologies, benchmarks serve as a first baseline, that needs to be improved on, to justify the use of more sophisticated modeling approaches. Commonly used benchmarks are, the day before (in the case of 24 hour day ahead forecasting) or the value before (in case of the robust- or persistent-predictor). Some modifications of those benchmarks like taking the average of previous days, or taking the same weekday one week before, can be found in the literature, and are found to be difficult to beat –depending on the evaluation criterion–. Therefore an intelligent benchmark choice can be a reasonable forecasting model itself. This can be seen in [17], where the authors used *K-nearest neighbors* (KNN) and refined its workings. For the task of forecasting 24 hours day ahead, the first step was to define a similarity measure based on which to evaluate the "goodness" of the forecast. As is mentioned in [8] commonly used pointwise error metrics have drawbacks, therefore the authors used the permuted error metric defined in [27]. Based on this error metric, the author found similar daily consumption profiles and took their successors. The successor days got merged, according to a defined merging function (that again used the permuted error metric as a minimization objective in the merging process), resulting in the final forecast. This approach builds upon the assumption, that similar days, have similar successors.

In [10] Hayes et al. give a comparison of various forecasting methods on varying aggregation levels. The show, that on the high aggregation level of substation level and higher MAPE rates of 1-3 % can be achieved, whereas on the low voltage level these MAPE increase dramatically depending on the considered data set. Models that are found to be useful at the higher aggregation, are not better performing than ”naive” forecasters. Especially when considering computation time, ease-of-use and periodic updating of the forecasting system, they question the suitability of sophisticated methodologies compared to the simpler approaches. The authors argue further for the limited use of pointwise error metrics in case of the low voltage level, and advocate for a probabilistic approach, delivering confidence intervals and ranges, rather than point forecasts, in which system operators can navigate.

Veit et al. [28] take state-of-the-art forecasting methods including ARIMA and ANN models and compare them critically to persistent benchmarks. The authors state very high MAPE errors depending on the data set. Furthermore, they argue, that without extensive tuning the models do often not outperform the persistent benchmark.

In [8] the author’s group households with similar consumption patterns into different clusters, based on certain statistical properties of the consumption profiles. This clustering leads to an increase in forecasting accuracy. Next to the clustering, the authors also looked at varying data resolutions and found that the bigger the regarded time intervals are, the better the forecasting models do perform.

Lusis et al. [29] investigate the effects of (a) incorporating calendar and weather variables, (b) data granularity and (c) length of training data on the forecasting accuracy of energy profiles of residential customers. The study is performed on a household stock with 27 households, and a 30-minute data resolution for a day ahead energy consumption forecast. Regression Trees, Support Vector Regression and Neural Network models are considered, with Support Vector Regression achieving the lowest normalized root mean square error (NRMSE). The calendar and weather variables did have very little predictive power on the forecasting performance. No consistent improvement was found in the models, that incorporated those variables in addition to historic consumption recordings. Decreasing the granularity from 30 to 60 and 120 minutes lead to more accurate forecasts. Interestingly the authors found, that no benefit for the forecasting system was drawn, by incorporating more training data then one year. The comparison if a training data set of one year, with a three-year-long training data set, found no differences in forecasting performance. The authors ascribe this finding to the changing and dynamic lifestyles of occupants. The consumption patterns three years ago, might not have any useful information, and one year might be enough data for reasonable inferences.

3.2.4 Neural Network Approaches

In many studies, neural network models are one of the best performing methodologies. Although the advantage of neural network models is not that apparent when considering high voltage level domains; the volatile nature of the low voltage energy demand profiles might favor the more flexible neural network models over other forecasting methods.

In recent years also recurrent neural networks got more attention in the realm of forecasting energy demand. In [25] an overview of different recurrent neural network architectures is given on three synthetic and three real-world tasks of forecasting time series. The different recurrent neural network architectures compared in this study are, (a) "vanilla" recurrent neural network (RNN), (b) gated recurrent neural network (GRU), (c) long short term memory network (LSTM), (d) echo state network and (e) NARX (non-linear autoregressive exogenous input) network. One of the considered real-world time series are energy consumption profiles on the high voltage level of several MWh. It was found that the performances of the various approaches do depend heavily on the task at hand. No recurrent model approach does perform "best" in every single task. However, the interesting finding of this extensive review is that the more sophisticated recurrent neural network architectures like the GRU or LSTM did not out-perform the more simple "vanilla" recurrent neural network. The authors argue that, although the LSTM and GRU models reach significant improvements over "vanilla" recurrent neural network in the realm of speech recognition and language translations, these advantages is not to be seen when forecasting energy consumption profiles, due to the less complex temporal dependencies in energy consumption time series compared to the dependencies encountered in speech and languages tasks.

Restricted Boltzmann Machines (RBM) –in particular Conditional and Factored Conditional RBM's– is another neural network-based modeling approach used by Mocanu et al. in [20]. Other then neural networks, RBM are trained in an unsupervised fashion trying to learn the "hidden" data representation in the data itself, and not learning based on true target values –which is supervised learning–. This learned data representation is then used to create forecasts, with the goal of generating forecasts coming from the same "distribution" as the real profiles it learned from. The authors found a superior performance of the proposed Factor Conditional RBM model compared to other considered ML-techniques, including ANN, SVM, and Recurrent Neural Networks.

An LSTM based sequence to sequence model is proposed in [30]. The authors compared an LSTM forecasting model with an LSTM based sequence to sequence model and found that for the very low resolution of 1 minute, the LSTM based sequence to sequence approach performed better than the normal LSTM model. For the hourly data resolution, this is not found. The sequence to sequence model is built on the separation of an encoder and decoder part of the forecasting system. The first part –the encoder– is taking the inputs and is constructing an output vector. Based on this output vector of the encoder, the second part –the decoder– has to infer the forecast. By training this sequence to sequence-structure the encoder is forced to represent the inputs in an optimal manner. The decoder then takes this compressed information to construct a reasonable forecast.

In [18] the authors proposed a Long Short Term Memory (LSTM) recurrent neural network forecasting framework and tested this approach against other machine learning models. They found that the more volatile daily consumption profiles are, the more advantageous the LSTM model is likely to be compared to other methods. The authors ascribe this finding to the fact, that the LSTM model has higher flexibility due to its numerous parameters, that can "learn" the erratic patterns better than the less flexible methods they investigated.

Two interesting concepts, first, the clustering of similar energy consumption profiles, and second, comparing the forecast on the aggregation profile of single households, with the aggregation of the single household forecasts are done in [23]. Peng et al. found, that the aggregation helps in getting better forecasts and that the aggregation of households belonging to a certain cluster did bring even more beneficial effects.

Yildiz et al. [3] performed a load forecasting study on 14 households. The authors investigated Artificial Neural Networks and Support Vector Regression models for this task. They use smart meter recordings, weather data as well as temporal variables as input to their models. The authors studied the effect of various data resolution by investigating 5, 15, 30 and 60 minute time intervals, as well as the effect of varying the forecasting horizon –1,6,12 and 24 hours–. Results show, that both parameters –the data resolution and the forecasting horizon of the investigated demand profiles– do have a great effect on the forecasting performances. Furthermore, daily profiles –belonging to a distinct household– are grouped into certain buckets, according to their volatility (measured by the standard deviation of the load profile), to train models particularly only on similar volatile load profiles. Energy consumption peaks grow more distinct in magnitude and variance the finer the data resolution gets. The forecasting models are doing poorly on predicting consumption peaks. Additionally to the often-used pointwise error metrics, the authors introduce an accuracy measure by which to assess the goodness of the various models, due to the inherent shortcomings of pointwise errors for the task of low-level forecasting.

The interesting concept of an Aggregation Error Curve (AEC) is described in [31], where the authors investigate the effect of various aggregation levels on the forecasting performance by presenting an intuitive scaling law. The study points out some of the main challenges arising in the field of forecasting when going from the high aggregation level to the low aggregation level. It emphasizes the "out-smoothing" effect that comes with aggregating over single profiles. This leads to a more periodic signal, thus allowing for more accurate forecasts. However, there is a point, where further aggregation stops being valuable in regard to forecasting performance.

A Long Short Term Memory (LSTM) network is used in [32] for the task of short term load forecasting on a university building. The LSTM approach is compared to SARIMA (Seasonal Auto-Regressive Integrated Moving Average) and NARX (non-linear autoregressive neural network with exogenous features) models. The forecast is performed for a 15-minute data resolution. The LSTM approach was the best performing of the considered models.

In [18] Kong et al. provide an analysis of LSTM networks for a day ahead forecast of residential customers. The effects of exogenous features (calendar variables) and the effect of data granularity is investigated. The LSTM approach is compared to the "mean" benchmark (statistical mean of on particular household for weekdays or weekends). The authors also study the effect of aggregating the individual demand prof. An interesting finding, is the fact, that the aggregation of individual forecasts is more accurate than the forecast of the aggregated profile. However, they additionally found that the superior performance of the LSTM network compared to the naive mean forecaster is stronger on the individual level, then on the aggregated level. This fact may be explained by the ability of the LSTM network to capture temporal dependencies in the individual household's consumption profiles. This advantage weakens when aggregating the profiles, although the aggregation leads to an "out-smoothing" effect of the profile itself, which might help the LSTM network with extracting more information from the rather noisy single household consumption patterns.

A recent study [23] investigates the question, of what is more important, the optimal model or the preprocessing step leading to good forecasts? The authors argue, that for the very volatile energy demand profiles of individual households, there is a lot of randomness in the energy consumption profiles –stemming from the unpredictability of the resident's behavior– leading the models astray. Various models ranging from linear regression, gradient boosted regression trees, multi-layer perceptron to more complex LSTM models are compared and historic smart meter recordings are used as input variables to the models. The analysis is performed on 1700 residential households with a data resolution of 30 minutes. One hour ahead and one day ahead are the considered forecasting horizons. Before diving into finding the most sophisticated model and tweaking the parameters of the model, it is recommended to spend time on increasing the predictability of the energy profiles in the first place. The authors state, that the optimal model might not be the most important choice in the task of forecasting, but to ensure the fact that the energy profiles are actually reasonably predictable. For this, they use the approximate entropy, a metric describing the "predictability" of time series data. The authors advocate for data processing steps prior to choosing the optimal model, to decrease the approximate entropy in the underlying consumption profiles. They show that this can be achieved by differencing the consumption profiles or by aggregating the individual profiles.

Every method -statistical or machine learning, parametric or non-parametric- has its advantages and disadvantages. One of the main points arguing against the use of neural networks was the high computational cost of these models. This drawback is decreasing with the development of computational power, especially the development of GPUs (graphical processing units) and of TPUs (tensor processing units). With faster inference times, more experiments can be conducted. Due to the availability of computational resources, many studies are now focusing on using neural network methodologies. However, in [33] the authors argue, that many of the proposed methodologies might often be over-parametrized, to generalize well, and that a rather simple neural network architecture might be sufficient for the task of forecasting energy demand. Therefore, the superior results of sophisticated neural network models have to be taken with care, as long as the understanding –or lack of understanding for that matter– of neural networks is taken into account. However, the authors state that although they are surprised by the many good results found when using over-parameterized neural network models not every study does fully investigate the probable over-fitting effect these kinds of models might have. Nevertheless, the authors are aware, that the problem of over-parameterization is not well understood in the realm of neural network research, and is a topic of ongoing research [34].

A conclusion from the many studies related to the topic of energy consumption forecasting might be, that there is no optimal model for every task. Not only is the

ideal model category –physical models, statistical models, machine learning models– unclear, but also the choice of the individual models itself. All this is to say, that there is current research effort on finding the optimal approach for forecasting at various voltage levels. However, it can be said that over the last couple of years, neural network models gained a lot of attention, and are consistently one of the best performing methodologies, on the high voltage level, as well as on the low voltage level. Choosing the appropriate model is only one of the many choices that have to be considered when dealing with the task of forecasting energy demand. Other considerations encountered in the literature are clustering, aggregating/disaggregating, varying temporal resolution, forecasting horizons and the incorporation of various predictor variables.
Based on the promising performance of neural network methodologies, which are repeatedly among the "best" performing models in many studies, this thesis is building on those findings and investigates the suitability of neural networks. In particular, a certain kind of neural network methodology, that has shown to be useful for the task of time series modeling and sequence learning.

3.3 Progress beyond the state of the art

This thesis is building on top of the existing research in the disciple on energy consumption forecasting and aims at evaluating the suitability of Recurrent Neural Networks –in particular Long Short Term Memory (LSTM) networks– for a 24-hour day-ahead energy consumption forecast on individual households. The analysis is performed on a distinct household stock in the city of Vienna, and the value of additional exogenous features is investigated. Furthermore, it is analyzed to what extent Long Short Term Memory networks are best applied to this task. For this matter, LSTM models are once trained on all households, and once on single distinct households. Investigating this question might give insights into which approach is more beneficial for yielding reliable forecasts.
The following listing mentions points in which this thesis is expanding on the current state of the art. The exact approaches and empirical questions this thesis is investigating are described in chapter 5 in more detail.

1. Comparison between (a) LSTM models that are trained with additional features, and (b) LSTM models only taking historical consumption profiles as input, to investigate the importance of the considered additional features.
2. Answering the question of what is the best use case of deploying LSTM models. Comparison of "General" models with "Individual" models. Is a "General" LSTM model, trained on the total household stock appropriate to forecast every household's energy demand, or is a tailored model for every single household the better option?
3. Investigating the effect of data resolution on forecasting performance, as well as on the importance of additional features. (a) 15-minute and (b) hourly time intervals are considered.
4. Not only regarding one pointwise error metric but also taking a second performance measure -daily energy consumption deviation- into account. Evaluating

the forecasting methods based on both metrics helps in getting a more nuanced assessment of the ”goodness” of competing methods.

5. Comparing feedforward neural networks to LSTM networks, to investigate the suitability of the sequential processing of the latter approach.
6. Comparing neural network methods to benchmarks. Is the deployment of neural network models justifiable when taking performance and computational complexity into account?

Chapter 4

Theoretical Background

This chapter provides the theoretical background behind the used neural network methodologies applied in this thesis. The main concepts are introduced, to get an understanding of the workings of an artificial neural network (ANN). Furthermore, some of the biggest challenges coming with the use of ANN are discussed. After this theoretical understanding, the succeeding chapter 5 then explains the usage of those methodologies tailored to the problem of day-ahead energy consumption forecasting.

4.1 Forecasting Definition

Naming Conventions

Here the problem of a day ahead forecast is described to facilitate the use of machine learning terminology and notation. In the realm of machine learning and especially for artificial neural networks it is common to call the inputs and outputs of a model *tensors*. The input to the models is commonly made out of a collection of *features*. Features are the predictive variables, based on which the models perform a forecast. Input tensors to the model are denoted as $\mathbf{X}$ and are commonly referred to as values, whereas the output tensors of the model are denoted as $\hat{\mathbf{y}}$, and are referred to as predictions. In a supervised machine learning problem, the available true values –based on which the models are trained and optimized– are denoted as $\mathbf{y}$ and are called targets. Depending on the dimension of the considered tensors, it might be easier to refer to them directly as matrices (a tensor of dimension two), vectors (a tensor of dimension one) or scalars. The particular dimension of the tensors is noted, when needed. The naming conventions can be summed up as follows:

- Inputs: $\mathbf{X}$: ***values***
- Outputs: $\hat{\mathbf{y}}$: ***predictions***
- Targets: $\mathbf{y}$: ***targets***

In a supervised machine learning problem, the inputs $\mathbf{X}$ and targets $\mathbf{y}$ get paired together. Every input gets assigned its corresponding target. In the task of forecasting the energy consumption for the next day, the target $\mathbf{y}$ is a tensor "holding" the energy consumption profile of the next day, which is to be predicted. The challenge is in processing and constructing the inputs $\mathbf{X}$ in a way, so that the neural network model is able to learn from the inputs $\mathbf{X}$ how to predict the targets $\mathbf{y}$ as close as possible. Artificial neural networks are known to be universal function approximators. In theory, an ANN is capable of mirroring any functional relationship between input and output pairs. In practice, however, there are certain obstacles that hinder neural networks to achieve that. These difficulties in applying neural networks are discussed at the end of this chapter. Applying machine learning models can be regarded as a functional relationship in the form of:

$$f(\mathbf{X}) = \hat{\mathbf{y}}$$

where the function $f()$ is the machine learning model. The goal of a machine learning system is to minimize the error of the prediction $\hat{\mathbf{y}}$ in regard to the truth $\mathbf{y}$. The internal mechanisms of artificial neural networks for finding those underlying relationship between $\mathbf{X}$ and $\mathbf{y}$ is explained further below.

How the inputs $\mathbf{X}$ and the corresponding targets $\mathbf{y}$ are processed and constructed to form training pairs for the models is discussed in the next chapter 5.

For the problem of forecasting a day-ahead energy consumption of individual households, the input tensor $\mathbf{X}$ needs to be processed in a form to "hold" useful information to enable the model to infer accurate estimation of $\mathbf{y}$. The prediction of the model $\hat{\mathbf{y}}$ has the same form (shape or dimension) as the target $\mathbf{y}$.

4.2 Problem Definition

The problem of forecasting the energy consumption for a day ahead is in the simplest form a multi-output regression problem. Depending on the data resolution, the model tries to find values of every time interval in the output space to be as close as possible to the true amount of electric energy consumed during this interval. The units of the amount of energy consumed per interval are kilowatt-hours (kWh). The starting error metric on which the forecasting performance is evaluated is the mean absolute error (MAE) 3.2. The MAE describes how far the model is off on average overall time steps from the true target value. A mean absolute error of 0.1 kWh of one forecast to the true target consumption profiles means, that on average for every interval of the regarded profile the model is off by 0.1 kWh.

The chosen error metric on which to evaluate the performance of different models is of importance. The error metric should convey an interpretable way of accessing the performance of the method. Furthermore, there are certain disadvantages that come with the use of a pointwise error metrics like the MAE. This is discussed in section 7.2.1 and throughout this thesis. The usage and reliance on one single error metric can yield outcomes, that might not be ideal in regard to the end-use case of the forecasting system. This is a problem in the forecasting field, and ongoing research is trying to develop good forecasting error metrics on the low voltage level for multi-step ahead forecasts. However, the diversity in household stocks and end-use cases, makes it challenging to get a consensus on what the best error metrics are for a particular task. Yilidiz et. al. [8] therefore propose a template for approaching forecasting studies on the low level, that researchers can use as a guidepost to develop further consistent research in this field.

4.3 Artificial Neural Networks

The models in this thesis are trained in a supervised way. Supervised machine learning means that the models have the true target available during training. During the training procedure, the neural network model tries to find the functional relationship between input $\mathbf{X}$ and target $\mathbf{y}$. The training procedure is an iterative process. The model gets an input $\mathbf{X}$ and calculates an output $\hat{\mathbf{y}}$. The calculations performed by a neural network are linear transformations in conjunction with non-linear activation functions. The prediction $\hat{\mathbf{y}}$ than gets compared with the true energy consumption profile $\mathbf{y}$. During this comparison step, the loss or error of the predictions gets calculated. The loss indicates how far off the prediction is from the true target. Based on this loss, the model updates its parameters, –weights and biases–, following the backpropagation algorithm, which changes the parameters ever so slightly in a stepwise fashion to decrease the loss of the forecast over time. This procedure is done for every single sample in the data set. A sample is an input/target pair $\mathbf{X}/\mathbf{y}$. By processing all samples of the data set for multiple times, the model is likely to find optimal parameters that minimize the loss on average for all provided samples.

4.3.1 The Artificial Neuron

The building block of every artificial neural network is the artificial neuron. An artificial neuron receives an input $\mathbf{X}$ and produces one output value.
The neuron takes the inputs and calculates a weighted sum of those input values, described by equation 4.1. Every neuron is equipped with weights w and one bias b. All the weights and biases of every neuron in a neural network are the "learnable" parameters that characterize the behavior of a neural network.

$$WeightedSum = \sum_{i=1}^{n} w_i x_i + b_0 \tag{4.1}$$

In equation 4.1 n refers to the size of the input. For instance, the input $\mathbf{X}$ might correspond to a days energy consumption profile encoded as a vector of size $n = 24$, where every element x_i of this vector holds the amount of energy consumed for a particular hour of the day. Each element x_i of the input $\mathbf{X}$ has one weight w_i associated with it. The bias term b_0 of a neuron is one more "learnable" parameter. The bias, however, is not directly associated with an input element x_i. This weighted sum then gets fed into an activation function to allow the neuron to model non-linearity.

Activation Functions To provide a neuron with the capability to model non-linear relationships between input and target, a non-linear activation function must be provided. This activation function "squashes" the result of the weighted sum into a range constrained by the output realm of the activation function. Some of the most common activation functions used in machine learning problems are listed below.

1. Logistic or Sigmoid Function: $f(x) = \sigma(x)$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

2. Tanh Function: $f(x) = tanh(x)$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.3}$$

3. Rectified Linear Unit (ReLU): $f(x) = relu(x)$

$$relu(x) = \begin{cases} 0 \text{ for } x \leq 0 \\ x \text{ for } x > 0 \end{cases} \tag{4.4}$$

4. Leaky Rectified Linear Unit: $f(x) = leakyrelu(x)$

$$leakyrelu(x) = \begin{cases} 0.01x \text{ for } x \leq 0 \\ x \text{ for } x > 0 \end{cases} \tag{4.5}$$

5. Exponential Linear Unit: $f(\alpha, x) = elu(\alpha, x)$

$$elu(\alpha, x) = \begin{cases} \alpha(e^x - 1) \text{ for } x \leq 0 \\ x \text{ for } x > 0 \end{cases} \tag{4.6}$$

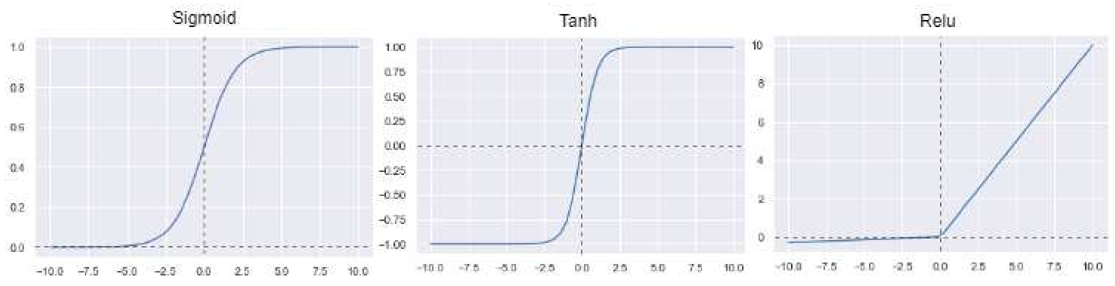

Figure 4.1: Sigmoid, Tanh and ReLU activation function

Each of these activation functions comes with its own characteristics, and therefore with distinct advantages and disadvantages. The role of the activation function is crucial for the learning process of one neuron, and therefore for the entire neural network. Up to this point one neuron gets the input $\mathbf{X}$, calculates the weighted sum according to 4.1 and inputs this intermediate result into an activation function. The resulting value is the output of the neuron and is determined by the weights w_i, the bias b_0 as well as the chosen activation function.
The formal processing performed by one neuron with the input tensor $\mathbf{X}$, the *sigmoid* function σ as activation function, $\mathbf{w}$ as its weight vector and b_0 as its bias is shown by equations 4.7 and 4.8:

$$NeuronOutput = \frac{1}{1 + e^{-(\sum_{i=1}^{n} w_i x_i - b_0)}} \tag{4.7}$$

or:

$$NeuronOutput = \sigma(\sum_{i=1}^{n} w_i x_i + b_0) \tag{4.8}$$

Figure 4.2 shows the informational flow of a single neuron.
This operation of one neuron is the essential building block of all neural network architectures.

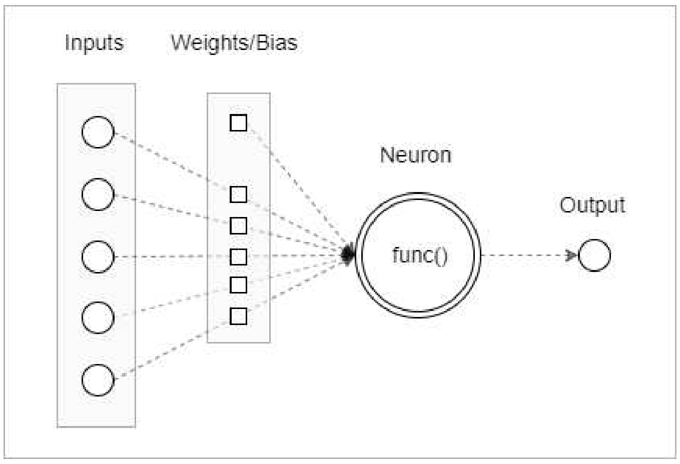

Figure 4.2: Single Neuron

4.3.2 Feedforward Neural Networks

The next building block after the single neuron is getting the individual neurons into a layer.

Neural Network layers: A layer in a neural network is formed by multiple neurons. Every neuron in a layer is receiving the same input values. The output of one layer is not a single value, but a vector holding all the single output values of its neurons. This vector processed by one layer is then the input to the neurons in the subsequent layer. When stacking numerous layers on top of each other, this is called *Deep Learning*.
Figure 4.3 shows a simple neural network, where five neurons are forming a layer. With the combination of multiple neurons to form a layer, and with the stacking of layers, an arbitrary neural network can be created. The most simple neural network is the so-called feedforward neural network. This network architecture is simply the stacking of various layers, where each layer consists of a certain number of neurons. In this thesis, when talking about neural networks, it is always that the network is fully-connected.

Fully-connected network: A fully-connected network means, that the output value at a certain layer gets fed to every single neuron in subsequent layer –but only to this layer–. The resulting output of this layer is then processed by every single neuron in its subsequent layer and so on. This fully-connection and the processing in a straight forward way, layer after layer refers to the neural network architecture of a *fully connected feedforward neural network* (FFNN).

Input and Output layers: The input layer of a neural network has to be adjusted to the size of the input $\mathbf{X}$ of the neural network. The output layer is fixed by the size of the target-output $\mathbf{y}$.

Hidden layers: The layers in between the input and the output layers are called hidden layers.

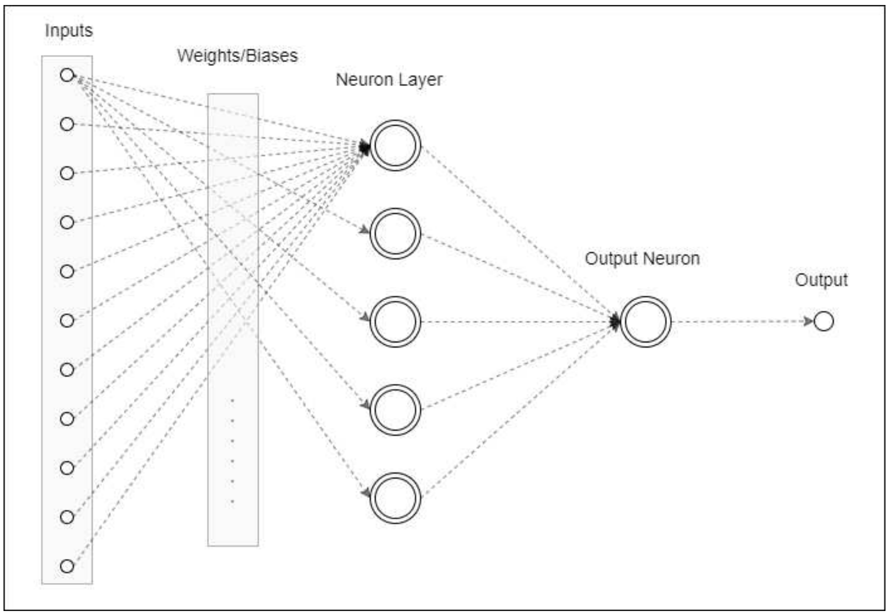

Figure 4.3: Fully connected feedforward Neural Network

Weight-matrices and Biases: With the increase in neurons comes also an increase in the number of parameters the network is equipped with. Every neuron has its set of weights w_i, which has to be equal to the size of the input it receives from the layer before. Every neuron additionally has one bias term b_0. Every connection in the network from one neuron to another, is weighted by one particular weight parameter w^n_{ij}, where i describes the i^{th} Neuron in the n^{th} layer and j describes the j^{th} Neuron in the $(n-1)^{th}$ layer. The indices $_{ij}$, therefore, encode the direction of the informational flow of one weight connection between two neurons. The biases in a network layer are commonly encoded with b^n_i, where i describes the i^{th} bias, and n refers to the n^{th} layer.

To illustrate this point figure 4.4 shows the naming convention of three particular connections between two subsequent layers. Every weight connection inside a neural network can be understood as a potential informational path, where the input can flow through. During training the phase the model might detect that certain informational pathways are especially relevant for accurate output. The weights corresponding to each connection gets adjusted, depending on the importance or the irrelevance of this informational pathway.

The fully-connection between the $(n-1)^{th}$ layer and the n^{th} layer is described by a weight matrix $\mathbf{W^n}$. The weight matrix $\mathbf{W^n}$ holds the weights of every connection between all neurons in those two layers. For instance the weight w^n_{ij} in the weight matrix $\mathbf{W^n}$, describes the weight connecting the j^{th} Neuron in the $(n-1)^{th}$ layer, with the i^{th} Neuron in the n^{th} layer.

All these parameters –weights w^n_{ij} and biases b^n_0– are characterizing the behavior of a neural network model. During the training process of a neural network, all its parameters are continually updated to finally emerge at an "optimal" set of parameters that describe the functional relationship between inputs $\mathbf{X}$ and targets $\mathbf{y}$ as close as possible.

