Machine Learning Approaches for Energy Forecasting

Dissertation zur Erlangung des
Doktorgrades der Naturwissenschaften
(Dr. rer. nat.) der Fakultät für Physik

der Universität Regensburg

vorgelegt von

Christian Lang

aus

Hof (Saale)

im Jahr 2021

Promotionsgesuch eingereicht am: 27.04.2021
Die Arbeit wurde angeleitet von: Prof. Dr. Elmar W. Lang

Prüfungsausschuss:

Vorsitzender:	Prof. Dr. Jascha Repp
1. Gutachter:	Prof. Dr. Elmar W. Lang
2. Gutachter:	Dr. Stefan Solbrig
weiterer Prüfer:	Dr. Alfred Jay Weymouth

Contents

List of Abbreviations

Adam	Optimisation algorithm utilising Adaptive Moment Estimation
AR	Auto-Regression
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	Autoregressive Integrated Moving Average with Exogenous Variables
ANN	Artificial Neural Network
CER	Commission of Energy Regulation Ireland
CHP	Combined Heat and Power Plant
CNN	Convolutional Neural Network
CO_2	Carbon Dioxide
EEG	Erneuerbare Energien Gesetz
GRU	Gated Recurrent Unit
IPCC	Intergovernmental Panel on Climate Change
IRE15	Aggregated load of 15 residential households
IRE40	Aggregated load of 40 residential households
IRE350	Aggregated load of 350 residential households
ISSDA	Irish Social Science Data Archive
LSTM	Long Short-Term Memory
LTLF	Long-term load forecast
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error

MLP	Multi Layer Perceptron
MSE	Mean Squared Error
MTLF	Medium-Term Load Forecast
Nadam	Nesterov-accelerated Adam optimser
NLP	Natural Language Processing
NN	Neural Network
OLS	Ordinary Least Square
PV	Photovoltaic
ReLU	Rectified Linear Unit
RLM	Reale Lastgangmessung
RNN	Recurrent Neural Network
SARIMA	Seasonal ARIMA
SLP	Single Layer Perceptron
STLF	Short-Term Load Forecast
SVM	Support Vector Machine
VSTLF	Very Short-Term Load Forecast

1 Introduction

1.1 Motivation

Climate change concerns us all, inevitably, undeniably.
The fight against anthropogenic climate change and its impacts is the biggest challenge humankind faces in the 21st century. There are several effect of the progressing climate change that can already be observed today. The increasing global temperature is causing glaciers and the polar caps to melt at record speed. The increased carbon dioxide content of the air is acidifying the oceans and rapidly changing the living environment of a large part of the ocean's creatures. Climate change and its impacts, however, not only effect nature and the living conditions of wildlife but also us humans. [1, 2]
Higher temperatures, for example, have negative health effects. Additionally, they are very likely the cause of an increasing amount and intensity of extreme weather events, both of which endanger human lives and the lives of wildlife in the same manner. Furthermore, a rising sea-level threatens to make densely populated coast lines uninhabitable. Those are only a few of the direct and indirect impacts of climate change on human life in the foreseeable future. They show that climate change concerns everybody, even if no direct impacts on daily life are yet recognisable, and stress that limiting the human impact on our climate should, therefore, be acted upon fiercely. [1, 2, 3]
Fighting climate change and developing innovative solutions to reduce the emission of harmful greenhouse gases is a challenge that concerns everybody. The scientific community, in particular, is called upon to come up with new ideas and concepts to combat this global challenge. This thesis is my humble attempt to contribute to the solution.

In the 19th century, the foundations for today's climate research were already laid. The mathematician and physicist Joseph Fourier discovered the greenhouse effect in 1824 [4]. Later, Tyndall [5] was able to measure the behaviour of several gases and vapours under infrared irradiation and Arrhenius [6] was able to describe the atmospheric greenhouse effect quantitatively.
The German natural scientist Alexander von Humboldt is said to be the first scientist who recognised the human influence on nature and (micro-)climate [7, 8]:

> "Veränderungen [...] welche der Mensch auf der Oberfläche des Festlandes durch das Fällen der Wälder, durch die Veränderung in der Vertheilung der Gewässer und durch die Entwicklung grosser Dampf- und Gasmassen [...] hervorbringt. Diese Veränderungen sind ohne Zweifel wichtiger, als man allgemein annimmt."

Today, the *Intergovernmental Panel on Climate Change* (IPCC) is investigating the causes and impacts of anthropogenic climate change, and advises on which global measures need to be taken in order to minimise those impacts. They regularly publish reports on updated findings concerning climate change and its implications. The latest report was published in 2013 and 2014 [2] and the next one is to be published in 2022.
The the fifth IPCC assessment report emphasised that the warming of atmosphere and oceans is "unequivocal" and that it is "extremely likely" that humans are the dominant cause of the warming since the 1950s. The reason for this is the progressing emission of greenhouse gases that increased the concentration of greenhouse gases in the atmosphere to the largest levels in 800 000 years. In a special report, published in 2018 [9], the IPCC offered pathways to limit the rise in global temperature. In a nutshell, all suggested scenarios call for a drastic reduction in the emission of greenhouse gases within the next fifty years.

The reports clearly show that the emission of greenhouse gases, like carbon dioxide, must be reduced drastically in order to slow down or stop the human-induced climate change. A large part of CO_2 emission originates from coal and gas powered power plants, which are corner stones of the electricity supply in many countries [1]. It is, therefore, necessary to develop alternatives for those power plants and eventually shut them down. In the last few decades, a multitude of technologies that are nowadays widely used and that utilise renewable energy sources to generate electricity, for example wind and solar radiation, have been developed. The challenge of integrating electricity from renewable sources and eventually completely replacing the electricity from fossil energy sources in our power-grid is based on the uncertainty in planning with and the volatility of the renewable energy sources. Therefore, new concepts that factor those disadvantages in and ultimately eliminate them must be developed. There are a lot of different concepts that deal with the challenge of integrating renewable energy sources into the energy market. What most concepts have in common is the need of accurate predictions of the possible energy production and the anticipated energy load.

This thesis focuses on predicting the electricity loads and, therefore, investigates different options for load forecasting utilising machine learning models. The performance and the feasibility of generating those forecasts are evaluated. In the

next section, an energy concept that improves the integration of renewable energy sources is introduced and the importance of accurate load forecasts is explained.

1.2 Project MAGGIE

The research of this thesis is embedded in the research project MAGGIE, which is funded by the Federal Ministry for Economic Affairs and Energy. The project addresses several objectives of federal initiatives regarding energy efficiency and the transition from fossil to renewable energy sources, for example the *6. Energieforschungsprogramm der Bundesregierung*, the *Leitinitiative Zukunftsstadt*, and the *Nationaler Aktionsplans Energieeffizienz der Bundesregierung* (NAPE) to name just a few.
The goals of the project are manifold. One focus is to reduce the electricity and heat consumption of apartment buildings, particularly in older buildings. Two potential options to achieve that are explored in MAGGIE. Firstly, to improve the structure of the building by improving the insulation and, secondly, to increase the efficiency of the systems which provide heat and electricity. Another research question of the project is how to better integrate renewable energies in the energy market with the help of decentralised energy management. For that purpose, new energy concepts with sector coupling and energy concepts for whole city quarters are developed through MAGGIE. [10]

Typical, broadly available renewable energy sources are wind, solar radiation, and biogas produced through fermentation of organic material. The first two sources are by far the biggest renewable contributors in the energy market, but they come with their own challenges. The energy allocation of power plants that rely on natural sources like solar radiation or wind cannot be planned as precisely as those which rely on "traditional" energy sources, like coal. For a coal power plant, for example, it is possible to accurately plan how much energy is provided at which time. In addition, it is possible to adjust that production schedule on short notice. Unfortunately, most renewable energy sources do not allow for those possibilities. The solar radiation fluctuates due to permanent changing cloud cover, precipitation, and concentration of aerosols. The wind speed and direction changes constantly and highly varies locally. All those factors not only perpetually change the power which can be gained from the wind and solar radiation, but also make it difficult to establish an accurate power forecast for renewable energy power plants. Without a reliable forecast, however, integrating renewable energies into the energy market is challenging. Power grid operators must ensure a nearly constant voltage and frequency of their power grids while increasing the share of more volatile renewable

energies in their energy mix. By 2050, at least 80 percent of the German electricity has to be contributed by renewable energy sources according to the EEG law [11]. In order to achieve that, methods have to be implemented that are, on one hand, capable of balancing the fluctuations in the electricity generation from renewable energy sources and, on the other hand, work better with uncertain forecasts. Only when those methods exist, can renewable energy power plants permanently replace power plants that use coal and natural gas, therefore increasing the sustainability in the power market. [12]

The volatile renewable energy sources, wind and solar radiation, are mostly used to produce electricity. In comparison to other energy forms, storing electric energy locally is costly either due to high losses (e.g. in electrolysis) or expensive equipment (e.g. batteries). The key to better integrating electricity from renewable sources into the power market, however, is to provide a barely fluctuating, stable energy source. One promising solution to achieve that is sector-coupling. It means connecting the different energy sectors of electricity, gas, and heat in a beneficial way and thereby utilising their advantages and compensating their disadvantages. Connecting energy sectors typically means creating the possibility to transform one energy form into another when it is profitable.

An example of sector-coupling is power-to-gas technology, which produces hydrogen or methane using electrolysis. Of course, using the electrical energy directly is generally more efficient than transforming it into gas. However, the advantages of gas over electricity are that it can be easily stored, that it can be easily transported, and that the German gas grid has an enormous storage capacity, which can be utilised. Nowadays, the electricity power grid which connects the big off-shore wind power plants at the northern German coast with the highly industrial areas in central and southern Germany is regularly overloaded during periods of strong winds along the whole German coastline. For that reason, wind power plants have to be shut down regularly and potential renewable electricity is lost, which then has to be produced in traditional power plants instead. The first facilities that use the superfluous electricity for power-to-gas so the energy can be used later, albeit with losses, are currently in use. The gas grid and gas storage can therefore be seen as the battery for renewable energies on the way to a more sustainable energy usage. [10, 12]

Through MAGGIE, the ideas of sector-coupling are applied on a more local level. In the project, a building from the *Baugenossenschaft Margaretenau* in western Regensburg was energetically renovated and evaluated, as well as equipped with a novel energy system that enables the implementation of the ideas of sector-coupling on a local scale. A novel central energy system was developed for the building which

provides heat (via heated process water and warm drinking water) and electricity. The system consists of a combined heat and power plant (CHP), a heat pump, a photovoltaic (PV) power system, and a heat buffer.
A CHP is similar to a generator, the difference being that both, the generated heat and electricity can be used. Hence, it has a very large efficiency. A heat pump uses electricity to generate heat by transferring thermal energy from a colder thermal reservoir to a warmer thermal reservoir. The PV system produces electricity from solar radiation. A scheme of the energy system can be seen in figure 1.1. In addition, a connection to the power grid can receive surplus electricity or provide electricity. The CHP and the heat pump both provide thermal energy, one from fuel the other from electricity. The heat demand of the residents can therefore either be met by using electricity or fuel, whereat the latter option additionally produces electricity. The electricity demand can also be met by two sources, the CHP and the PV system. All parts of the system are monitored continuously and can be controlled remotely and independently by a central control unit. Therefore, it is possible to shift the production and consumption of heat and electricity in time and between the participants of the system by heating and using the water of the heat buffer at the needed times. This allows for optimisation of the machine schedules depending on an optimisation target, for example a cost reduction or a high degree of self-sufficiency.
The advantage of the system becomes clear when two factors are taken into consideration. First, the system is meant to be scalable up to whole city districts. That means it can consist of more than one CHP, heat pump, etc., that are then connected to each other and are all controlled by the central control unit. And second, the feed-in and consumption from the power grid has to be reported to the power grid operator in a 15 minute grid on the previous day, if the installed power of the system exceeds a certain value. When multiple apartment buildings are supplied using CHPs, the installed power of the resulting energy system surpasses this threshold. At the time the machine schedules are calculated, the electricity and heat demand of the residents and the electricity production by the PV system are only predictions and therefore inaccurate. Deviations in production and consumption, however, can be compensated for with this system by the CHP and the heat pump utilising the possibility to store and access thermal energy. Thereby, the declared electricity schedule for the power grid can be met despite deviations of the consumption and production forecasts. This facilitates the integration of PV systems into local energy system that provide a decentralised energy supply. [13, 10, 12]

The energy system can furthermore be used to act in a grid-supportive manner. When too much or too little energy availability in the power-grid is impending, the system can help stabilise the grid by compensating for the energy difference. With the system it is possible to reserve capacities of the thermal energy storage for either

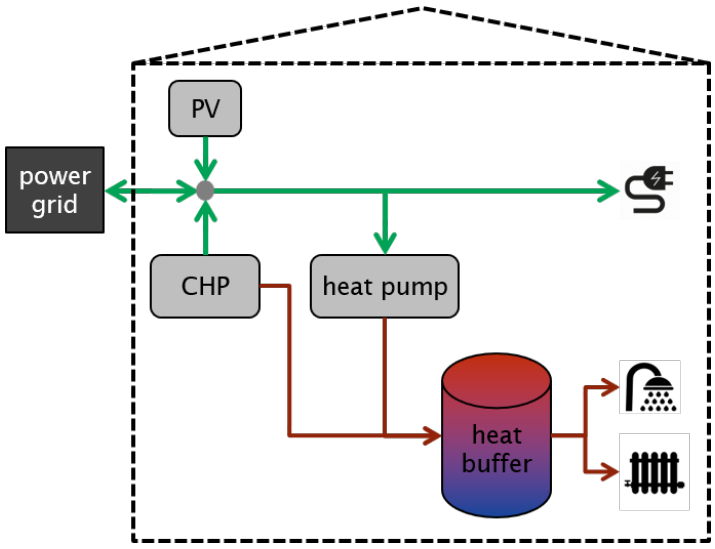


Figure 1.1: Simplified schema of the energy system that is installed in the Margaretenau for the project MAGGIE. It shows how the different components of the energy system are connected. The green arrows indicate the possibility of electricity transfer, the red arrows of heat transfer via heated water. The components are completely interconnected to enable variability in production and consumption of the components.

extracting electricity from the power grid and transforming it into thermal energy with the heat pump or feeding-in electricity from the CHP. Naturally, this concept is only applicable in a larger scale. When several similar systems act together as a so-called *virtual power plant*, they can assist in stabilising the power-grid. The influence of one system on the power grid is simply too small. Initial trials of the concept [14, 15] have recently been successfully conducted. [10, 12]
When a virtual power plant acts grid-supportive, it enables the power-grid operator to react on short notice to deviations from the expected electricity production in the grid. Those deviations can, for example, be caused by wind power plants that generate less electricity than anticipated. With the virtual power plant at their disposal, the grid operator can utilise it to generate and feed in the lacking electricity. The virtual power plant, thus, allows for compensating the inevitable uncertainties the wind power plants introduce and stabilises the power grid. Without the possibility of adjusting the electricity production of a power plant on short notice, the integration of renewable energy sources would not be possible. Today, the only power plants that are able to spontaneously adjust their schedules are gas and, to a certain extent, coal power plants. Those, however, must be replaced in order to reduce the emission of greenhouse gases that fuel climate change.

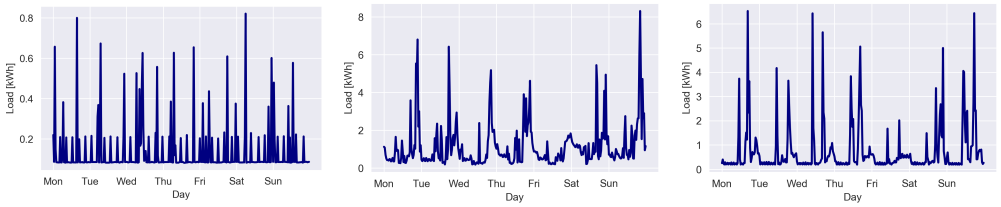
The crucial part in operating the local power system in project MAGGIE is planning and calculating the machine schedules in a way that both ensures the availability of electricity and heat for the residents at all times and achieves a maximum efficiency regarding the optimisation target. The optimisation must particularly take the volatility of the electricity and heat consumption of the residents into account. In big grids of entire city districts or even entire regions, the average consumption, the so-called standard load curve, can simply be used as load forecast. In single apartment buildings, however, the standard load curve is not feasible as a forecast due to the high volatility and the big influence that changes in behaviour of single households have on the total load of the building. Consequently, the quality of the optimisation depends strongly on the accuracy of the load forecasts and more sophisticated methods of computing forecasts need to be applied.

1.3 Dataset

In this work, the goal is to develop an algorithm for accurately forecasting electricity loads. The focus lies on electricity and not heat forecasts mainly for one reason - a lack of storage capacity. A surplus or underproduction of heat can be easily compensated internally by the installed central heat puffer and, therefore, barely compromises the reliability of energy supply, whereas a discrepancy between electricity consumption and production can only be compensated by either exchanging electricity with the power grid, resulting in large fees due to the deviation of the declared electricity schedule, or by producing undesired and unscheduled heat with the CHP or heat pump, which in return negatively influences the upcoming schedules. Therefore, knowing the future electricity load as precisely as possible is of paramount importance. The algorithm to be developed for predicting electricity load can naturally be applied to forecasting the heat demand as well.

As one requirement for the algorithm is to be scalable in order to be applicable from single apartment buildings to whole city districts, the data must represent all these different scenarios.
The plan was that such data was to be collected by smart meters in the building during the project duration of MAGGIE. In Germany, they are also called *RLM-Zähler*, which stands for *registrierende Lastgangsmessung*. Unfortunately, due to delays in the renovation of the historical apartment building in the Margaretenau, changes and readjustments of the project road map and objectives, and the bankruptcy of an essential project partner, this data could not be collected during the project duration or, respectively, during the making of this thesis.

Therefore, I had to fall back on publicly available data. The data should represent the electricity demand on a local or building level.
The data eventually used originates from the *Smart Meter Trial* in Ireland conducted by the *Commission for Energy Regulation (CER)*. It is made available by the *Irish Social Science Data Archive (ISSDA)* on their homepage [16]. The dataset contains roughly 5000 electricity load time series of single consumers, recorded for a period of 17.5 months from July 14th 2009 until the end of the year 2010. The data was collected in a 30-minute grid. 3273 of those time series are labelled as residential, the remaining series are either from businesses or unlabelled. No further information was given about the type of housing, the number of inhabitants, or the location mainly due to privacy reasons. Figure 1.2 illustrates how diverse the load curves of individual residential households can be.



(a) This load series displays no day/night patterns, probably no resident is home. The regular spikes originate presumably from a regular consumer, e.g. a fridge.

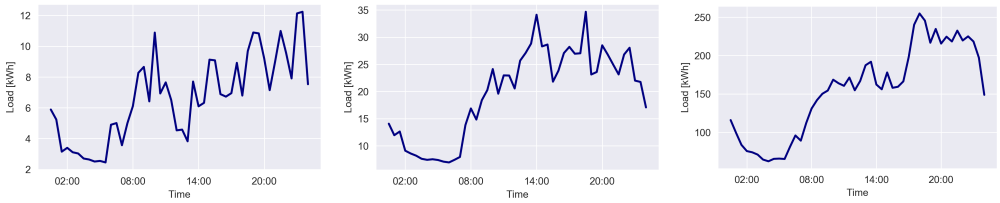
(b) In this load series, the day and night difference is recognisable. The consumption of the different days, however, varies significantly.

(c) This load series shows a regular behaviour, with Monday to Thursday being particularly similar. Moreover, the behaviour on the weekend differs clearly from the weekdays.

Figure 1.2: The three graphs show electricity loads of individual residential households and demonstrate how different the load of the same week can look like. All time series show a large volatility and about the origins of the occurring load spikes can only be speculated.

For the energy system described in the previous section, only the accumulated electricity load of the whole building is of significance. Hence, only the accumulated load needs to be predicted. Processing the electricity consumption of single households would also be extremely questionable in terms of data and privacy protection. With the *Smart Meter Trial* dataset such accumulated load time series can be easily created by adding together the load time series of an arbitrary number of residential buildings.
In order to have time series similar to the ones that would have occurred in the MAGGIE project, three different aggregated load time series were created and used during the development of a forecasting model: a series of 15 households, 40 households, and 350 households. The households were chosen randomly. The 15 household

load time series corresponds to the building that is renovated and equipped with the new energy system in the project MAGGIE, which consists of around that number of apartments. There are larger apartment buildings in the city district as well, which are represented by the second time series. Finally, the 350 household load resembles the whole city district of Margaretenau. In the following the three load time series are referred to as IRE15, IRE40, and IRE350. An extract of the aggregated loads is shown in figure 1.3.



(a) Combined load of 15 households. **(b)** Combined load of 40 households. **(c)** Combined load of 350 households.

Figure 1.3: The graphs show the consumption of the different aggregated electricity consumption time series on the first day of the dataset. It can clearly be seen that the volatility the series exhibits decreases with a higher aggregation level.

The computations performed during the research into time series forecasting models were conducted on two different machines. For the less costly and extensive computations an ordinary laptop with four kernels was used. Mainly the fitting and training of the linear and regression methods, as well as several preliminary tests were conducted on the laptop. The more costly and time-consuming computations were performed on a gaming computer with an Intel i7 processor containing eight cores (16 threads) running with a clock frequency of 3.80 GHz. The computations were distributed on all threads. Utilising the graphics board for training the neural networks did not decrease the computation duration. A training epoch of the convolutional neural networks used in this thesis utilising the aggregated electricity load datasets typically took between six and twelve seconds, depending on the complexity of the network.

1.4 Research Objectives

The detailed description of the project MAGGIE and its objectives stressed the importance of electricity load forecasting. Computing forecasts is essential for effectively planning and optimising the scheduling of the local energy system. Until now predicting the aggregated electricity load of single buildings or small accumulations of buildings has not been a necessity. Therefore, methods for computing those forecasts are still limited.
This thesis offers a comparative study of different machine learning methods that can be applied to time series forecasting. Additionally, a novel forecasting method, which utilises convolutional neural networks, is developed in the course of this thesis and the development process is portrayed in detail. An evaluation of the ability to compute accurate forecasts for the aggregated electricity load time series from different numbers of consumers is conducted for all models. They are compared to each other and the advantages and disadvantages of each model are illustrated.

The subjects covered in this thesis are discussed in the following order.
After the introduction, the relevant literature is presented in chapter 2, in order to get an overview of the research area. Afterwards, chapter 3 covers the different methods that are applied in this thesis. First, neural networks are introduced in 3.1 and the development from a single layer perceptron to deep neural networks and how they are trained is illustrated in the sections 3.1.2 to 3.1.5. Subsequently, convolutional neural networks are discussed in section 3.1.6. After concluding with neural networks, established machine learning methods are introduced - first, different regression models in 3.2.1, followed by recurrent neural networks in 3.2.2, and finally tree-based methods in 3.2.3. At the end of this chapter, in 3.3, how the different methods are applied to time series forecasting is explained. Chapter 4 includes and discusses the evaluation results. At the beginning, how the datasets are split in 4.1 and which features are used for calculating the forecasts in 4.2 is described. Afterwards, the forecast performance of the established models is presented: first, for simple baseline models in 4.3, followed by the fully-connected neural networks in 4.4, recurrent neural networks in 4.5, and the tree-based models in 4.6. This is followed by the detailed representation of the development process of the novel model, which is based on convolutional neural networks, in 4.7. The final models are presented in 4.7.6, are analysed in depth in 4.7.7, and are compared with the established models in 4.8. In the conclusive chapter 5, the findings of the work are summarised, discussed with respect to the current research, and limitations and future works are described.

2 Literature Survey

This chapter offers an overview of the literature that deals with electricity load forecasting. It first presents studies and methods that deal with the broad subject of load forecasting. Afterwards, the literature that addresses forecasting similar to the data in this thesis is reviewed.

Forecasting the electric power demand is essential for all participants in the electric industry. Forecasts are necessary for power grid development as well as for energy distribution planning and enable low-carbon energy technologies to be better integrated into the electric power market. Accurate load forecasts are also extremely important for the power dispatch scheduling of power grid operators and energy suppliers. Creating electricity load forecasts, however, is a complex and highly nonlinear problem that can be influenced by weather, location, social, and economic factors [17]. Due to the various applications of electricity load forecasting, it is common to divide forecasts according to their forecast horizon into four categories [18]. The first category is *very short term load forecast* (VSTLF) which includes forecasts up to an hour. A forecast horizon of one hour to several days is called *short-term load forecast* (STLF). A *mid-term load forecast* (MTLF) is a forecast from a month to one year and, finally, the *long-therm load forecast* (LTLF) includes all forecasts with a forecast horizon of one year and longer.
LTLFs are mostly used for assessing the future degree of capacity utilisation of the power grid, in order to identify possible congestions. Based on the possible congestions, the development of the electric power grid and the power plant infrastructure are planned. Linear regression and auto-regression methods are commonly used for LTLF. The *Autoregressive integrated Moving Average* (ARIMA) algorithm and its extensions, the seasonal ARIMA (SARIMA) and ARIMA with exogenous inputs (ARIMAX), are constantly applied to that task [19, 20]. Other, different regression methods are also regularly investigated in the literature [21]. Machine Learning methods and bio-inspired optimisation techniques like *Genetic Algorithms* [22] and *Artificial Neural Networks* (ANN) [23, 24] are used as well, in order to generate long-term forecasts.
The challenges in creating MTLFs are similar to LTLFs, which means the applied methods are similar as well. Different variations of ARIMA are used in the literature. Hybrid models of ARIMA and machine learning algorithms, e.g. ANNs, are

also sometimes used [20].
A majority of the publications deal with STLF and VSTLF, as there is a large number of possible applications for the forecasts. Short-term predictions are crucial for electricity suppliers, electricity traders, and for transmission and distribution planning. The whole bandwidth of statistical and machine learning methods are applied to the task. Autoregressive (AR) [25, 26], ARIMA [27, 28], ARIMAX, and SARIMA [28] models have already been used in different variations for years, [29], as have simple regression models [30, 31] or exponential smoothing [32, 33]. The most recent progress using these models stems mostly from a larger availability of data, the integration of external variables (e.g. weather data), and combinations of different statistical methods. Linear models work well for predicting a few time steps, however, they reach their limitations when a larger number of time steps must be predicted, due to their inability to represent non-linear dependencies, which become more influential for longer forecasts. Hence, they provide good forecasts when either the forecast horizon or the resolution are small. Therefore, the main areas of application are VSTLFs and forecasts of low resolution in all forecast horizons that consist of only a few time steps.
In addition, a large variety of machine learning models have been employed to create forecasts. Most frequently used are fully-connected NN [34, 35, 36, 37] and SVMs [38, 39, 40]. Tree-based regression models have been successfully applied to STLF as well [41, 42]. In recent years, mostly due to advances in computing capacities, recurrent neural networks (RNNs) [43, 44], long short-term memories (LSTMs) [45], genetic algorithms [46], and hybrid models [47, 48] were the focus of the research community and resulted in further improvements of the forecast quality.