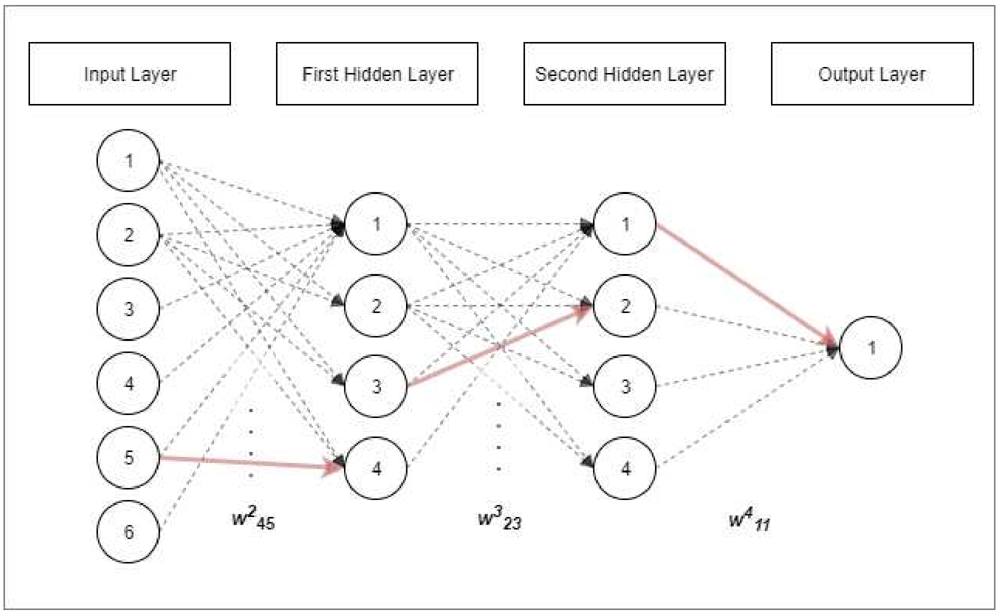

Figure 4.4: Weight connections between layers in a Neural Network

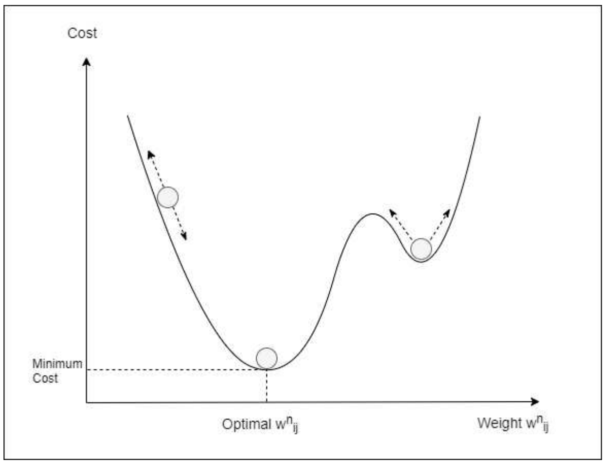

Figure 4.5: Finding the optimal parameter by continually applying gradient descent

Finding the optimal parameters: The most common process for finding the "optimal" set of parameters in neural networks, is the backpropagation algorithm. This algorithm relies on the gradient of a Cost- or Loss-function with respect to every individual parameter of the network. The Loss-function is a chosen function that describes, by how far the prediction of the neural network $\mathbf{\hat{y}}$ is off from the true value $\mathbf{y}$. The common notation of the output of the Loss-function is $\mathbf{C}(\mathbf{y}, \mathbf{\hat{y}})$. Many Loss-functions work well for different machine learning tasks. For regression problems (as in this thesis) the mean absolute error or mean squared error Loss-functions are common choices. With the gradient-descent step (moving in the opposite direction of the gradient of the Loss-function with respect to the parameters) the parameters get continually nudged into the direction, that minimizes the loss $\mathbf{C}(\mathbf{y}, \mathbf{\hat{y}})$ –often referred to as the cost– over time. This process describes the neural network's capability to "learn". In a supervised setting a neural network receives input, this input gets processed through the network and is finally returning an output. This output of the neural network $\mathbf{\hat{y}}$ then gets "compared" with the actual true value $\mathbf{y}$. The chosen Loss-function then determines, by how far off the output is from the true value, and returns the cost $\mathbf{C}(\mathbf{y}, \mathbf{\hat{y}})$. This cost is the starting point for the updating process of the various parameters. The gradient $\nabla\mathbf{C}(\mathbf{y}, \mathbf{\hat{y}})$ of the cost with respect to the parameters then gives the opposite direction (depending on the definition of the Loss-function) in which to update the parameters. Therefore, the quantities of interest in the updating process of the parameters are:
(a) $\frac{\partial \mathbf{C}(\mathbf{y},\mathbf{\hat{y}})}{\partial \mathbf{w^n_{ij}}}$, and (b) $\frac{\partial \mathbf{C}(\mathbf{y},\mathbf{\hat{y}})}{\partial \mathbf{b^n_j}}$. These two terms describe by how much the cost changes, when the weights and biases are "wiggled" by a very small amount. The step for updating the parameters is described by 4.9 for the weights and by 4.10 for the biases respectively.

$$w^n_{ij(new)} = w^n_{ij(old)} - \eta \frac{\partial C(y,\hat{y})}{\partial w^n_{ij}} \tag{4.9}$$

$$b^n_{j(new)} = b^n_{j(old)} - \eta \frac{\partial C(y,\hat{y})}{\partial b^n_j} \tag{4.10}$$

η in these equations refers to the learning rate of the neural network. The learning rate describes the magnitude of the parameters update. It is one of the most important hyper-parameters for the learning ability of neural network models. Choosing the learning rate too big can lead to "overshooting" the optimal solution for the parameters. On the other hand, choosing the learning rate too small can lead the neural network to get stuck in local minima of its Loss-function, or taking a very long time to converge to the "optimal" solution. Figure 4.5 shows a graphical illustration of the updating process of one parameter. This is, of course, an oversimplification. The curvature of a neural network can be extremely complex, and is formed by thousand or up to millions of parameters.

A lot of research effort is undertaken to develop algorithms, that find the best parameters in a computationally efficient way. These algorithms are referred to as optimization algorithms, many of the well known rely on the basic gradient descent steps outlined above. Various optimization algorithms commonly used for neural networks are the *AdamOptimizer*, *RMSProp*, *AdaGrad* and *SGD*. Where each one has modifications to the workings of the simple gradient descent approach. The most prominent advances are (a) Momentum and (b) adaptive learning rate. For a more in-depth discussion on optimization algorithms, the interested reader is referred to [35, 36, 37].

A few notes on the Backpropagation Algorithm: A comprehensive discussion of the backpropagation algorithm is out of the scope of this chapter, however, a few core ideas are mentioned here to help in understanding the main concepts of this operation. The interested reader is referred to [38, 39]. Especially in [39] an example of the detailed calculations performed during backpropagation in a simple neural network with two input neurons, two hidden neurons, and two output neurons is given. The main ideas to keep in mind when trying to understand the backpropagation algorithm are: (a) informational flow forward through the network in functional stages, (b) the chain rule for decomposing the functional stages, and (c) backward flow through those functional stages and finding the partial derivatives. To arrive at the final output of a neural network the input has to flow through all neurons in the network. The final output depends on the first place on the outputs of the neurons present in the output layer. However, the output of every neuron in the output layer depends on the output of every neuron in the layer before, and so on, until reaching the input layer. Because of this, the final result can be seen as the output of a big composition function with various parts belonging to different layers and its neurons. When asking about the partial derivative of a Loss-function of the neural network in respect to an individual parameter, as in $\frac{\partial \mathbf{C}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{w}^{\mathbf{n}}_{\mathbf{ij}}}$, it is important to keep in mind, that there are many intermediate steps involved to arrive at this result. One single parameter w^n_{ij} influences the weighted sum operation performed in its neuron. The altered result of the weighted sum, in turn, alters the output of the neuron after traversing through the activation function. It becomes clear, that altering a single parameter w^n_{ij} leads to a cascade of changes in the information flow of the network. The backpropagation algorithm deals with these many interrelations in the network by applying the chain rule step by step for every operation in the path of $\frac{\partial \mathbf{C}(\mathbf{y},\hat{\mathbf{y}})}{\partial w^n_{ij}}$.
Only backpropagating one connection back from the neural networks output –that is finding the partial derivative of the Loss-function with respect to a weight parameter in the final output layer by applying the chain rule– this term becomes: $\frac{\partial \mathbf{C}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{a}(\mathbf{z})} \frac{\partial \mathbf{a}(\mathbf{z})}{\partial \mathbf{z}(\mathbf{w})} \frac{\partial \mathbf{z}(\mathbf{w})}{\partial w^n_{ij}}$, where $\mathbf{a}(\mathbf{z})$ refers to the output of one neuron after the activation function, and $\mathbf{z}(\mathbf{w})$ refers to the results of the weighted sum operation prior to the activation function, $\mathbf{w}$ refers to the set of weights (weight-vector) holding the individual weights of a certain neuron in the output layer, and w^n_{ij} refers to one of those individual weights. Every single one of these partial derivatives needs to be calculated and then multiplied to arrive at the rate of change of the cost regarding one single parameter w^n_{ij} in the output layer. This describes only one connection. In neural networks, the number of parameters easily exceeds 50.000 and often reaches the millions. All of these connections have to be taken care of by the backpropagation algorithm. This results in massive composite functions. However, the building blocks for performing the backpropagation algorithm are: (a) separating the calculations into its parts, (b) applying the chain rule to find all partial derivatives of every dependent calculation in the path and (c) calculating those partial derivatives. It is because of these huge numbers of calculations necessary for training a neural

network, that for a long time they were not the preferred machine learning model in many domains. Only with the increase in computational power, through the development of powerful processing units, this shifted, and neural networks are now widely used.

FFNN Conclusion: The paragraphs above describe the workings of a fully connected feedforward neural network in rough terms and do not cover every aspect of even the simplest neural networks. However, it should provide a basic understanding and an intuitive way to think about the processing steps a neural network undertakes to "learn" and find the underlying functional relationship between input $\mathbf{X}$ and target output $\mathbf{y}$. For an in-depth explanation of the inner workings of neural networks, the interested reader is referred to [37, 36, 38, 40].
Neural networks are known as universal function approximators. They are conjunctions of linear transformations and non-linear activation functions. These consecutive conjunctions allow neural networks in ever more growing complexity to model functional relationships between input and output. Neural networks are applied in many domains, ranging from its deployment in the field of quantum physics, where they are used to model the state wave function of quantum states, to language translation applications present in every modern smartphone. They are especially beneficial, if the work of creating good features/predictors for models is difficult due to lack of knowledge, or simply not feasible due to the amount of data. On the flip-side of things, this is also one of the biggest shortcomings of using neural networks in the first place. They are "black-boxes" and they are hard to interpret. Just because they are modeled and inspired by the workings of the human brain does not guarantee, that the model makes decisions that are aligned with the human reasoned choice. Examples of this can be seen in the work of image classification, where when a model, that is trained for a long time on bananas and can predict those with a high degree of accuracy, suddenly predicts a picture of a banana as a car. The only change was, that in the image –that the model classified as a car– the pixel values were minimally changed, what lead to a wrong prediction, although humans still see a banana. All that is to say, that neural network models function to a certain extent as black boxes and their inner workings are often non-intuitive, and not fully understood.
With the most essential building block -the neuron- a neural network with an arbitrary number of layers and neurons per layer can be built. Advanced neural network methods are constructed from this one building block. With the rough understanding of the workings of a normal fully connected feedforward neural network, the next section gives an introduction to the more sophisticated recurrent neural network.

4.3.3 Recurrent Neural Networks

Recurrent neural networks (RNN) are similar to feedforward neural networks. The main distinction is, that feedforward neural networks do only allow for informational flow in one direction –from the input layer to the output layer–. Recurrent neural networks, on the other hand, allow for information to flow backward. Because of this kind of processing, recurrent neural networks are capable of holding "memory". Every processing step holds some information from all the preceding steps due to the "back-feeding" loop inside a recurrent neural network. This capability of having a

"memory" makes recurrent neural network especially useful for tasks where sequential data is considered, and where there is a temporal dependency in the sequence. Language translation, speech recognition, time-series predictions, are some of the applications where recurrent neural networks often outperform feedforward neural networks due to this memorization capability.
The total input tensor to a recurrent neural network $\mathbf{X}$ is best thought of as a sequence of single input tensors $\mathbf{x_t}$. The single inputs of the sequence are sequentially ordered or have a temporal relationship. The single inputs are usually referred to by $\mathbf{x_t}$, where t describes the considered time step of the single input.
The "back-feeding" loop of a recurrent neural network is achieved by the fact, that the RNN does not only process the input $\mathbf{x_t}$ at time step t but also its output $\mathbf{y_{t-1}}$ from the time step before.
The easiest way to think about this fact is to start with a single recurrent neuron. It is helpful to picture the recurrent operation in a recurrent neuron by "unrolling" it through time, which is regarding every processing time step anew as is shown on the right in figure 4.6. However, the weights w and the bias b of a recurrent neuron are shared over all time steps. The recurrent neuron performs its mathematical operation not only on the input $\mathbf{x_t}$, but on a combination (stacking together both values) of the input $\mathbf{x_t}$ at time step t and the single output y_{t-1} at time step $t-1$.

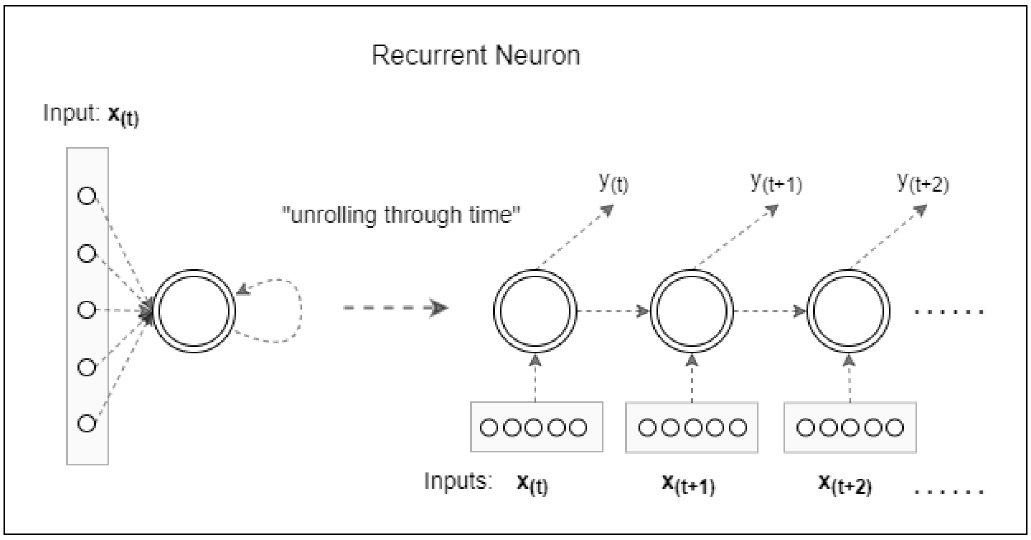

Figure 4.6: Recurrent Neuron on the left and its "unrolled" version on the right.

The operation performed by a single recurrent neuron is described by equations 4.11 and 4.12.

$$RecurrentNeuronOutput \rightarrow y_t = \phi(\sum_{i=1}^{n} w_i \begin{bmatrix} \mathbf{x_t} \\ y_{t-1} \end{bmatrix}_i + b_0) \tag{4.11}$$

or in vector form:

$$RecurrentNeuronOutput \rightarrow y_t = \phi(\mathbf{w} \begin{bmatrix} \mathbf{x_t} \\ y_{t-1} \end{bmatrix} + b_0) \tag{4.12}$$

Where the function $\phi()$ is a chosen activation function. The main difference therefore from the operation of a normal artificial neuron described in equation 4.8, is the

stacking of the input $\mathbf{x_t}$ with the output y_{t-1}. Because of the stacking of those two values, also the weights $\mathbf{w}$ have to change accordingly. When regarding a single recurrent neuron, the weights $\mathbf{w}$ do change, as far as they now have one more weight connection corresponding to the single output produced by the recurrent neuron. For the very first processing step, where $t = 1$ the value for $y_{t=0}$ is commonly initialized to 0 –because there is no preceding step yet–.
This single recurrent neuron functions as the building block for more advanced architectures of recurrent neural networks. For getting a recurrent layer many recurrent neurons are taken together. The output of the recurrent layer is not a single value, but a vector with the same length as there are recurrent neurons in the layer. Every recurrent neuron in this layer then gets not only the input at every time step $\mathbf{x_t}$ but also the output of the recurrent layer from the time step before $\mathbf{y_{t-1}}$.
Figure 4.7 shows a recurrent neural network layer, where the five recurrent neurons each have one output value. These five output values are taken together to form the recurrent output vector. This recurrent output vector is then stacked together with the new input at the next time step.

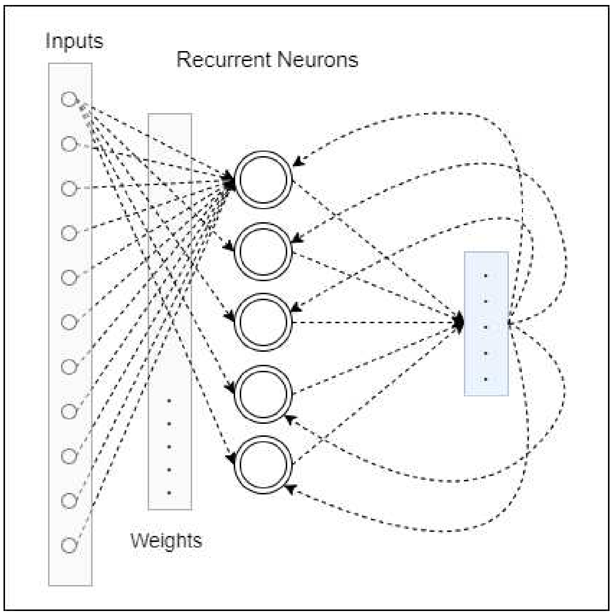

Figure 4.7: Back-feeding inside a Recurrent Neural Network layer

The operation performed by one recurrent layer is defined by equations 4.13 and 4.14. The common notation for the weight matrices $\mathbf{W_x}$ and $\mathbf{W_y}$ is used. $\mathbf{W_x}$ is the matrix holding the weights, connecting every input $\mathbf{x_t}$ to the neurons. Every row in this matrix $\mathbf{W_x}$ holds the set of weights $\mathbf{w}$ corresponding to one of the neurons in the layer. The matrix $\mathbf{W_y}$, on the other hand, does connect the "back-feeded" output vector $\mathbf{y_{t-1}}$ with the layer. Again, every row in the matrix $\mathbf{W_y}$ corresponds to a set of weights belonging to one recurrent neuron.

$$RecurrentLayerOutput \rightarrow \mathbf{y_t} = \phi(\mathbf{W_x x_t} + \mathbf{W_y y_{t-1}} + \mathbf{b}) \tag{4.13}$$

The more compact form of this equation is:

$$RecurrentLayerOutput \rightarrow \mathbf{y_t} = \phi(\left[\mathbf{W_x}, \mathbf{W_y}\right] \begin{bmatrix} \mathbf{x_t} \\ \mathbf{y_{t-1}} \end{bmatrix} + \mathbf{b}) \tag{4.14}$$

The activation function $\phi()$ is processing the values of the output of the recurrent layer element-wise.

Explanatory Example: All this notation is a little bit confusing, and it might not be obvious how a recurrent network performs one processing step. To clarify the concept and show the dimensions of the various vectors and matrices involved, a short example of one processing step is given below.
With $\mathbf{X}$ being one input for the recurrent neural network. This can be a sentence, audio-clip, time-series, every kind of data where the elements hold a temporal dependency and the sequential processing capability of the recurrent neural network is suitable. For this example, $\mathbf{X}$ is a time series, where every element of $\mathbf{X}$ is a vector $\mathbf{x_t}$ of size 10 holding the information of this particular time step t. A recurrent neural network with 5 recurrent neurons is chosen for this task. How is the informational flow from an arbitrary time step t to the next time step $t+1$ in the network?
The size of the inputs $\mathbf{x_t}$ of every time step t is 10. The size of the recurring inputs (the "back-feeding" outputs of the time step before) is 5, because their are 5 recurrent neurons. The stacked weight matrix $[\mathbf{W_x}, \mathbf{W_y}]$ in equation 4.14 has therefore dimensions $(5, 15)$. Every one of the 5 rows corresponds to the set of 15 weights $\mathbf{w}$ of one recurrent neuron in the layer. The first ten columns in the stacked weight matrix $[\mathbf{W_x}, \mathbf{W_y}]$ are holding the weights responsible for the inputs $\mathbf{x_t}$ with size 10. The remaining five columns are holding the weights, responsible for the recurrent output $\mathbf{y_{t-1}}$ from the previous time step. The stacked vector $\begin{bmatrix} \mathbf{x_t} \\ \mathbf{y_{t-1}} \end{bmatrix}$ is of length 15. The bias vector $\mathbf{b}$ has size 5, holding one bias b_0 for every recurrent neuron.
With this setup, equation 4.14 becomes

$$\mathbf{y_t} = \phi(\left[\mathbf{W_x}, \mathbf{W_y}\right]_{[5\times 15]} \begin{bmatrix} \mathbf{x_t} \\ \mathbf{y_{t-1}} \end{bmatrix}_{[15\times 1]} + \mathbf{b}_{[5\times 1]}) \tag{4.15}$$

resulting in the output vector $\mathbf{y_t}$ of length 5 with values bound by the constraints of the chosen activation function $\phi()$. For the next processing step along the input sequence this output vector $\mathbf{y_t}$ then again gets stacked with the input $\mathbf{x_{t+1}}$ and the calculation 4.15 is performed again. This is done until the end of the sequence $\mathbf{X}$ is reached.
Depending on the task for which the recurrent network is deployed, a specific Loss-function then calculates the error of the network after the sequence got processed. This error then, as in the case of the feedforward neural network, is the starting point for the updating process of the parameters characterising the network. The process for finding the optimal set of parameters is similar to the one in feedforward neural network, where the methodology of gradient descent with the backpropagation algorithm is followed. In recurrent neural networks however, the process of backpropagating the gradient signal through the network is called backpropagation through time or (BPTT). Because of this, it is helpful to think of recurrent neural networks in the "unrolled" version as is shown in figure 4.8. When unrolling the recurrent neural network through time, the BPTT algorithm can then be regarded as the normal backpropagation algorithm as is the case in a feedforward neural network –with the difference, that the weights and biases are shared over all time steps–.

Hidden state $\mathbf{h_t}$: The hidden state of recurrent neural networks h_t refers to its internal state, that "holds" the memory and is fed back to the network in a recurrent fashion. In a simple recurrent neural network described above this hidden state $\mathbf{h_t}$ is the same as the processed output at every time step $\mathbf{y_t}$. However, this those not have to be the case. In the next section a more powerful recurrent neural network architecture is introduced, where a cell state $\mathbf{c_t}$ and a hidden state $\mathbf{h_t}$ equip the network with more powerful memorization capabilities.

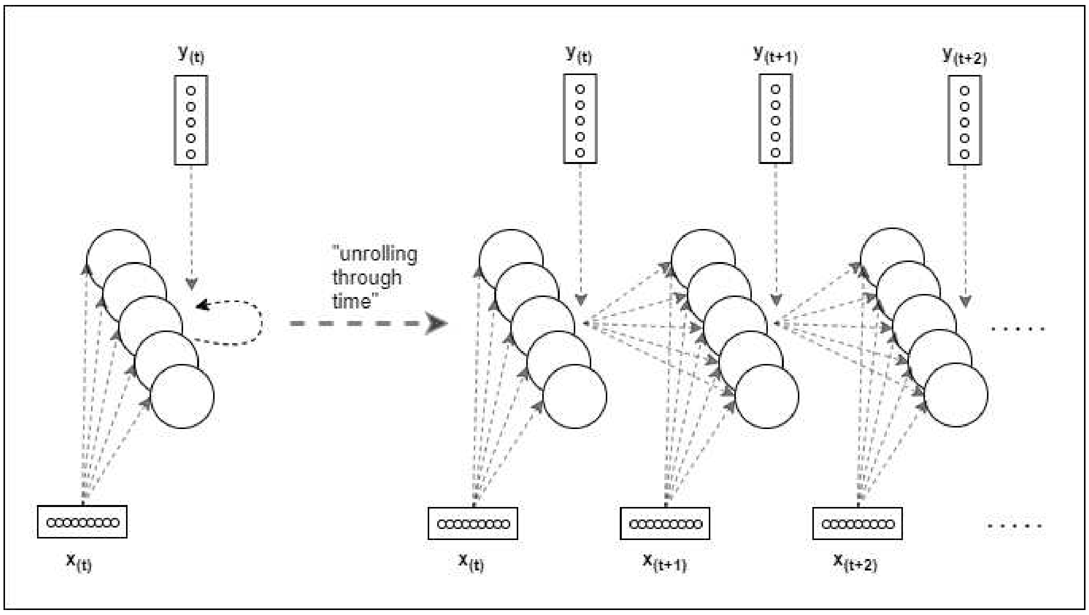

Figure 4.8: Recurrent Neural Network. On the left the "recurrent" version, on the right the "unrolled" version.

Conclusion Recurrent Neural Networks: Recurrent neural networks are a special kind of neural networks. Other then feedforward neural networks that process data in a feedforward way, –running the inputs straight from the first input layer to the last output layer–; recurrent neural networks process the data sequentially. The word recurrent stems from the fact, that for each single input in the input sequences, the same operations are performed again and again. The learned weights and biases in a recurrent neural network are shared over all time steps. An intuitive way to think about recurrent neural networks is by "unrolling" them through time. For the length of the input sequences, there is one layer per time step. When processing a sequence of length ten, a recurrent neural network can be thought of, as a ten layer deep neural network that does not only receive the current time steps single input, but also the information from the layer before, aka. the time step(s) before. The weights and biases connecting the layers however, are the same.
The recurrent neural network works well on temporal dependent data, where the sequential ordering of its elements is import. However, it has one major drawback. In theory RNNs can learn temporal dependencies in a sequence of arbitrary length. In practice however, the problem of the *vanishing gradient* arises, –the fact that the weight update based on the gradient descent step gets smaller the longer the sequence is–.

The effect of the vanishing gradient problem is, that the signal from the gradient of the initial cost calculated by the Loss-function at the last layer does not reach the earlier layers in the network, but only the parameters in the later layers get updated. For this reason, in long sequences, the information from the start of the sequence might get lost completely, and the parameters get only updated based on the information gathered form the latter layers. The solution to this vanishing gradient problem encountered by RNNs, was proposed by Hochreiter [41] through the development of the Long Short Term Memory (LSTM) Network. Due to its inner structure –the gates, a hidden state, and a cell state– this particular modification of the RNN can hold and update information for very long sequences without suffering from the vanishing gradient problem.
In the next section, this kind of recurrent neural network is introduced and its inner workings are explained. It is the architecture used in the empirical part of this thesis.

4.3.4 Long Short Term Memory Networks

To circumvent the vanishing gradient problem of RNNs when working on longer sequences, the LSTM network relies on a special architecture. The main difference is, that the LSTM network has an inner cell state $\mathbf{c_t}$ and a inner hidden state $\mathbf{h_t}$. Whereas the hidden state $\mathbf{h_t}$ of the LSTM network is similar to the hidden state in a RNN, and can be thought of as holding short term memory; the cell state $\mathbf{c_t}$ can be though of as holding long term memory. This more ”robust” memory allows for backpropagation of the gradient signal over much longer sequences. The internal cell state $\mathbf{c_t}$ of the LSTM network gets updated with every step. This continual updating process of the cell state is performed by specific *gates*. In particular there are three gates in a LSTM network responsible for the updating of the cell state. The three gates are:

- Forget gate: f_g
- Input gate: i_g
- Output gate: o_g

Each one of these gates is on its own a recurrent layer with its set of parameters (weights and biases), that are learned during the training process of the network. Every gate provides the LSTM network with a different capability of modifying the internal cell state, –which acts as the information carrier over many time steps–. A helpful analogy is to see the cell state as a conveyor belt, that transports information from previous time steps through the network. The information form this conveyor belt can then either be taken to create an output or it can be modified to change the internal ”knowledge” of the network. The gates are modifying the cell state with every time step. These modifications enable the LSTM to either overwrite the cell state, retrieve information from it, or keep it as it is for the next time step.

Inner Workings of a LSTM network: The workings of a LSTM network are explained in the following paragraph according to the image 4.9. Although the architecture of a LSTM can vary depending on the implementation details, this form of an LSTM network is a common LSTM architecture.

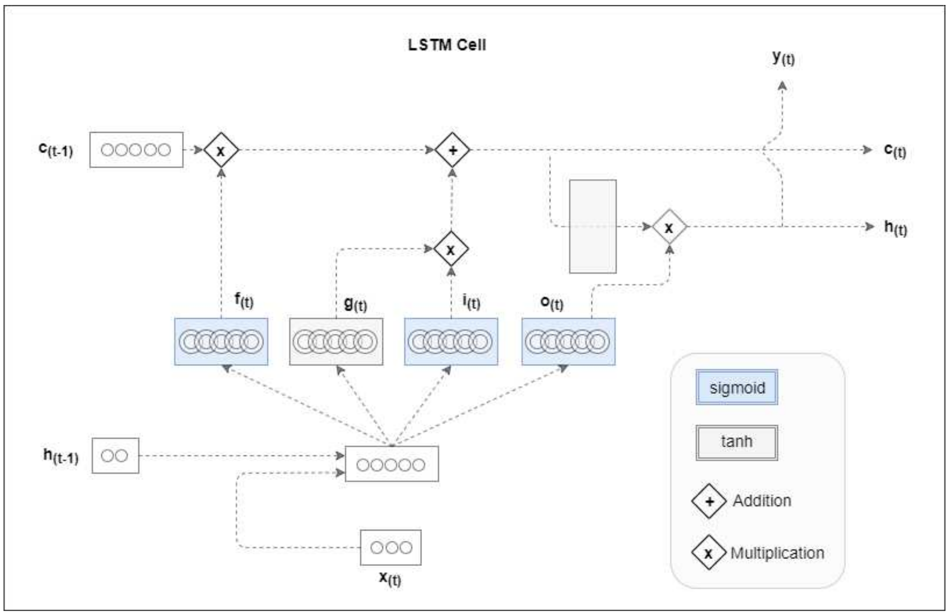

Figure 4.9: Image of an LSTM cell.

There are four layers in this LSTM network $f_{(t)}, g_{(t)}, i_{(t)}$ and $o_{(t)}$. Three of these four layers are the aforementioned gates equipped with the sigmoid/logistic activation function $\sigma()$, namely the forget gate $f_{(t)}$, the input gate $i_{(t)}$ and the output gate $o_{(t)}$. As in the case with a RNN, the first processing step is in concatenating the current input $\mathbf{x_t}$ with the hidden state from the time step before $\mathbf{h_{t-1}}$. The resulting vector $\begin{bmatrix}\mathbf{x_t}\\ \mathbf{h_{t-1}}\end{bmatrix}$ is then transported to every of the four layers and modified by them. The layers are learned during the supervised training phase of the network. These layers therefore learn the best strategy to modify the internal cell state. Each gate is a kind of filtering pipeline with different internal workings. The gates $(f_{(t)}, i_{(t)}, o_{(t)})$ are all equipped with the sigmoid/logistic activation function $\sigma()$. Values returned by the σ activation function range between 0 and 1. The gates are therefore can be thought of as "closed" for values near zero and "open" for values closer to one.

The length of the cell state and the hidden state of the LSTM model is determined by the number of neurons specified in the four different gate layers. The number of neurons is the same in every gate layer. The dimension of the parameters are fixed by the length of the input $\mathbf{x_t}$ and the length of the hidden state $\mathbf{h_t}$. The weights and biases of every gate layer are shared over all time steps of the sequence, however every gate layer has its own unique set of parameters.