As can be noticed by the great variety of methods covered by recent publications, no superior model or algorithm for time series prediction has emerged yet. This perception is shared by several literature surveys [29]. Moreover, the data used in the majority of the mentioned publications are electricity loads of a large aggregation level, meaning consumption of whole cities, districts, or counties. Methods that work for these datasets usually cannot be easily transferred to data of a smaller aggregation level, for example to the electricity consumption of city districts or single apartment buildings. However, the initial goal of this thesis is to provide electricity demand forecasts for one apartment building in the Margaretenau in Regensburg and, in the long term, for at least a part of or maybe even the whole quarter of Margaretenau.
As could already be seen in section 1.3, the behaviour of time series of different aggregation levels differ widely, in particular in regard to the volatility of the time series. When accumulating the consumption of a large number of households, e.g. of one city, there is hardly any recognisable noise left in the load series, as the

influence of the individual behaviour of one household is negligible. Therefore, statistical models perform well on that kind of load time series, which is also reflected in the results of the existing research. Even naïve forecast models that utilise the historic average load or apply simple regressions perform sufficiently well on those datasets [49].
The amount of publications that deal with load time series of smaller aggregation levels are in the minority. In the publications that work with data of lower aggregation levels, due to the volatile nature of the data, mostly machine learning methods are applied to create the forecasts, as they are generally better suited to find and utilise underlying patterns in the data.
Additionally, it is noteworthy that a lot of the studies only aim to predict one time step or the daily average consumption, in particular for time series of lower aggregation levels. The reason might be that, due to the volatility, it is difficult to produce good forecasts for a longer period. The problem at hand in the project MAGGIE, however, demands a load prediction for one-and-a-half days.

While the research situation for electricity load forecasts with a prediction horizon of one day or more is still sparse, the demand for exactly those predictions is rising. Especially in the energy sector, load forecasts have become increasingly important, due to the integration of renewable energy sources and the extension sector-coupling. Furthermore, short-term predictions for more than a day with an hourly or sub-hourly sampling rate are also of interest in other fields, e.g. in medicine or economics, and hopefully methods developed for the energy sector can also be of use in those fields.
Due to the limited number of publications on this topic and the rising interest in it, investigating short-term energy load forecasting with machine learning methods is necessary to not only enable the energy supply concept of the project MAGGIE, but to advance the scientific discourse in this area.
The focus of the research presented in this thesis has been on generating forecasts with Convolutional Neural Networks. At the time I commenced the exploration of using CNNs for STLF, no publications on this subject were available. Due to their structure and characteristics, it seemed promising to apply CNNs to one-dimensional time series data in order to generate load forecasts.
CNNs have already been very successfully applied to image classification and other tasks that require the handling two-dimensional data [50, 51, 52] before. Only recently have CNNs appeared in publication for time series analysis, for example for the purpose of classification [53, 54]. Since the beginning of my research, a few publications have been released from researchers that apply CNNs for load forecasting as well. In 2017, for the first time, a few research results that predict time series with CNNs were published. Hosein et.al. [55] and Dong et.al. [17], however, predict merely one time step and are therefore not applicable for STLF. Only in one

publication, in the study of Amarasinghe et.al. [56], was a load time series of more than one day forecasted. The authors discovered that, with a CNN structure, better predictions on a building-level than with the widely-used support vector regression can be produced, and that time series forecasting with CNNs is a promising approach. A few other studies that did STLF used a CNN as the first stage of a more complex algorithm. He et.al. [57] use CNNs to pre-process the data as input for a recurrent neural network, and Li et.al. [58] extract features of the time series with a CNN based on which the data is clustered before computing predictions with a fully-connected neural network. In the following years until now, the number of publications concerning electricity load forecasting rose and it has become more widely recognised that it is promising to use CNNs for STLF. However, CNNs have mostly been used as part of bigger, more complex network and algorithm structures. For example, Zahid et.al. [59] utilised CNNs as a first step for feature extracting, while Deng et.al. [60] incorporated them into a complex network that was used to create a 3-day forecast of the electricity load of Ireland. Cai et.al. [61] constructed a gated CNN unit, similar to LSTMs and employed it for load forecasting, and Kim et.al. [62] used CNNs together with LSTMs. All these studies show that, with the application of convolutional networks, improvements in STLF are possible. Kuo et.al. [63] compared different machine learning methods for electricity load forecasting, including a convolutional network. Their results show that proper forecasts for more than 24 hours can be produced by machine learning methods like tree-based regression, ANNs, CNNs, and LSTMs. A comparison has shown that the CNN produces the most accurate forecasts. The dataset the authors used was of a larger aggregation level than used in this thesis. Their results, however, conform with the experience gained from working with the dataset from Ireland and confirm the forecasting capability of convolutional neural networks.

In summary, it can be said that research interest in forecasting electricity loads has only started increasing recently, due to the changes the energy sector is undergoing right now. Therefore, the amount of publications concerning this topic is still rather sparse and the research has naturally been applying well-established methods first. In the last few years, machine learning methods gained more attention and have since been outperforming statistical methods. For the author of this thesis, the next logical step seemed to be applying convolutional neural networks to the task of STLF, which is a novel approach, as a survey of the existing literature suggests that it has not been tried before. Since then, a few publications concerning electricity load forecasting utilising CNNs have been published and support the effectiveness of this method.

3 Methods

In recent years, different, intriguing applications of machine learning models drew the attention of the interested public to this research field. However, machine learning models have already been investigated for decades by researchers who laid the foundation for the development of models which are, for example, able to defeat the world champion in "Go" [64]; can be used in smart-phone apps for image-based plant identification [65]; or operate autonomous driving vehicles [66]. The potential machine learning models offer for data-processing is understood and more and more areas of application are arising.
Historically, image processing was one of the first applications of machine learning models and has since been thoroughly studied. Much of the progress in the research area has been driven by the development of increasingly accurate models for image classification, object detection and so forth. However, machine learning techniques and models are not in any way limited to the application to two-dimensional data. As a result of advancing digitisation and digital transformation in the industrial sector and also in many areas of daily life, the amount of digitally available information is constantly increasing. This opens up opportunities for processing time series with machine learning methods, as much of the information is time-dependent. Typical applications are monitoring of machines, e.g. in an industrial setting in order to recognise wear or increase reliability and efficiency, and the analysis and prediction of energy flows in order grant grid stability and plan machine schedules. In comparison to processing two-dimensional data, the development of machine learning models for time series is still in its infancy.
The research conducted during the course of this thesis is concerned with the possibilities of predicting time series using machine learning models. The time series used in this work are electricity loads of residential households. The predictions of the electricity consumption are intended to facilitate the optimal production and allocation of electrical and thermal energy in a power supply system.

This chapter initially describes the evolution and working principles of neural networks - the best known machine learning model - in 3.1. First, the biological inspiration of neural networks is highlighted in 3.1.1, and then their development from the perceptron in 3.1.2 to the deep neural networks in 3.1.3 is traced. In addition, the

training algorithm of neural networks is outlined in 3.1.4 and convolutional neural networks, which are a specialised type of neural networks, are discussed in 3.1.6.
Afterwards, further, established machine learning methods in the field of time series forecasting are introduced in 3.2. Firstly, 3.2.1 introduces several regression models. Secondly, recurrent neural networks are discussed in 3.2.2, before tree-based models are presented in 3.2.3.
Finally, the application of the previously presented methods on time series forecasting are discussed in 3.3.

3.1 Neural Networks

Artificial neural networks are a powerful machine learning model that has been applied to a multitude of tasks. Neural network models became hugely popular in the last two decades, in particular, with the increase in available computing power and the simultaneous decrease in the cost of the same. The most prevalent tasks for those models are classification and regression, the models are, however, also applied to transcription, machine translation, anomaly detection, de-noising, and many more. [67]

In the following, the single layer perceptron as the first abstraction of a living neuron and the development of that simple model to the deep neural network that is used today is described. Additionally, the back-propagation algorithm that neural networks are trained with is presented. Furthermore, the working principle and structure of a convolutional neural network are discussed, as this network type is predominantly used and analysed in the late part of this work.

3.1.1 Biological background

The artificial Neural Network (NN) used in Machine Learning are models based on the nervous system of animals and humans, in particular on their brains. The complexity of the brain naturally can not be represented by the models, but the attempt to reproduce the processes of the brain was the starting point of the research into artificial neural networks. Therefore, a brief summary of the information processing in the brain follows.

The central nervous system consists of the brain, the different sensory systems, and the motorical system. Most of the information processing takes place in the brain while only some pre-processing happens outside of the brain, e.g. in the retina.
The most integral part of the nervous systems are the neurons. There are roughly 10^{11} neurons in the human brain. A neuron is a cell that accumulates and transmits

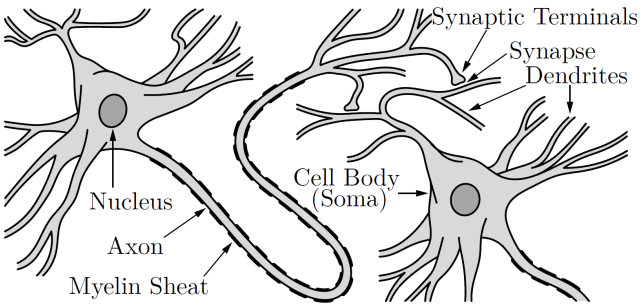


Figure 3.1: Structure of a prototypical biological neuron. Crucial parts of the neuron are annotated. The neurons are electrically interconnected by their synapses and can transport information through electrical signals. [68]

electrical activity. The prototypical neuron and its components are shown in figure 3.1. The axons are the fixed paths through which neurons interact. The axon of one neuron with its terminal buttons leads to the dendrites of other neurons. The gaps between them are only a few nanometres and are called synapses. An average neuron of an adult has between 1.000 and 10.000 synapses. The communication between the neurons works mostly through neurotransmitters. The change of the electrical potential is accumulated in the cell body. When it reaches a threshold, it is again transmitted through neurotransmitters by the terminal buttons to other cells and changes their electrical potential. In addition to these chemical synapses, there are also electrical synapses that conduct the electrical nerve impulses directly through gap junctions. They are mostly found in neural systems that require a fast transmission speed, like reflexes for instance. [68, 69]

3.1.2 Perceptron

In 1943, McCulloch & Pitts [70] were the first to propose a mathematical model for an artificial neuron. They described a neuron as a unit with an arbitrary number of inputs N, a threshold θ, an inhibitory input i, and a binary output. The output y of the unit is calculated by

$$y = \begin{cases} 1 & \text{if } \sum_{k=0}^{N} x_k > \theta \text{ and } i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

and represents a unit step function or Heaviside step function that is shifted by θ. With this, they tried to mimic how a single neuron in the brain works, it is either activated or not. With their model, it is already possible to implement most boolean

functions. However, the parameters of their network had to be designed manually.

3.1.2.1 Single Layer Perceptron

Rosenblatt [71] extended that concept further and allowed for adaptation of the neurons during a learning process based on the idea of the plasticity of neurons suggested by Hebb [72]. The result was the Single Layer Perceptron (SLP), which is the basis of all modern neural networks. The SLP corresponds to a binary model that can separate two linearly separable classes. The output y for an input vector $\mathbf{x}$ of this artificial neuron is

$$y(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T\boldsymbol{x} + b\right) \tag{3.2}$$

with the bias b and the non-linear activation function

$$\sigma(a) = \begin{cases} 0, & a \geq 0 \\ 1, & a < 0 \end{cases} \quad . \tag{3.3}$$

The values of the weight vector $\boldsymbol{w}$ are changed during training according to the so-called *delta rule*:

$$w_i^{\text{new}} = w_i^{\text{old}} + \Delta w \tag{3.4}$$

$$\text{with} \quad \Delta w_j = \eta(t - y)x_j \ . \tag{3.5}$$

η denotes the learning rate, which is typically between 0 and 1, and t is the desired output of the neuron. Rosenblatt proved that the weights will always converge to the correct network weights if a solution exists when using his learning rule for the perceptron. [71, 73, 74]

The SLP, however, is inherently limited due to its incapability to describe non-linearly separable classification problems, for example the XOR function, as was first shown by Minsky and Papert [75]. The transfer of the known learning rule to more complex programme architectures proved to be non-trivial.

3.1.2.2 Multi Layer Perceptron

The development of the *error back-propagation* (see section 3.1.4) and the Multi Layer Perceptron (MLP) in the 1980s ignited the interest of the scientific community in machine learning again.
A MLP consists of several artificial neurons that are organised into layers. It is

able to thereby perform binary classification tasks on datasets which are not linearly separable. It can be shown that all boolean functions, no matter the number of inputs, can be represented by a combination of neurons organised in only two layers. Using logical equivalences, the function is divided into several linearly separable boolean functions that are represented by the neurons of the first layer. The neurons in the second layer recombine those functions to the desired boolean function.

A MLP can be best described as a directed graph. Its nodes are organised into layers. The nodes of one layer are only connected to the nodes in the next layer. Usually, the nodes are fully-connected, which means all nodes in one layer are connected to all nodes in the consecutive layer. The first layer of the MLP is the input layer. It receives the input and therefore has as many nodes as input values. Inputs can be for example measured values or extracted features. A feature is a numeric value or a vector of values which represents a property of the object about which the model is to make a prediction or analysis. Features are often engineered in such way that they contain a maximum of information, so the dimension of the input remains small. The output layer is the last layer of the MLP and returns the calculated result. There can be more than one node in the output layer. All edges of the graph are given a trainable weight.
There can be an arbitrary number of additional nodes organised into several layers between the input and the output layer. Those nodes are called hidden as they are not visible to or accessible by external systems. The hidden nodes and the output nodes consist of the same artificial neuron as the SLP, hence the name Multi Layer Perceptron (MLP). That also means that calculations take place only in those nodes, whereas no processing happens at the input nodes, as their only task is to introduce the input vector to the graph. [68, 76]

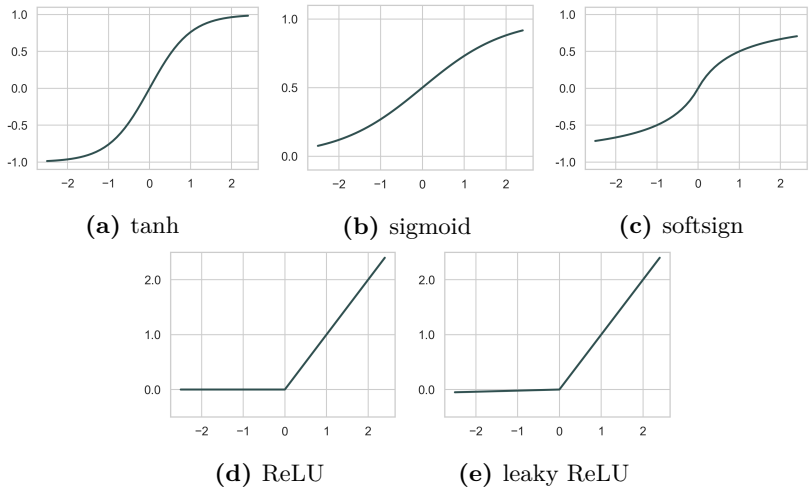
3.1.3 From MLPs to Deep Neural Networks

A fully-connected feed-forward neural network is a more general form of the multilayer perceptron. The neurons are fully-connected, just as they are in a MLP. The significant difference, however, is the activation function of the neurons. In contrast to SLPs and MLPs, which per definition have the Heaviside step function as activation function, the neurons of neural networks can use any continuous, differentiable function. When non-linear functions are used as activation functions, the network can represent non-linear correlations between its input and output more easily. The differentiability is a crucial property of a neuron's activation function, as it allows for the neural network to be trained with the back-propagation algorithm (details in section 3.1.4).

Typical activation functions are:

- tanh $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- sigmoid $f(x) = \frac{1}{1+e^{-x}}$
- softsign $f(x) = \frac{x}{1+|x|}$
- Rectified Linear Unit (ReLU) $f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
- Leaky ReLU $f(x) = \begin{cases} 0.01x & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$

The ReLU and Leaky ReLU are presented here, as they have proven very effective although they are not differentiable at $x = 0$. They can, however, be used in neural networks because software implementations for neural network training usually only use the one-sided derivatives, which exist for both functions. The introduced inaccuracy may be justified by understanding that the training on a digital computer is subjected to numerical errors anyway.



(a) tanh **(b)** sigmoid **(c)** softsign

(d) ReLU **(e)** leaky ReLU

Figure 3.2: (a) to (e) show plots of commonly used activation functions in neural networks.

In addition to the inputs from other nodes, each neuron of a neural network possesses a bias input. The bias input is always present and its value adapts during the training. It can be interpreted as an additional, constantly active input with the value $x_\mathrm{B} = +1$ that is connected to the neuron j of the layer l with a trainable weight $b_j^{(l)}$. Hence, a change of the bias input of a neuron corresponds to a shift of

its activation function. Thus, the bias defines the initial activation of the neuron when no inputs are applied. It is a critical feature, as it is crucial for an effective error back-propagation (more in sec. 3.1.4), and therefore for an effective training. The bias ensures that the activation of a neuron has such a magnitude that the gradient of the activation function is not vanishing and the back-propagation can be carried out effectively. [77, 68, 76]

The computation of the result of a neural network is carried out layer-wise from the input layer $l = 0$ to the output layer $l = L$. At each neuron of the layer $(l + 1)$, the total excitation $a_j^{(l+1)}$ is calculated as a weighted sum of the outputs of the previous layer l. Then, by applying the activation function f_{act}, the output of the neuron j is computed.
For the total excitation, that gives:

$$a_j^{(l+1)} = \sum_{i=1}^{M^{(l)}} w_{ij}^{(l+1)} y_i^{(l)} + b_j^{(l+1)} \tag{3.6}$$

and for the output of the neuron j in layer $(l + 1)$:

$$y_j^{(l+1)} = f_{\text{act}}\left(a_j^{(l+1)}\right) \tag{3.7}$$

$w_{ij}^{(l+1)}$ represents the weight of the connection from neuron i in layer l to neuron j in layer $(l + 1)$; $M^{(l)}$ the number of neurons in layer l; $b_j^{(l)}$ the bias weight of the neuron j in layer l, with $0 \leq j \leq M^{(l)}$, $l = 0, 1, \ldots, L$.

Those equations can also be represented as a matrix multiplication:

$$\boldsymbol{W}^{(l+1)} = \left(w_{ij}^{(l+1)}\right)^{\mathrm{T}} \tag{3.8}$$

$$\boldsymbol{y}^{(l+1)} = f_{\text{act}}\left(\boldsymbol{W}^{(l+1)} \boldsymbol{y}^{(l)}\right) \tag{3.9}$$

The bias weight is integrated into the weight matrix $\boldsymbol{W}^{(l+1)}$.

Thus, a neural network is a non-linear function from a set of input variables to a set of output variables controlled by the matrix $\boldsymbol{W}$ of adjustable parameters.
An exemplary network is shown in figure 3.3. [77, 76]

Based on the presented formalism, the activation of every neuron in a neural network of an arbitrary number of layers can be calculated. This allows for the development of deep neural networks consisting of a multitude of hidden layers.

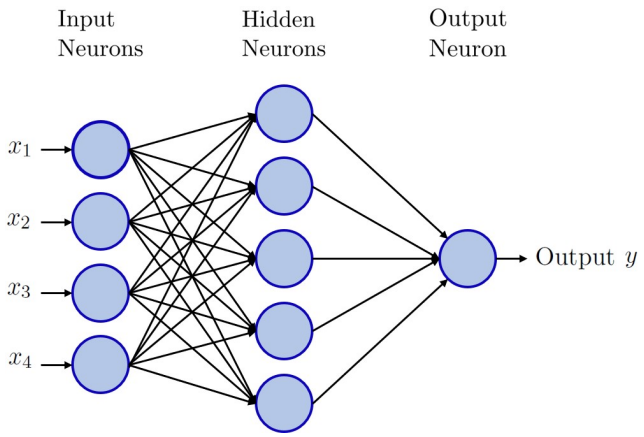


Figure 3.3: A very simple fully-connected neural network with one layer of hidden neurons. The activation functions of the hidden neurons and of the output neuron are differentiable. Every neuron of a layer, that is not part of the input layer, is connected to every neuron of the previous layer, hence fully-connected. Each connection possesses a trainable weight.

3.1.4 Training a Neural Network via Back-Propagation

For the human brain, learning means it is adapting its structure and modus operandi based on experiences had. When a human learns something, a stimulus of some kind is necessary. This can either be an external stimulus, which needs to be identified, or the feedback from an executed action. For the brain to learn, several similar stimuli are necessary, including the errors made in the identification or execution. Based on those errors, the brain adapts and eventually learns to master the assigned task. The tasks can be anything from learning to speak to identifying tastes and mastering a new motor function.
During learning, the connections between the neurons and the rate of firing of the neurons are adapted in such a way that the brain is able to grasp and represent the correlations between its inputs and between the inputs and outputs.

Artificial neural networks are less complex than the brain. In contrast to the brain, in NNs the neurons only have forward connections and the only trainable parameters are the weights w_{ij} between the neurons.
Before the training of a NN starts, the network structure is defined and the network is initialised with random weights. During the training, which corresponds to the learning process of the brain, the weights of the connections are adapted based on the error between the desired and the computed output of the network. That process is repeated iteratively until the network is able to produce the desired output.

Hence, a cost function or loss function that measures the error between the output $\boldsymbol{o}$ of the network and the target $\boldsymbol{t}$ needs to be specified. Based on that error, the weights are adapted. Depending on the learning task, different cost functions are applied. The most common ones are:

- mean squared error (MSE)
- mean absolute error (MAE)
- mean absolute percentage error (MAPE)
- cross entropy or log loss
- Kullback Leibler Divergence (KL Divergence)

The first three loss functions are usually applied to regression problems, the latter two to classification problems. [67]

The adjustments made during the training of the NN are based on the error of the cost function. Typically, the *error back-propagation* algorithm is used to adapt the weights of the NN. It is similar to the *delta rule*, for SLP presented in section 3.1.2.1 . The idea is to present an input vector $\boldsymbol{y}_{\text{in}}$ to the NN, compute the output $\boldsymbol{o} = \boldsymbol{y}^{(L)}$, and calculate the error ϵ with the cost function. When using, for example, the MSE as cost function, that means:

$$\epsilon_{\text{MSE}} = (\boldsymbol{t} - \boldsymbol{o})^2 = \sum_{i=1}^{M^{(L)}} (t_i - o_i)^2 \tag{3.10}$$

$$= \sum_{i=1}^{M^{(L)}} \left(t_i - y_i^{(L)}\right)^2 \tag{3.11}$$

The error of the loss function is then back-propagated starting from the output layer through the entire network towards the input layer. During the back-propagation, the trainable weights are adjusted layer-wise in order to reduce the error. The most common optimisation algorithm for training neural networks is the *gradient descent*. It adjusts the weight as following:

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial \epsilon}{\partial w_{ij}^{(l)}} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)} \tag{3.12}$$

η indicates the step size or *learning rate*, which has to be set manually.