Equations 4.16 to 4.21 describe the informational flow throughout a LSTM network.

$$ForgetGate \rightarrow \mathbf{f_{(t)}} = \sigma([\mathbf{W_{xf}}, \mathbf{W_{hf}}]\begin{bmatrix}\mathbf{x_t}\\ \mathbf{h_{t-1}}\end{bmatrix} + \mathbf{b_f}) \tag{4.16}$$

$$InputGate \rightarrow \mathbf{i_{(t)}} = \sigma([\mathbf{W_{xi}}, \mathbf{W_{hi}}]\begin{bmatrix}\mathbf{x_t}\\ \mathbf{h_{t-1}}\end{bmatrix} + \mathbf{b_i}) \tag{4.17}$$

$$OutputGate \rightarrow \mathbf{o_{(t)}} = \sigma([\mathbf{W_{xo}}, \mathbf{W_{ho}}]\begin{bmatrix}\mathbf{x_t}\\ \mathbf{h_{t-1}}\end{bmatrix} + \mathbf{b_0}) \tag{4.18}$$

$$\mathit{Layer} \rightarrow \mathbf{g_{(t)}} = \tanh(\left[\mathbf{W_{xg}}, \mathbf{W_{hg}}\right] \begin{bmatrix} \mathbf{x_t} \\ \mathbf{h_{t-1}} \end{bmatrix} + \mathbf{b_g}) \tag{4.19}$$

$$\mathit{CellState} \rightarrow \mathbf{c_{(t)}} = \mathbf{f_{(t)}} \otimes \mathbf{c_{(t-1)}} + \mathbf{i_{(t)}} \otimes \mathbf{g_{(t)}} \tag{4.20}$$

$$\mathit{HiddenState} = \mathit{OutputState} \rightarrow \mathbf{h_{(t)}} = \mathbf{y_{(t)}} = \mathbf{o_{(t)}} \otimes \tanh(\mathbf{c_{(t)}}) \tag{4.21}$$

The explanation of the various processing steps is giving bellow:
After forming the concatenated vector $\begin{bmatrix} \mathbf{x_t} \\ \mathbf{h_{t-1}} \end{bmatrix}$ from the input at the current time step $\mathbf{x_t}$, and the hidden state of the previous time step $\mathbf{h_{t-1}}$; the first processing step of the LSTM given by equation 4.16. The vector $\begin{bmatrix} \mathbf{x_t} \\ \mathbf{h_{t-1}} \end{bmatrix}$ gets modified by the forget gate $\mathbf{f_t}$. In the forget gate this vector, gets multiplied with the weight matrix $\mathbf{W_{xf}}$ (holding the weight connections of every recurrent neuron in its rows). The non-linear sigmoid/logistic activation function $\sigma()$ then squashes the resulting vector from the matrix-vector (plus bias term $\mathbf{b_f}$) operation into the range between 0 and 1. The last step performed by the forget gate $\mathbf{f_t}$ is in multiplying this vector -holding only values between 0 and 1 element-wise with the cell state $\mathbf{c_t}$. An entry in the cell state gets decreased or deleted (is forgotten), if multiplied by a value close to zero or actually zero, whereas it stays (is remembered) if multiplied by a value close to or exactly 1. The forget gate therefore acts as a guard, who is looking at the new input, and according to the ”importance” of the new information, modifies the cell state to forget or keep information from the previous time steps.
Step two of the LSTM is described by equation 4.17. This modification to the cell state is performed by the input gate $\mathbf{i_t}$. Similar to the forget gate, it starts with the concatenated vector $\begin{bmatrix} \mathbf{x_t} \\ \mathbf{h_{t-1}} \end{bmatrix}$. The resulting vector is multiplied with the weight matrix $\mathbf{W_{xi}}$ of the input gate added with the bias terms $\mathbf{b_i}$ and the resulting vector is squashed again by the sigmoid activation function into the range between 0 and 1. The next step is where the input gate differs from the forget gate. A second layer is part of this processing step, namely the layer $\mathbf{g_t}$. Its operation is described by equation 4.19. The parameters in this layer $-\mathbf{W_{xg}}$ and $\mathbf{b_g}-$ process the concatenated vector the same way as before, but instead of using an sigmoid activation function $\sigma()$, this layer uses the tanh() activation function, to squash the values into the range between -1 and 1. The output vectors of (a) the sigmoid path and (b) the tanh path of this step are then multiplied element-wise resulting in a vector with values between -1 and 1, where some of the entries where filtered out by the sigmoid path. Finally, this vector gets added to the cell state $\mathbf{c_t}$. While the sigmoid path does the same as the forget gate; in combination with the tanh path it gives the LSTM the capability of adding relevant information to the cell state. The LSTM is therefore able to look at the new information, and decide if the incoming information is valuable or not. Through addition or subtraction of values between -1 and 1 it can modify the cell state in this step.
The next processing step involves the output gate $\mathbf{o_t}$. This gate does not change the cell state directly, however it calculates the new hidden state; and is therefore influencing the cell state for the next time step. To calculate the new hidden state, the output gate performs the same operations as the forget gate and the input

gate, although with its own set of parameters –$\mathbf{W_{xo}}$ and $\mathbf{b_o}$–, described by equation 4.18. To alter the hidden state $\mathbf{h_t}$ for the next time step, the output gate gets multiplied element-wise with the tanh() activated cell state –the cell state that is already modified by the forget gate and the input gate according to equation 4.20–. The element-wise multiplication of the output gate with the tanh() activated cell state is described in equation 4.21. The output gate therefore decides which parts of the long term memory of the cell state should be outputted at the current time step.

With all those operations a LSTM network is capable of deleting, adding, and retrieving information at every time step from the cell state. Through separating the idea of memory into a long term part –the cell state $\mathbf{c_t}$- and a short term part -the hidden state $\mathbf{h_t}$– the LSTM is capable of learning long term dependencies in the data.

The search for the optimal set of parameters in a LSTM network is performed in the same way as in a normal RNN. By unrolling the LSTM network through time, the backpropagation through time (BPTT) algorithm can be used as the updating procedure. Iteratively, the backpropagation algorithm is used to backpropagate the gradient signal throughout the network. The gradient descent step then nudges the parameters into the direction, that decreases the cost of the LSTM network over time. It is important to keep the individual functional stages of the network in mind, when applying the backpropagation process. In the case of a LSTM network the total composite function yielding the output of the network, is even more interrelated and complex then in the case of a feedforward neural network. For an example of how the backpropagation works in an LSTM and the numerous steps involved, the reader is referred to [42], where an example of the backpropagation algorithm applied to an LSTM network over three time steps is given.

Conclusion of LSTM networks: LSTM networks are a modification to the original recurrent neural network. Both networks are capable to hold information in memory, due to there sequential processing of data and the back-feeding of information. For long sequences recurrent neural networks suffers from the vanishing gradient problem. The vanishing gradient problem is an often encountered effect, when working on longer sequences, stemming from the fact, that the backpropagation signal diminishes, and only the latter layers get updated. LSTM networks are known not to suffer from this problem, due to their internal structure of having a cell state, a hidden state, as well as three gates: forget-, input- and update-gate. These gates enable the network to modify the cell state.

An interesting view on the workings of LSTM networks and there specific gate convention is by Francois Chollet , the Founder of Keras –a deep learning library [43]–. In his book [44] he argues, that the gates are not doing what humans name them to do. The reason why the LSTM network performs better then usual recurrent neural networks is because the LSTM is (because of the gates) equipped with more representational power, that is suitable for long term sequences. It is not that the forget gate is crucial or the input gate is the important part. It is just, that the representational power of an LSTM network is different then the one of a RNN. This is also pointed out by the fact, that Gated Recurrent Unit (GRU) networks (a more recent modification to recurrent neural networks then the LSTM network), that does not have the same gate architecture as the LSTM, does have in many learning *tasks similar or superior performances then the LSTM. The GRU does only have one gate. It is all about the representational power that the model is equipped with.*

4.3.5 Problems with Neural Networks

Neural networks are one of the most powerful machine learning models. Especially due to their often numerous parameters in the thousands or millions, these models can explore a wide range of functional relationships in the data. Neural networks are extremely flexible and do not need sophisticated pre-processing or feature engineering of the data before using them. One of the biggest strengths of neural networks is that they are capable of extracting hierarchical features by themselves. However, some disadvantages come along with the power and flexibility of neural network. Some of the challenges and problems arising when working with neural networks are mentioned in the following.

Hyperparameters

Neural networks are characterized by their parameters (weights and biases). These parameters are affecting the informational flow throughout the network, from the input until arriving at the final output. In addition to that, there are many more parameters, that need to be chosen to enable the network to learn properly in the first place. The parameters that are not "learned" during the training phase of the model, but need to be specified in advance are called hyperparameters. The choice of the optimal set of hyperparameters is not trivial. Some of the most important hyperparameters include:

- Number of hidden layers: The number of hidden layers is responsible for the level of abstraction a neural network can learn. The deeper a neural network is, the more hierarchical structures it can find. Each layer allows the network to represent the functional relationship from input to output, based on the found relations in the previous layers. This enables to network to build upon its findings. Especially in the case of computer vision this shows to be extremely valuable. For instance transfer learning describes the fact, that neural network layers, that are learned on a specific use case, can be applied for other use cases as well, if they share some similarities. In computer vision this means, that although a neural network (in computer vision mostly convolutional neural networks) is trained to detect cars in an images; earlier layers of this trained network might be valuable also for the task of recognising faces. This is because earlier layers extract general features and functional relationships, that apply to a wide range of use cases. The deeper the network, the more specific the latter layers and the extracted features become.

- Number of neurons per layer: The number of neurons per layer can be seen as describing the representational power of every layer. It is a common choice to decrease the number of neurons with every advancing layer, therefore forcing the network to find hierarchical representations from layer to layer.

- Training batch size: The training batch size refers to the number of data samples that are fed to the model at once. After one such batch the model changes its parameters following the backpropagation and gradient descent steps. The batch size can be chosen between 1, that is, the parameters get updated after every single sample, and the total size of the data set n. Common choices of the batch size are $16, 32, 64, 128$. The size of the batch size therefore describes, how many samples should be regarded before updating the parameters. The updates of the parameters are then calculated so that they decrease the cost of the neural network for the processed batch of samples. This does not automatically mean, that the updates are a good choice, leading to minimizing the cost of the network for the totality of all samples. However, with many iterations, this process usually converges to a good solution.

- Number of Epochs: One epoch refers in machine learning terminology to one processing of the total available training data. After the fact, that every single sample in the training data is fed through the network once, one epoch is completed. By processing the available samples for many epochs, the goal of the model is to eventually find the parameters (by continually updating them into a certain direction), that describe the (true) functional relationship between input and output. Therefore, the more epochs a model trains, the more it can search for underlying relations in the data. However, too much training time can lead the model to find relations, that are not "true" and only random. This then hinders the model on making good decisions on unseen data, that is, the models generalizability is poor.
- Optimization algorithm: There are many optimization algorithms to choose from. Some of the most common are: Stochastic Gradient Descent (SGD), Adam Optimizer, AdaGrad, RMSProp. There are no clear rules which optimizer is the best choice for different kinds of problems. Often the differing optimization algorithms are performing similar. All of the mentioned optimizations algorithms are modifications to the idea of gradient descent.
- Learning rate: The learning rate has a big influence on the training process of the neural network. It is usually chosen to be a default value, depending on the choice of the optimization algorithm. Choosing the learning rate too big, might lead to overshooting the "optimal" set of parameters of the model, whereas choosing the learning rate too small might lead to non-convergence. Adaptive learning rates are often applied, to adjust the learning rate alongside with the advancement of the training process of the model. The learning rate might decrease with time as the model gets closer to the optimal set of parameters, therefore updating more precisely and enhancing the likelihood of finding a optimal solution.
- Activation Function: The choice of activation functions in a neural network can have a big effect on its learning process. The output range and slope (when computing the derivatives) are the influential factors of the activation function. Activation functions equip the network with non-linearity, and certain constraints. Inside those constraints the neural network has to search for the optimal parameters for solving the task at hand.

Batch Normalization, and *Parameter Initialization* are other hyperparameter choices that can have a significant effect on the performance of a neural network. The "ideal" choices of all of these hyperparameters is usually to be found by testing a wide range of different neural network architectures and different sets of optimizers/learning rates and batch sizes. The goal is to find the neural network, that can generalize well to ”unseen” data samples, after the training phase of the model is complete.

Overfitting

Generalizability is the ultimate goal of every machine learning model. This means that the model is not only capable to predict the samples it has seen during training with accuracy, but also, that it can transfer the performance to new samples (from the same or similar distribution) it has not seen before. It might be the case that the models are very good when assessing their performance on the training data, however when evaluating the models on new data the performance is poor. This discrepancy between performance on the training data and new data is called overfitting. This happens, when machine learning models find a functional relationship in the training data that is random and has nothing to do with the actual true underlying relations from inputs to targets.
Neural networks are especially prone to overfitting due to their often high number of parameters. The flexibility that comes with the many paths a neural networks can use to describe and formulate relations in the data is making overfitting likely. To mitigate the problem of overfitting, and increase the generalizability of the network, certain regularization methods can be applied.
Regularization methods are reducing the overfitting effect during the training phase and increase the generalizability of the model. However, it is not clear from the beginning which regularization technique is the best or helpful. It highly depends on the data itself. The ”best” approach for training a model is often, to first provide the model with many parameters, so it has enough representational power to model the input target relations precisely. If this then leads to overfitting, the next step is to prune the model down. This pruning is done by using regularization techniques, or the reduction in model parameters. Especially in Deep Learning settings the common approach is first: overfitting the training data, to see if the representational ability of the model is sufficient for the task. If this is the case, the next step is in using regularization. These regularization methods constrain the model parameters, so that the model is less likely to fit its parameters to describe relations in the data, that are purely random. Typically regularizations like the *L1, L2* or *Elastic Net* –a combination of the first two–, penalize parameters, if their norm is getting to big during the training phase. If the model finds, that a certain connection –either coming from the input, or from successive computations– is of high importance for a accurate prediction, then the model is going to weight this connection stronger with every training step. However, this connection might not describe the real relationship in the data, and might be purely by chance, –either because of the data itself, or because of the random initialization phase of the model–, thus resulting in overfitting. To avoid this over-weighing of certain connections, the regularization methods, keep weights from getting too big. This regularization on the other hand, can hinder the model from accurately describing the relationship in the data as it is. It might be the case that a certain connection in the overall model informational path is of high importance, but the regularization hinders this connection from being fully exploited. Therefore, neural network models are often trained and tested first without any regularization. After comparing the performance on the train- and test-set the generalizability of the model can be assessed. If the generalizability is

weak, that is the model is overfitting on the train-data; regularization methods can be applied, and the performances on the different data sets gets compared again. This incremental experimentation for finding the model, that is the most suitable for the task at hand is very tricky, and involves a lot of additional hyperparameters tuning.

Regularization

There are mainly three ways to incorporate regularization in neural networks, to mitigate the effect of overfitting.

- Penalizing Weight Connections: This regularization technique penalizes weights that are becoming to big during the training phase. It is tuning back the importance of certain connections, so that no single path in the neural network gets to much emphasis. Common regularization methods falling into this approach are the Ridge- (L1), Lasso- (L2) and Elastic Net- (a combination of both) regularization. All methods work by adding an extra regularization term (the sum of all weight values) to the Loss-function of the network. Therefore, the usual optimization steps for minimizing the cost, also tries to minimize the regularization term, therefore penalize weight values that are becoming to big.

- Dropout: This regularization methods drops a certain percentage –determined by the hyperparameter choice of the dropout probability– of neurons in the network for every processing step. This methods can therefore be seen as a kind of ensemble learning technique, where with every training step, a different model gets trained, and in the end all different models are used for the outcome. By dropping certain connections with every training step, the neural network is forced to explore a wide range of possible paths, and is not able to rely on one particular path alone. A common choice of the dropout probability is between 0.25 and 0.5, meaning that 25% to 50% of neurons are randomly shut down for every training step. For the final predictions all neurons are then activated.

- Data Augmentation: This method relies not on changing the internal structure of the neural network, but in changing the inputs to the model. The method is especially useful when the number of available training samples is limited. The augmentation of the input data does two things (a) increases the amount of data for the model, and (b) forces the model to look at a broader range of inputs and therefore explore more possibilities.

4.3.6 Conclusion Theoretical Background

This chapter provided the theoretical background to have a basic understanding of the workings of neural network models. All neural networks can be broken down into the most basic building block, the neuron. By stacking neurons into layers, and layers into deeper neural networks, powerful models can be built. These neural network models should have enough representational power to model the underlying functional relationship form input to target. Feedforward neural networks and recurrent neural networks differ in their way of processing the input. Recurrent connections equip recurrent neural networks with a kind of memory, making them useful for temporal dependent data. In the next chapter the implementation of recurrent neural networks and feedforward neural networks for the task of forecasting electric energy consumption of individual households is described.

Chapter 5

Experimental Setup

This chapter explains how the neural network methodologies introduced in the previous chapter are deployed for the task of low level day ahead energy consumption forecasting. The various neural network models are introduced, and the approach for comparing and evaluating the differing methods is explained. The main empirical questions this thesis tries to answer are emphasised. The data, on which this analysis is performed gets introduced, and the various data processing steps are discussed.

5.1 Problem Definition

The objective of this thesis is to deploy, evaluate and asses the performance of neural network methods on the task of forecasting the energy consumption of individual household for one day ahead. The forecasting goal –that is the energy consumption profile for the next day– starts at 00:00 in the morning and ends at midnight. This 24 hour period is the forecasting target of the different models.
In this thesis the considered energy consumption recordings are analysed for two sampling rates (a) 15 minutes and (b) 1 hour. The goal is to create a energy consumption forecast for the next day with the same resolution. The target $\mathbf{y}$ is therefore an array of length 96 in the case of 15 minute time intervals, and of length 24 in the case of hourly intervals. Each entry in the array is corresponding to the amount of energy consumed during a distinct time interval. The unit of the energy recordings is kilowatt hours [kWh].
The differing models are outputting a prediction for the next days energy consumption profile $\hat{\mathbf{y}}$.
Every model is getting a certain input, consisting of information about the historic energy consumption, as well as –depending on the model– additional features.
Based on the provided input, the model tries to forecast the energy consumption for the next day of a particular household.
The exact processing of the various inputs for the different models is explained in section 5.3.

5.1.1 What makes it challenging?

The task of forecasting the electricity demand of individual households is extremely challenging due to the high volatility and randomness in the energy consumption curves of individual residents. Figure 5.1 shows the electricity demand profiles of three customers within one week in April 2018 with 15 minute time resolution.

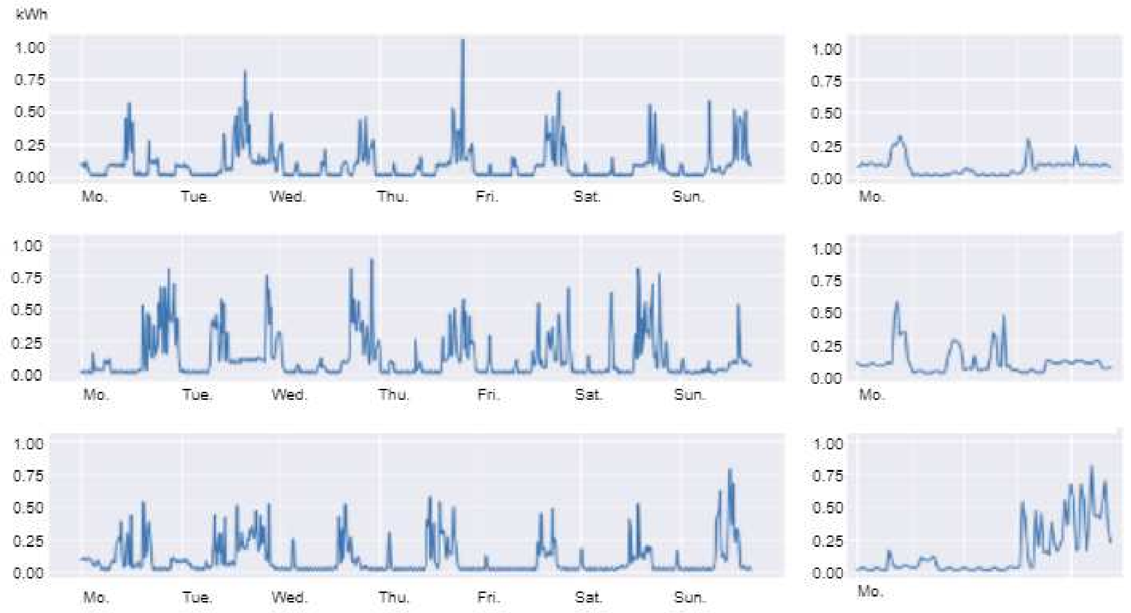

Figure 5.1: Electricity demand profiles of three customers. On the left are the consumption profiles for a whole week (Monday-Sunday). On the right is the following day (Monday) of the same household.

Time of day, weather conditions, information of appliances, costumers or dwelling specific information have been found to have an impact on the energy consumption patterns of occupants, and therefore could be valuable for the forecast, if provided as input the the model. However, the biggest share accounting for the amount of energy consumed is the residents lifestyle [45].
Therefore, by analysing the historic consumption patterns, as well as exogenous factors, machine learning models might be able to pick up certain characteristics that result in reasonable energy consumption forecasts. This is no trivial task, as the consumption patterns vary drastically from household to household. Even one household has big variations in its electric energy consumption on a day to day basis.

5.2 The Approach

This section outlines the main questions this thesis is trying to answer. The experimental setup to answer those questions, the used data and the data processing methods are explained.

5.2.1 Questions to answer

In the following, the main questions are stated and the process for answering them is elaborated.

- **Question Number 1:**

Are recurrent neural networks useful? In particular, the usefulness of LSTM networks for low-level energy consumption forecasting is investigated. To answer this question, LSTM models get compared to feedforward neural networks. This comparison shows if the memory capability and the sequential processing of LSTM networks are advantageous compared to the processing of a feedforward neural network. Both neural network approaches, the LSTM models as well as the feedforward models get compared to benchmarks commonly used in forecasting studies. Using benchmarks as a comparison gives insights if the use of computational expensive neural networks is actually justifiable.

- **Question Number 2:**

  What is the effect of additional features on the forecasting performance? To answer this question two slightly different LSTM networks get compared: One LSTM model –*LSTM all*– gets input, carrying information about: Historic energy consumption data, autoregressive shifts, temporal information (time of day, day of the week, holiday) and meta-information (apartment size). The second LSTM model –*LSTM one*– only receives the historic energy consumption as input. A comparison of the performance of these two models, once with and once without the additional information identifies the impact of the additional features on the forecasting quality.

- **Question Number 3:**

  What is the best use case of deploying an LSTM network? *General* vs. *Individual* approach? To investigate this question, one LSTM model gets trained on all available data, that is it learns based on all households. This model called the *General* model then gets compared with LSTM models that are only trained on the data of one particular household. These models are called *Individual* models. A comparison of the forecasting quality of both methods on various data sets identifies the most suitable approach. The *LSTM all* models are used for this comparison.

- **Question Number 4:**

  What is the effect of the data granularity? In the literature, several different forecast horizons and sampling rates of the data are regarded. In [3] it is shown that the shorter the forecasting horizon and the lower the data resolution, the better artificial neural networks (ANN) are able to perform on pointwise predictions. This effect is mainly attributed to the out-smoothing of energy consumption profiles when going from smaller to bigger time intervals. For this reason, two different data resolutions of the energy consumption recordings are regarded in this thesis. (a) 15 minute time intervals, and (b) hourly time intervals. The above-mentioned questions 1, 2 and 3 are investigated for both time resolutions. This approach gives insights into (a) the effect, that the data resolution has on the forecasting performance, and (b) if and how the answers to the questions 1-3 do change with the variation in data granularity.

5.2.2 Implemented Models

The models deployed to answer the above questions are described in the following. For question 1 and 2 neural network models, as well as benchmarks, get compared to each other. Question number 3 uses LSTM models, that are trained with additional features.

Naming of Neural Network Models:

- **LSTM all**: This refers to the LSTM approach that is trained with additional features. The model is trained with the following input information: the historic energy consumption, autoregressive shifts of the historic consumption (the hour before, a day before, week(s) before), calendric information (time of day, day of the week, holiday) and meta-information (apartment size).
- **LSTM one**: Describes the LSTM approach that is trained only with historic energy consumption.
- **FFNN**: This refers to the feedforward neural network model that also only has the historic energy consumption as input to the model. The FFNN, therefore, varies from the LSTM one model not by the input it receives, but by the way the input is processed.

Naming of Benchmark Models: Comparing the neural network models to the benchmarks gives insights about the usefulness of neural network approaches. These rather simple benchmarks are found to be difficult to beat by various machine learning and statistical model techniques in the realm of forecasting the electric energy consumption of individual households.

- **"naive"** Benchmark: This benchmark takes the day before as the prediction for the next day. Often encountered as the persistent benchmark in the literature.
- **"soph"** Benchmark: This benchmark takes the same day the target day one week before, as its predictions. For example, if the next day to be predicted is a Monday, the "soph" benchmark will use the last Monday as its prediction. The naming convention "soph" comes from the abbreviation of sophisticated.
- **"mean"** Benchmark: This benchmark takes the average consumption of the week prior to the target day. For instance, if Monday is to be predicted, the "mean" benchmark computes the average energy consumption per time interval of the prior week (Monday to Sunday) and takes this as its forecast.

5.2.3 Error and Performance Metric

To evaluate the forecasting quality of the different methods mentioned, two metrics are used in this thesis.

The Error Metric

In this thesis the first error metric, based on which to judge the performance of the varying models is the *Mean Absolute Error* or *MAE* described by equation 5.1, where n refers to the length of the forecast. n is either 24 in case of hourly time intervals, or 96 in case of 15 minute time intervals.

$$MeanAbsoluteError = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{5.1}$$

This pointwise error metric is the starting point for evaluating forecasting quality. This error gives an intuitive first impression of the goodness of the forecast and, describes by how far off on average the forecast $\hat{\mathbf{y}}$ is from the true energy consumption $\mathbf{y}$ per time interval. However, this pointwise error metric has certain disadvantages and might not be appropriate for a definitive judgment of the forecast. A more in-depth discussion, why this error metric alone does not allow for a conclusive evaluation of the models, is provided in section 7.2.1.

The Performance Metric

Due to the fact, that the pointwise *MAE* error metric might not capture the performance of the different models in full detail, a second performance measure is introduced here. In combination with the *MAE* this performance metric named *Daily Energy Consumption Deviation*, describes the forecast in a way, that (together with the MAE) allows for a more nuanced judgment of the differing forecasting performances.

$$Daily\ Energy\ Consumption\ Deviation = |\sum_{t=1}^{n}\hat{y}_t - \sum_{t=1}^{n}y_t| \tag{5.2}$$

Where n again is referring to the length of the forecast, and t is referring to a particular time step. Therefore, $\sum_{t=1}^{n}\hat{y}_t$ is the predicted total amount of energy consumed for one day, and $\sum_{t=1}^{n}y_t$ is the true total amount of energy consumed for that day.

5.3 Data and Data Processing

The household stock on which this analysis is performed on, is from Aspern Seestadt Vienna.[1] Although this thesis is aimed at investigating the suitability of neural network forecasting approaches for the P2PQ project mentioned in section 2.2; the smart meter recordings from the households belonging to the P2PQ project where limited in quality and time duration. Therefore, a data set with the smart meter recordings coming from Aspern Seestadt is used. The households in both cases a comparable in size and demographics. For this reason, insights found on the Aspern Seestadt data, are likely transferable to similar apartments in the P2PQ project.
There are a total of 127 households ids, with 117 unique dwellings over a three year time period from January 2016 to December 2018. There are more households ids then dwellings, due to the fact, that if residents move out during the regarded time period, the incoming new resident gets a unique id, although living in the same dwelling. The size of the different dwellings ranges from 40 to 120 square meters. Not all households have smart meter recordings over the total three year period. The baseline consumption is very low, due to either low consumption patterns or high vacancy.

[1]The data is provided by the ASCR, the Aspern Smart City Research Group.

5.3.1 Distinct Data sets

For the investigation of the empirical questions, six distinct data sets are chosen. Every data set consists of one *Train*- and one *Test*- partition. Each data set is made out of samples. Every sample is to be understood to form one input/target pair for the neural network models. The exact processing of those samples is explained in the coming sections. The naming convention of those data sets for the rest of the thesis and the motivation behind choosing them is given below:

- **"Simple"**: This data set includes the samples, of all households that have valid energy consumption recordings for at least five months. This data set got filtered only for missing values. Missing values do mess up the training process of the neural network models, and samples including missing values are therefore not regarded.

- **"Extra"**: This data set is a modification of the "Simple" data set. It performs an extra filtering step. Not only samples with missing values and non-valid recordings get discarded, but also samples, that do not satisfy certain minimum consumption criteria. These criteria filter for samples that have at least a moderate amount of consumption happening. A modest energy consumption threshold of 2 kWh per day on average is used, as well as at least one energy consumption peak of 0.5 kWh for the hourly data, and 0.125 kWh for the 15-minute data. If those criteria are met by the processed sample, it is part of the "Extra" data set; if not it gets dropped, and is only part of the "Simple" data set.

  The choice of this extra filtering step based on minimum consumption criteria is motivated by the fact, that a lot of very low consumption profiles are part of the overall "Simple" data set. These low consumption profiles, that might not really convey consumption habits of residents, but just reflect the consumption profiles of vacant dwellings for long time periods, with the only consumption recorded being the self-consumption of the smart metering device, or standby consumption of appliances. By comparing neural network models trained on the "Simple" and on the "Extra" data set, it might be seen if the inclusion of very low energy consumption profiles messes up the neural network models, and if there is a change in models forecasting behavior/performance when they get dropped. It is to be expected, that the exclusion of the lower consumption profiles does lead the models to train on more "real" consumption days. For this reason, models trained on the "Extra" data set are expected to forecast higher overall energy consumptions than the models trained on the "Simple" data set. By applying this filtering step, roughly 25% percent of samples do not meet the criteria for the minimum consumption and are therefore not considered in the "Extra" data set.

- **"X23"**, **"X70"**, **"X111"** and **"X115"** - These data sets corresponds to the samples of households with id X23, X70, X111, X115.

  The motivation behind selecting these particularly four households is, that they met certain criteria, that made them useful to be included as individual household data sets. There has to be a minimum amount of data available for a reasonable evaluation, therefore households with not at least two and a half years of correct recorded consumption profiles got discarded. Furthermore, households should have a regular consumption happening, to be able to draw conclusions, if the proposed methodology is of any use, for households that show steady and realistic consumption behavior. There were more households in the data set meeting those criteria, choosing four households is seen as a reasonable trade-off between interpretability of the results and being able to investigate the questions. However, it has to be mentioned that including only those four individual households does not allow for a conclusive answer. The limited amount of data available for individual households is an issue. Inferring distinctive answers from this limited amount of data is not valid. However, it is investigated, if the amount of samples, especially in the case of individual household data sets, is sufficient for the task of forecasting based on neural network methodologies.

In summary: There are six distinct data sets for the analysis. Two, -*"Simple"*, *"Extra"*-, that include all households, that have of at least five months of correct consumption recordings. And four data sets of individual households -*"X23", "X70", "X111"* and *"X115"*-. It has to be mentioned, however, that the choice of these six data sets, the filter criterion for the *"Extra"* as well as the assumptions for the individual households -*"X23", "X70", "X111", "X115"* are somehow arbitrary. Therefore using different data sets and different filtering criteria can lead to different outcomes of the analysis. Figure 5.2 shows the daily energy consumption distribution of the different data sets. It can be seen, that the filtering step for minimum energy consumption threshold nudges the "Extra" data set to the right. Moreover, it can be seen, that the four individual households show a higher consumption than the average daily energy consumption of all households. The individual households also vary in their distribution.

Train/Test Split

For assessing the performance of the neural network models, the available data sets are each split into two parts (a) *Train-* partition and (b) *Test-* partition.
The models are trained on the train set and after the training process, -during which the models find the optimal set of parameters describing the functional relationship between input and target-, the model is evaluated on the test set. the performance of the models on the test set gives insights if the models are able to generalize well to new unseen samples.
The train/test split ration is set to 0.75. Therefore 75% of each data set belongs to its train partition and 25 percent make up the test partition. Every household is split based on this ratio. Only households with at least five months of available data are included for this analysis.
The train/test split is done during the data processing phase, before the actual filtering for missing values or for minimum consumption is performed. Therefore, the final data set sizes are slightly off, from the exact 0.75/0.25 split ratio. The split is done with regard to keeping the temporal order. That means, the train set is before the test set. Another methodology on which to perform the train/test split could be to split based on households itself. For instance, choosing all consumption profiles from the first 80 households of the total data set and use the residual households for

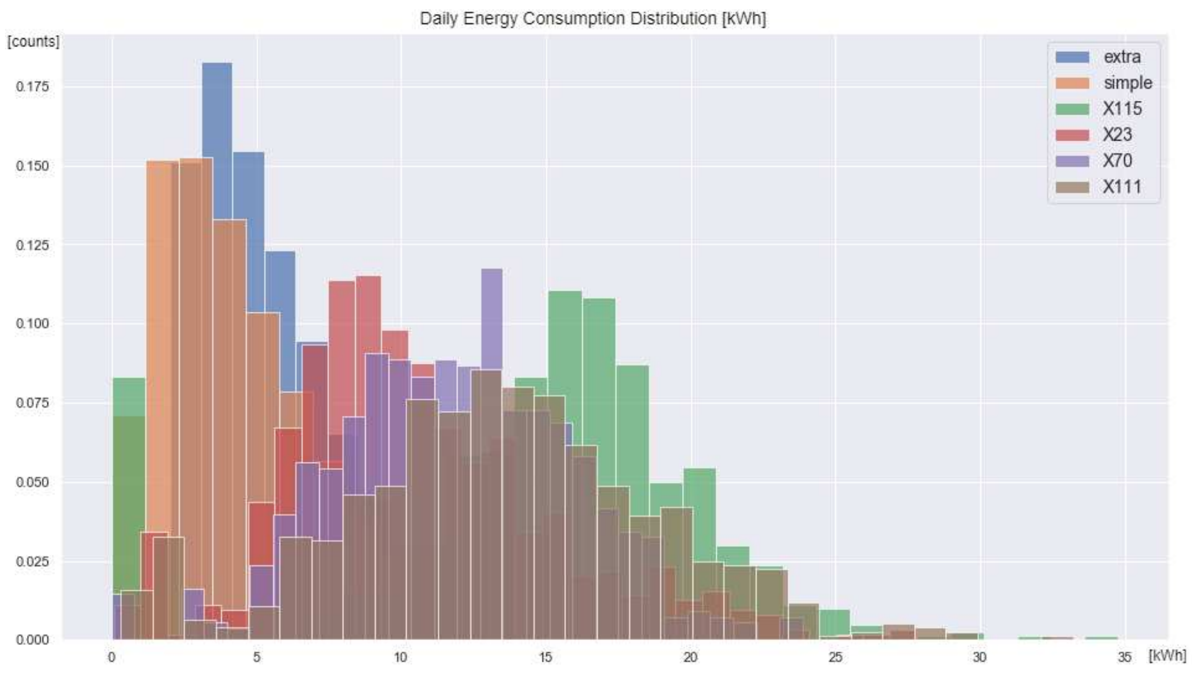

Figure 5.2: Daily Energy Consumption Distribution of the six distinct data sets. The y-axis is describing the normalized count of days over the totality of each data set, that falls into the corresponding bin. The x-axis describes the amount of energy consumed per day in [kWh].

the test set. This split method, though would not allow for a per household model comparison -*General vs. Individual*- and is therefore not used.

How big are the data sets?

The six distinct data sets, -*"Simple", "Extra", "X23", "X70", "X111", "X115"* are each split into a *Train* and *Test* partition. After this split, the amount of data -that is the number of samples- making up the *Train*- and *Test*- partition of every data set is shown in table 5.1.

Time Interval	Partition	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
15 Minute	Train	55.000	40.000	700	600	650	600
	Test	20.000	8.800	220	230	180	150
Hour	Train	50.000	44.000	650	650	650	650
	Test	23.000	8.600	260	230	210	210

Table 5.1: Size of the train and a test partition of each distinct data set for 15 minute and hourly time intervals.

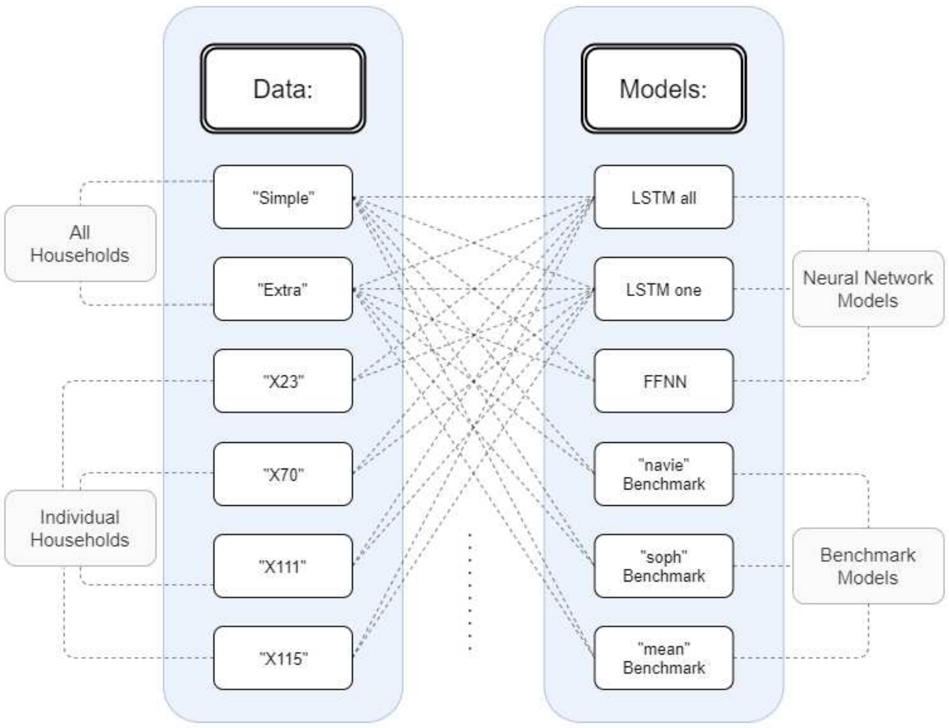

Figure 5.3: Graphical illustration of the different forecasting models and different data sets on which this analysis is performed. Each neural network model is trained on every of the six data sets.

5.3.2 What is one sample made of?

Input/Target

What does the input to the different models look like and how is the data processed? To start, the energy consumption profiles of every individual household in the Aspern Seestadt household stock are processed, to get the pure –only the electric energy consumption recordings, from the smart meter– data for every household. The processing steps are undertaken to enrich the data with useful information, that might hold predictive power, and can be utilized by the models to output accurate energy forecasts. The bulk of the processing steps is then only used by the *LSTM all* model, whereas the *LSTM one*, as well as the *FFNN* model, do only take the historic energy consumption as input. The following steps are performed for every household:

- temporal encoding: In this step, the temporal aspect of the consumption profile is regarded, and the temporal information gets extracted. Temporal information might have an effect on the forecasting ability of the model. Features generated in this step include:
  - hour of the day
  - day of the week
  - month of the year

  All these features might carry predictive power for the models, to perform reasonable forecasts. The information, if the considered day is a Sunday in winter might have useful information about the consumption patterns, when compared to a weekday during a different season.

- holiday encoding: This step includes boolean features, if the day is a holiday, or if the next day is going to be a holiday. The consumption habits are likely to change on holidays, therefore these two boolean features are included.
- autoregressive features: Autoregressive features, are temporal shifts in energy consumption to represent certain lags in the energy demand profile of the household. The temporal shift considered are:
  - one hour before
  - one day before
  - one week before, as well as two weeks before, three weeks and four weeks before.

  It has to be said, that the autoregressive shift performed here, might not be the optimal shifts for all households. It might be that one household has different optimal lags, that have the most predictive power. Overall, the above-mentioned shifts should include additional information about the consumption habits of the households.
- meta information - The last is the inclusion of *meta* information. In this step information from a survey about the inhabitants of the Aspern Seestadt household stock is used to enrich the input with further details. Information included in the survey are:
  - size of the dwelling
  - educational status
  - average age of residents
  - technology affinity
  - appliance information
  - household income

  Unfortunately, many answers are sparse, and only the dwellings size was available for all households. Therefore, the size of the dwelling is used as the only *meta* feature in this step.

These steps are responsible for enriching the input to the *LSTM all* model. The other neural network approaches process only the historic energy consumption feature, this is schematically shown in figure 5.4. Up to this point, the data is only enriched, but not yet usable by the neural network models. In the following paragraphs, the process for getting the data into the right shape, so it can be used by the neural network models is described.

Look-back Window: The look-back window describes the amount of time that is regarded from the historic energy consumption profiles of each household. For example, a look-back window of one day means, that one day gets sampled and is used as input to the model. This selected time horizon is then enriched with

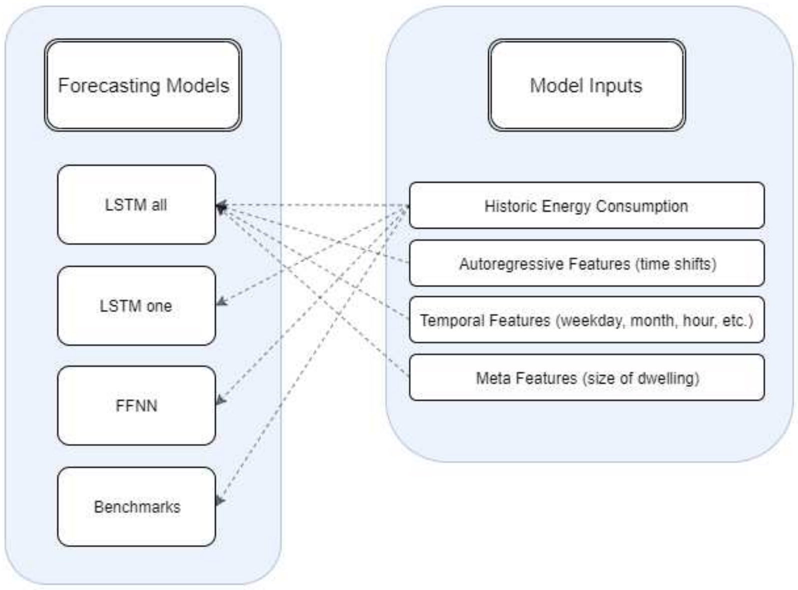
Forecasting Models
LSTM all
LSTM one
FFNN
Benchmarks
Model Inputs
Historic Energy Consumption
Autoregressive Features (time shifts)
Temporal Features (weekday, month, hour, etc.)
Meta Features (size of dwelling)

Figure 5.4: The differing forecasting models with its different inputs

additional features, according to the processing steps explained above, to make up the input of one sample. In this thesis, the look-back window is chosen to be one week. This choice is motivated by the fact, that a whole week might carry useful information about the consumption characteristics of a household. For 15-minute data resolution, this results in input with length 672, and for hourly time intervals, the inputs have length 196.

Sample: One input plus its corresponding target output form one sample. The different samples are processed by a rolling window approach of one day. Every sample is made of the input, –a week of historic energy consumption recordings, (and in the case of the *LSTM all* model, additional features)–, and one target, –the true target day, following the input week.

Shape of samples: The different models, –*LSTM all*, *LSTM one*, and FFNN–, have different inputs. The *LSTM all* processes the historic consumption profiles as well as the additional features. This model has, therefore, –in the case of 15-minute data–, an input of shape (672, 53). The first dimension of 672, describes the number of time steps this input is made out of. Each of those 672-time steps then holds a feature vector of size 53, that captures the energy consumed at this time step, plus all the additional features. The 53 values in the feature vector are as follows:

- energy consumption values: the energy consumed at this time step, one hour before, one day before, one week before, two weeks before, three weeks before and four weeks before. Resulting in seven features corresponding to historic energy consumption.
- temporal data: hour of the day, day of the week, the month of the year. These features are one-hot encoded, resulting in 24 (for hours) + 7 (for days) + 12 (for months) features. In total 43 features dealing accounting for the temporal data.

- holiday: two boolean features describe if the present day is a holiday and if the next day is a holiday, resulting in two features dealing with holiday encoding.
- meta feature: one feature in the 53-dimensional feature vector is held by the scaled size of the household, resulting in one feature dealing with meta-information.

All these features make up the 53-dimensional feature vector, that is present for every single time-step of the 672 in the case of 15-minute data, and the 196-time steps in case of hourly data. The *LSTM all* model uses all available information, -historic consumptions, autoregressive, temporal, holiday and meta features-. The *LSTM one* model only regards the energy consumption as input, –no additional features–, but still processes the input in a sequential way. The *FFNN* model also only regards historic energy consumption. However, the processing of the input is not sequential, as in the case of the LSTM models. The shapes of the inputs and outputs for 15 minute and hourly data of the various models are shown in table 5.2.

Time Interval	Model	Input	Target
15 Minute	LSTM all	(672, 53)	(96,)
	LSTM one	(672, 1)	(96,)
	FFNN	(672,)	(96,)
Hour	LSTM all	(168, 53)	(24,)
	LSTM one	(168, 1)	(24,)
	FFNN	(168,)	(24,)

Table 5.2: Shape of input and target arrays for the three different neural network models.

5.4 Problems and Explanations

Up to this point the main empirical questions, this thesis tries to investigate are stated. The data sets on which the final analysis is performed and the way they get processed is mentioned. However, there are caveats in the processing of the data and in the setup for comparing the various models. This section is meant to emphasis some of the potential difficulties and help in fostering the understanding of the various decisions included in this study.

Data Cleaning and Filtering

The data cleaning procedure used in the filtering and processing stage of this work is rather rude. Samples got processed in the described way to form input/target pairs. After the samples are created, the cleaning step is performed. Every sample, where either the input or the target has missing values, is dropped from the data set. This leads to a 30% loss of samples from the ”raw” data set to the *”Simple”* data set. This is a significant loss of data. This rude cleaning and dropping of samples could be mitigated by data insertion techniques, or by interpolation methods. However, due to the fact, that the missing values stretched over various time periods in the energy consumption recordings throughout the total household stock, no appropriate insertion technique was found, that could insert reliable data, and therefore not mess up the training inference of the models. For this reason, the rude cleaning approach is chosen.

Scaling

Scaling in the machine learning domain is dealing with the problem of putting the values (input and target) into a preferable range of the particular machine learning models. Some machine learning techniques are very sensitive to the range of values, others are more robust. Neural networks are known to be more sensible, and it is preferable to get the range of the input and target values into an ideal range so that the information can flow optimally throughout the network. However, there is no upfront ideal solution to this task. One of the most obvious problems emerges, if the output of the activation function in the last layer, is not suited for the range of the output values. That is if the activation function constrains the output values, and the real value is outside of this limit. For instance, the sigmoid function bounds its output to a range between 0 and 1. However, the target values need not be in this range. When this is the case, it becomes clear that this model would not be able to learn optimally and predict accurately. For this reason, the first obvious benefit of scaling is getting the target values into the output range of the preferred activation function.

The other important point for scaling is the fact, that the signal from the gradient flow is more stable by scaling the input and target. Usually both, the input and target values, get scaled into either (a) the range between 0 and 1, –this can be achieved via Min-Max-scaling–, or (b) get the values *standard scaled*, forcing the values into a normal distribution with 0 mean and unit variance.

The problem of scaling in this thesis arises from the fact, that the overall consumption is very low, with most energy consumption values near 0 kWh. Therefore, when applying one of standard scaling methods like Min-Max-, Robust- or standard-scaling, the resulting range of outputs either does not fall into the optimal range for neural networks or is very skewed. After different scaling methods were analyzed, the approach of not using any scaling is used. Due to the fact, that all energy consumption values recorded of the total household stock over a three year period do not exceed 1.5 kWh in the case of 15 minute time intervals and 4.0 kWh in the case of hourly intervals, this approach is chosen as a reasonable option.

Comparison of Models

A fair comparison between the various neural network models is hard to accomplish. First, there is the comparison between the three neural network models *LSTM all*, *LSTM one* and *FFNN*; and second, there is the comparison of the *LSTM all* models in terms of regarding the *General* and the *Individual* model. Furthermore, there is a big difference in the size of the various data sets, as well as the difference in the data points when regarding 15 minute or hourly time intervals.

The optimal model architecture, appropriate for one particular data set, characterized by data resolution and size of the data set, might not be the optimal architecture for a different set of samples.

Therefore it is not appropriate to use the same architecture for the *LSTM all, LSTM one* or *FFNN* model on all data sets, disregarding the size of the available data. Because a valid comparison between the various use cases of the models is difficult, a set of heuristics is used, to make a reasonable comparison possible.
For this reason, in this thesis, there are 12 different model architectures proposed and trained. The first distinction is made in terms of data resolution. Is the model trained on (a) 15 minute time intervals or (b) on hourly time intervals. The second distinction is based on the data set it is trained on: is the model trained on one of the *"General"* data sets, that is either *"Simple"* or *"Extra"*, or (b) on one of the *Individual* data sets: *"X23"*, *"X70"*, *"X111"* and *"X115"*. For every one of these four cases: Data resolution either 15 minute time interval or hourly time interval, and data set either belonging to the *General* or *Individual* category, there are the three different neural network models: *LSTM all*, *LSTM one* and *FFNN* resulting in twelve models. Every model has one hidden layer.

Model Architecture

The main difference in those 12 models is the number of neurons in the single hidden layer. With the number of neurons in the hidden layer, the number of parameters of every model can be set. The main heuristic on which to compare the models is chosen to be the number of parameters, every model has. Therefore, when comparing the three different neural network models in the case of 15 minute time intervals and *General* data set, all three models are chosen to have a similar magnitude of parameters, resulting in a comparable representational power. However, due to the fact, that the *LSTM all* model does consider the additional features, its number of parameters is bigger then that of the *LSTM one* model, although both having the same number of neurons in the hidden layer. The number of neurons in the *FFNN* model are then chosen, so that its number of parameters, is comparable to the number of parameters of the *LSTM one* model.
The next heuristic is for choosing the starting point of the number of parameters in the first place: The number of parameters of the *LSTM one* model trained on a *General* data set is chosen, so that it is close to the number of samples in the *Train*-partition of the *"Simple"* data set. This means the *LSTM one* model trained on a *General* data set has about 50.000 parameters. Going from there, the number of parameters gets cut in half when considering *Individual* data set. Furthermore, the step down of the number of parameters going from the 15-minute time interval to the hourly time interval is again by a factor of two. Therefore, the models trained on a *Individual* data set and hourly data resolution have approximately a fourth of the parameters of that are trained on a *General* data set and 15-minute data resolution. The number of neurons in the hidden layer and the corresponding number of parameters of each model for the different cases is shown in table 5.3.
A rule of thumb, to mitigate overfitting, is to build a model with roughly the same number of parameters as are training samples available for the problem. This is a very rough rule, and more than matching the number of parameters to the number of training samples, it is supposed to help in cases where there is limited data available. In the case of limited data, a model that has drastically more parameters then training samples, is more likely to overfit. Due to its representational power, it is able to find relationships in the input to output pairs, that are not true and might be found purely by chance. The bigger the number of parameters, and simultaneously

Time Interval	Approach	Model Type	Num. of Neurons	Num. of Parameters
15 Minute	General	LSTM all	90	60.576
		LSTM one	90	41.856
		FFNN	50	38.546
	Individual	LSTM all	60	33.216
		LSTM one	60	20.736
		FFNN	25	19.321
Hour	General	LSTM all	50	22.024
		LSTM one	50	11.624
		FFNN	50	9.674
	Individual	LSTM all	30	10.824
		LSTM one	30	4.584
		FFNN	25	4.849

Table 5.3: Number of model parameters

a relatively small data set size, increases the chance of finding random patterns in the data. The relation of parameters to the number of training samples is not the only factor playing a role in overfitting; other factors include regularization methods, training time, or data diversity itself. For this reason, neural network models trained on one of the *Individual* data sets are chosen to have fewer parameters then the models trained on the *General* data sets, by a factor of two. The three models *LSTM all*, *LSTM one* and *FFNN* have the same architecture each for the two total data sets *"simple"* and *"extra"*, as well as for the four individual household data sets *"X23", "X70", "X111", "X115"*. This is done because the *General* data sets have roughly the same size, as have the household data sets, shown in table 5.1. Furthermore, a distinction of the model architecture for the 15-minute data and the hourly resolution data is made. This is because the finer resolution comes with more energy consumption recordings causing a richer representation of the input and the target. The increase in the number of recordings when looking at the hourly resolution data to the 15-minute resolution data is by a factor of four. For a fair comparison of the models and to see what effect the time resolution has one the overall forecasting performance, the models working with the 15-minute resolution are chosen to have roughly 2 times more parameters then the hourly resolution data.

Number of hidden layers Another heuristic is used for the choice of the number of hidden layers. For many applications, a simple model with only one hidden layer is a reasonable starting point, and the choice of the number of neurons in the one hidden layer should be between the number of inputs and the number of outputs, to allow for an abstraction of the feature space in the hidden layer and ultimately use those abstraction found in the hidden layer to produce the final output. For this reason, the model architectures for all neural network models are chosen to include one hidden layer. However, the choice of the number of neurons for the hidden layer does not allow for its size to be between the input and the output sizes. This is because more weight is given for the decision to match the number of parameters (of the *LSTM one* model) roughly with the number of samples (in the *"Simple"* Train-partition), and scale all other models in relation to this according to the rules mentioned above.

Hyperparameter Search The above-mentioned choices for the number of parameters, number of layers as well as the number of neurons per layer might seem a little bit "unsophisticated". Through the empirical work of this thesis, it was found, that more "sophisticated" models, equipped with more layers, regularization methods -L1, L2, elastic net, dropout, recurrent dropout-, batch-normalization did not increase the model performances. Due to the inherently challenging nature of forecasting energy consumption profiles of individual households, it was found, that an increase in "sophistication" of the models did not come with an increase in model performance. Other hyperparameter choices were also investigated, like different optimizers, -RMSProp, SGD, Adam, AdaGrad,-, different batch sizes during the training phase -8,16,32,64-, shuffling of the data, number of epochs -25,50,75,100,150- and different choices of activation functions –tanh, relu, elu, linear–.

Choices of Hyperparameters

Many different sets of hyperparameters where explored during the modeling phase of this analysis. More sophisticated models, with an additional number of hidden layers, regularization and batch normalization where included. Experimentation with different activation functions and Loss-functions were performed. In the end, a rather modest architecture and set of hyperparameters are chosen for this thesis. Some of the main explorations, its effect, and the final hyperparameter choices are listed below:

- Activation Function: The activation function in the LSTM layers is chosen, as is default in the LSTM implementation in *Keras* to be the tanh. The activation function in the *FFNN* hidden layer is chosen to be the *relu*. The activation function for all neural network models in the output layer is chosen to be the *linear*. These choices of the activation function showed the most stability in the training phase. The *elu* and *relu* were tried as the activation function in the output layer, but both lead, depending on the used batch size, to unstable training. The *relu*, *elu* and *sigmoid* were also tried for the LSTM layer, however, they did not result in more reliable forecasts.

- Number of hidden layers: The number of hidden layers, were tested to range from $1-5$. Due to the modest amount of data samples to start with, and the fact, that a single hidden layer is sufficient for many tasks, only one hidden layer is chosen. The increase in layers did not bring performance gain.

- Batch Normalization: This did bring a slight performance gain for the *FFNN* models; but not for the *LSTM* models. It is not used in the final model architecture.

- Regularization: Varying regularization methods were tried during the modeling phase. Dropout, applied to all layers did lead to slightly more stable results, depending on the chosen batch size. Lasso-, Ridge- or Elastic Net-regularization did not increase the accuracy of the models. This can be due to the fact, that the data of individual household energy consumption profiles is very volatile to start with. These erratic and volatile profiles function as a kind of regularization themselves, hindering the models from overfitting. For this reason, no regularization, but a rather moderate model architecture is chosen.

- Batch size: This hyperparameter had a big effect on the behavior of the models. The batch size was chosen to $8, 16, 32$ and 64. The batch size of 32 did show the most reliable and repeatable outputs. 32 is therefore chosen as the batch size.

- Optimizer: Adam, AdaGrad, RMSProp, and SGD were tried as the optimization algorithms for finding the optimal set of parameters in the model. No significant performance differences, as well as computational time differences, were observed. The Adam optimizer is chosen as the optimizer for the models.

- Number of Epochs: The number of epochs clearly has one of the biggest effects on the final model behavior. In the modeling phase, the models got trained for 25 epochs at a time, up to 150 epochs, and the varying performance and forecasting characteristic got analyzed after every 25-epoch step. It could be argued, that after only a few epochs overfitting was starting to take place, because the error (the mean absolute error) on the test set started to rise, whereas the error on the train set kept going down. The first two epochs come with a significant reduction in the mean absolute error. After those first two initial epochs, the error decreases very slightly. However, the model's forecasts become more reliable with more training time. For instance, the training error does decrease only slightly from epoch five up to epoch twenty, but the resulting forecasts are in the first case more erratic and are often in the negatives. Therefore, although the error does not change that much, the model's behavior does seem to adapt longer then the training error might indicate. In the end, the number of epochs is chosen to 50 for all models and all data sets.

- Cost-function: The used Loss-functions (or Cost-function) were the, mean squared error Loss-function, and the mean absolute error Loss-function. Both are point-wise error functions. They showed slight differences in the forecasting characteristics. At first, during the analysis only the mean-absolute-error was chosen as the Loss-function, describing the amount of energy (in units of kWh) that the models are off on average over all time steps. This choice was taken because the first main error metric based on which to judge the performance of the various forecasting approaches is the mean absolute error metric 5.1. However, because the pointwise error metric does not explain the "goodness" of a forecast in full detail, especially for the low voltage level, the performance metric *Daily Energy Consumption Deviation* is adopted as a second evaluation metric. When taking both metrics into account, for judging the "goodness" of a forecast it was found, that the models trained based on minimizing the mean absolute error Loss-function, did achieve better results then the models trained on the mean squared error Loss-function, when evaluated with the mean absolute error metric. However, after taking both metrics into account, the models trained via the mean squared error Loss-function look like the better choice in many cases, resulting in more accurate predictions of the amount of energy that gets consumed per day. For this reason, all neural network models are once trained with the mean absolute error and once with the mean squared error Loss-function to show this difference in model behavior.

All in this chapter mentioned neural network models got implemented in *Keras*, the Python Deep Learning Library [43].

5.5 Goal of this analysis

In this chapter, the main empirical questions were stated. The experimental setup, the data, and its processing into input/target samples, as well as potential difficulties, were discussed. Before moving to the results in the next chapter, the following listing gives a short summary of the main concepts this thesis trying to investigate by the setup discussed in this chapter:

- Is it valuable to use recurrent neural network methodologies for the task of forecasting day-ahead electric energy consumption on the level of individual households?
- Is targeting "Individual" households via their own models reasonable, or is a "General" model better suited to forecast the energy consumption of households?
- Are additional features -temporal, weather, demographical, appliance, sensor-useful for getting more accurate forecasts? This thesis analyses temporal, autoregressive and the size of the dwelling, as the additional features.
- Taking computational complexity, system complexity, and future development into account, what is the most reasonable method? Neural network approaches, or simple benchmarks?
- Is there a certain point -regarding the data resolution, consumption aggregation,- where it is found that considering additional features yields superior energy consumption forecasts?
- Is the sequential processing of the recurrent neural network models superior to the "normal" processing of feedforward neural networks?
- Are "simple" benchmark methods competitive in predicting next days electric energy consumption?

Not all questions can be answered definitive and conclusive. Nevertheless, this thesis tries to provide some insights and ongoing development in the challenging research field of forecasting energy consumption on the low level of individual households.

Chapter 6

Results

6.1 Explanation of Validation and Results

This chapter shows the results regarding the empirical questions outlined in the chapter above. The results get presented in the following way:
For both time resolutions (a) 15 minute time intervals and (b) hourly time intervals, the three questions 1 to 3 of section 5.2.1 are investigated and the results are shown. Every question is analyzed in terms of assessing the different models and their performances –based on two different metrics– on the six distinct data sets.
In the first step, the performance of the three different neural network models *LSTM all*, *LSTM one* and *FFNN*, as well as the three benchmark models *"naive"*, *"soph"* and *"mean"* get assessed on the *Train*- and *Test*-partition of the *Individual* data set (*"X23"*) based on the mean absolute error, shown in equation 5.1. After that, the results are shown for the *Test*-partitions of all six distinct data sets. (Every of the six distinct data sets has its six neural network models (three with the mean absolute error Loss-function, and three with the mean squared error Loss-function) trained on it, therefore 36 neural network models are part of this analysis step).
Due to the fact, that the pointwise error metric does not explain the goodness of the different methods in full detail; in the second step the same models and benchmarks get assessed based on the performance metric *Daily Energy Consumption Deviation*, shown in equation 5.2. Again, first, the results on the *Train*- and *Test*-partition of the *"X23"* data set get compared before the results on the *Test*-partition of every distinct data set are shown.
By taking both metrics into account, more information is regarded for comparing the various methodologies. These first two steps help to answer questions 1 and 2, about the value of recurrent neural networks and the impact of additional features.
The third analysis step is investigating the *"General"* vs. *"Individual"* approach of *LSTM all* models. For this, the twelve *LSTM all* models (for every of the six distinct data sets two *LSTM all* models are trained; one with the mean absolute error Loss-function, the other with the mean squared error Loss-function) get assessed again first based on the pointwise *MAE* error metric, and second based on the *Daily Energy Consumption Deviation* performance metric. With this step question 3, about which approach (*"General"* or *"Individual"*) is more useful, can be answered.
This concludes the analysis of the empirical questions for the 15 minute time interval data. The same presentation of the results is then performed for the hourly time intervals. By looking at the differences in performance relation of the various models

from the 15 minute time intervals to the hourly time intervals, the effect of data granularity gets exposed and question 4, about the impact of data resolution on forecasting performance can be investigated.
A paraphrasing of the question outlined in section 5.2.1 is stated below to help the reader keep the general idea in mind while going through the result sections:

- Question 1: LSTM method "better" then feedforward neural networks?
- Question 2: Are the additional features useful for more accurate forecasts?
- Question 3: "General" vs "Individual" model?
- Question 4: What is the effect of data granularity?