The gradient of the error with respect to the weight $w_{ij}^{(l)}$ in the layer l computes for one training sample $(\boldsymbol{y}_{\text{in}}, \boldsymbol{t})$ as follows. In (3.14) it is assumed that the cost function is still the MSE:

$$\frac{\partial \epsilon}{\partial w_{ij}^{(l)}} = \frac{\partial \epsilon}{\partial y_j^{(l)}} \frac{\partial y_j^{(l)}}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial w_{ij}^{(l)}} \tag{3.13}$$

$$= \frac{\partial}{\partial y_j^{(l)}} \left(\sum_i \left(t_i - y_i^{(L)} \right)^2 \right) \frac{\partial}{\partial a_j^{(l)}} f_{\text{act}} \left(a_j^{(l)} \right) \frac{\partial}{\partial w_{ij}^{(l)}} \left(\sum_i w_{ij}^{(l)} y_i^{(l-1)} - b_j^{(l)} \right) \tag{3.14}$$

To facilitate the representation of the derivative of the cost function, the gradient in the next step is calculated for a weight in the output layer L. Naturally, the calculation is possible for any neuron in the hidden layers by applying the chain rule.

$$\frac{\partial \epsilon}{\partial w_{ij}^{(L)}} = -2 \left(t_i - y_i^{(L)} \right) \frac{\partial f_{\text{act}} \left(a_j^{(L)} \right)}{\partial a_j^{(L)}} y_i^{(L-1)} \tag{3.15}$$

That means the correction term $\Delta w_{ij}^{(l)}$ in (3.12) can generally be written as

$$\Delta w_{ij}^{(l)} = \eta \frac{\partial \epsilon}{\partial y_j^{(l)}} \frac{\partial y_j^{(l)}}{\partial a_j^{(l)}} y_i^{(l-1)} \tag{3.16}$$

and for weights in the output layer when employing the MSE as cost function it simplifies to

$$\Delta w_{ij}^{(L)} = -2\eta \left(t_i - y_i^{(L)} \right) f'_{\text{act}} \left(a_j^{(L)} \right) y_i^{(L-1)} \quad . \tag{3.17}$$

Often, the correction term is presented in the style of the delta rule (see (3.4)) to emphasise their similarity:

$$\Delta w_{ij}^{(l)} = \eta \delta_j^{(l)} y_i^{(l-1)} \tag{3.18}$$

$$\text{with} \quad \delta_j^{(l)} = \frac{\partial \epsilon}{\partial y_j^{(l)}} \frac{\partial y_j^{(l)}}{\partial a_j^{(l)}} \tag{3.19}$$

In (3.13) it can be seen why it is necessary that the cost function and the activation funcation are differentiable. Additionally, it is important that their derivatives do not vanish when a minimum of the cost function has not been reached yet.
With the Heaviside step function as activation function, error back-propagation cannot be successfully applied as its derivative is either zero or does not exist. Therefore, back-propagation is not suitable for SLPs and MLPs. [76]

The training of neural networks is executed iteratively. One needs a whole set of training samples, and those samples are presented one after another to the network. Mainly, the order of the samples is random; they can, however, also have a specific order. After each training sample, a correction term for every weight of the network is computed according to (3.16).
If those terms are added to the weights directly afterwards, the type of training is called *stochastic gradient descent* or *online training*, because each sample gives a noisy estimate of the average gradient over all examples. The next training sample is then applied to the network with the newly adapted weights. Besides the online training, there is the *batch training*, which represents another variation of the gradient descent.
Some authors distinguish between a batch and a mini-batch, with a batch including all training samples and, therefore, having the same size as the training set, whereas a mini-batch is smaller than the training set and contains only a certain number of samples taken from the training set. In this work, mini-batches are simply referred to as batches.
For the batch training, the training set is divided into batches. A batch can include 1 to N samples, with N representing the total number of training samples. For each sample of the batch, the correction terms are calculated and saved. After all samples of a batch have been presented to the exact same network and all correction terms are determined, the average correction term for each weight is calculated and then applied to the weights. The errors of the next batch are then computed using the updated NN. With a batch size of one, the batch training equals the online training. [68, 78]
Both training methods have their advantages and disadvantages. Online training is often much faster. Additionally, the estimate of the gradient using only one sample is noisier than using a whole batch. Noise during the training can be desired in order to explore different minima of the cost function. In contrast to online training, the gradient is less noisy when applying batch training. Thus, the NN converges better as less fluctuations are present. Furthermore, by removing the noise in the data, sometimes a better generalisation can be reached. All these factors have to be taken into account when the size of the batches is determined. [79, 68]

After computing the error and correction term of each training sample, the network is typically not trained very well yet, no matter the training method. For that reason, the network is trained using the training set several times successively. Each training cycle is called an *epoch*. Depending on the learning task, the number of epochs necessary to achieve good results varies. [68] A neural network can barely be trained in a way that it reaches the global minimum of the cost function, as typically numerous stationary points and inequivalent minima of the cost function exist. The training

result depends strongly on the initial network parameters and the training method. It is, however, not necessary to find the global minimum. The goal of training is usually to reach a sufficiently good local minimum. [77]

In addition to the online training and batch training, various variations of the gradient descent exist that try to improve and speed up the training process. They all use the back-propagated error of the weight, but calculate the correction term $w_{ij}^{(l)}$ differently.
One of the biggest challenges for training is to find the most effective learning rate η, which is not too large in order to ensure convergence, but still large enough for the weights to converge efficiently. Another key challenge is to minimise the highly non-convex error function. The best established gradient descent optimisation algorithms besides the stochastic gradient descent and the mini-batch gradient descent are [77, 80]:

- gradient descent using momentum [81]
- Nesterov accelerated gradient descent [82, 83]
- Adagrad [84]
- Adadelta [85]
- RMSprop [86]
- Adam [87]
- Nadam [88]
- AMSGrad [89]

The gradient descent using momentum adds a fraction γ of the previous correction term to the current correction term. Therefore, convergence can be reached faster:

$$\Delta w_{ij}^{(l)}(t+1) = \eta \delta_j^{(l)} y_i^{(l-1)} + \gamma \Delta w_{ij}^{(l)}(t) \tag{3.20}$$

The Nesterov accelerated gradient descent uses momentum as well. The order of calculating the gradient and applying the momentum term, however, differs. Instead of calculating the gradient of the error function with the current weights $w_{ij}^{(l)}$ and then adding it together with the momentum term, the momentum term is first added to the weights and the gradient correction term is calculated afterwards. Poor updates of the correction terms can thereby be avoided. The Nesterov accelerated gradient descent is more robust and converges faster in convex regions of the error function.

The optimisation algorithm based on adaptive moment estimation (ADAM) computes an adaptive learning rate for each parameter (similar to Adagrad). In addition, it uses exponentially decaying averages of the first and second order momentum terms to obtain the correction terms.
The *Nesterov-Accelerated Adam* (NADAM) adds, as the name suggests, the Nesterov-momentum into the Adam algorithm. [80]

All the described methods in this section represent or apply to *supervised learning*. Supervised learning means that for each sample, the desired output is known and that the adoption of the weights takes place on the basis of the error between desired output and computed output of the model.
There are also other categories of training, for example *unsupervised training* [78, 90], which are not discussed in this thesis.

3.1.5 Network Structure Selection

The crucial first step when applying a neural network is to choose a network architecture that can deliver good results for the task on hand. That means the network should allow for low errors while not offering too many trainable weights. The more weights a network contains and the deeper it is, the longer the training takes, and the more challenging it is.

First, the term "layer" should be defined more specifically, as there are different interpretations of it in the literature. In section 3.1.2.2, hidden units (or hidden neurons) were already defined as units that cannot be reached or interacted with from the outside. Moreover, it was said that the units are organised into layers. The terminology for feed-forward neural networks used throughout this thesis from here on does not count the layers of units. Instead, it counts the number of layers of trainable weights. For example, a network that consists of one layer with N input units and one layer with M outputs units is called a one-layer-network as it has one layer of weights in between the units. That layer has N inputs and M outputs. The network shown in figure 3.3 therefore represents a two-layer-network.
As the weights are the only trainable variables of an artificial neural network, the information gained during training is solely stored in them. This definition of a layer is therefore consistent with the information storage within a NN.

When designing the architecture of a neural network, one has to make several crucial choices. The only inevitable parameters of the NN are the number of input neurons and the number of output neurons, everything else is variable.
How deep or shallow the network is going to be, meaning of how many layers it consists, must be decided. Then, one has to decide for each layer how many in- and

outputs it has. That decision determines the number of trainable variables of the network. In addition, the type of activation function used in the neurons has to be chosen. It is even possible to use different activation functions for different neurons or layers.

The properties of NNs for approximating different functions have been extensively studied. Thereby, it was proven that two-layer neural networks can approximate every logical function [91], every non-linear function [92], and every classification [93] to arbitrary accuracy when the number of hidden units chosen is large enough. Neural networks are therefore *universal approximators*.

Even though a shallow two-layer network is theoretically sufficient for most problems, it does not mean it is the most efficient network structure for the given problem. *Deep neural networks*, which consist of several hidden layers, can offer very effective solutions due to their hierarchical structure. They can often achieve the same approximation accuracy as shallow networks with less nodes. It is desirable to have the least amount of artificial neurons possible because the training of a smaller network requires less computational resources.
The challenge with deep neural networks on the other hand is that it is more difficult to train them properly. The reason being that is the error back-propagation. In order to calculate the correction term for a weight in an early layer of a deep neural network, the error has to be back-propagated from the output through several layers. At each neuron on the way, the activation function and a multiplication with a weight vector are applied. That can lead to the *vanishing gradient problem*, which means that practically no information about the error reaches the early layers and those layers, therefore, can not be trained effectively. The reason for that is that the error term of an early layer (see (3.13)) includes the derivative of the activation function several times, once for every consecutive layer, due to the chain rule. Typical activation functions often only exhibit small values for their derivative. Thus, the back-propagated error gets smaller with each layer, which means the NN has to be trained with more samples in order to train the weights of the early layers effectively.
Some methods that ensure effective training and lessen the error reduction during back-propagation of deep neural networks, e.g. batch-normalisation or residual networks have since evolved. [77, 76]
Hence, it is a fine line between having a network structure that is too deep or too shallow, and the advantages and disadvantages of additional layers should be considered carefully.

The activation function of the output neurons is a parameter with only limited choices for the user. It is already determined by the nature of the data and the assumed distribution. For standard regression problems, the activation function

should be the identity. For binary classification problems, the sigmoid function is usually used, and for a multi-class classification problem, the softmax function. [77]

Further consideration has to be given to the number of trainable parameters in a neural network. It is not reasonable to have too many trainable weights in a network. When there are too many weights compared to the amount of training samples, the network potentially memorises the training samples. The goal of machine learning algorithms, in particular neural networks, is to learn generalising and not to purely learn the existing data. By changing the weights of the network in order to minimise the cost function for the training samples, it is assumed thereby that the cost function for unknown samples, that are not used for training, is also reducing. Hence, the NN should be able to properly process unseen samples that originate from the same source. Therefore, the available samples are usually split into a training, a test, and sometimes a validation set.
The test set is used to evaluate the model performance after the training is completed. A large performance discrepancy between training and test set indicates that the sought generalisation property of the model was not accomplished.
The validation set is used to inspect the training progress during training. When the error of the training set is still decreasing with progressing training, however, the error of the validation set is constant or increasing, over-fitting occurs and the training should be terminated. *Over-fitting* describes the process when an estimator, due to its high capacity of trainable variables, memorises its training samples instead of finding underlying patterns and correlations. Typically, a model that was over-fitted exhibits a very low error on its training samples, but a large error on unseen samples from the validation or test set. In order to avoid over-fitting, the number of trainable weights of a model chosen should not be too large. [77, 67, 76, 78]

3.1.6 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), in the way they are used today, were first introduced by LeCun et al. [94] for zip code recognition. Since then, they have been further developed and are now the standard for image and pattern recognition. They have also been successfully applied in other domains, for example, for object tracking, scene labelling, text detection and recognition, and many more.
The structure of the first CNNs was inspired by early findings about the visual system. It is similar to the hierarchical arrangement of cells in the visual cortex ventral pathway. [78, 13, 95]

Modern convolutional networks are almost exclusively applied to image processing, particularly to image and pattern recognition. Hence, their input consists of a two-dimensional array or, in case of a colour image, of three two-dimensional arrays. Each pixel of the image corresponds to one value of the input array. The following introduction into the working principles of CNNs focuses therefore on the case of two-dimensional data, but can also be abstracted for the one-dimensional case

A typical convolutional network is assembled with three different layer types: convolutional layers, sampling layers, and fully-connected layers.
A convolutional layer processes its input using kernels and produces a two-dimensional feature map, sometimes also called an activation map. Each unit of the resulting feature map is only computed from a local patch of the input array. In addition, all units of the same feature map share the same trainable weights. Usually the number of weights, and thus the number of input nodes each neuron is connected to, is much smaller than the number of total inputs. This weight matrix is called kernel, filter, or filter-bank. The spatial extent of those sparse connections is called the receptive field. That means each value in the feature map represents only the subset of input values within its receptive field (see fig. 3.4). Additionally, the neurons have the same structure as the neurons of a fully-connected layer. A convolutional layer can have more than one kernel and hence produce several differing feature maps at the same time. The resulting feature maps are stacked together in the depth dimension and form the output.
The convolutional operation in a neural network can be described with the following function. Let $F : \mathbb{Z}^2 \to \mathbb{R}$ be a discrete function which represents the two-dimensional input data of a convolutional layer. $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$ is a kernel of the size $(2r + 1)$ and $w : \Omega_r \to \mathbb{R}$ the discrete weight matrix. The convolutional operation $*$ can then for a neural network be defined as:

$$(F * w)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})w(\mathbf{t}) \tag{3.21}$$

The reason for that architecture is that in images and similar array data, neighbouring values are often highly correlated. They form patterns and motifs. Those patterns, however, can occur in different locations of the input array. By applying a filter to the whole input image, it can detect the same pattern in every part of the image. That filtering operation is a discrete convolution of the input with the filter, hence the name convolutional layer. The values of the feature map indicate where in the input array the pattern associated with the kernel and similar patterns occur.