6.2 15 Minute Analysis

In this section, the first three out of the four questions get discussed for the energy recordings with 15 minute time intervals.

Input/Output: One input/target sample for the *LSTM all* model has the form of $(672, 53)/(96,)$. The two dimensional array with shape $(672, 53)$ is the input to the model. Every one of the 672 time steps holds a feature vector with 53 features, consisting of the encoded data as describe in subsection 5.3.2. An input sequence $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \ldots, \boldsymbol{x_{672}}]$ is given to the *LSTM all* models. Each feature vector per time step -$[\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_{672}}]$- holds 53 feature values. One $\boldsymbol{x_t}$ is made up of $[f_1, f_2, \ldots, f_{53}]$. For the *LSTM one* model the input sequence has the same length 672, however it does not include the full 53 elements of the features vector, but only the one feature regarding the historic energy consumption. One $\boldsymbol{x_t}$ therefore only contains $[f_1]$, where f_1 is the value of the energy consumed at the time step t. Other then the *LSTM* models -which process the input in a sequential order- the *FFNN* does processes it all at once. The input to the *FFNN* model is therefore a single vector $\boldsymbol{x}$ of length 672, where each element is the same as in the case of the *LSTM one* model, namely the feature f_1 holding the amount of energy consumed per time interval. The output for all models *LSTM all, LSTM one* and *FFNN* is a vector with length 96 of the form output $\hat{\boldsymbol{y}} = [y_1, y_2, \ldots, y_{96}]$. Where each element y_t of the prediction vector $\hat{\boldsymbol{y}}$ contains the predicted amount of energy consumed in the corresponding 15 minute time interval.
In summary: The inputs for the different models are of the form: $\boldsymbol{X} = (672, 53)$ for the *LSTM all*, $\boldsymbol{X} = (672, 1)$ for the *LSTM one* and $\boldsymbol{x} = (672,)$ for the *FFNN* model. The target as well as the output of every model is a vector $\hat{\boldsymbol{y}}$ with length $(96,)$.

6.2.1 Value of Recurrent Neural Networks, and the Impact of Additional Features

This subsection sets out to answer, if the LSTM approach is actually useful, and if the utilization of additional features helps in getting more accurate forecasts.

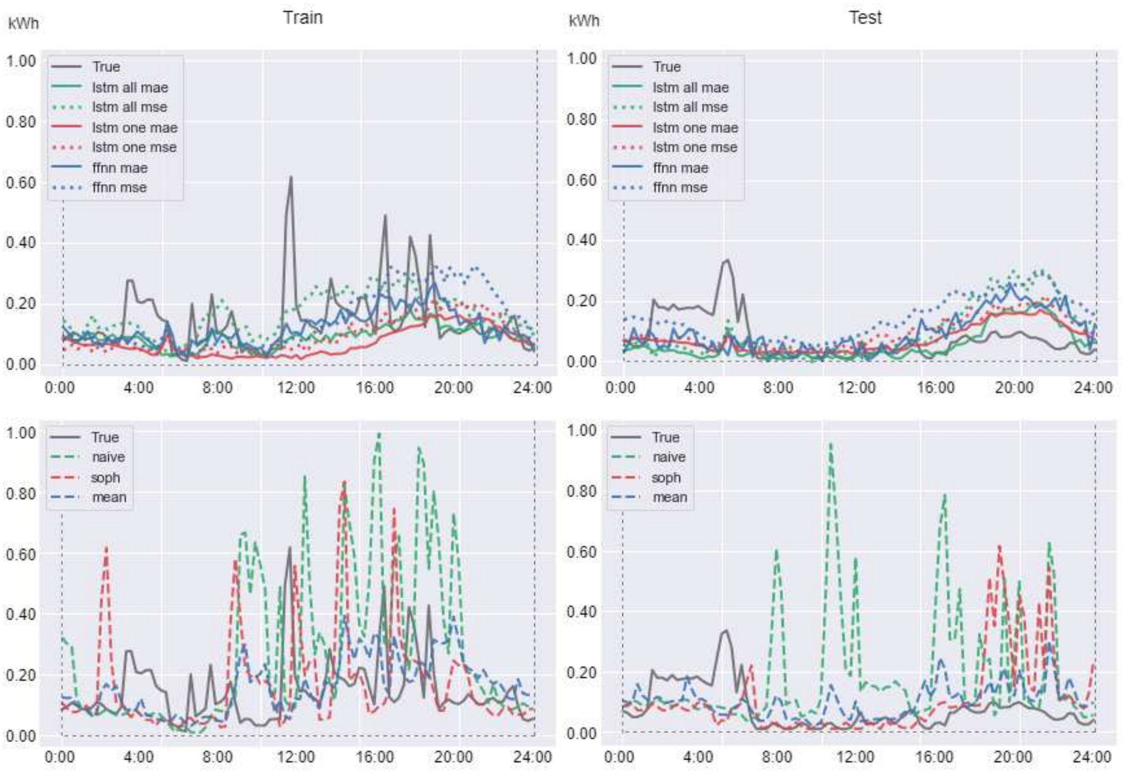

Figure 6.1: Train and Test predictions of energy consumption profiles of various models. The two images on the left are the predictions of the neural network models (top image) and the benchmark models (bottom image) for the same day, taken from the *Train*-partition of data set ”X23”. The two images on the right side are the predictions on a day from the *Test*-partition of data set ”X23”. The neural network predictions are separated from the benchmark predictions, for visual purposes. The three neural network approaches (*LSTM all*, *LSTM one* and *FFNN*) are once trained with the mean absolute error Loss-function (mae, solid line) and once with the mean squared error Loss-function (mse, dotted line), resulting in six predictions. The true day is colored in grey. In the upper images the *LSTM all* models are green, the *LSTM one* and *FFNN* models are coloured in red and blue respectively. For the bottom images the three differing benchmarks ”naive”, ”soph” and ”mean” (dashed lines) are coloured green, red and blue.

Mean Absolute Error Performance

To answer these questions the three neural network models *LSTM all, LSTM one* and *FFNN* –once with the mean absolute error Loss-function and once with the mean squared error Loss-function–, are trained for 50 epochs on the *Train*-partition of every of the six distinct data sets, resulting in 36 unique models. After the training phase, the neural network models and benchmark models are evaluated on the *Test*-partitions of the data sets. First the models are assessed based on the point wise *MAE* error metric (see equation 5.1).

Prediction Characteristics: By looking at the predictions of the different proposed methods, it can be inferred that the NN-models are not able to predict consumption peaks accurately. Figure 6.1 shows the forecasts of the NN-models and the different benchmarks. The two images on the left show a day from the train partition of the data set "X23". The two images on the right show the more interesting case of an out-of-sample day, which is a day from the test partition, on which the "Generalizability" of the models can be tested. The neural network models trained with the mean squared error Loss-function (dotted line) predict an overall slightly higher consumption curve than their mean absolute error Loss-function (solid line) trained counterparts. The "naive" and "soph" benchmark models are very erratic for this particular day. The *"mean"* benchmark has a smoother forecast. This is a forecasting pattern that can be observed over a wide range of days. The neural network models are not able to predict peaks well. The mean squared error Loss-function models tend to predict higher overall consumption curves than their mean absolute error Loss-function counterparts. The "naive" and the "soph" benchmark are highly volatile (simply mirroring prior days). The "mean" benchmark tends to return smoother forecasts, due to the averaging of the seven previous days. No definite conclusions can be drawn from one figure or the visual inspection of several. However, it gives insights into the overall behavior of the neural network models. For low consumption days, and relatively smooth profiles the neural network models are able to pick up this consumption pattern. But they have trouble predicting peaks in energy consumption. This is a problem when training and evaluating models based on a point-wise error metric, due to the double penalty error. This problem and its effects are discussed in section 7.2.1. Furthermore, it can be seen that the "mean" benchmark performs well in these cases. This emphasizes the point, that it is difficult in the realm of individual households to beat those kinds of rather simple benchmarks.

MAE Performance on one data set: In figure 6.2 a comparison of the different neural network models and benchmarks is given. The left bar chart explains the performance of the models on the training set, and the right explains the performance on the test set. The error scale ranging from 0 to 0.12 [kWh] per 15-minute time interval is the same for both charts. The actual chart of interest is the right one. It shows the forecasting performance on data samples, that the models did not see during training. It can be seen that the mean absolute error on the train partition is lowest for the *FFNN* model trained with the mean absolute error Loss-function. However, when looking at the test partition the *FFNN*'s error increases. This is an example of overfitting. Interestingly the overfitting effect seems to be more prevalent

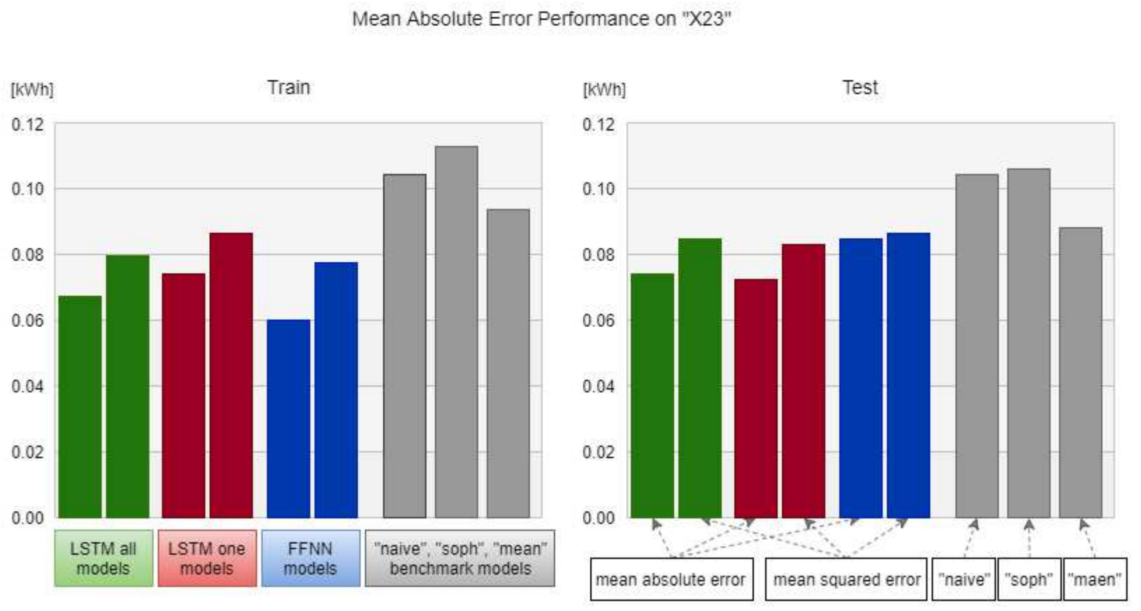

Figure 6.2: Mean Absolute Error performance of different NN-models and benchmarks on *Train*-partition (left) and *Test*- partition (right) of the "X23" data set. The two green bars represent the *LSTM all* models, the red bars the *LSTM one* models and the blue bars the *FFNN* models respectively. The three benchmarks "naive", "soph" and "mean" are represented by the grey bars from left to right. The left bar of each NN-model pair -green, red and blue- is representing the model that is trained with the mean absolute error Loss-function. The right bar shows the MAE results of the same model but trained with mean squared error Loss-function. The lower the bar the better, as the height of the bar indicates the amount of energy (in [kWh]) the model is off in a 15 minute time interval on average over the total *Train*- or *Test*-partition.

for the *FFNN* model then for the other two neural network approaches *LSTM all* and *LSTM one*, although all are trained for the same number of epochs. On the *Test*-partition, the *LSTM all* and *LSTM one* models trained with the mean absolute error Loss-function are the best performers showing similar results with an average error of 0.074 kWh per time step interval (see the column representing "X23" in table 6.1). From the performance on this particular data set it can be said that based on the mean absolute error, the *LSTM* approach does merit a small benefit compared to the *FFNN* model. However, there is no difference in performance between the *LSTM all* model and the *LSTM one* model on the test partition. From these findings, it can be concluded that for this particular data set, and with this time resolution, the approach of additional features does not help in getting a better forecast of the energy consumption for the next day, when evaluated based on the *MAE*.
Comparing the neural network models with the benchmarks, it can be said that the best neural network model with a *MAE* of 0.074 is roughly 43% better relative to the *"naive"* or *"soph"* benchmark. Whereas the *"mean"* benchmarks error of 0.089 kWh is about 20% worse relative to the best neural network models on this particular data set.

NN-models and Benchmark Comparison: For an intuitive comparison between the neural network approaches and the benchmarks, a comparison metric is introduced here and described the relative performance difference of the best neural network model to the best benchmark model. This metric is named *Performance Difference* in this analysis. It is to be understood as an intuitive way to compare the NN-methodologies to the benchmarks. The *Performance Difference* will be used for both performance metrics (the *MAE* and the *Daily Energy Consumption Deviation*) based on which the quality of the different forecasting models is evaluated. Furthermore, the *Performance Difference* does show how the performance relation of NN-models to benchmarks differs from data set to data set, as well as how this relation does change when going from 15 minute to hourly time intervals. In case of the above example considering the data set *"X23"* the best neural network models on the *Test*-partition are the *LSTM all (mae)* and *LSTM one (mae)* with a *MAE* of 0.074 kWh. The abbreviations mae and mse in parenthesis describe the two different Loss-function which were used to train the models. The best benchmark in this example is the *"mean"* benchmark with a *MAE* of 0.089 kWh. This results in a performance difference of 20.3% when calculated according to equation 6.1.

$$Performance\ Difference = -\frac{best_{nn} - best_{bm}}{minimum(best_{nn}, best_{bm})} * 100\% \tag{6.1}$$

Where $best_{nn}$ is the error value of the best neural network model and $best_{bm}$ is the error value of the best benchmark model. Looking at this *Performance Difference* of the best performing neural network model compared to the best benchmark model, gives insights if there is a benefit in using the more sophisticated and computational expansive neural network methods. A big *Performance Difference* indicates superior performance of the neural network models, whereas a negative *Performance Difference* means the benchmarks are better performing. The *Performance Difference* does vary for each data set. It might found that for certain data sets the simpler benchmarks are sufficient, whereas for other data sets (household with more volatile consumption patterns) the NN-models are yielding substantial better performance. The *Performance Difference* is meant to function as a rough and quick measure to judge the suitability of complex NN-models compared to simpler benchmark approaches.

The analysis above does only look at the *"X23"* data set. In the next step, all six distinct data sets are regarded. The results are shown for the *Test*-partition of every data set in table 6.1.

Mean Absolute Error Performance						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
LSTM all (mae)	0.031	0.039	0.074	0.071	0.104	0.091
LSTM one (mae)	0.032	0.038	0.074	0.070	0.101	0.092
FFNN (mae)	0.031	0.039	0.086	0.081	0.122	0.098
LSTM all (mse)	0.037	0.045	0.086	0.072	0.109	0.096
LSTM one (mse)	0.037	0.045	0.084	0.074	0.109	0.100
FFNN (mse)	0.037	0.044	0.087	0.084	0.115	0.103
"naive" bm	0.043	0.055	0.106	0.098	0.140	0.116
"soph" bm	0.044	0.055	0.107	0.098	0.139	0.124
"mean" bm	0.037	0.046	0.089	0.079	0.112	0.098

Table 6.1: Mean absolute error performances of the various models (rows) on the six distinct data sets (columns). The first three rows tinted in grey are the three neural network models trained with the mean absolute error Loss-function, indicated with the (mae). The following three neural network models are trained with the mean squared error Loss-function, marked with the (mse). The last three rows show the results form the three benchmarks. The blue coloured cell of every column indicates the "best" performer on this particular data set according to the lowest *MAE*. The red tinted cell shows the "worst" performer on every data set respectively.

Summary of mean absolute error Results on 15-minute data: In table 6.1 it can be seen that the neural network models tend to have similar behavior on all data sets. Especially the *LSTM all* and the *LSTM one* models are very alike. The three neural network models trained to minimize the mean absolute error Loss-function have a smaller *MAE* then their mse counterparts. The neural network models are better performing then the benchmarks based on the *MAE*. The *LSTM one (mae)* model might be the "best" performer in total, however, the superiority is minor, especially when compared to the *LSTM all (mae)*. No benefit of regarding additional features can be seen. However, the *LSTM* approach does seem to come with a small performance gain compared to the *FFNN* approach, especially when considering the four individual data sets *"X23"*, *"X70"*, *"X111"* and *"X115"*. This means that the consideration of additional features does not help in yielding better forecasts of energy consumption when evaluated based on the mean absolute error.

The sequential processing of the input in the form of the *LSTM all* and *LSTM one* models, however, seems to be beneficial compared to the processing of the *FFNN* model. Furthermore, the neural networks have better forecasts than the benchmarks; between 21% and 7% relative less error, as can be seen in table 6.2 showing the *Performance Difference* for the six data sets.

	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
PerformanceDifference	19.4%	21.1%	20.3%	12.9%	10.9%	7.7%

Table 6.2: Performance difference of the best neural network model to the best benchmark model for every data set according to equation 6.1.

Caveat: It has to be noted, that in this thesis the benchmarks do not really stand a chance in beating the proposed NN-models when evaluated on the mean absolute error. This is because the benchmarks are formed out of historic energy consumption recordings, that come from the same input-feature space that is available to the NN-models. So while the *"naive"* or the *"soph"* benchmark might be reasonable benchmarks to choose, it is likely, that if these benchmarks are the "best" models based on the pointwise error metric *MAE*, then the NN-methods would be able to detect this relationship themselves. If for instance the "best" method really is the *"naive"* forecast, then any of the NN-models would have to take the last 96 or 24 entries in its input. The same goes for the *"soph"* and the *"mean"* benchmark. Those benchmarks are formed from the historic consumption up to a week, that is also available to the NN-models. For this reason, it would be surprising, if the benchmark models would actually be able to beat the NN-models when evaluated based on pointwise *MAE* error, on which the NN-models are distinctly trained on. However, this does not mean, that the benchmarks are not a valid choice. It is clear from the literature on forecasting [27, 33], that one error metric alone does not describe the "goodness" of any forecasting method in full detail. Therefore, in the next section, the *Daily Energy Consumption Deviation* is used and forms a second measure for evaluating the goodness of a forecast.

Daily Energy Consumption Deviation Performance

Motivation for a additional metric: One error metric alone might not explain the overall performance of different forecasting methodologies in full detail. For this reason, it is useful to have a second performance metric bound to the end-use-case of the forecast, based on which to evaluate the models, in addition to the pointwise error.

When only regarding the pointwise error metric it can be seen in table 6.1 that the NN-models perform "better" then the benchmarks (especially those trained with the mean absolute error Loss-function). However, this is only one error metric, and therefore only provides one way to evaluate and look at the problem. Another useful measure to consider is the overall amount of electric energy consumed per day, and how the predicted amount is deviating from the actual true amount of energy consumed every day.

When only considering the pointwise *MAE* error metric, the "best" performing model –based on this error–, might not be useful at all. Especially at the low voltage level, where the demand profiles of individual households are very volatile. Taking the pointwise error metric as the only measure for evaluating the forecasting performance, might result in a forecast, that does not help for the actual end-use-case, although they might be the "best" models based on the chosen error metric [17, 27]. This fact is important in the field of forecasting energy consumption of individual households and will be discussed in detail in section 7.2.1.

Total Deviation: The *Daily Energy Consumption Deviation* does explain the average deviation from the true consumption each day. However, it does not explain if the consumption is either too high or too low, but only that it is off by this amount per day. For this reason, an additional metric is introduced here catch over- or under-prediction over the totality of a data set. Equation 6.2 describes this new additional metric named *Total Deviation*.

$$\text{Total Deviation} = \frac{\text{TotalAmount}_{true} - \text{TotalAmount}_{pred}}{\text{TotalAmount}_{true}} * 100\% \qquad (6.2)$$

Where $TotalAmount_{true}$ is $\sum_{d=1}^{n}\sum_{i=1}^{96} y_i$, and describes the sum of total amount of electric energy consumed over the whole data set partition. Respectively, $TotalAmount_{pred}$ is $\sum_{d=1}^{n}\sum_{i=1}^{96} \hat{y}_i$ and describes the predicted sum of electric energy consumed over the same data set. d is the index for days in the data set partition with length n. The index i is describing the time step interval of every day. Here for the 15 minute time intervals the length of $\mathbf{y}$ and $\hat{\mathbf{y}}$ is 96. For the hourly time intervals, this is then 24.
For example, the total amount of electric energy consumed by a particular household over the total time period of the test data set might be 1000 kWh. The total predicted amount, however, sums up to only 900 kWh. In this case the value for the *Total Deviation* results in 10% according to equation 6.2. A positive *Total Deviation* value, therefore, describes too little predicted energy usage compared to the truth, whereas a positive *Total Deviation* value is the result of too high energy consumption predictions over the totality of a data set partition.

Interpretation of metrics

The main metrics based on which to evaluate the quality and accuracy of a forecast in this work are the *MAE* 5.1 and the *Daily Energy Consumption Deviation* 5.2. The two additional metrics, the *Performance Difference* 6.1 and the *Total Deviation* 6.2 are introduced to allow for a more nuanced distinctions of the various model performances, and to answer the empirical questions from more angles. The *Performance Difference* describes the relations between the best neural network model and the best benchmark on every data set (the relation is taken for both main metrics the *MAE* and the *Daily Energy Consumption Deviation*) The *Total Deviation* takes the totality of the different data sets into account. Therefore a more robust assessment of the competing methods is possible. The *MAE* describes how far off the forecasts are on average per time interval; the *Daily Energy Consumption Deviation* describes

how far off the forecasts are on average per day –both these metrics do not account for over- or under-prediction–. The *Total Deviation* describes how far off the sum of all energy consumption forecasts are, from the true total amount of energy consumed for the whole data set. The *Total Deviation* metric therefore does account for over- or under-prediction.
A reasonable forecast does perform well on the average absolute deviation from the true demand per time interval. In addition to that, the deviation from the predicted total amount of energy consumed per day to the true daily energy consumption is also of interest. Where the first pointwise error metric explains by how far the model is off on average per time interval, the *Daily Energy Consumption Deviation Error* 5.2 explains by how far the model is off on average per day. This might seem redundant, however it is important to keep the *double penalty error* in mind (see section 7.2.1). A predicted energy consumption profile, that is actually a very good forecast, but only slightly shifted in time, and therefore not exactly overlapping with the actual consumption happening, will get a very high error when evaluated based on the pointwise *MAE* error. However, the total amount of energy consumed for this particular forecast might be accurate. On the other hand, a model predicting only a very low consumption curve might perform "better" then the shifted forecast, due to the *double penalty* effect, but when taking the daily amount of energy consumed into account this low consumption forecast might not be a reasonable forecast at all. Therefore, it is helpful to look at different error metrics –with the end-use case in mind–, and not only one error metric to evaluate the models. Relying just on one single metric might yield sub-optimal results. For this reason, the *Daily Energy Consumption Deviation* 5.2 is included as the second main metric in this thesis, to attain a more robust evaluation. By considering the two main metrics, and the two additional metrics, a more nuanced discussion of the four questions 5.2.1 emerges.

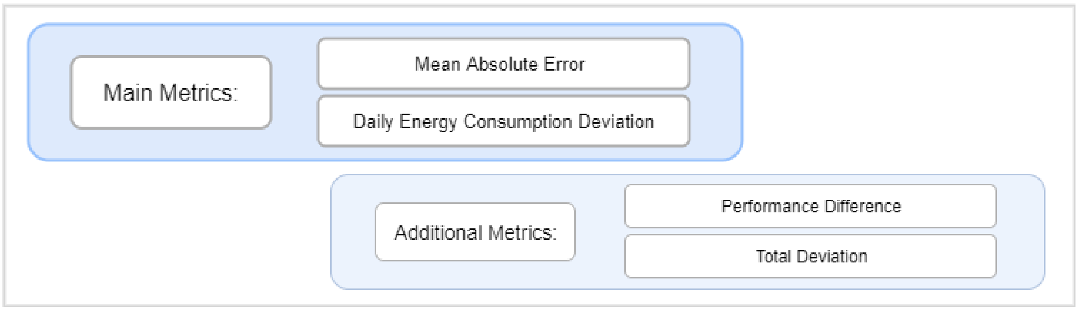

Figure 6.3: Overview of the metrics for evaluating the different forecasting approaches. The two main metrics are the mean absolute error or *MAE* (equation 5.1) and the *Daily Energy Consumption Deviation* (see equation 5.2). The two additional metrics are the *Performance Difference* (see equation 6.1), and the *Total Deviation* (see equation 6.2).

Daily Energy Consumption Deviation Results

When looking at the mean absolute error 6.1 it seems as if the *LSTM all* approach does not justify the additional features, and the *LSTM* approaches do yield slightly better forecasts then the *FFNN* models. Furthermore, the neural network methods seem to be better than the benchmarks. How do these assumptions change when taking the *Daily Energy Consumption Deviation* into account?

First, the two bar charts in figure 6.4 show the *Daily Energy Consumption Deviation* performance on the *Train*- and *Test*-partition of the data set *"X23"*.

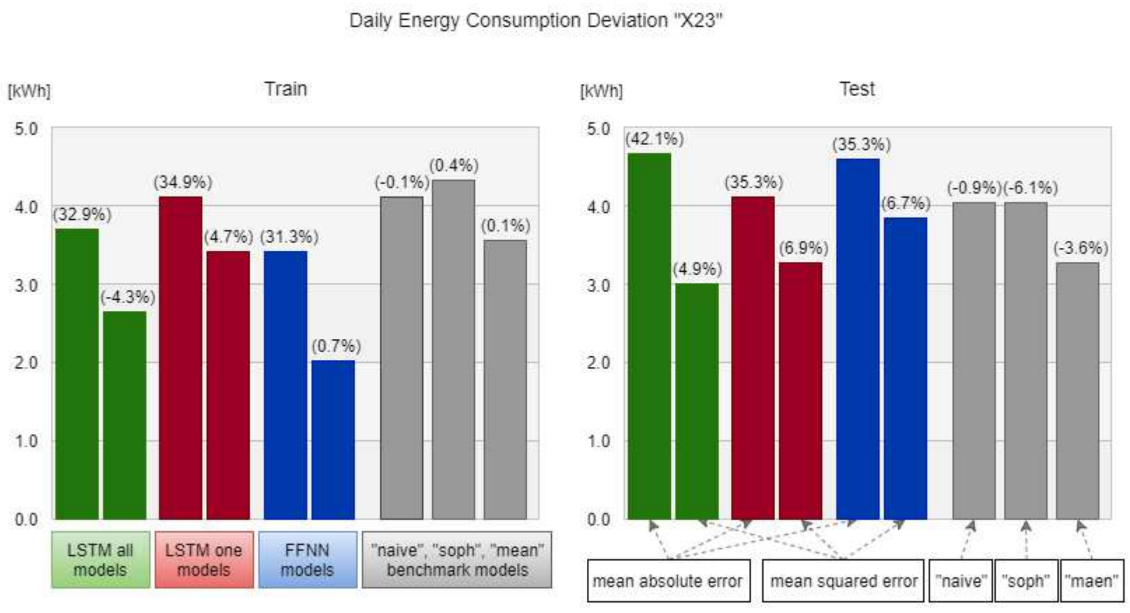

Figure 6.4: *Daily Energy Consumption Deviation* performance on data set *"X23"*. The bar chart on the left is showing the performance of the various models on the *Train*-partition. The right image shows the results on the *Test*-partition. The y-axis has the same scaling for both charts, ranging from 0 to 5 kWh. The color-coding, groups the neural network methods. The left bar shows the results from the mean absolute error Loss-function trained model, the right bar the mean squared error Loss-function model respectively. The grey bars represent the three benchmarks "naive", "soph" and "mean" from left to right. The percentage value in parenthesis on top of every bar shows the *Total Deviation* performance –the relative deviation of predicted energy consumed to actual energy consumed over the total data set partition–, according to equation 6.2.

Next, the *Test*-partitions of all six distinct data sets are considered. Table 6.3 shows the *Daily Energy Consumption Deviation* results as well as the *Total Deviation* results of all forecasting approaches on every data set.
One of the most significant findings here is the poor performance of the neural network models trained with the mean absolute error Loss-function. All three of the proposed neural network models are strongly under-predicting the total amount of electric energy consumed over the period of the total data set, as is indicated by the high and positive *Total Deviation* results. Whereas the mean squared error Loss-function models are not suffering from this under-prediction.

The *Performance Difference* (equation 6.1) between the best NN-model and the best benchmark models for the case of the *Daily Energy Consumption Deviation* are shown in table 6.4.
The "best" performing neural network methods in regard to the mean absolute error (those trained by minimizing the mean absolute error Loss-function), do not perform

Average Daily Energy Consumption Deviation Performance						
	”Simple”	”Extra”	”X23”	”X70”	”X111”	”X115”
LSTM all (mae)	1.8; (31.7%)	2.2; (29.8%)	4.7; (42.1%)	4.8; (36.2%)	6.1; (35.4%)	4.1; (19.9%)
LSTM one (mae)	2.1; (39.9%)	2.2; (30.8%)	4.1; (35.3%)	4.6; (34.7%)	6.2; (38.7%)	3.3; (7.1%)
FFNN (mae)	1.9; (35.4%)	2.5; (39.4%)	4.6; (35.3%)	3.7; (21.3%)	5.9; (24.0%)	3.6; (8.7%)
LSTM all (mse)	1.5; (-0.6%)	1.8; (2.9%)	3.0; (5.0%)	3.1; (17.4%)	4.1; (4.9%)	3.4; (2.4%)
LSTM one (mse)	1.6; (-4.7%)	1.8; (-0.1%)	3.3; (6.9%)	2.5; (7.5%)	4.0; (2.8%)	3.6; (-8.3%)
FFNN (mse)	1.6; (-6.3%)	1.7; (6.6%)	3.8; (6.7%)	3.3; (-6.7%)	5.3; (8.0%)	4.0; (-10.1%)
”naive” bm	1.7; (-0.2%)	2.1; (1.6%)	4.0; (-0.9%)	2.9; (-1.0%)	4.9; (-0.2%)	4.1; (-1.2%)
”soph” bm	1.9; (-1.08)	2.2; (2.5%)	4.1; (-6.1%)	3.0; (0.1%)	4.9; (0.4%)	5.3; (-1.8%)
”mean” bm	1.5; (-0.7%)	1.7; (2.4%)	3.3; (-3.6%)	2.8; (-1.5%)	3.7; (0.2%)	3.7; (-2.0%)

Table 6.3: *Daily Energy Consumption Deviation* performance of the various models on the *Test*-partition of the six distinct data sets in the case of 15 minute time resolution. The first three rows tinted in grey, represent the neural network models trained with the mean absolute error Loss-function (mae). The following three rows are the same models, but trained with the mean squared error Loss-function (mse). The last three rows represent the results obtained by the three benchmarks. The first numerical value in every cell describes the Average Daily Energy Consumption Deviation in [kWh]. That is, on average the models are off, by this amount of energy in units of [kWh] per day. The percentage value in parenthesis is the *Total Deviation* calculated by 6.2, and refers to how much off the model is predicting the total amount of energy consumed over the total *Test*-partition of the data set. The sign indicates if the model is predicting more then the true consumed amount, or less. A negative sign indicates a over-prediction of the total energy, whereas a positive sign indicates that the true amount is bigger then the predicted amount. The cell of the best performing model on every data set is coloured in blue, and the cell of the worst performing model is coloured in red.