A pooling layer reduces the spatial dimensionality of its input using a pooling operation. Examples of pooling operations are calculating the arithmetic mean or the

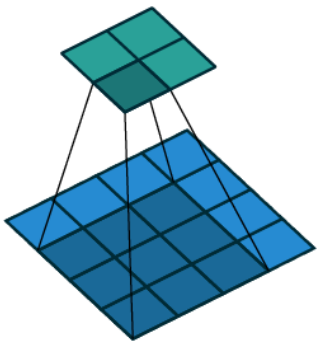


Figure 3.4: The image illustrates the receptive field in a CNN on a small exemplary convolutional layer. The two-dimensional input of the convolutional layer is depicted in blue and the resulting feature map in green. The darker area of the input represents the receptive field of the darker value of the feature map when a 3x3 kernel is applied. The receptive fields increases with every subsequent convolutional layer. [96]

maximum of several values. Each output node of a pooling layer is only connected to a local two-dimensional patch of input nodes, similar to a convolutional layer. In contrast to a convolutional layer, the inputs are not weighted. Depending on the stride size, also called step size or increment, and the padding method, pooling layers can reduce the dimensionality of its input. Another function of pooling layers is to make the representation of the patterns invariant to small translations.
A fully-connected layer of a CNN has the same structure as the fully-connected feed-forward layer discussed in section 3.1.3. [67, 78, 95]

The network architecture of a CNN is structured into stages. The first stage consists of one or more convolutional layer(s) and a pooling layer. Several of those stages are usually stacked in series. The last stage consists of several fully-connected layers. Those layers take the information from the last pooling layer, combine it, and compute the final output. That can, for example, be the affiliation to the known classes in case of a classification problem. The output layer for classifications consists of as many neurons as classes and the activation function is usually a softmax function. The values of each neuron correspond to the probability that the input image belongs to the designated class. [67, 78]

As mentioned before, the concept of CNNs is based on the working principle of the mammalian vision system. The first neuroscientific findings about that were awarded with a Nobel prize in 1981 [97] and had a great influence on early deep learning models.
The first convolutional layer of a CNN extracts only low-level features like edges,

corners, and slopes. Those are then, with each convolutional layer, combined into more complex, higher-level features. Corners, edges, and slopes combine into shapes, shapes form parts, and objects are assembled from parts. Even though a single output neuron of a layer has only a limited receptive field, by using a deep architecture, the number of pixels of the original input image each output neuron indirectly interacts with increases with every layer. Eventually, due to the hierarchical feature extraction, the neurons are connected with all pixels of the input image. The pooling layers in between the convolutional layers serve two purposes. First, they allow for small variances in location and appearance of the elements. That is particularly important as the distances and angles between features in images of the same object can vary. The second reason for the pooling layer is to reduce complexity of the NN through dimension reduction, which, in return, decreases computational load during training and the likelihood of over-fitting. [67, 78]

In a nutshell, three characteristics of a CNN distinguish CNNs from fully-connected feed-forward networks: sparse interactions, parameter sharing, and equivariant representations. It is those characteristics that makes them very suitable for image processing.
In comparison to traditional NNs, where each input of a layer interacts with each output, CNNs have a sparse interaction between the neurons. Furthermore, in convolutional layers, each output node of a feature map shares the same weights, in contrast to fully-connected layers, where the weights are independent from each other. Due to the shared parameters, the convolutional layers are equivariant to translation of features in the input. They are, however, not equivariant to other transformations as rotation or change in scale. Moreover, these three properties reduce the computational and memory requirements of CNNs drastically in comparison to fully-connected NNs, and improve efficiency because there are significantly fewer trainable parameters in CNNs.
The training of CNNs, however, works in the same manner as described for fully-connected NNs in section 3.1.4. [67, 78]

3.2 Established Machine Learning Methods for Forecasting

This section introduces some established machine learning methods for time series forecasting. Most of them are subsequently applied to the forcast task in chapter 4. Firstly, three regression methods are described, followed by recurrent neural networks. Finally, tree-based methods are discussed.

3.2.1 Regression Models

In the following, three linear regression models are introduced. Linear methods have already been used for time-series forecasting and analysis for a long time, as they can mostly be computed with a small computational cost.
The presented methods are amongst the most widely used approaches for forecasting. [98] They are exponential smoothing, ridge regression, and ARIMA.

3.2.1.1 Exponential Smoothing

Forecasting with *Exponential Smoothing* was already proposed in the late 1950's, but is still successfully applied today. A time series forecast with Exponential Smoothing consists of a weighted sum of historic values. The weights decay exponentially, meaning more recent values have larger associated weights. A time-series forecast for the time t can be written as:

$$y_t = \alpha y_{t-1} + \alpha\left(1-\alpha\right) y_{t-2} + \alpha\left(1-\alpha\right)^2 y_{t-3} + \cdots \tag{3.22}$$

with $0 \leq \alpha \leq 1$. The parameter α describes the rate at which the weights decline and is the only trainable parameter of the model. It is typically estimated by minimising the sum of squared errors. In addition, a few extensions of exponential smoothing exist: a seasonal exponential smoothing that can capture known seasonalities in the data or a double exponential smoothing that can represent a trend in the data, for example [98].

3.2.1.2 Ridge Regression

Linear Regression is an approach that models the relationship between a dependent variable y and an explanatory variable x. If more than one explanatory variables

are used, it is called multiple linear regression. The equation of the linear regression is given in its general form by:

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.23}$$

with $\mathbf{y}$ being the dependent variable, X the explanatory or independent variables, $\boldsymbol{\epsilon}$ the error term, and $\boldsymbol{\beta}$ the model coefficients.
The coefficients are usually estimated by applying the ordinary least square (OLS) method. The estimated coefficients $\hat{\boldsymbol{\beta}}$ are the result of:

$$\hat{\boldsymbol{\beta}} = \text{argmin}_\beta \left\| \mathbf{Y} - X\boldsymbol{\beta} \right\|^2 \tag{3.24}$$

Ridge Regression is a variation of the linear regression with an adjusted cost function containing a regularisation term. One problem when fitting a linear regression model with several inputs is multicollinearity. That occurs when the input variables are correlated to each other to a certain extent, which is regularly the case for machine learning applications. By introducing additional bias, large coefficients are penalised, which makes the model more stable. The coefficients are estimated according to

$$\hat{\boldsymbol{\beta}} = \text{argmin}_\beta \left(\left\| \mathbf{Y} - X\boldsymbol{\beta} \right\|^2 + \lambda \left\| \boldsymbol{\beta} \right\|^2 \right) \tag{3.25}$$

with λ being the penalty parameter or, respectively, the Lagrange multiplier for the constraint. [99, 100]

3.2.1.3 ARIMA Models

The acronym ARIMA stands for *Auto-Regressive Integrated Moving Average*. ARIMA models aim to model the time series by describing the autocorrelation in the series. As the name suggests, ARIMA models consist of three different processes. It is a combination of an auto-regressive model, a moving average model, and an integration.

An *Auto-Regressive* (AR) model forecasts the value y_t of a time series as a linear combination of past values of the time series. Hence, it is a regression of the variable against itself. An AR model can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \tag{3.26}$$

with ϵ_t being the white noise at the time t and c a constant. The formula describes an AR(p), which means it includes lagged values up to the order p. In

order to represent a stationary time series, the parameters must fulfil some constraints, for an AR(2) model for example: $-1 < \phi_2 < 1$, $\phi_1 + \phi_2 < 1$, $\phi_2 - \phi_1 < 1$ [98].

A *Moving Average* (MA) model uses past forecast errors for modelling the time series, rather than lagged values. An MA(q) model can be written as:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{3.27}$$

where q represents the order of lagged errors that are included, c is a constant, ϵ_t is white noise, and ϵ_{t-i} with $i < t$ are the past forecast errors of the model at time $t - i$. The ϵ_{t-i}, however, are not observed errors. They are deviations from the original time series, if the value at time $t - i$ was predicted with the existing MA model [98].

ARIMA models should only be applied to stationary time series. A time series $\{y_t\}$ is stationary if, for all s, the distribution of $(y_t, \ldots, y_{t+s})$ does not depend on t, which means the properties of the time series do not change with time. If a time series is not stationary, it is differentiated d times until stationarity is achieved. In the case of a discrete time series, differentiating is not possible. Therefore, the time series is differenced, which means the difference between consecutive values is computed [98].

When combining these three processes, one obtains the ARIMA model:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t \tag{3.28}$$

where y'_t is the d times differenced time series. The equation represents an ARIMA(p, d, q) model, with p being the order of the auto-regressive part, q the order of the moving average part, and d the degree differencing.
The parameters of ARIMA models are typically fitted using the *Maximum Likelihood Estimation*, which estimates the parameters of a probability distribution by maximising a likelihood function. A likelihood function is a function that uses the parameters of the probability distribution as variables. By maximising the likelihood function, the distribution parameters that are most probable for the employed statistical model can be found. The fitted model can then be applied for time series forecasting.

In a nutshell, autoregressive moving average (ARMA) methods compute the forecast as a combination of a linear regression on lagged values of the time series, which is the autoregressive part, and of a moving average of previous deviations from the expectation value. The ARIMA algorithm works with the differentiated time series, as stationariness, which is a requirement for the ARMA algorithm to work, can be achieved by differentiating.

3.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are neural networks that are designed to deal with sequential data. Typical sequential information would be, for example, speech, text, and music. Typically, for this kind of data, the single data points are not independent from each other and have a given order. In contrast to a fully-connected feed-forward NN, which handles all input variables simultaneously, a RNN processes only one variable at a time.

In a RNN, the computed output of a neuron is fed to the neuron(s) in the subsequent layer and additionally back to itself. This feedback loop is called "internal state" and acts as a memory for the neuron. Thus, the output of the neuron at a given time t depends not only on the input at time t, but also on the internal state of the neuron at time t and, thereby, on its previous inputs. Information about the previous inputs and computations is kept in the internal state. Thereby, RNNs are better equipped to capture temporal dependencies within the data than feed-forward NNs, due to their design. [101, 78]
Recurrent networks are trained analogous to feed-forward networks using back-propa-gation. Additionally, *back-propagation through time* is applied, which means the feedback loops are unfolded for the whole sequence of input data and then trained similarly to feed-forward networks (see fig. 3.5). In practice, however, RNNs effectively make use of only the information from a few steps back during the training, due to the vanishing gradient problem. The gradient, which determines the effect of an input variable on the weight adoption, gets smaller with every passing through the neuron, or, respectively, through the activation function. After a few passings, the gradient is so small that it has hardly any influence on the training process any more. [101, 68]

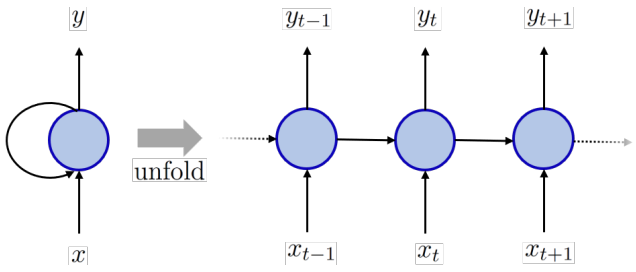


Figure 3.5: Structure of a one-neuron recurrent neural network. On the left side, the feedback loop is visible, which feeds back the computed output of the neuron as input for the next computation step. The right side represents the unfolded network.

Several variations of recurrent networks exist. Some improve the vanishing gradient problem, others have special architectures adapted to the characteristic of data. Well-established modifications are *Gated Recurrent Units* (GRUs) and *Long Short-Term Memories* (LSTMs). The GRU and LSTM are units that replace the classic neuron in the network. Both of them consist of a cell with regulators, called gates, that control the change of the internal state and the flow of information in and out of the cell without necessarily feeding the input through an activation function. Thereby, the vanishing-gradient problem during the back-propagation is diminished and long-term dependencies can be better represented. [102, 103]
Typical areas of application for RNNs and their variations are natural language processing, machine translation, and time series prediction. [101, 102, 78]

3.2.3 Tree-based Models

Another successfully applied method for time series forecasting is the *tree-based regression*. The tree-based regression method is an ensemble learning technique where the model consists of several decision trees. In the case of a regression, they are called regression trees. The regression result is the aggregation of the output of the individual trees.
A decision tree is a set of nodes and edges. Each node has two outgoing edges, called children. The final nodes of the tree are called leaves and store the final results of the trees. At the split nodes, a test function is applied to the input variables. The test function is chosen in such a manner that it splits the samples in two optimal sub-sets according to a split criterion (e.g. the gini impurity, the variance of the sub-set, or the least-squares criterion). [104, 105]

There are two prevalent methods of generating an ensemble of regression trees - bagging and boosting.
When bagging is applied, each prospective regression tree is built based on a sub-set of size n of the training set. These sub-samples are called bootstraps and are chosen randomly, with replacement from the dataset. Additionally, only a pre-defined number m_{try} of randomly selected variables of the bootstraps are taken into account for calculating the best possible split at each node. The aggregation is done by averaging the output of all regression trees. An ensemble of trees trained with bagging is called *Random Forest*. The advantage of the Random Forest regression is that the trees are de-correlated and that due to the ensemble of trees, the method is insensitive to noise less prone to over-fitting [106].
Boosting is an additive method of building the regression trees. The most common modification of boosting is gradient boosting. The single trees of the gradient boosting algorithm are fitted to achieve the best split. The trees, however, are built

successively, based on the prediction error of the already existing ensemble of trees. After training a regression tree, the tree is first scaled with a shrinkage parameter η (similar to the learning rate in neural networks) and then added to the ensemble. Lastly, the training dataset is evaluated on the updated ensemble model and and the pseudo-residuals are calculated, which constitute the training data for the next tree. The pseudo-residuals are the deviations between the prediction of the existing ensemble of regression trees and the desired output. [107, 108]

3.3 Applying Machine Learning Methods to Time Series Forecasting

This section describes how the above mentioned machine learning methods are applied and adapted for time series forecasting in this thesis. First, the application of the variety of regression models to time series forecasting is discussed, followed by the tree-based models. Finally, it is introduced how convolutional neural networks are applied to predicting time series.