	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
PerformanceDifference	0.0%	0.0%	10.0%	12.0%	-8.1%	12.1%

Table 6.4: *Performance Difference* calculated by equation 6.1 for the case of the *Daily Energy Consumption Deviation*.

well when evaluated based on the *Daily Energy Consumption Deviation*. They are actually often the "worst" performers. Clearly, the neural network models trained by minimizing the mean squared error Loss-function are more accurate in this regard. Furthermore, they are slightly better than the *"naive"* and the *"soph"* benchmark, however, the *"mean"* benchmark achieves two out of six times the lowest average deviation. The *LSTM* approaches are more accurate than the feedforward neural network, in the case of the (mse) models. When comparing the *LSTM all* approach with the *LSTM one* no consistent difference emerges and it can be said, that the additional features do not help in better forecasting the energy demand for the next day in this setup.
As is found in the literature [10, 3], it seems that the additional information from the extra features does not help for the task of forecasting energy consumption on the low-level. At this level, the main driving force is the behavior of occupants which does not depend too much on exogenous factors as does the consumption on higher voltage levels, where temporal information and weather information show to increase forecasting performance. With all that said, it is important to keep in mind, that this conclusion can only be drawn on the regarded data sets and with the 15-minute resolution and the used models. When combining the results from both metrics: (a) the pointwise mean absolute error (see table 6.1), and (b) the *Daily Energy Consumption Deviation* performance metric (see table 6.3), it becomes clear, that the first error metric (a) does not explain the "goodness" of a forecasting methodology in full detail. Only considering the performance based on the *MAE* metric, might lead to the assumption, that NN-models trained with the mean absolute error Loss-function are the most reasonable choice. However, when considering the *Daily Energy Consumption Deviation* and *Total Deviation* results, the NN-models trained with the mean squared error Loss-function or the *"mean"* benchmark are the better choice.

6.2.2 "General" vs. "Individual" Approach

After investigating the first two questions about the suitability of LSTM networks and the impact of additional features for the 15 minute time intervals in the section above; this section focuses on the third question about the optimal deployment of *LSTM all* models. To answer this question, twelve *LSTM all* models –two *LSTM all* models for every data set, trained with the two different Loss-function–, get compared on the different data sets by the pointwise mean absolute error *MAE*, and the *Daily Energy Consumption Deviation* performance metric.
The goal of this exploration is to see if a "General" model is suitable for forecasting a wide range of households, or if the better option is to have specific models for every household.
The impact of additional features might vary in those two approaches. Maybe the information does help when focusing on one specific household at a time with rela-

tively consistent consumption patterns (”Individual” model), whereas the additional information is not valuable by the models trained on all households due to the variability in occupant lifestyles. Another interesting idea to explore with this setup is if a model trained on one particular household is able to forecast other households as well? If this is the case, it might indicate, that similar households can be found, and dedicated models trained on this group of similar households might provide superior forecasts.

Mean Absolute Error Performance

First the *MAE* performance of the twelve different *LSTM all* models on the *Test*-partitions of the six distinct data sets is considered. Table 6.5 shows the mean absolute error in units of [kWh] for all *LSTM all* models on the six *Test*-partitions.

Mean Absolute Error Performance						
	”Simple”	”Extra”	”X23”	”X70”	”X111”	”X115”
LSTM all (”Simple”)	0.031	0.039	0.078	0.074	0.109	0.094
LSTM all (”Extra”)	0.032	0.039	0.075	0.073	0.109	0.096
LSTM all (”X23”)	0.061	0.064	0.074	0.100	0.114	0.111
LSTM all (”X70”)	0.062	0.063	0.111	0.071	0.116	0.101
LSTM all (”X111”)	0.066	0.065	0.091	0.086	0.104	0.105
LSTM all (”X115”)	0.088	0.083	0.110	0.095	0.134	0.091
LSTM all (”Simple”)	0.037	0.045	0.085	0.081	0.117	0.099
LSTM all (”Extra”)	0.039	0.045	0.090	0.080	0.117	0.104
LSTM all (”X23”)	0.084	0.085	0.086	0.114	0.119	0.109
LSTM all (”X70”)	0.077	0.075	0.106	0.072	0.121	0.101
LSTM all (”X111”)	0.098	0.090	0.097	0.095	0.109	0.102
LSTM all (”X115)	0.096	0.095	0.117	0.115	0.141	0.096

Table 6.5: Mean Absolute Error performance of the twelve different *LSTM all* models on the 15 minute data sets in units of [kWh]. The cell of the best performing model for every data set is coloured in blue, the cell of the worst performing model is coloured in red respectively. The first six rows tinted in grey represent the *LSTM all* models that are trained with the mean absolute error Loss-function. The data set description in parenthesis is describing the data set on which the model was trained on for 50 epochs. The last six rows are the *LSTM all* models trained with the mean squared error Loss-function.

Although the ”Individual” models (those models trained particularly on one household only) are for every data set the best performing method. The ”General” models LSTM all (”Simple”) and LSTM all (”Extra”) are very close or even identical to the one model that is trained only on this one household. From this, it might be concluded that the ”General” approach –that is regarding the total household stock– is an equivalent choice with the ”Individual” approach. However, when taking the second performance metric into account a more nuanced answer to the question can be given.

Daily Energy Consumption Deviation Performance

Table 6.6 shows the *Daily Energy Consumption Deviation* and the *Total Deviation* results from the 12 different *LSTM all* models.
To answer the question if the "General" or the "Individual" approach is more suitable, it is paramount to consider both evaluation metrics. If only considering the mean absolute error performance from table **??** it looks as if the *LSTM all* models trained on one of the two *"General"* data sets –*"Simple"* or *"Extra"*– are as good as the models dedicated to there own household. However, when taking the *Daily Energy Consumption Deviation* into account 6.6, the *Individual* models are better in forecasting the overall daily consumption of their households. The *Total Deviation* value is nearly always smaller for one of the *"General" LSTM all* models, then for the *"Individual"* models. All this is said about the NN-models that are trained with the mean squared error Loss-function, which fare substantially better than the mean absolute error Loss-function models.

6.2.3 Summary of 15 Minute Data

- *Question Number 1 and Question Number 2*: From the findings of this analysis it can be said that the use of the *LSTM* methods does yield slightly better forecasts than the *FFNN* methods, which might indicate the suitability of sequential processing of former methods. However, the *LSTM all* approach does not perform better than the *LSTM one* approach, indicating that the additional features do no help in getting more accurate forecasts. Although the neural network models are better than the benchmarks when looking at the pointwise error metric *MAE*, when regarding the *Daily Consumption Deviation* performance metric the case for using neural network models weakens, as the *"mean"* benchmark often achieves similar or better results.

- *Question Number 3*: From the comparison of the different *LSTM all* models on the six data sets it is found, that based on the mean absolute error *MAE*, it appears as if the *LSTM all* models trained on the *"General"* data sets –*"Simple"* and *"Extra"*– perform better then the *"Individual"* models on the two *"General"* data sets, and equally well on the *"Individual"* data sets –*"X23", "X70", "X111", "X115"*–. Furthermore, it can be said, that the individual trained models do not generalize well to other household data sets. However, by additionally looking at the *Daily Energy Consumption Deviation* and the *Total Deviation* it shows that the individual trained models, are better then the general trained models, in forecasting the amount of daily energy consumption for the distinct data set. Because of this, when taking both error measures into account, it might be said, that the best use case for *LSTM all* models might be to have dedicated (mse) models for every household. This, of course, depends highly on the amount of available energy consumption recordings of the individual households.

Daily Energy Consumption Deviation Performance						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
LSTM all ("Simple")	1.80; (31.8%)	2.34; (33.9%)	4.16; (31.9%)	5.37; (40.4%)	5.82; (30.2%)	4.33; (16.6%)
LSTM all ("Extra")	1.78; (24.1%)	2.18; (29.8%)	3.91; (33.2%)	3.83; (24.6%)	5.41; (24.5%)	4.82; (18.3%)
LSTM all ("X23")	2.66; (-3.9%)	2.76; (6.4%)	4.68; (42.1%)	3.58; (22.9%)	6.33; (39.5%)	7.40; (47.2%)
LSTM all ("X70")	2.90; (23.7%)	2.87; (20,1%)	4.37; (10.4%)	4.79; (36.2%)	5.05; (26.3%)	6.80; (42.3%)
LSTM all ("X111")	3.42; (45.2%)	3.39; (39.7%)	4.79; (40.4%)	4.62; (32.1%)	6.09; (35.4%)	7.52; (47.8%)
LSTM all ("X115")	4.86; (20.3%)	4.16; (-2.0%)	3.96; (-7.7%)	3.20; (-16.7%)	4.26; (-15.3%)	4.05; (19.9%)
LSTM all ("Simple")	1.53; (-0.6%)	1.78; (3.6%)	3.05; (8.9%)	3.40; (3.4%)	4.89; (0.0%)	3.55; (-4.8%)
LSTM all ("Extra")	1.75; (-6.3%)	1.80; (2.9%)	3.53; (1.5%)	3.13; (10.6%)	4.73; (0.3%)	3.82; (1.0%)
LSTM all ("X23")	4.65; (-75.7%)	4.37; (-52.9%)	3.00; (5.0%)	3.06; (-8.3%)	4.05; (15.3%)	4.74; (22.8%)
LSTM all ("X70")	3.26; (45.7%)	3.22; (34.6%)	3.83; (18.3%)	3.08; (17.4%)	4.30; (9.5%)	6.14; (37.8%)
LSTM all ("X111")	7.33; (116.9%)	6.16; (69.0%)	5.69; (49.4%)	3.17; (-8.9%)	4.10; (4.9%)	5.30; (31.5%)
LSTM all ("X115)	6.09; (-47.9%)	5.76; (-55.2%)	3.97; (-23.8%)	5.87; (-45.2%)	5.16; (-29.2%)	3.36; (2.4%)

Table 6.6: *Daily Consumption Deviation* performance of the twelve different *LSTM all* models on the 15 minute data. The columns represent the six distinct data sets. The data set encoding in parenthesis next to the *LSTM all* in every row, refers to the data set, on which the model was trained on. The first six rows (hued in grey) show the *LSTM all* models trained with the mean absolute error Loss-function. The remaining six rows (hued in white) are the models trained with the mean squared error Loss-function. The best performing and worst performing model is coloured in blue and red respectively for every data set. The value in parenthesis next to the *Daily Energy Consumption Deviation* shows the *Total Deviation* result of every model, described by equation 6.2

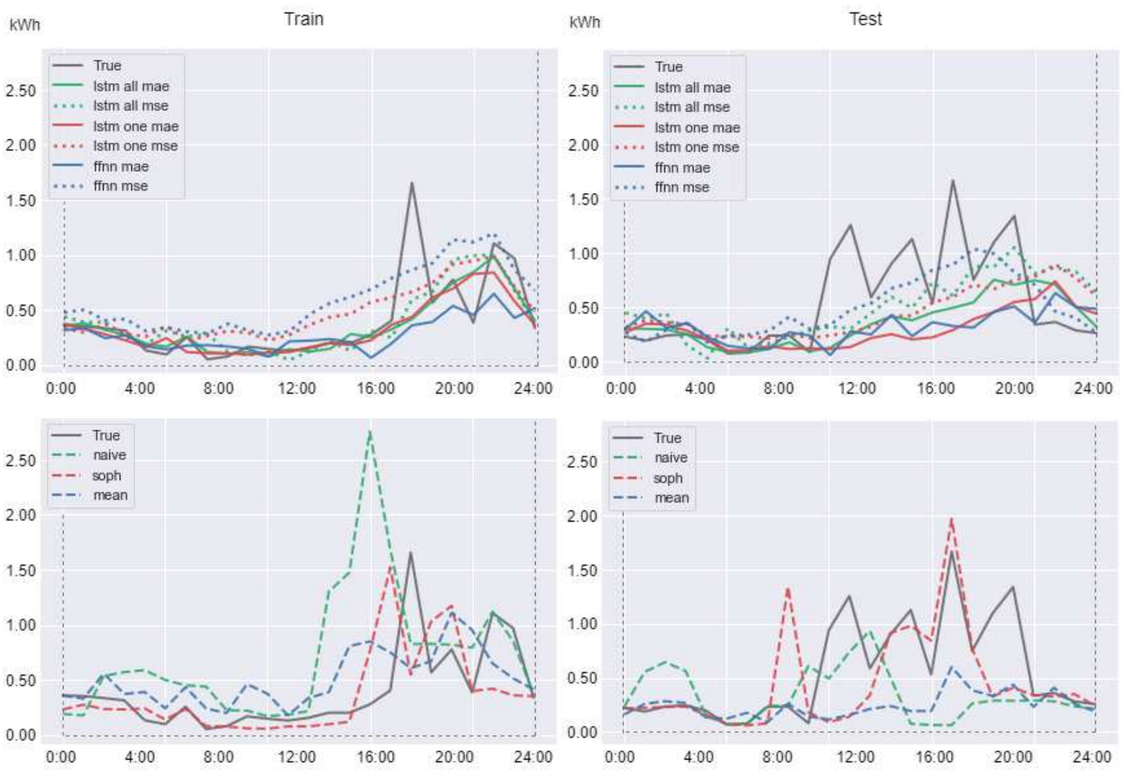

Figure 6.5: Predicted consumption profiles of various models. The two images on the left side are the predictions of the neural network models (top image) and the benchmark models (bottom image) for the same day, taken from the *Train*-partition of data set "X23". The two images on the right side are the predictions on a day from the *Test*-partition of data set *"X23"*. The neural network predictions are separated from the benchmark predictions, for visual purposes. The three neural network approaches (*LSTM all*, *LSTM one* and *FFNN*) are once trained with the mean absolute error Loss-function (mae, solid line) and once with the mean squared error Loss-function (mse, dotted line), resulting in six predictions. The true day is coloured in grey. In the upper images the *LSTM all* models are green. The *LSTM one* and *FFNN* models are coloured in red and blue respectively. For the bottom images, the three differing benchmarks *"naive"*, *"soph"* and *"mean"* (dashed lines) are coloured green, red and blue respectively.

6.3 Hour Analysis

After investigating the first three empirical questions for the case of the 15 minute time intervals, this section shows the same analysis for the hourly aggregated consumption profiles. Comparing the findings of both data resolutions, gives insights into the last of the four questions, regarding the impact of data granularity on the forecasting quality.

The inputs for the different models in the case of the hourly time interval are of the form: $\boldsymbol{X} = (168, 53)$ for the *LSTM all*, $\boldsymbol{X} = (168, 1)$ for the *LSTM one* and $\boldsymbol{x} = (168,)$ for the *FFNN* model. The output is for every model the a vector $\boldsymbol{y}$ with length $(24,)$.

In various studies, it is found that the granularity of the consumption recordings has a big impact on the forecasting performance of different predictive models [8, 22]. Models, that perform "best" on a certain data set with distinct characteristics are often not the optimal choice any more when consumption patterns with different characteristics are investigated. Increasing the regarded time interval of the examined energy consumption recordings is known as a way, to increase the predictability of time series. With the increase of the time intervals, the randomness and volatility in the consumption patterns might get reduced, and the predictive power from additional features might weigh differently, then for the 15-minute case.

6.3.1 Value of Recurrent Neural Networks, and the Impact of Additional Features

In the 15-minute analysis, some of the key findings are, that the *LSTM* approach is slightly superior to the *FFNN* approach. Using additional features does not result in better forecasts. And the "Individual" approach seems better than the "General" approach. How do these findings change with the "out-smoothing" of the consumption pattern by increasing the considered time interval to hourly?

Mean Absolute Error Performance

The performance analysis on the hourly data starts with looking at the point-wise *MAE*. Six different neural network models (once trained with the mean absolute error Loss-function and once trained with the mean squared error Loss-function) are compared to each other, and the three benchmarks. The architecture of the NN-models changed regarding the architecture of the models trained on the 15-minute data. Due to the decrease in data points, the architecture of the models is adjusted. The number of parameters is reduced by a factor of 2 (see table 5.3) to reduce the risk of over-parametrization and overfitting. The two bar charts in figure **??** show the performance of the different neural network models *LSTM all*, *LSTM one* and *FFNN* as well as the three benchmark models *"naive"*, *"soph"* and *"mean"* on the *Train*- and *Test*-partition of data set *"X23"*.

A similar pattern as in the case of the 15-minute data can be seen. An overfitting effect is taking place, as the error on the *Train*-partition is for the *LSTM all* and *FFNN* models smaller than the error on the *Test*-partition. Interestingly, the overfitting effect is not observed for the *LSTM one* models. The performance of the different neural network models is very similar on the *Test*-partition of data set "X23".

Table 6.7 shows the *MAE* results of the various models on the *Test*-partition of each data set.

Increasing the time intervals does not alter the performance relation of *LSTM all* model to *LSTM one* and to the *FFNN* model respectively. The *LSTM* models are performing very similar, and slightly better than the *FFNN* models. The *LSTM all* model is not better than the *LSTM one* model, indicating that, also in the case of the hourly time interval, the additional features do not help in getting more accurate

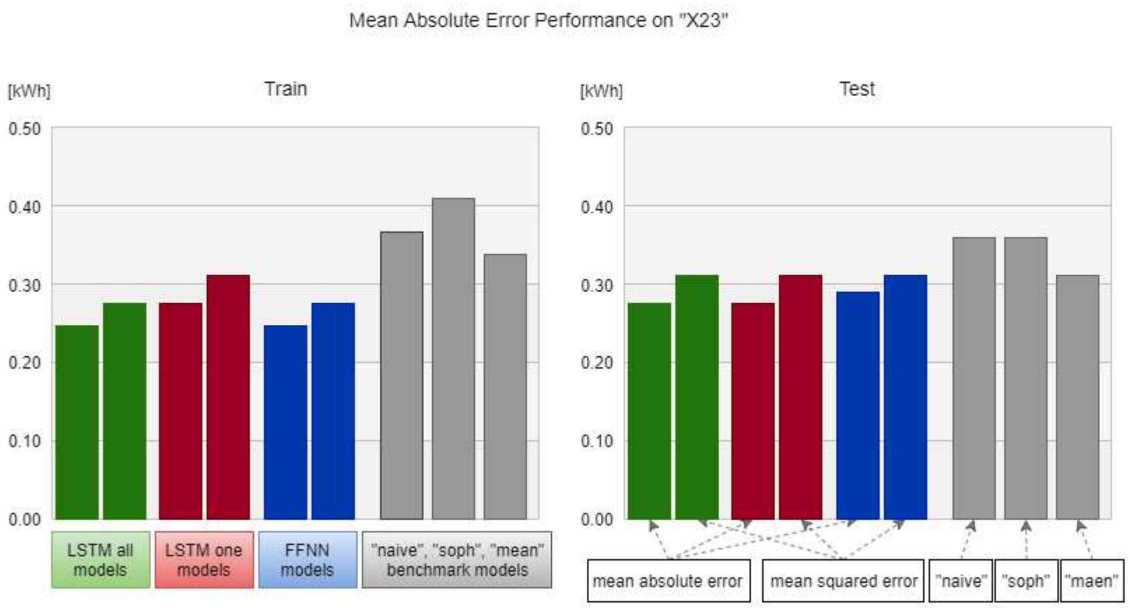

Figure 6.6: Mean Absolute Error performance of different models and benchmarks on *Train*-partition (left) and *Test*-partition (right) on the *"X23"* data set. The two green bars represent the *LSTM all* models, the red bars the *LSTM one* models and the blue bars the *FFNN* models respectively. The three benchmarks *"naive"*, *"soph"* and *"mean"* are represented by the grey bars from left to right. The left bar of each NN-model pair –green, red and blue– represents the model, that is trained with the mean absolute error Loss-function. The right bar is shows the *MAE* results of the same model, but trained with mean squared error Loss-function.

Mean Absolute Error Performance						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
LSTM all (mae)	0.110	0.142	0.277	0.373	0.377	0.305
LSTM one (mae)	0.108	0.140	0.274	0.366	0.375	0.304
FFNN (mae)	0.110	0.142	0.285	0.379	0.387	0.313
LSTM all (mse)	0.127	0.162	0.309	0.380	0.386	0.317
LSTM one (mse)	0.121	0.158	0.313	0.343	0.392	0.317
FFNN (mse)	0.132	0.167	0.313	0.369	0.407	0.330
"naive" bm	0.149	0.190	0.357	0.461	0.498	0.378
"soph" bm	0.153	0.190	0.360	0.459	0.499	0.394
"mean" bm	0.138	0.158	0.309	0.367	0.400	0.310

Table 6.7: Mean absolute error performances of the various models (rows) on the six distinct data sets (columns). The first three neural network models are trained with the mean absolute error Loss-function, indicated with the (mae). The following three neural network models are trained with the mean squared error Loss-function respectively, marked with the (mse). The blue coloured cell shows the "best" performer, the red coloured cell shows the "worst" performer on every data set.

forecasts.
Based on these *MAE* results it looks as if the increase of the time intervals from 15 minutes to one hour does not change the performance relations from *LSTM* models to *FFNN* and benchmarks respectively. As in the case of the 15 minute time intervals, the *LSTM all* approach is not advantageous compared to the *LSTM one* method, which means, that the additional features do not help in getting more accurate forecasts based on these data sets and the used model architectures. The *LSTM one* model is, in fact, the best performing models for all data sets. In five out of the six data sets the *LSTM one (mae)* model is the best forecaster based on the *MAE* metric, and on data set "X70" the *LSTM one (mse)* achieves the lowest *MAE*. Other than in the case of the 15 minute time intervals, where the best performing models are found in each NN-methodology, shown in table 6.1. These results are rather surprising and do not support the hypotheses, that with bigger time intervals and the out-smoothing of the volatile consumption patterns, additional features could become more valuable. The results indicate the opposite, as the *LSTM one* models do now perform better for every data set.

	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
PerformanceDifference	32.4%	12.9%	12.8%	7.0%	6.7%	2.0%

Table 6.8: *Performance Difference* of the best NN-model and the best benchmark models based on the *MAE* metric, according to equation 6.1.

By comparing table 6.8 with table 6.2 no drastic shift in performance relation can be observed when going from the 15 minute to the hourly time interval. Arguably, the case for using NN-models weakens, because the *Performance Difference* value decreases in five out of six data sets.

Daily Energy Consumption Deviation Performance

Figure 6.7 shows the performance of the different models and the benchmarks on the *Train*- and *Test*-partition of data set "X23".

Table 6.9 shows the *MAE* results of the various forecasting methods on the *Test*-partition of all data sets.
By comparing the *Performance Difference* based on the *Daily Energy Consumption Deviation* for both time resolutions (15 minute 6.4, and hourly 6.10), a small shift in favour of the NN-model can be observed when considering the bigger time intervals. This could indicate, that with bigger time intervals, the benefit of using NN-models compared to simpler benchmarks does increase, based on the *Daily Energy Consumption Deviation*. This shift in favor of the NN-models is not to be found when the *MAE* is the metric based on which the comparison is done.

6.3.2 "General" vs. "Individual" Approach

To answer the third question about the best deployment method of a *LSTM all* model for the case of the hourly data resolution, the performances of the twelve *LSTM all* models –six mae-trained and six mse-trained– are compared on all six

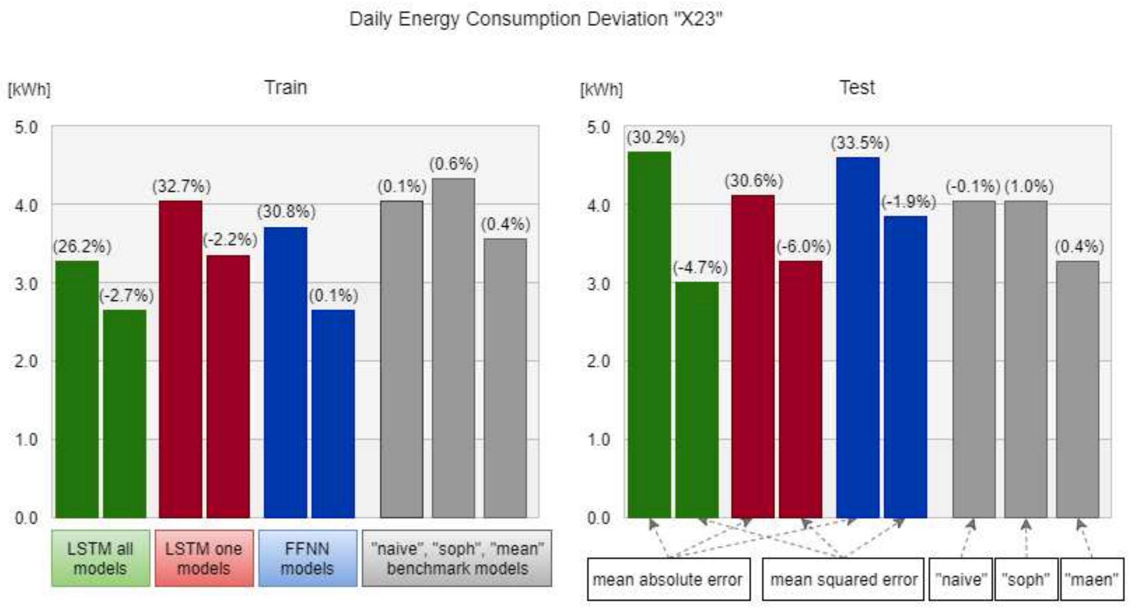

Figure 6.7: *Daily Energy Consumption Deviation* performance on data set *"X23"*. The bar chart on the left is showing the performance of the various models on the *Train*-partition. The right image is showing the results on the *Test*-partition of the data set. The y-axis has the same scaling for both charts, ranging from 0 to 5 kWh. The color coding groups the neural network methods. The left bar shows the mean absolute error Loss-function trained model, the right bar the mean squared error Loss-function trained model respectively. The grey bars represent the three benchmarks *"naive"*, *"soph"* and *"mean"* from left to right. The percentage value in parenthesis on top of every bar shows the *Total Deviation* results –the relative deviation of predicted energy consumed to actual energy consumed over the total data set partition–, according to equation 6.2.

Average Daily Energy Consumption Deviation Performance						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
LSTM all (mae)	1.73; (27.0%)	2.07; (27.8%)	3.93; (30.2%)	6.27; (44.3%)	5.09; (21.3%)	3.10; (2.2%)
LSTM one (mae)	1.71; (27.3%)	2.12; (30.3%)	3.85; (30.6%)	6.27; (44.5%)	4.81; (20.0%)	3.23; (7.2%)
FFNN (mae)	1.67; (25.2%)	1.90; (23.7%)	4.19; (33.5%)	6.40; (46.9%)	4.92; (17.8%)	3.17; (2.4%)
LSTM all (mse)	1.55; (2.0%)	1.71; (0.8%)	3.17; (-4.7%)	4.31; (16.4%)	4.24; (3.9%)	3.28; (-8.7%)
LSTM one (mse)	1.48; (5.0%)	1.65; (1.0%)	3.56; (-6.0%)	3.78; (15.3%)	4.26; (0.9%)	3.25; (-9.4%)
FFNN (mse)	1.62; (-6.9%)	1.79; (-6.1%)	3.50; (-1.9%)	4.67; (18.6%)	5.01; (4.4%)	3.45; (-6.9%)
"naive" bm	1.79; (-2.4%)	2.12; (2.3%)	4.04; (-0.1%)	5.51; (20.1%)	5.54; (-0.3%)	4.06; (-0.2%)
"soph" bm	1.94; (-3.2%)	2.17; (3.5%)	3.97; (1.0%)	5.65; (21.0%)	5.63; (-1.6%)	4.48; (-0.5%)
"mean" bm	1.57; (-2.8%)	1.70; (3.4%)	3.25; (0.4%)	4.65; (20.5%)	4.22; (-1.3%)	3.26; (-0.5%)

Table 6.9: *Daily Energy Consumption Deviation* performance of the various models on the *Test*-partition of the six distinct data sets in case of hourly time resolution. The first three rows tinted in grey, represent the neural network models trained with the mean absolute error Loss-function (mae). The following three rows are the same models, but trained with the mean squared error Loss-function (mse). The last three rows represent the results obtained by the three benchmarks. The first numerical value in every cell describes the *Daily Energy Consumption Deviation* in [kWh]. The percentage value in parenthesis is the *Total Deviation* value calculated by equation 6.2, and refers to how much off the model is predicting the total amount of energy consumed over the total data set. The best performing model on every data set is coloured in blue, and the worst performing model is coloured in red.

	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
PerformanceDifference	6.1%	3.0%	2.5%	23.0%	-0.9%	5.2%

Table 6.10: *Performance Difference* of the *Daily Energy Consumption Deviation* metric describing the relation between best NN-model and best benchmark model calculated by 6.1.

data sets, based on the *MAE* metric as well as the *Daily Energy Consumption Deviation* metric. It is investigated if one of the "General" models –trained either on the *"Extra"* or *"Simple"* data set–, or the approach of using "Individual" models –trained on *"X23"*, *"X70"*, *"X111"* or *"X115"*– is the more reasonable approach.

Mean Absolute Error Performance

Table 6.11 shows the performance of the twelve different *LSTM all* models on the *Test*-partitions of all six data sets.

Mean Absolute Error Performance						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
LSTM all ("Simple")	0.110	0.143	0.284	0.280	0.390	0.318
LSTM all ("Extra")	0.115	0.142	0.279	0.274	0.386	0.310
LSTM all ("X23")	0.184	0.206	0.277	0.354	0.414	0.398
LSTM all ("X70")	0.178	0.198	0.344	0.265	0.437	0.344
LSTM all ("X111")	0.218	0.236	0.340	0.342	0.377	0.370
LSTM all ("X115")	0.220	0.240	0.421	0.311	0.455	0.305
LSTM all ("Simple")	0.127	0.158	0.306	0.288	0.409	0.312
LSTM all ("Extra")	0.143	0.162	0.320	0.295	0.412	0.318
LSTM all ("X23")	0.250	0.272	0.309	0.386	0.420	0.402
LSTM all ("X70")	0.197	0.218	0.368	0.270	0.444	0.337
LSTM all ("X111")	0.220	0.241	0.338	0.351	0.386	0.350
LSTM all ("X115)	0.231	0.250	0.423	0.362	0.490	0.317

Table 6.11: Mean Absolute Error Performance of the twelve different *LSTM all* models on all six data sets for hourly time intervals. The best performing model for every data set is coloured in blue, the worst performing model is coloured in red. The first six rows tinted in grey represent the *LSTM all* models that are trained with the mean absolute error Loss-function. The data set description inside the parenthesis is describing the data set on which the model was trained on for 50 epochs. The last six rows are the *LSTM all* models trained with the mean squared error Loss-function.