Generally, a dataset of measured, historic target values with their correspondent input values is necessary for the training of any machine learning algorithm. The input can consist of measured values or engineered features. The more data available, the better the results of the training usually are. Additionally, it is essential that the dataset represents the whole spectrum of possible target and input values. The dataset is then split into a training and test set, and, if desired, a validation set. Each of the individual sets should represent the whole dataset as well as possible to avoid biases.
As the names suggest, the training set is used to train the machine learning model and the test set to evaluate the performance of the model after the training is complete, using data the model has not processed before. Validation sets are often used to measure the training progress, for cross-validation during the training, or to adjust model parameters during the training. If the test set were to be used for evaluation or regulation during the training process, the evaluation of the training success with the test set would not be independent from the training process. It could depict the training as more successful than it actually was.

3.3.1 Forecasting with Regression Methods

In in 3.2.1, several regression methods have been introduced. While exponential smoothing and ARIMA are regression methods explicitly developed for time series

data, the ridge regression is a more general approach that can be applied to data of any dimensionality. ARIMA and exponential smoothing are both auto-regressive methods, which means they compute a result based on preceding values of the time series. A ridge regression model can, in principle, process any one-dimensional input variable for the computation of the output. However, it makes hardly sense to use the complete time series as input for a ridge regression forecast model, due to redundancy in the relatively high-resolution data. Engineered features, that represent the meaningful information of the historic time series, are instead utilised as input for the ridge regression model. The forecast is then computed as linear combination of those features. The crucial step in developing a regression model is, therefore, the development of the input features.
Similarly to the ridge regression model, a fully-connected neural network can be employed as regression model as well. The key difference between a ridge-regression and a neural network model, however, is that the computed forecast constitutes a non-linear regression of the model inputs when applying a NN model. Both models can either compute a univariate or a multivariate regression.

3.3.2 Forecasting with Tree-Based Models

Tree-based models are an established method for load forecasting and are regularly deployed to that task [41, 42]. Their small computational cost and the explainability of decision trees make that method a favourite of the industry.

Before an electricity load forecast can be created with a tree-based regression model, the input variables have to be defined. Typically, the inputs of regression trees are meaningful features rather than the entire load time series. The main reason for this is that tree-based methods tend to dramatically over-fit, essentially memorising the training data if too many variables are presented to the model. The effect is amplified when the variables contain redundant information, which would be the case if, for example, the load series of the past week was the input. Instead, engineered features, which contain much condensed information, are used as input to the regression tree. The feature selection and feature engineering are the most crucial part in the development of a tree-based model for load time series prediction.
It is common practice to predict each data point individually and independently, because some features are usually specific to a point in time. Predicting the electricity load for the next 36 hours, as necessary in the project MAGGIE, means that 72 individual forecasts must be computed with the model. Due to the forecast horizon of 36 hours, the load from the previous day at the same time of day is not available for all predictions. Therefore, only loads from 48 hours ago and before can be used for the feature engineering.

During the development of the tree regression models, several engineered features have been tested. The features that eventually proved to be effective can be divided into three groups: consumption features, weather features, and date features. Details about feature-engineering and the used features can be found in section 4.2. Depending on the dataset, a selection of the features is used as input of the model.

The algorithm used for training the regression model is *XGBoost* [108], which is short for *extreme gradient boosting*. It is one of the most widely used classification and regression frameworks, primarily due to the successes achieved in machine learning competitions [109, 110, 111]. As the name indicates, the ensemble of regression trees is generated using gradient boosting. Furthermore, the algorithm includes some modifications to other tree-based methods that enhance the speed and effectiveness of the training, for example awareness of sparse data and parallelisation of the tree construction.
In order to obtain the best hyper parameters for the dataset on hand, a small grid search of the most crucial hyper parameters is conducted as the first step of the training. To evaluate the variations in the hyper parameters, a 4-fold cross validation is performed and the best parameters are eventually applied to train the final model on the complete training set. Without the cross-validation during the training, hyper parameters representing a more complex model give the best training result. However, that model would not be able to generalise well. Due to its complex structures with numerous trainable parameters, it would only achieve good results on the training data by essentially memorising the data. Through the cross-validation, over-fitting can be detected and avoided. The squared error is chosen as loss function and the other hyper parameters are set to their default values.

3.3.3 Forecasting with Convolutional Neural Networks

In order to create electricity load forecasts with CNNs, the classic structure of a Convolutional Neural Network with two-dimensional inputs and filter banks (as described in section 3.1.6) is adapted for the one-dimensional case of a time series. When forecasting electricity loads, the input is a one-dimensional time series. Hence, the filter-banks and possible pooling operators are to be one-dimensional as well. The last stage of the network is identical to the case of a two-dimensional input. It consists of fully-connected layers that combine the information from the earlier layers and compute the output.

The convolutional function from sec 3.1.6 changes in the one-dimensional case to

$$(F * w)(n) = \sum_{s+t=n} F(s)w(t) \tag{3.29}$$

with the weight function $w : \Omega_r \to \mathbb{R},\ w(i) = w_i$ that represents the weight vector $\boldsymbol{w}$ of a kernel $\Omega_r = [-r, r] \cap \mathbb{Z}$. $F : \mathbb{Z} \to \mathbb{R},\ F(i) = y_i^{(l)}$ represents the one-dimensional input data from the previous layer of neurons l with $y_i^{(l)}$ being the output of the ith neuron in layer l or, respectively, the ith value of the input series. This gives for the total excitation of the jth value in layer $(l + 1)$

$$a_j^{(l+1)} = \left(F * w^{(l+1)}\right)(j) \tag{3.30}$$

$w^{(l+1)}$ indicates here that the weight vector from layer $(l+1)$ is used.

The basic idea for why time series forecasting should work just as well or better with neural networks that include convolutional operations in comparison to established methods and fully-connected NNs is that meaningful features are extracted by the convolutional layer(s). If it were not for the convolutional layers, the information contained in these features would either have to be extracted and fed to the model manually, or would be omitted. In order to extract features from the time series, the very first layer of the neural network is a one-dimensional convolution layer. The first convolutional layer should operate directly on the time series, because otherwise the sequential character and information of the data is lost. The benefits of a convolutional layer are that dependencies and correlations of adjacent data points are exploited and that the temporal developments of detected patterns can be observed. An arbitrary number of further convolutional layers, or layers that conserve the temporal dependencies of the data points, e.g. pooling layers, can be added subsequently. The concluding part of the model consists of fully-connected layers. The size and depth of the fully-connected part of the neural network model can be chosen at will. The outputs from the convolutional part are treated as features, which means they are handled alike and independently from each other. This results in the loss of the temporal information. The extracted features are combined in the fully-connected layers and basically a non-linear regression that eventually results in the forecast is computed, which is yielded by the output layer of the NN. The whole neural network is still called a convolutional neural network, as the crucial parts of the network are the convolutional layers.
The kernels of a convolutional layer are able to identify patterns in its input series. Those located patterns, represented in feature maps, are then either used as input for further convolutional or pooling layers, or they constitute the input features of the fully-connected layer(s). Based on the position, the intensity, frequency, and

combinations of those patterns, the neural network identifies the electricity consumption behaviour and also changes in the behaviour in time. This information is utilised for the computation of the prediction.
In contrast to fully-connected neural networks and other machine learning methods which rely on engineered features, the neural network extracts the significant information of the time series itself, identifies hidden structures, and creates the forecast utilising this information. Naturally, external features can still be added in order to further improve the forecast quality, for example weather data.

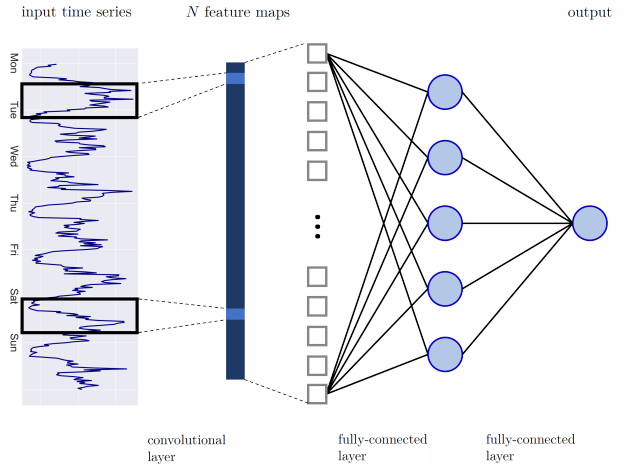


Figure 3.6: The schema demonstrates the functional principle of a one-dimensional CNN. It depicts a simple one-dimensional CNN with one convolutional layer and without pooling layers and a single point output.

As the number of neurons in the output layer of a neural network, and therefore the size of the output vector, can be chosen freely, two different ways of creating a forecast with a CNN are possible.
The forecast for the complete forecast horizon can be created simultaneously by setting the number of output neurons to the length h of the forecast horizon. That means that the prediction is created directly from the set of input variables. The second way to generate a forecast is iteratively. That means the value for only one time step is predicted by the network. The predicted point is then appended to the input time series and the first data point is cut off in order to maintain the shape of the input time series. This procedure is repeated until h data points have been predicted.

4 Results and Discussion

This chapter discusses the forecast results of the different machine learning methods and presents the development of the novel forecasting model based on convolutional neural networks in detail.
First, the used dataset is analysed and the division into the training, validation, and test set is discussed. In the subsequent section, the features that are used for the different models are introduced and discussed. Thereafter, three linear forecast models are trained in order to create a baseline for the machine learning models. Then, fully-connected networks, recurrent networks and tree-based networks are applied to the task of electricity load forecasting. The development of the tree-based model is presented in detail. For the large remainder of this chapter, the development of the CNN model is outlined. First, the model is developed from a simple structure to a more complex structure using only the available dataset. Afterwards, several external features are incorporated into the model and their influence is investigated before the two models are combined. Finally, the performance of the different forecast models are compared and a conclusion is drawn.

In order to evaluate and compare different machine learning models, an error measure must be chosen. For achieving the best forecast result, that error measure is also applied as cost function during the training of the different models. The error measure used throughout this thesis is the mean squared error (MSE):

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (t_i - o_i)^2 \tag{4.1}$$

For all N time steps within the forecast horizon, t_i describes the ith target value and o_i the corresponding output value of the model.
When applying the MSE, the deviation from the target value is squared, which results in a strong emphasis being on large errors and outliers. They are undesired because it is of particular importance to avoid large errors between the predicted and actual electricity consumption. Small deviations can be compensated by the energy system, by utilising the heat buffer and adapting the machine schedules on short notice. Large deviations from the forecast, however, potentially cannot be compensated, e.g. when the heat buffer state does not allow for a large heat transfer. Moreover, it is possible that a large heat transfer from or into the heat

buffer compromises the further schedule of the energy system. Therefore, large forecast errors should be avoided in the electricity forecasts, for which reason the MSE is used as error measurement.

Additionally, the *mean absolute percentage error* (MAPE) is sometimes used as well:

$$\text{MAPE} = \sum_{i=0}^{N} \left| \frac{o_i - t_i}{t_i} \right| \times 100 \tag{4.2}$$

It describes the deviation between the forecast o_i and the actual load t_i as percentage of the target. Therefore, the MAPE allows to better compare the errors of target loads that exhibit a different magnitude.

The performance of all models is measured by computing a 72 point forecast with the data available at the beginning of the forecast period. This implies that for the last predicted data point, the earliest available consumption data is 36 hours old. When, in the following, it is referred to the prediction error of a dataset, this means that as many consecutive 36-hour forecasts have been computed as the dataset and the model allow for, and the error of these single forecasts are averaged.

4.1 Partitioning of the Datasets

This section discusses how the three datasets are split into a training, validation, and test set and discusses why the division is done in this way. It also discusses the different characteristics of the three sets and the disadvantages caused by the small dataset. It is furthermore outlined from which data source the temperature data originates.

Before a machine learning model is trained, the dataset is usually split into a training, a validation, and a test set. The model is then trained on the training data. During the training process, the success of the training is measured with the validation set. By doing this, over-fitting is recognised and the training can be terminated. The model performance on the validation set is also used to evaluate changes in hyper-parameters. The test set is eventually used to evaluate the success of the development and the training of the model. As the model has neither processed the training data before nor have the hyper-parameters of it been adjusted based on the data in the test set, the evaluation of the final model with the test set offers an unbiased result. When the performance is similar to the validation set, the machine learning model has successfully learned to generalise.