The "Individual" models trained with the mean absolute error Loss-function are consistently the best model. However, the "General" models *LSTM all ("Simple")* and *LSTM all ("Extra")* are achieving the best results on the *"General"* data sets *"Simple"* and *"Extra"*, and are only slightly behind then the dedicated "Individual" models on the single household data sets. Interestingly, as in the case of the 15 minute time intervals, the "Individual" models due not generalize well to the other unseen households.

When looking at the daily energy consumption distribution of the different data sets in figure 5.2 it shows, that *"X70"* and *"X111"* are more similar in regard of the shape of the consumption distribution then to the other two households *"X23"* and *"X115"*. However, it is not found, that the models trained on each of those two rather similar households do perform well on the other household, nor are there performances similar to the remaining data sets. These results do not support the hypothesis that models trained on similar households, can achieve reasonable forecasts for each other. This negates the idea that a model trained on one particular household might be able to generalize well to a household with a similar shape of daily energy consumption distribution. This statement, however, is based on the visual similarity of the shape of the energy consumption distribution of the six distinct data sets.

The same relations are apparent for the hourly data as they are for the 15-minute data. The "Individual" models trained particularly on one household are the "best" performers in every case. However, as is the case in the 15-minute data, the "General" models do perform only slightly worse than the "Individual" models on the single household data sets. Therefore, the use of "Individual" models might not be justified when making the decision based on these results. Because of the additional complexity that would arise via the training of an individual model for every household in a household stock, it is preferable to use only one "General" model for all those households; provided the "General" model does forecast reasonable. As with the 15-minute data, at this point it looks as if the "General" model *LSTM all ("Extra")* might indeed be the best choice for the case of using a *LSTM all* model.

In the next step, the *Daily Energy Consumption Deviation* performance metric gets included.

The question is if there is information in the input sample of every household, that enables a "General" model to learn enough, to forecast the energy consumption for the next day, regardless of the considered household. Is a neural network model, in this case, an LSTM model, able to pick up consumption patterns that are universal to all households? Are there any universal relations from input to output, so that models can learn these patterns, and forecast the next day's energy consumption. Or are households too diverse, so that there are no universal relations that govern the consumption patterns. It is likely that for any individual household there are such relations and patterns. However, they might vary drastically from household to household, and this might hinder a "General" model from exploiting this kind of information. Universal patterns in the data samples from input to output would allow the model to find common relations in the samples, regardless of the considered household. For instance, regardless of the amount of energy that is recorded and fed to the models as input, the behavior that might take place in numerous households like a small consumption spike in the morning then low consumption during the afternoon, and a spike in energy consumption in early evening hours. Furthermore, when taking information about the weekday into account, a universal pattern might be, that the spike in morning consumption does shift to later hours on weekends. This universal patterns could be exploited by the model if they are occurring often enough throughout the processed samples.

Daily Energy Consumption Deviation Performance

Based on the pointwise mean absolute error, the use of "Individual" models does not seem to be justifiable. Is the "General" model still the more reasonable choice when considering the *Daily Energy Consumption Deviation* metric? Due to the fact, that the individual households consumption patterns stray from the general consumption distributions, it is to be expected, that the "Individual" models have different internal weightings of there input parameters than the "General" models, which would suggest that a tailored model to every single household is better able to exploit specific consumption patterns.

Table 6.12 shows, that the "General" models do perform worse than the "Individual" models on the single household data sets. The "Individual" models are consistently the "best" performers for the particular data set on which they were trained. Again the NN-models trained with the mean absolute error Loss-function are performing poorly based on the *Daily Energy Consumption Deviation* and the *Total Deviation* metrics.

6.3.3 Summary of Hourly Data

For the hourly time interval, the first two questions regarding the suitability of the*LSTM* approach and the effect of additional features can be answered as follows:

- After considering the mean absolute error performance of the models on the six distinct data sets it can be said, that the *LSTM* models –*LSTM all* and *LSTM one*– do outperform the *FFNN*, which indicates, that for the regarded data sets in this analysis, the sequential processing of the historic energy consumption does help in achieving more accurate forecasts based on the pointwise *MAE*. When also taking the second performance metric into account, the *LSTM* models trained with the mean squared error Loss-function are much better in forecasting the daily energy consumption as well as the total energy consumed over the whole data set partition. The *"mean"* benchmark is close and in some cases even better than the best NN-model, based on the *Daily Energy Consumption Deviation* metric.

- The *LSTM all* method did not perform "better" then the *LSTM one* when evaluated based on the *MAE* or the *Daily Energy Consumption Deviation*. This indicates that the additional features, as in the case of the 15-minute data, do not help in getting more accurate forecasts. This might be due to the fact, that there is no predictive power in the regarded additional features, or that the randomness and volatility in the data is still the dominant factor. Therefore, performing actions for increasing the predictability of energy consumption profiles might be a valuable step, to decrease the "noise" in the data and extract the "signal" coming from additional information. The finding, that there is no performance gain for the *LSTM all* models, when regarding the bigger time intervals, is rather surprising, as the out-smoothing of the consumption profiles was expected to benefit this approach.

Daily Energy Consumption Deviation Performance						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"
LSTM all ("Simple")	1.73; (27.0%)	2.22; (30.6%)	4.05; (24.3%)	4.44; (24.1%)	5.70; (28.9%)	4.33; (17.6%)
LSTM all ("Extra")	1.73; (20.1%)	2.07; (27.8%)	3.96; (29.4%)	4.11; (22.2%)	5.78; (27.8%)	4.17; (15.0%)
LSTM all ("X23")	2.06; (5.8%)	2.21; (16.2%)	3.93; (30.2%)	5.12; (35.2%)	6.65; (39.1%)	6.84; (43.2%)
LSTM all ("X70")	2.12; (15.2%)	2.40; (20.8%)	4.05; (18.8%)	4.09; (25.2%)	5.54; (27.8%)	5.38; (30.9%)
LSTM all ("X111")	3.05; (6.0%)	3.06; (7.1%)	3.76; (17.5%)	4.23; (17.0%)	5.09; (21.3%)	5.38; (30.9%)
LSTM all ("X115")	3.17; (-42.7%)	3.17; (-27.1%)	4.84; (-26.4%)	3.22; (-2.0%)	4.46; (-1.0%)	3.1; (2.2%)
LSTM all ("Simple")	1.55; (2.0%)	1.77; (8.1%)	3.63; (6.5%)	3.34; (5.6%)	5.11; (3.2%)	3.56; (6.6%)
LSTM all ("Extra")	1.85; (-13.2%)	1.71; (0.8%)	3.61; (-2.6%)	3.38; (-0.3%)	4.98; (4.0%)	3.50; (-2.2%)
LSTM all ("X23")	3.51; (-49.3%)	3.56; (-35.8%)	3.17; (-4.7%)	3.90; (3.0%)	4.61; (15.1%)	4.80; (23.8%)
LSTM all ("X70")	2.38; (10.8%)	2.58; (14.0%)	4.13; (17.7%)	2.81; (5.2%)	4.66; (13.9%)	4.68; (24.7%)
LSTM all ("X111")	3.02; (-12.7%)	3.17; (-6.6%)	3.81; (13.0%)	3.48; (-7.0%)	4.24; (3.9%)	4.09; (18.1%)
LSTM all ("X115)	3.24; (-41.0%)	3.44; (-28.8%)	5.11; (-30.4%)	4.19; (-17.1%)	4.84; (-13.5%)	3.28; (-8.7%)

Table 6.12: *Daily Energy Consumption Deviation* performance of the twelve different *LSTM all* models on the *Test*-partition of the six data sets with hourly time intervals. The data set encoding in parenthesis next to the *LSTM all* in every row, refers to the data set, on which the model was trained on. The first six rows (hued in grey) show the *LSTM all* models trained with the mean absolute error Loss-function. The remaining six rows (hued in white) are the models trained with the mean squared error Loss-function. The "best" performing and "worst" performing models are tinted in blue and red respectively. The percentage value in parenthesis in every cell shows the *Total Devation* result (see equation 6.2).

- *"General" vs. "Individual" approach*: When only considering the mean absolute error in table6.11 it seems as if the "General" models trained on the data set *"Simple"* and *"Extra"* are as good as the "Individual" models trained and evaluated on the individual household data sets. This might suggest, that when taking the additional complexity of training many "Individual" models into account, the "General" approach is more appropriate. The big advantage with having one "General" model that can forecast on a wide range of households, is the fact, that when a "new" household with no historic energy consumption recordings yet available needs to be predicted. In this case, the "General" model only needs the enough smart meter recordings to process one input. On the other hand, an "Individual" model would need many months or even years of energy consumption recordings to let a neural network model have sufficient amounts of data to make reasonable forecasts on this one household. In this regard, the "General" approach has a big advantage. Still, it can be seen that the "General" model –although equal when regarding the mean absolute error– is not as good as the "Individual" models when the *Daily Energy Consumption Deviation* is additionally taken into account. Depending on the end-use case and the weighing of the different metrics, a decision can be made, which approach is more reasonable.

- Interestingly there is no actual difference to be found –based on the chosen metrics– in the performance of the two "General" models. Both the *LSTM all* trained on the *"Simple"* data set and the *LSTM all* model trained on the *"Extra"* data set have very similar results for both metrics –the *MAE* and the *Daily Energy Consumption Deviation*–. This is surprising and might conclude, that the extra filtering step of going from the *"Simple"* data set to the *"Extra"* data set, via the minimum consumption filtering criteria does not change the behavior of the *LSTM all* models. Although this filtering step dropped roughly 20% of the samples with very low consumption, this step does not help the *LSTM all* model trained on *"Extra"* to be more accurate in the daily amount of energy consumed then the *LSTM all* model trained on the *"Simple"* data set.

- The "Individual" models are better than the "General" models in forecasting the next day's energy consumption on the individual household data sets. Therefore, the "Individual" models are preferable from the standpoint of pure performance measure –according to both metrics–. However, it has to be analyzed if the performance advantage is big enough to justify the additional complexity, that comes with deploying "Individual" models.

6.4 Comparison of 15 Minute and Hour analysis

By comparing the findings and insights gathered from different time resolutions, the fourth question about the effect of data granularity on forecasting performance can be investigated. The main differences and similarities between those two resolutions are summed up in the listing below:

- The *LSTM all* approach does not out-perform the *LSTM one* approach in both cases. Therefore, it can be said, that the additional features considered in this thesis do not help in getting more accurate day-ahead energy consumption forecasts for individual households. This statement is only valid given the proposed neural network architecture, the input processing as well as the metrics based on which the evaluation is performed.

- The *LSTM* approach with its sequential processing of the inputs does yield better performances than the *FFNN* approach.

- The *"Individual"* model approach does lead to better forecasts than the *"General"* model approach. However, the advantage is small. Interestingly the *"General"* models are often more accurate based on the *Total Deviation* metric.

- The *"mean"* benchmark seems to be a good trade-off between "goodness" of the forecast and simplicity of deployment.

- The mean squared error Loss-function trained models are better than their mean absolute error Loss-function trained counterparts, especially for the evaluation based on the *Daily Energy Consumption Deviation* and the *Total Deviation*, where the mean absolute error Loss-function models are off by a wide margin.

- A difference is the fact, that the *LSTM one* methods are consistently the best models when considering the hourly data. In the case of the 15 minute time intervals, the best performing models are more mixed. These results negate the hypothesis, that with an increase of the time intervals and the "out-smoothing" of the consumption profiles, the additional features can be exploited more effectively.

- One of the overall findings for both time resolutions is, that the filtering process: -going from *"Simple"* data set -filtered only for NaNs to the *"Extra"* data set did not yield a substantial performance difference. The similar performance of the models trained on *"Simple"* and *"Extra"* was not to be expected. The assumption for the filtering step was in the first place, to drop very low consumption profiles, to allow the models to predict higher consumption. Neither for the 15-minute data sets nor the hourly data sets was it found, that the models trained on the *"Extra"* data set predicted significantly higher consumption than the models trained on the *"Simple"* data set. This finding is surprising. It might be because the filtering threshold is chosen to conservative, or that the models find different representations for making their forecasts. This is one of the biggest shortcomings of neural network models, that they are hard to interpret and are somewhat black-box models. To shine some light on the internal behavior of the neural network models it might be reasonable to change the inputs to the models in small ways and observe the change in the output accordingly. With this, it might be possible to find the most important input pathways, based on which the models make their characteristic predictions.

- The results obtained from the *Performance Difference* regarding both metrics, are mixed. No distinctive performance gain or drop of the NN-models compared to the benchmark models is observed when varying the time resolution.

- Consistently it is was found, that the NN-model trained with the mean absolute error Loss-function are the best models when only considering the *MAE* for evaluation. However, these models are suffering from the under-prediction bias (see section 7.2.1). This results in low energy consumption forecasts. As can be seen in tables 6.3 and 6.9 the *Total Deviation* value in parenthesis of the mae-trained models is very high and positive. Which indicates substantial under-prediction over the total data set. For the NN-models trained with the mean squared error Loss-function, the under-prediction bias is not a problem.

- It has to be kept in mind, that this empirical study does not allow for definitive answers about whether neural network models and in particular LSTM networks are suitable for the task of low-level energy consumption forecasting. The findings of this thesis are specific for the analyzed household stock, as well as the proposed neural network architecture with the described hyperparameter choices. Additionally, the processing of the model input/target pairs plays a major role in the outcomes.

- The inclusion of the additional features, (auto-regressive features, temporal features, dwelling size) as input to the *LSTM all* model does not come with an increase in forecasting quality. This can mean two things: either there is no actual underlying relationship in the additional features to the true target day, meaning the additional features do not hold any predictive power, or it could be that there is too much noise in the data (high variability and randomness in the consumption profiles) for the LSTM model to extract the underlying relationship. However, the process of increasing the time intervals to out-smooth randomness in the consumption profiles and allow for the exploitation of additional features did not result in better performances of the *LSTM all* models.

6.4.1 Results Overview

This final section of the chapter tries to digest the findings and insights in regard to model performances into helpful and memorable representations. The two overview tables shown here aim to represent the summaries and points made above in a concise and intuitive way. The first table 6.13 captures the various models performances regarding the two metrics *MAE* and *Daily Energy Consumption Deviation* for both time resolutions. This table encompasses the first two empirical questions this thesis set out to answer, regarding the suitability of the *LSTM* approach and the value of additional features. The second overview table 6.14 sums up the results concerned with question three, about the best deployment of *LSTM all* models, *"General"* vs. *"Individual"*. Both tables allow for a comparison of the findings, in respect of the data resolution. The upper part shows the analysis for the 15 minute time intervals, the lower part for the hourly time resolution. A simple scoring scheme is used to allow for a comparison of the different forecasting methodologies.
The scoring scheme is the following: The first value in every cell describes the performance based on the *MAE* metric. The second value describes the performance based on the *Daily Energy Consumption Deviation*. The possible scores for each metric are: ++, +, , -, and –. The best model based on the regarded metric gets ++, the second and third best model get a +. The worst model receives the –. The

second and third-worst models get a -. The models in between receive a . The final score of every model in the rightmost column is defined by adding overall scored points in the row, where ++ = 2, + = 1,   = 0, - = -1, and − = -2.
These overview tables, therefore function as a kind of summary of the results chapter. Furthermore, the rough scoring scheme allows for an ordering of the competing forecasting methods and their performances for the conducted analysis.

Performance Overview Neural Networks and Benchmarks							
	15 minute data sets						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115	Score
LSTM all	++, -	+, -	++, −	+, −	+, -	++, -	1
LSTM one	+, −	++, -	++, -	++, -	++, −	+, ++	5
FFNN	+, ∼	+, −	∼, -	∼, -	-, -	∼, +	-3
LSTM all	∼, ++	∼, +	∼, ++	+, ∼	+, +	+, +	10
LSTM one	∼, +	∼, +	+, +	∼, ++	+, +	∼, +	9
FFNN	-, +	∼, ++	∼, ∼	-, ∼	∼, ∼	-, ∼	0
"naive" bm	-, ∼	−, ∼	-, ∼	−, +	−, ∼	-, ∼	-8
"soph" bm	−, ∼	−, ∼	−, ∼	−, ∼	-, ∼	−, −	-13
"mean" bm	∼,++	-, ++	-, +	∼, +	∼, ++	∼, ∼	6

	hourly data sets						
	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"	Score
LSTM all	+, -	+, ∼	+, ∼	∼, -	+, -	+, ++	4
LSTM one	++, ∼	++, -	++,∼	+, -	++, ∼	++, +	10
FFNN	+, ∼	+, ∼	+, −	∼, −	∼, ∼	∼, +	0
LSTM all	∼, +	∼, +	∼, ++	-, +	+, +	∼, ∼	6
LSTM one	∼, ++	∼, ++	∼, ∼	++, ++	∼, +	∼, ∼	9
FFNN	∼, ∼	-, ∼	∼, +	∼, ∼	-, ∼	-, -	-3
"naive" bm	-, -	−, -	-, -	−, ∼	-, -	-, -	-13
"soph" bm	−, −	−, −	−, -	-, ∼	−, −	−, −	-20
"mean" bm	-, +	∼, +	∼, +	+, +	∼, ++	+, ∼	7

Table 6.13: Performance overview of different forecasting approaches on the six distinct data sets and two different data resolutions. The first three rows (grey background) are representing the three NN-models trained with the mean absolute error Loss-Function. The following three rows (white background) are the same NN-models but trained with the mean squared error Loss-Function. The row of the best overall model –scoring the most points, according to the scoring scheme– is tinted in blue. The worst performing models has its row tinted red.

Note on Performance Overview Table 6.14: Other then overview table 6.13, this table does not aim for a comparison of the models itself, but rather on the comparison of the *"General"* with the *"Individual"* approach. An interesting point to see in this table is the fact, that the diagonal entries (hued in light blue for visual purposes) representing the *"Individual"* models do perform best. When summing the scores of those diagonal entries, the score value reaches 15 (in the 15-minute case) and 18 (in the hourly case) when the "best" –with the most points, according to the scoring scheme– *"Individual"* model (either trained with mean absolute error Loss-function or with the mean squared error Loss-Function) is chosen. This comparison of the sum of diagonal scores to the overall best model *LSTM all ("Simple")* score would represent the "Individual" vs. "General" approach. Another interesting finding can be seen, as the *"Individual"* models are poorly performing on *"Individual"* data sets on which there are not specifically trained.

Performance Overview of LSTM all models							
LSTM all	***15 minute data sets***						
trained on:	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"	Score
("Simple")	++, +	++, +	+, ∼	+, -	+, -	+, ∼	8
("Extra")	+, +	++, +	+, ∼	+, ∼	+, -	+, ∼	8
("X23")	∼, ∼	∼, ∼	++, -	-, ∼	∼, −	−, -	-5
("X70")	∼, ∼	∼, ∼	-, -	++, -	∼, ∼	∼, -	-2
("X111")	∼, ∼	∼, ∼	∼, -	∼, -	++, -	-, −	-4
("X115")	-, -	∼, -	-, ∼	∼, ∼	-, +	++, +	-1
("Simple")	+, ++	+, ++	+, +	∼, ∼	∼, ∼	∼, +	9
("Extra")	+, +	+, +	∼, +	∼, +	∼, ∼	-, +	6
("X23")	-, -	-, -	∼, ++	-, ++	-, ++	-, ∼	-1
("X70")	∼, ∼	-, ∼	-, +	+, +	-, +	∼, -	0
("X111")	−, −	-, −	∼, −	-, +	+, +	∼, ∼	-7
("X115")	-, -	−, -	−, ∼	−, −	−, ∼	+, ++	-10

LSTM all	***hourly data sets***						
trained on:	"Simple"	"Extra"	"X23"	"X70"	"X111"	"X115"	Score
("Simple")	++, +	+, ∼	+, -	+, -	+, -	∼, ∼	4
("Extra")	+, +	++, +	+, ∼	+, ∼	+, -	+, ∼	8
("X23")	∼, ∼	∼, +	++, ∼	-, −	∼, −	-, −	-5
("X70")	∼, ∼	∼, ∼	-, ∼	++, ∼	-, -	∼, -	-2
("X111")	∼, -	∼, ∼	∼, +	∼, -	++, ∼	-, -	-1
("X115")	-, -	-, -	-, -	∼, +	∼, +	++, ++	0
("Simple")	+, ++	+, +	+, +	∼, +	∼, ∼	+, +	10
("Extra")	+, +	+, ++	∼, +	∼, +	∼, ∼	∼, +	8
("X23")	−, −	−, −	∼, ++	−, ∼	-, +	−, -	-11
("X70")	∼, ∼	∼, ∼	-, -	+, ++	-, +	∼, ∼	1
("X111")	-, ∼	-, -	∼, ∼	-, ∼	+, ++	-, ∼	-2
("X115")	-, -	-, -	−, −	-, -	−, ∼	+, +	-10

Table 6.14: Performance overview of all twelve *LSTM all* models for the 15 minute (top) and the hourly data resolution (bottom). The very left column represents the 12 different *LSTM all* models. The data set description in parenthesis tells the data set on which the *LSTM all* model got trained on for 50 epochs. The first six rows (grey background) are the *LSTM all* models trained with the mean absolute error Loss-function. The six following models (white background) are trained with the mean squared error Loss-function. The row of the best overall model –scoring the most points, according to the scoring scheme– is tinted in blue. The worst performing models has its row tinted red. The diagonal entries are representing the *"Individual"* approach, and are tinted in light blue for visual purposes.

Chapter 7

Discussion

This final chapter first discusses some of the decisions made in this thesis defining the experimental setup. The effect of these decisions on the results, and hence how changing them might lead to different outcomes is elaborated. After that, potential pitfalls worth considering when dealing with the task of low-level energy forecasting are discussed. Finally, an outlook is given on interesting concepts and ideas that could be explored in further studies.

7.1 Decisions

7.1.1 Model Architectures and Hyperparameter Choices

As stated in chapter 5 it is difficult to define a fair comparison between different models, and different methodologies. One possible idea might be to use the same model architecture –the same number of hidden layers, the same number of hidden neurons, the same activation functions, etc.– for all data sets. However, this is likely to lead to an unjust comparison. The final choices of the architecture and hyperparameters are therefore based on certain neural network heuristics and resulted in the 12 different models as described in section 5.4. Some of the points worth considering regarding model architecture and hyperparameters choices are mentioned in the listing below:

- The reduction of data recordings from the 15-minute interval to the hourly interval is by the factor of four, the reduction in parameters for the respective models is roughly by a factor of two and therefore not exactly matching the reduction in data points. This might lead the models trained on the hourly data to be more likely to over-fit. The representational complexity per sample is different in the two data resolution, but the representational power of the models those not scale by exactly the same amount.
- The ”optimal” model architecture is likely to vary for the different data sets; the two *”General”* data sets:*”Simple”*, *”Extra”* and the four individual data sets: *”X23”*, *”X70”*, *”X111”* and *”X115”*. The best possible case would be to tailor the models architecture and its hyperparameters to each distinct data set.
- Input processing: The way the inputs get processed might not be the optimal choice for all households for getting a good forecast. The choice for

taking a whole week as a look-back window (and additionally incorporating auto-regressive time shifts up to 4 weeks –resulting in energy consumption information ranging back up to 5 weeks in the case of the *LSTM all* model)–, is made to exploit a big time window and learn from its consumption patterns. This might not be the optimal look-back size. And as is found in this thesis, the additional information concerned with the auto-regressive shift up to four weeks did not help the *LSTM all* models to perform superior then the *LSTM one* models, which only processed one week's information. The effect, the length of the look-back window does have on forecasting quality can be analyzed by varying its size, and comparing the models forecasting performances. It might be found, that distinct forecasting methodologies work optimally with different look-back windows. Furthermore, encoding features in a different way might help the models exploit additional information more efficiently. The raw historic energy consumption could be represented for example by the average amount of energy consumed over the last days, or other statistical quantities like the minimum and maximum consumption per a defined time interval. These quantities could be used as additional features, or as the main informational representation.

- Feature Importance: Attention has to be drawn to the fact, that in this thesis the impact of additional features is not analyzed on a feature to feature basis. What this means, is that the impact of the various features did not get analyzed on an individual level. The conclusion, obtained from the results, that the additional features did not help the *LSTM all* models to forecast better than the *LSTM one* models, allows for the statement that no advantage was to be found when all additional features were incorporated at the same time. However, is not valid to conclude, that each additional feature on their own does not help in getting better forecasts. To inspect the importance of one feature and its effect on the forecasting quality in more detail, the models have to be trained on various feature sets. By comparing models trained with all available features, to models trained with all but one, the effect of the left-out features on the forecasting performance can be analyzed. Trimming the number of used features up or down step by step and comparing the performances after each feature inclusion or exclusion shows the actual impact of the individual feature in more detail. For example, it might be found that the incorporation of feature A does help in getting better forecasts. But when feature A and additionally feature B are used, the forecasting performance might drop. It is not valid to state from the outcome of the last evaluation that neither feature A nor feature B do hold predictive power. It is only valid to conclude, that the incorporation of both features A and B in combination did not hold predictive power. Therefore, the conclusions drawn in this thesis about the feature importance can only be made in regard to the totality of all additional features. To analyze the impact of the auto-regressive shifts, or the dwelling size on its own, a more nuanced features analysis has to be performed.

- Over-parametrization: A big issue with neural network models is the huge number of parameters, that come along with a moderately sized model. Therefore, a balance between the amount and diversity of the data, with the number of parameters needs to be achieved. However, this balance is not fully understood. In [33] it is shown that the over-parametrization of models, i.e. using model architectures that lead to too many parameters compared to the amount of data, does often result in good models with robust generalizability. It might be due to the fact, that the high volatility in the energy consumption patterns acts as a potent regularization tool itself, and therefore the many parameters are not able to over-fit as might be suspected. The models in this thesis are equipped with many parameters compared to the number of input samples, especially in the case of the individual models.

- An additional difficulty emerges in the fact, that overfitting in the realm of multi-step forecasting can not be spotted as distinctly as in other common machine learning tasks, like classification or single point regression. As is elaborated in section 5.4, the Loss of the NN-models drops rapidly during the first three epochs in the training phase and then continues to decrease very slowly. For some data sets, the *MAE* started to rise on its *Test*-partition after a few epochs. However, through visual inspection of the predicted consumption profiles, it could be said, that the longer the training time, the more "reasonable" the forecast became on the *Test*-partition. This boils down to the crux of the problem in forecasting low-level energy consumption profiles, as the error metric based on which to evaluate the quality of a forecast might not explain the "goodness" of the forecast satisfyingly.

7.1.2 Data set selection

The selection of the *"Simple"* data set is an obvious choice. It incorporates all households (that have at least five months of consumption recordings, that enable the input/target processing) into the data set. The other choices, the filtering step to arrive at data set *"Extra"*, and the choices for the four individual households, are not that obvious. With the filtering step, the effect of dropping very low consumption profiles from the data was investigated. However, the thresholds were chosen based on domain knowledge and general assumptions about electricity consumption in single households. The "Individual" four data sets are selected based on the statistics on their historic consumption recordings. All four households had long periods of non-missing smart meter recording available, as well as relatively high overall consumption, compared to the total household stock. Because of this, the comparison of *"General"* vs. *"Individual" LSTM all* models is only performed with regard to those four households with relatively high energy consumption.

7.2 Future Work & Outlook

7.2.1 Error Metric

As is mentioned throughout this thesis, there is ongoing development in the field of forecasting energy consumption for the residential level. No consensus is yet achieved, on what might be the best error metrics to use. The end-use cases for different studies around the world are differing in forecasting horizon, household stock, data resolution, aggregation level, and forecasting evaluations. Due to these many

differences, it is hard to establish a definitive consensus on what is the best choice for evaluating forecasting performance given the differences in the above-mentioned points. Nevertheless, with more research in this area, driven by (a) the need to make the transition to a more efficient grid, and (b) the data avalanche coming with the smart meter and sensor readings; the scientific community will develop error measures tailored to certain end-use cases and modeling approaches. With this development, detailed advantages and disadvantages of certain methodologies will become more obvious and the adoption of proper methods and evaluation metrics becomes clearer.
Many studies and research activities working on the task of forecasting low-level energy consumption, evaluate the "goodness" of the obtained forecasts via a pointwise error metric. Often times this is either the mean absolute percentage error $MAPE$, the root means squared error $RMSE$ or the mean absolute error MAE. There is nothing wrong with using these error metrics as a calculation on which to evaluate the performances of differing approaches. However, it is important to be aware of the shortcomings of a pointwise error metric in the realm of forecasting energy consumption profiles. Knowledge about the potential disadvantage of these error metrics allows constructing additional metrics on which to evaluate the "goodness" of the forecasts. These extra metrics can be tailored explicitly to the task and its final use case. Therefore, the pointwise error metric can be used as a starting point, but might not be the best choice when used in isolation.

Unfortunately, there is no one-solution fits all metrics that can be applied to give a reasonable description of the "best" forecast. As is shown in [8, 27], the error metrics have to be tailored to:

1. the end-use case of the problem at hand
2. to the particular household stock that is considered in the study

Therefore, there is no single error metric that will always be the optimal choice. However, an effort is undertaken to establish consensus in the realm of forecasting low-level consumption to help make good decisions about error metrics as well as establishing baseline data sets on which different methodologies and metrics can be tested and compared.

Problem with a pointwise error metric: The main problem with a pointwise error metric in the realm of forecasting –especially for low level, multi-step ahead– is the *Double Penalty Error*.

Double Penalty Error

The *Double Penalty Error* describes the situation, in which the true consumption curve and the predicted consumption curves are actually pretty similar in shape and magnitude. The two profiles, however, have a time shift compared to each other. A simple illustration of the *Double Penalty Error* is shown in figure 7.1. In the most extreme case, both consumption curves are identical, but the predicted curve is always one time step before or after the actual true consumption. This forecast would probably make perfect sense for a lot of use cases, especially when the time interval is relatively short. However, calculating a point-wise error metric, -that only compares the true value y_t^{true} and predicted value $y_t^{forecast}$ at time step t-, results in a high error. Not only does the pointwise error metric penalize the forecast for being wrong at the time step t, but also for the next time step $t+1$, therefore the term *Double Penalty Error*. To circumvent this problem various alternative error metrics are introduced in the literature. In [27, 17] the authors use a modified *permuted error metric* where they make a slight adjustment to the pointwise error metric as far, as they not only compare the true value y_t^{true} with the predicted value at time step t but also with the forecasts at predefined shifts to the left (past) and to the right (future) of the actual time step. For instance, by specifying a sliding window of three, the true value is compared to the predicted values at time steps $y_{t-3}, y_{t-2}, y_{t-1}, y_t, y_{t+1}, y_{t+2}, y_{t+3}$. The time step with the closest values -out of those seven- to the true value at time step t is then taken to calculate the point-wise error and this is the error of the forecast for time step t. A further modification of this *permuted error metric* is to put weight on the forecasting error, according to the distance it is off. That is, the closer the forecast value is to the true value in regard to their time steps respectively, the better. By using this adjusted error metric it is possible to circumvent the most extreme cases of *Double Penalty Error*. It has to be noted, that this error metric, –as many others (accuracy measures, dynamic time warping, Pearson correlation)– has its advantages and disadvantages. One of the main issues with using the *permuted error metric* comes with the additional complexity of specifying the optimal window size, and the weights according to every time step. It is not obvious from the start, what might be an ideal choice of these parameters, and certainly depends on the data resolution and the end-use case. For instance, for the task of optimizing the charging and discharging cycle of a battery energy management systems, the weights with respect to particular time shifts regarding past and future might be different, then when considering the task of scheduling and providing flexibilities on the energy market via the aggregation of heat pumps, or the optimal targeting of a fleet of electric vehicles.

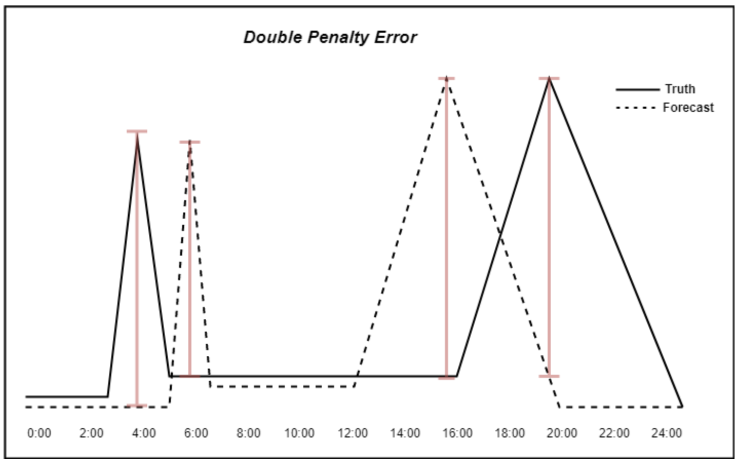

Figure 7.1: Illustration of the *Double Penalty Error*

Underprediction Bias

The training mechanism of neural network models is driven by minimizing a certain Loss-function. Therefore, models that are encouraged to minimize a pointwise error metric are trying to avoid the *Double Penalty Error* by not predicting consumption peaks. Resulting in models that only learn moderate consumption characteristics. This is certainly not advantageous for forecasting systems, where one of the more important systems estimates is the peak load of the grid. However, models trained based on the task of minimizing a pointwise error metric are doing poorly for forecasting peak energy consumptions. This further results in models with a *underprediction bias*, -that is models, which predict too low consumption profiles overall. This is especially true for the neural network models trained with the mean absolute error Loss-function. To investigate the *underprediction bias* for the proposed NN-models, the difference from the predicted amount of daily energy minus the true amount of daily energy consumed is analyzed. Other then the absolute difference regarded in tables 6.3 and 6.9 describing the *Daily Energy Consumption Deviation*; for visualizing the *underprediction bias* the real difference is taken.

$$\textit{Real Daily Difference} = \sum_{i=1}^{24/96} \hat{y}_i - \sum_{i=1}^{24/96} y_i \tag{7.1}$$

Where the upper limit of the sums is either 24 for the hourly time intervals or 96 for the 15 minute time intervals.
The boxplots in figures 7.2 and 7.3 show the magnitude of the *underprediction bias* on the *Test*-partition of two different data sets –*"Extra"* and *"X23"*– for the neural network models for both data resolutions (figure 7.2 15 minute time intervals, and figure 7.3 hourly time intervals). The boxplots are showing the distribution of the *Real Daily Difference* values for the various models (see equation 7.1). It can be seen, that the daily energy consumption forecasts of the NN-models trained with the mean absolute error Loss-function are consistently under predicting. Interestingly this is not to be found for the models that are trained to minimize the mean squared error Loss-function.
When considering only the mean absolute error as the main evaluation metric, the NN-models trained with the mean absolute error Loss-function seem to be the best performers. However, these models are suffering severely from the *underprediction bias*. Interestingly, this effect is not to be found for the neural network models trained with the mean squared error Loss-function. This is a surprising finding. Both Loss-functions are minimizing a pointwise error, therefore their different forecasting behavior might not be obvious. The difference in their forecasts can be attributed to the different parameter updating process during the training phase, resulting from the different starting point, described by the mean absolute error or the mean squared error.

To conclude this section on the choice of error metric: There is no one universal optimal error metric for all energy consumption forecasting tasks. A pointwise error metric is a useful starting point. However, there are certain shortcomings in these metrics. Therefore, some custom error metric might be developed, tailored to the task at hand, to better explain the quality of a forecast. Error metrics developed by the research community can be chosen as either appropriate support metrics or as the main metric based on which to evaluate the goodness of competing methods.
The most reasonable approach, therefore, might be in defining upfront what a good forecast would look like. Does it have to be as exact as possible at every time step? Is it more important to be close to the true amount of consumption happening over

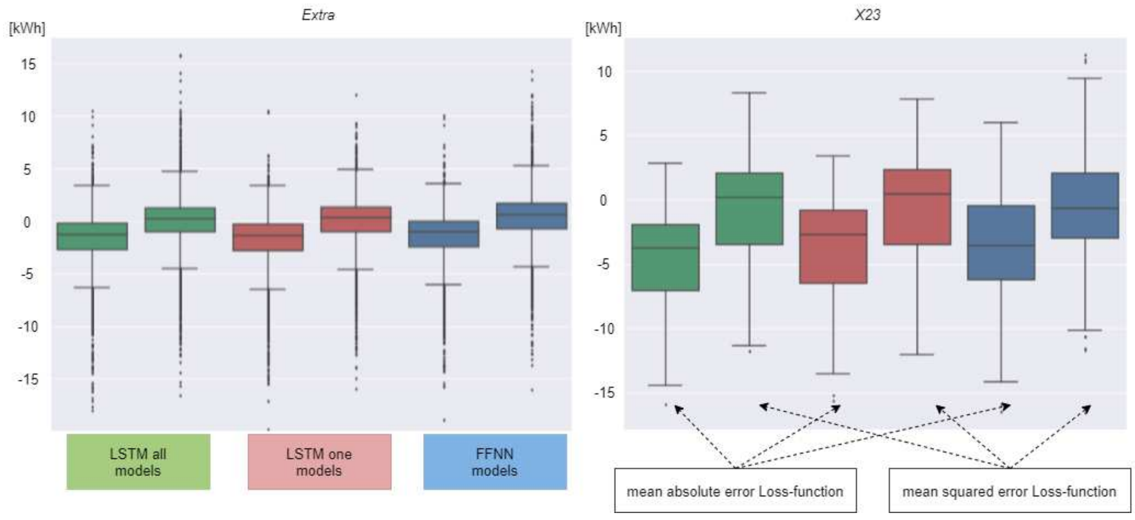

Figure 7.2: *Underprediction bias* on the *Test*-partitions of two different data sets (*"Extra"* on the left, and *"X23"* on the right). The boxplots show the distribution of the *Real Daily Difference* values in unit of [kWh] for the 15 minute time intervals. The three NN-model approaches –*LSTM all* (green), *LSTM one* (red) and *FFNN* (blue)– are grouped in coloured pairs. The left boxplot of each pair shows the distribution of the *Real Daily Difference* results obtained by the mean absolute error Loss-function trained model. The right boxplot shows the mse-trained counterpart.

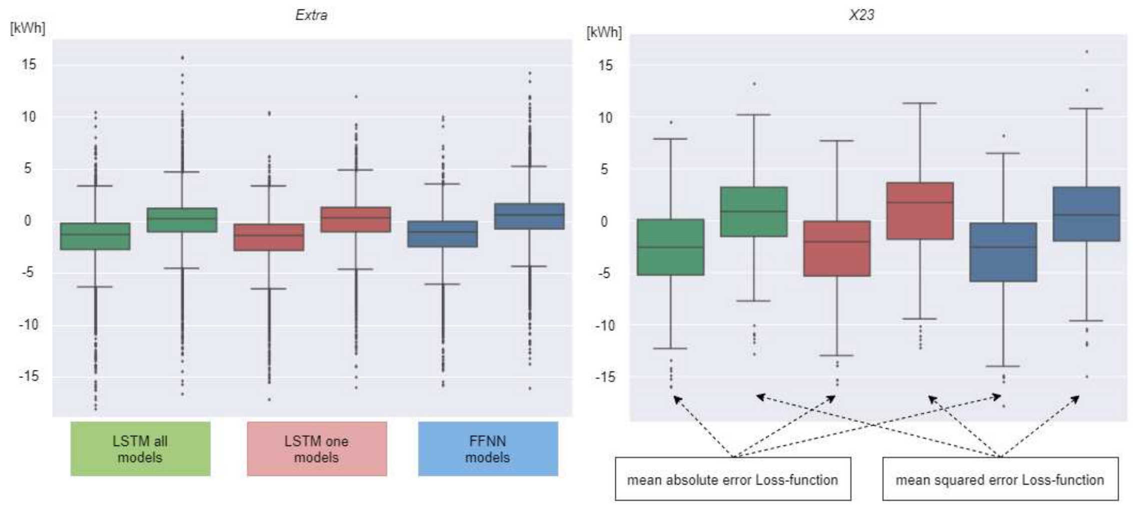

Figure 7.3: *Underprediction bias* on the *Test*-partitions of two different data sets (*"Extra"* on the left, and *"X23"* on the right). The boxplots show the distribution of the *Real Daily Difference* values in unit of [kWh] for the hourly time intervals. The three NN-model approaches –*LSTM all* (green), *LSTM one* (red) and *FFNN* (blue)– are grouped in coloured pairs. The left boxplot of each pair shows the distribution of the *Real Daily Difference* results obtained by the mean absolute error Loss-function trained model. The right boxplot shows the mse-trained counterpart.

a certain period? Is the most important factor, to capture peaks in the consumption profiles, and timely deviations are perfectly acceptable as long as they fall into a certain range? With a clear idea of what a good forecast has to deliver, the error metrics most appropriate for capturing this desired behavior can be chosen. Either one error metric is sufficient, or different metrics get included and combined to arrive at a reasonable judgment. With this, different metrics might get weighted differently based on their importance regarding the end-use case, resulting in a nuanced assessment.

7.2.2 Increasing Predictability

The biggest challenge in forecasting low-level energy consumption of single households is the volatility of the energy demand profiles itself. The main influencing factor of energy consumption on the low level is the behavior of the residents. This behavior certainly has some periodicity and regularity, nevertheless, it is very erratic, and therefore poses big challenges for getting accurate forecasts, especially when regarding a single household and considering short time intervals. Different actions can be taken to mitigate this problem and help to increase the predictability of the consumption profiles. In the following, four possible actions to achieve this are listed, and provide valuable paths for future studies, for combining and extending the work of this thesis.

Granularity This aspect is investigated via the two considered time intervals in this project. (a) 15 minute time intervals and (b) hourly time intervals. A number of studies have investigated the question of what effect time resolution does have on the predictability of the consumption profiles, and therefore on the forecasting performance [8, 3]. The main effect, an increase in the considered time interval does have on the consumption profiles is the out-smoothing of the consumption curve. Lifestyle is the driving factor of electric energy consumption at the household level. Although lifestyle certainly has its regularities and habitual actions, like doing many things the same way, especially during workdays, the exact time of energy consumption might vary. When considering short time intervals like 15 minutes this random behavior becomes obvious. The bigger the time intervals are chosen, more of the frequent actions fall into this interval. For instance, the energy consumption in the first three morning hours might be pretty stable for differing households at workdays and weekends, however, the exact time at which appliances of daily use are powered might vary in time. This intuitive, but very limited understanding of human energy consumption can be taken into account when regarding bigger time intervals. Again, it is important to tailor the considered time intervals to the end-use case. Therefore the forecasts for a four hour time interval will probably be more accurate then the ones obtained with the same time period but evaluated every 15-minute interval, however, it will not be useful for the task of obtaining optimal electricity prizes. The idea is that the bigger the considered time intervals are, the smoother the consumption profiles become, which therefore increases the predictability.

Aggregation Another possible action to increase the predictability of energy consumption profiles is to aggregate the consumption of households together [8, 3]. This is due to (a) that the baseline consumption of the aggregated profile is higher overall, decreasing the effect of the double penalty error at every time step, and (b) the aggregation of multiple consumption profiles leads to the out smoothing of the erratic consumption of one single household.
An interesting finding in the work of [18] is, that when comparing (a) the forecast of an aggregated energy consumption profile of a certain number of households to (b) the aggregation of individual forecasts of the single households (that were aggregated), the aggregation of the household forecasts performed better than the forecast of the aggregated household profile. This might indicate that there is valuable information in individual households, although they are small and the effect is not that severe. However, this information can be advantageous when aggregating all of those small informational gains to an overall forecast. Whereas forecasts based only on aggregated consumption do not have the potential of incorporating information about individual households.

Filtering Another idea of dealing with the fact of changing the raw energy consumption profiles to make them more predicable is the act of filtering the profiles according to certain expert knowledge or predefined criteria. This point falls more into the realm of data processing and data cleaning. This step, however, can help models learn to predict more reasonable consumption patterns. Outlier detection and outlier dropping fall into this category of actions. Rare events that are very extreme in either direction might not help, when incorporated in the training phase of numerous machine learning models. Especially neural networks are prone to outliers, that is they can be strongly affected by single very unusual training samples; whereas Support Vector Machines (SVM) are known to be more robust in dealing with outliers.
Knowledge about certain data collection methods and imputing practices, managing missing data in a reasonable way and filtering data based on certain statistical characteristics, can all be applied in this step. However, it is crucial to not process the data in a form that -although the models are now able to predict more reasonable- most of the data gets actually discarded, and does not yield an overall benefit in the eye of a long-term deployment of a system. For this reason, a solution must be thought of, for how to deal with samples, that are dropped during a certain filtering step.
In the case of this thesis, the filtering step described in the chapter 5 -based on the minimum consumption threshold filtering- drops a significant portion from the *"Simple"* data set. These dropped samples can be regarded as low-consumption samples, as they do not make the cut for the minimum consumption threshold. A separate minimum-consumption model can be used (either a neural network model, a simple benchmark, or a different approach altogether) that is suitable for these low-consumption samples. During inference time, when a forecast is needed, if the input gets filtered -by the energy threshold criteria- into the low-consumption data set the minimum-consumption model performs the forecast for this input.

Clustering A more sophisticated idea for dealing with the predictability of energy consumption is clustering. The main idea can be summed up as follows: Cluster or group ”similar” households together, forming a set of consumption profiles with ”similar” characteristics, and then train models for every distinctly found cluster. Therefore enabling the models to pick up on characteristic behavior of the households present in the distinct clusters. This idea assumes that different energy consumption behaviors mess up the models, resulting in models not being able to forecast properly.

The crux in clustering is in the measure of ”similarity” and the decision of what is the best metric to perform the clustering on. The most common approach found in the literature is the well known *K-Means* clustering algorithm. This algorithm calculates its similarity score based on the euclidean distance. Therefore this similarity measure is a pointwise metric. As in the case of the pointwise error metric, this has certain disadvantages, when grouping or clustering is performed based on this measure. For instance, the *K-means* algorithm would give two similar consumption profiles, that are shifted against each other in time, a very low similarity score. When clustering based on the *K-Means*, the number of clusters the algorithm will search for has to be predefined. It is not obvious what the optimal number of clusters is in advance. The *elbow method* or *silhouette method* are processing techniques that help to find the optimal number of clusters when applying the *K-Means* algorithm. Other common similarity measures that take the shape of the consumption profiles into account are the Pearson correlation or dynamic time warping. Here the dominating factor is not the pointwise closeness at each time step, but the overall similarity in shape and curvature.

Hierarchical clustering is a similar clustering method as the *K-means*, however, no upfront decision about the number of clusters has to be made. Hierarchical clustering comes with a slight increase in complexity, due to the fact, that the kind of linkage-method has to be selected, based on which clusters are formed during the clustering phase of the algorithm. Every cluster has the goal to minimize the average distance of all samples in the cluster to its center. When a new sample gets processed, it gets assigned to the cluster it is ”closest” to, where ”closest” needs to be specified by the similarity measure.

The goal of performing clustering in the first place is to find households that depict similar energy consumption patterns. With grouping them together, each cluster can then be targeted more specifically. This more specific targeting could mean using different forecasting methods for different clusters based on their consumption characteristics.

Not only the similarity measure, based on which to perform the clustering, needs to be chosen; but also the information based on which the clustering is performed needs to be considered. The clustering can be performed on (a) The historic energy consumption profiles of the various households. (b) Statistical quantities describing the historic energy consumption of the households (mean, min, max, number of outliers). (c) Demographic and personal information (educational status, age, number of people living in the household, appliances in the household). Furthermore, a decision has to be made if the clustering gets performed on the household- or on the sample- level. That is, should each sample be clustered individually, or should the households be clustered beforehand, and all samples obtain from a particular household belong to this cluster. An effort is undertaken to find the most promising information based on which to cluster, and then find the ideal clustering method for the chosen information.

Combination of Clustering and Classification: This is a further development of the idea of clustering. The first step -clustering households together based on a defined similarity measure- stays the same. After such clusters are formed, samples belonging to a particular cluster get marked as being a "member" of this cluster. The next step is using information about households, demographics, appliance information, etc., to observe which information has the most relevance for classifying the household into the correct cluster. Although the clustering might be performed on the energy consumption profiles, the classification step takes different information into account and can show which information is highly associated with a particular cluster. With this insight, questionnaires can be build to ask residents the most important questions, that were found to influence the energy consumption behavior the most, like –educational status, relationship status, appliances, age of residents– and so on.

All of the above-mentioned steps aim at increasing the predictability of the underlying consumption profiles. In [18] the authors declared the processing of the energy consumption profiles to make them more predictable, as the most important step; a necessity before diving into the choices and tweaking of sophisticated forecasting methodologies. No matter how well the model might be able to forecast, and how well it is tuned to the task at hand, this all is to no avail, if the underlying data is extremely volatile, where randomness and stochastic behavior are the dominant factors in the profile. Therefore, with the above-mentioned steps, the predictability of energy consumption profiles can be increased, allowing the subsequent modeling phase to tailor the models effectively.
With the clear end-use case in mind, and the knowledge about what a good forecast actually looks like, one, two or a combination of all of the above steps can be used to build a robust forecasting system. For some use cases, it will be suitable to regard bigger time intervals and aggregate over a certain number of households. Clustering can be applied beforehand, to only aggregate households that are belonging to the same cluster. In the end, the optimal forecasting model and processing steps need to be tailored to the specific use case. The final forecast then gets assessed based on the appropriate error- and performance-metrics. Keeping all those steps aligned, does increase the probability to achieve a robust and reliable forecasting system.

7.2.3 More Ideas

After exploring the most common ideas for increasing the predictability of energy consumption profiles, this section explores some other ideas that might help with the task of getting more reliable, stable and overall more accurate forecasts of the electric energy consumption of individual households.

More "meta" Information This thesis set out to answer the question if the consideration of additional information does help with the prediction of the next day's energy consumption of individual households. The results obtained indicate that the additional features considered for the *LSTM all* models do not help in getting better forecasts. As pointed out, this might be because: (a) the combination of the additional features do not carry useful information, (b) the models are not

able to utilize the additional information, due to their parameterization and learning phase, or a combination of both. Information that is not directly obtainable from the historic energy consumption profiles and its temporal ordering is often referred to as "meta"-data. The size of the dwelling, for instance, is one feature that falls into the category of "meta" information. Whereas the auto-regressive features and the temporal feature encodings do not.
Ongoing research is investigating what information is the most important for forecasting the low-level energy consumption. Weather data, temporal data were found to help for the high voltage level, but its feature importance decreases the lower the aggregation level gets.
Incorporating more relevant "meta"-features regarding the household's energy consumption, is likely to lead to better forecasts. Valuable data could include information about the buildings (e.g., age, insulation, heating system, size), appliances and sensory input (like noise signals, moving sensors), controllable loads, ownership of electric vehicles and other high consumption devices like HVAC, or information about the inhabitants itself (e.g. age, profession).

Tailored Cost- / Loss-Function in Neural Networks Additional error and performance metrics do help in making better assessments about forecasting quality. However, the underlying behavior of the neural network models itself does not change, as long as the NN-models are trained via a gradient descent algorithm to minimize the pointwise Loss-functions.
Creating a custom Loss-function for neural networks tailored to the task of low-level energy consumption forecasting could be a worthwhile endeavor. Neural networks are very powerful machine learning models, they are capable of approximating any functional relationship there is -provided they have the representational power needed-. However, neural network models do only what they are told to do. That is, in the case of this thesis, the *LSTM*, and the *FFNN* models, are trained to minimize two particular Loss-functions (mean absolute error Loss-function and the mean squared error Loss-function), over the totality of the training partitions. With the gradient descent algorithm, the weights/parameters of the models are continually adjusted and updated in a direction to minimize the output of the specified Loss-function. However, what if the Loss-function is not the best target, to begin with? Although the models are the "best" models when evaluated based on the mean absolute error metric, or mean squared error metric respectively –based on which they are trained–, this does not mean that they are the overall "best" model, when a more nuanced assessment with various useful performance metrics is performed. The idea of using a customized Loss-function is to harness the power of neural networks for finding the most optimal weights and parameters, based on an "optimal" Loss-function tailored to the task.
There are certain constraints that a Loss-function of neural networks has to obey. When the updating process is based on gradient descent, the most obvious one is that the Loss-function has to be differentiable with respect to the parameters characterizing the model. Another aspect has to do with the implementation of the Loss-function itself. The available Loss-functions in deep learning libraries are highly optimized for the fast processing of large amounts of samples. If a custom Loss-function is poorly designed, the computational overhead could make the back-propagation process very slow. However, if those points are considered, a custom

loss metric might harness the power of neural networks for finding the relationship from input to target and return forecasts that are ”best”, according to the new custom loss metric.
It can be the case, that by providing a different loss metric to the neural network models, the importance of additional values does change. That is, although there is information in the additional features considered, this information can not be exploited due to the constraints given by the loss metric to minimize the pointwise error. However, when not being constrained by the pointwise error, but being able to operate in a different ”frame”, the relationship might become obvious and features predictive power changes. However, as with the error metric, -based on which the forecasts are evaluated- there might not exist the ”perfect” Loss-function -based on which to train neural network models-.

Cumulative Profiles Using the cumulative energy consumption profiles as the target profile instead of the original consumption profiles might yield improvements in the forecasting models. The cumulative profile is formed by adding the daily prior total consumption to the current consumption value at time step t. This results in an ever-increasing curve, holding the total amount of energy consumed over the day in its last value. By regarding the cumulative curves, one minor drawback of considering the original profiles is mitigated, namely that the information of the daily amount of energy consumed is not directly accounted for. When taking the cumulative profile into account though, this issue is taken care of, as the amount of consumption of every day is then provided to the model in its final value corresponding to the last time step. It might be interesting to see how changing of the target profile –from raw energy consumption profiles to the cumulative profiles– does change the behavior of the neural network models, and if it results in more accurate forecasts.

Building a robust Forecasting System: To design a robust and reliable forecasting system many individual steps have to be considered. The first step is to clearly define the end-use case of the forecasting system. This clarity is essential to specify and decide on what makes up a good forecast. This allows for the appropriate choice of an error metric or a combination of metrics that make a good assessment of the forecasting quality. The next step might lie in trying to increase the predictability of the considered consumption profiles. This, of course, is constrained by the end-use case of the forecasting task. After exploiting the steps for increasing the predictability, the subsequent modeling phase tries to find and fine-tune the best models. Each of those steps is getting attention in the research community. Combining the best and most reasonable approaches increases the chances of getting robust forecasting systems.

Chapter 8

Conclusion and Ending Statement

This thesis explored the topic of energy consumption forecasting for individual households. Why forecasting is an important field of current research, and how machine learning plays a supportive role in the transition of the power grid is discussed. Different neural network methodologies were compared and evaluated. Long Short Term Memory network models were tested against feedforward neural network models. Testing those NN-models against three benchmarks gave insights about the usefulness of sophisticated machine learning approaches. Feature importance was investigated by the incorporation of additional features for the *LSTM all* model. The effect of data granularity was examined by looking at both 15 minute and hourly sampled consumption recordings. The challenges emerging in the field of low-level energy consumption forecasting were investigated. Especially the consequence of using pointwise error metrics for evaluating the quality of a forecast is reviewed. The importance of knowing the advantages and disadvantages of certain error metrics is pointed out, as well as alternative error, and performance metrics were mentioned. Finally, interesting concepts and ideas are mentioned to expand on the work of low-level energy consumption forecasting.
Combining powerful machine learning models with expert domain knowledge and the steady growth of data acquisition is providing opportunities and big challenges for the advancement of the energy grid. Finding optimal solutions to the arising challenges by optimizing every aspect of a forecasting system will be a crucial development to a more sustainable and efficient energy grid in the coming years.

Bibliography

[1] Energie Control Austria, "Bericht zur Einführung von intelligenten Messgeräten in Österreich," 2019. https://www.e-control.at/marktteilnehmer/strom/smart-metering/monitoring, Accessed: 2019-11-30.

[2] European Commission, "Smart grids and meters," 2019. https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters/overview, Publised: 31 July 2014, Last update: 29 October 2019. Accessed: 2019-11-30.

[3] B. Yildiz, J. I. Bilbao, J. Dore, and A. B. Sproul, "Short-term forecasting of individual household electricity loads with investigating impact of data resolution and forecast horizon," *Renewable Energy and Environmental Sustainability*, vol. 3, p. 3, 2018.

[4] SMARTGRIDS Austria: Die österreichische Technologieplattform zum Thema Smart Grids, "Was sind smart grids?." https://www.smartgrids.at/smart-grids.html, 2019. Accessed: 2019-11-30.

[5] European Commission, "Regulation of the european parliament and the council on the internal market for electricity," 2017.

[6] European Commission, "Directive of the european parliament and of the council on common rules for the internal market in electricity," 2017.

[7] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana, "A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart buildings," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1460–1495, 2014.

[8] B. Yildiz, J. Bilbao, J. Dore, and A. Sproul, "Recent advances in the analysis of residential electricity consumption and applications of smart meter data," *Applied Energy*, vol. 208, pp. 402–427, 2017.

[9] H. Hamedmoghadam, N. Joorabloo, and M. Jalili, "Australia's long-term electricity demand forecasting using deep neural networks," *arXiv preprint arXiv:1801.02148*, 2018.

[10] B. Hayes, J. Gruber, and M. Prodanovic, "Short-term load forecasting at the local level using smart meter data," in *2015 IEEE Eindhoven PowerTech*, pp. 1–6, IEEE, 2015.

[11] S. Ryu, J. Noh, and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, p. 3, 2017.

[12] A. Rahman, V. Srikumar, and A. D. Smith, "Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks," *Applied energy*, vol. 212, pp. 372–385, 2018.

[13] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, 2015.

[14] J. W. Taylor and P. E. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2213–2219, 2007.

[15] Y.-c. LI, T.-j. FANG, E.-k. YU, *et al.*, "Study of support vector machines for short-term load forecasting [j]," *Proceedings of the Csee*, vol. 6, p. 10, 2003.

[16] X. Wu, J. He, T. Yip, N. Lu, *et al.*, "A two-stage random forest method for short-term load forecasting," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2016.

[17] O. Valgaev, F. Kupzog, and H. Schmeck, "Designing k-nearest neighbors model for low voltage load forecasting," in *2017 IEEE Power & Energy Society General Meeting*, pp. 1–5, IEEE, 2017.

[18] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid*, 2017.

[19] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2018.

[20] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustainable Energy, Grids and Networks*, vol. 6, pp. 91–99, 2016.

[21] N. G. Paterakis, E. Mocanu, M. Gibescu, B. Stappers, and W. van Alst, "Deep learning versus traditional machine learning methods for aggregated energy demand prediction," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–6, IEEE, 2017.

[22] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087–1088, 2018.

[23] Y. Peng, Y. Wang, X. Lu, H. Li, D. Shi, Z. Wang, and J. Li, "Short-term load forecasting at different aggregation levels with predictability analysis," *arXiv preprint arXiv:1903.10679*, 2019.

[24] D. Alberg and M. Last, "Short-term load forecasting in smart meters with sliding window-based arima algorithms," *Vietnam Journal of Computer Science*, vol. 5, no. 3-4, pp. 241–249, 2018.

[25] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, “An overview and comparative analysis of recurrent neural networks for short term load forecasting,” *arXiv preprint arXiv:1705.04378*, 2017.

[26] B. Yildiz, J. I. Bilbao, and A. B. Sproul, “A review and analysis of regression and machine learning models on commercial building electricity load forecasting,” *Renewable and Sustainable Energy Reviews*, vol. 73, pp. 1104–1122, 2017.

[27] S. Haben, J. Ward, D. V. Greetham, C. Singleton, and P. Grindrod, “A new error measure for forecasts of household-level, high resolution electrical energy consumption,” *International Journal of Forecasting*, vol. 30, no. 2, pp. 246–256, 2014.

[28] A. Veit, C. Goebel, R. Tidke, C. Doblander, and H.-A. Jacobsen, “Household electricity demand forecasting–benchmarking state-of-the-art methods,” *arXiv preprint arXiv:1404.0200*, 2014.

[29] P. Lusis, K. R. Khalilpour, L. Andrew, and A. Liebman, “Short-term residential load forecasting: Impact of calendar effects and forecast granularity,” *Applied Energy*, vol. 205, pp. 654–669, 2017.

[30] D. L. Marino, K. Amarasinghe, and M. Manic, “Building energy load forecasting using deep neural networks,” in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 7046–7051, IEEE, 2016.

[31] R. Sevlian and R. Rajagopal, “A scaling law for short term load forecasting on varying levels of aggregation,” *International Journal of Electrical Power & Energy Systems*, vol. 98, pp. 350–361, 2018.

[32] J. Zheng, C. Xu, Z. Zhang, and X. Li, “Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network,” in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2017.

[33] H. S. Hippert, C. E. Pedreira, and R. C. Souza, “Neural networks for short-term load forecasting: A review and evaluation,” *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.

[34] Y. Shin and G. E. Karniadakis, “Trainability and data-dependent initialization of over-parameterized relu neural networks,” *arXiv preprint arXiv:1907.09696*, 2019.

[35] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[36] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* ” O’Reilly Media, Inc.”, 2017.

[37] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[38] M. A. Nielsen, *Neural Networks and Deep Learning”*. Determination Press, 2015.

[39] M. Mazur, “A step by step backpropagation example.” https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/, 2015. Accessed: 2019-11-30.

[40] V. M. Sebastian Raschka, *Python Machine Learning*. Packt Publishing Ltd., 2017.

[41] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] J. D. S. Abraham Kang, “A lstm backpropagation example.” https://www.kdnuggets.com/2019/05/understanding-backpropagation-applied-lstm.html, 2018. Accessed: 2019-11-30.

[43] F. Chollet, “Keras: The python deep learning library.” https://keras.io/. Accessed: 2019-11-26.

[44] F. Chollet, *Deep Learning with Python*. Greenwich, CT, USA: Manning Publications Co., 1st ed., 2017.

[45] R. V. Jones, A. Fuertes, and K. J. Lomas, “The socio-economic, dwelling and appliance related factors affecting electricity consumption in domestic buildings,” *Renewable and Sustainable Energy Reviews*, vol. 43, pp. 901–917, 2015.