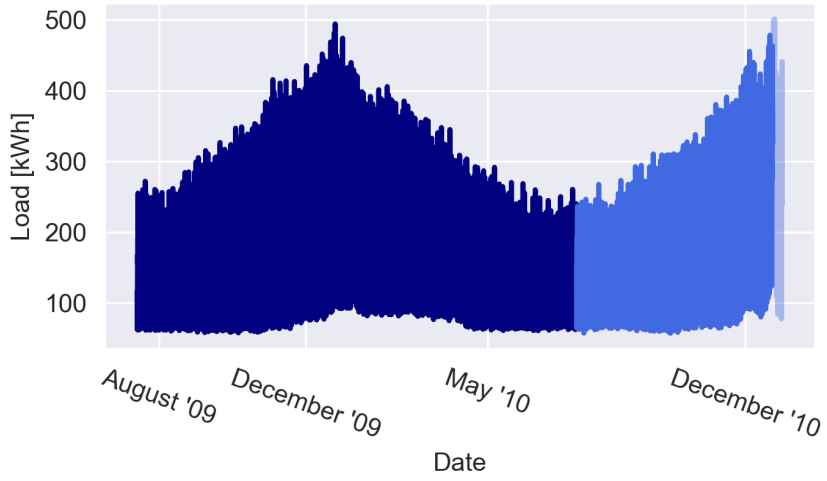
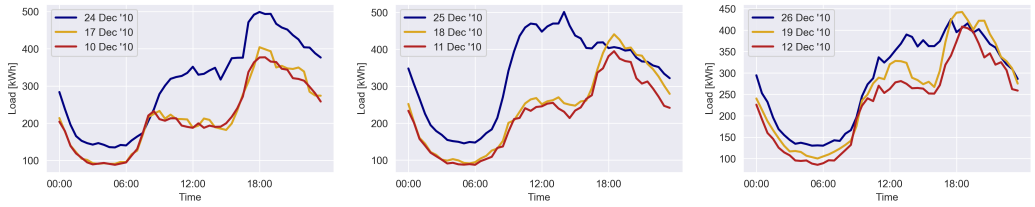


Figure 4.1: Plot of the complete IRE350 dataset. It is obvious that the electricity consumption varies strongly with the seasons. The largest load is reached at Christmas 2009 and Christmas 2010. The colours indicate how the dataset is split. The training set is depicted in darker blue, the test set in a brighter blue, and the excluded Christmas Days are shown in a transparent blue.

In order for the subsets to fulfil their respective tasks, each dataset must be representative of the complete dataset. This implies that the range of input values and the range of target values of the samples in the subset should be comparable to the complete dataset. Furthermore, the distribution of samples in the sets should be identical with the expected distribution of samples coming from the data source. Successfully trained machine learning models are able to recognise and represent underlying processes and correlations of the data source, which is not possible if the distribution of the training set does not coincide with that of the data source.

The dataset from the CER Smart Meter trial includes electricity loads from mid-July 2009 until the end of the year 2010. In order to have an evenly distributed training set in which no season or part of the year is under- or over-represented, the training dataset consists of the load of exactly one whole year: from 14 June 2009, 00:30, until 14 June 2010, 00:00 . The remaining data consists only of samples from half a year. It is impossible to divide half a year of electricity loads into two unbiased sets for the validation and testing. Therefore, creating a validation set is refrained from. Instead, the remainder of the data from 14 June 2010, 00:30 until the end of the year is consolidated to the test set. In case validation during the training

is necessary, the test set is used. During the development of the machine learning models, however, it became apparent that none of the models is able to properly predict the consumption for the Christmas days (24, 25, and 26 December). The electricity consumption behaviour during Christmas differs significantly from the rest of the year, as is illustrated in figure 4.2 . Therefore, the three Christmas days of 2010 are not considered in the remainder of this thesis when evaluating the model performance. In general, it should be possible for a machine learning model to identify and correctly predict the consumption for Christmas, especially since a feature to better identify the Christmas days can be created and passed on to the model. The limiting factor for properly processing the Christmas days is the limited amount of data. As the full dataset consists of less than two years of data, the largest balanced training set consists of one year of data, as explained above. Hence, the training set contains only one Christmas, which is insufficient for an effective training of the model concerning recognising and predicting the Christmas days. In that case, it is more promising to use a naïve model to predict the Christmas days. Repeating the load from last year's Christmas, for example, gives quite good results and outperforms all tested machine learning models. It is symptomatic of machine learning models trained on small datasets that they do not perform well on events of the dataset that only occur once or very scarcely in training set, as too few differing training samples of the same event are available. This does not enable the model to learn to generalise.



(a) The load of 24 December 2010 and of two additional days for comparison.

(b) The load of 25 December 2010 and of two additional days for comparison.

(c) The load of 26 December 2010 and of two additional days for comparison.

Figure 4.2: The figures show the consumption during Christmas 2010 of the IRE350 dataset. For comparison, the consumption of the same weekdays of the two prior weeks is plotted as well. It is clearly recognisable that the electricity consumption behaviour during Christmas differs from the rest of the year, which seems logical because it is a widely celebrated holiday of the year. The load curves of the Christmas days exhibit a different shape and a higher daily mean.

The goal of this thesis is to provide a machine learning model that can be applied for electricity load consumption. As briefly mentioned in section 1.2, one of the constraints when creating forecasts for the energy market is that the feed-in into and the consumption from the power grid must be announced to the power grid operator on the day before, in the early afternoon. In order to calculate the machine schedules for the different components of the energy system and, thereby, the feed-in and consumption, the state of the energy system's buffer storage must be known as exactly as possible. Its state, in turn, depends on the previous consumption and machine schedules. Hence, the load forecast needs not only be computed for 24 hours, but for around 36 hours. Therefore, the forecast horizon h used throughout this thesis is $h = 36\,\mathrm{h}$. The sampling rate of most data in the energy market is 15 minutes. The data from the CER Smart Meter Trial, however, is only available at a 30-minute sampling rate. For the data to remain coherent, the forecast is computed with a 30-minute sampling rate, too, which means a forecast is comprised of 72 data points.

All proposed models are tested on the three datasets of a varying aggregation level. The sets consist of 15, 40, and 350 randomly selected households and, thereby, represents different areas of application: a small apartment building, a large apartment building, and a whole city district. It is not certain, though, that the selected households are from the same part of the country. They can therefore experience different external influences and consequently exhibit different behaviour. However, even the inhabitants of apartments in the same building can have different social and economic backgrounds, which leads to varying behaviour. The unknown location of the single households should, hence, not constitute an issue, as different behaviour of different tenants is to be expected.
The machine learning model that is developed should be applicable to the datasets of all three aggregation levels, because one of the MAGGIE project goals is to create an energy system that is scalable. When analysing the three datasets, one recognises that the approximate daily consumption pattern remains similar, but the volatility of the data increases drastically with a decreasing aggregation level. This has already been observed in figure 1.3. The larger the aggregation level, the smaller the influence individual households and devices, and the smaller the influence of random deviations by the residents from their regular behaviour. Electricity loads of large aggregation levels are more well-behaved and can therefore be more easily analysed and presumably also predicted. Therefore, three individual models are trained and optimised for each dataset. At the end of this chapter, whether the models perform when they compute forecasts of diverging aggregation levels with respect to the dataset they are trained on is analysed.

The electricity loads of the three datasets show a clear dependency on the seasons (see fig. 4.1). The daily peak consumption and the daily load variance change with the season. In order to analyse and pre-process the data, historical temperature data from Met Éireann, Ireland's National Meteorological Service is used [112]. Due to data privacy, the locations of the single households from which the used aggregated datasets originates are unknown. According the *Irish Social Science Data Archive* [16], which provides the CER Smart Meter Trial data, the data includes homes from all over Ireland. The temperature data used was recorded by the weather station "Phoenix Park" in Dublin. This station was chosen because 1.9 million people live in the greater Dublin area, which is a large share of the total Irish population of roughly five million people. The probability that a good share of the households that account for the different aggregated datasets is from the Dublin area, and therefore experience similar weather to the weather station, is large. The assumption that the temperature data is approximately correct is supported by the fact that the Republic of Ireland is about $70\,000\,\mathrm{km}^2$ roughly the size of Bavaria. The temperature should, in most occasions, not differ too much within the country. However, when analysing the data and the results, on should keep in mind that the temperature set used is not completely accurate for the aggregated datasets.

4.2 Feature Engineering

Many machine learning models use features as input variables instead of whole time series in the case of one-dimensional data, or images in the case of two-dimensional data. Engineering these features is an integral part of developing a successful machine learning model. In this section, all the engineered features that are used in one of the models discussed hereafter are introduced and the reasoning in utilising these features is explained.

The features can roughly be divided into groups based on: past electricity consumption, environmental variables, and the date or time.
The *consumption features* contain information that is extracted from recent historic loads. That means they represent the consumption of the last few days. The first feature of this group is the load from exactly 48 hours before the time for which a forecast is to be created. Additionally, past consumptions from one week, two weeks, and three weeks ago are also used as features. The motivation for using these features is that the load at a specific time of the day is most likely similar to the load of the same time on a previous day, especially on the same day of the week. Without any irregularities and changing external influences, the daily consumption patterns repeat every seven days. The loads from 48 hours ago are utilised for computing the forecast, because load values that are closer to the predicted load values are not

necessarily available for all time steps of the forecast. The forecast horizon in the project MAGGIE is 36 hours which implies that the time difference between the most recent measured load value and the last expected data point of the forecast is 36 hours. Hence, the loads from 24 hours ago are not available for the complete forecast horizon. Further consumption features are statistical key-performance indicators from the last completely available day. They include the minimum and maximum, the mean, the standard deviation, and the 0.25- and 0.75-percentiles.
The *weather features* contain information that is related to the weather. The ambient temperature has a presumably large influence on the electricity consumption. In addition to the predicted temperature for the time and date to be predicted, the temperatures three hours prior and thereafter are also provided, in order to identify temperature trends. The reasoning for including the temperature is that a correlation between the average daily consumption and the average daily temperature is noticeable (see fig 4.3). That correlation is plausible because the colder the ambient temperature, the more probable it is that the residents spend time in their apartments. Hence, they consume more energy by using electric devices in their households.

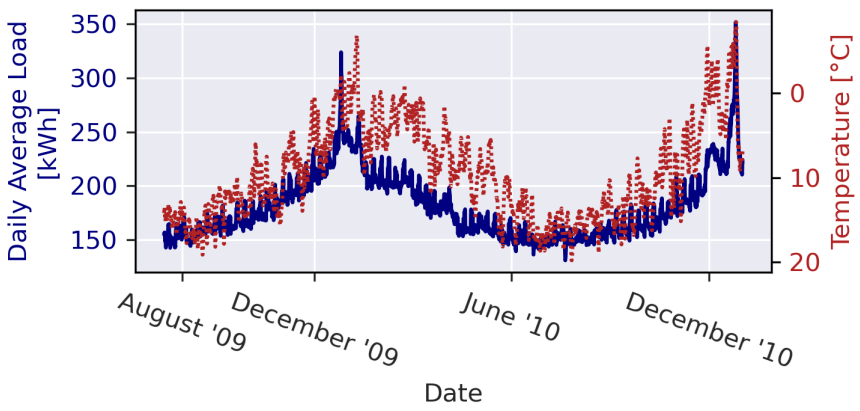


Figure 4.3: The graph shows the daily average electricity consumption in blue and the daily average temperature in red of the complete dataset. The y-axis of the temperature is inverted. The temperature curve and the load curve display a similar development, which indicates a negative correlation between the two. Additionally, regular fluctuations in the load curve can be seen that exhibit a seven day frequency, which stems from the difference in consumption between weekdays and weekends.

Then, there are the *date features*, that offer information which is connected to the date for which the forecast is computed. There are re-occurring cycles in the electricity consumption and the date features enable the models to place the date of the forecast within these cycles. The existing cycles are the 24-hour daily cycle, the seven-day weekly cycle and the 365-day yearly cycle. Information about the existence of these cycles and the location of the value to be predicted within the cycles is used for creating features. They include the hour of the day, the day of the week, the day of the year, and the month of the year. It has already been mentioned that there are differences between the average daily consumption pattern of workdays and weekends. Therefore, an additional feature is created that states whether the predicted value is on a weekend or a workday. Thus, this feature offers temporal context for the forecast computation. A similar consumption behaviour to on weekends is observed on bank holidays. Therefore, a feature that states whether the prediction is for a bank holiday is created as well. Furthermore, the course of the sun on the corresponding day is computed and can be used to determine the time of sunset and sunrise and the amount of daily sunshine hours. They are created, as the necessity of electric lighting when the sun has set clearly effects the electricity consumption. The course of the sun is calculated for the city of Dublin and is used for the complete dataset, as no exact location of the consumers is available.

In addition to the inclusion of the features into machine learning models, the formatting of the features matters for the success of the model as well. Therefore, in the following, a short introduction to how features can be formatted is given.
Designed features usually have the form of a number or a vector. Categorical information, which means only a limited number of defined states exist, is usually one-hot-encoded. A one-hot-encoded feature is presented to the machine learning model as a vector whose length equals the amount of possible feature states. Each entry is assigned to exactly one state, can only assume values of 0 and 1, and indicates which state is active. Categorical information can also be integer-encoded. That means an integer value is assigned to each state of the feature.
It is particularly important for neural networks that the features are one-hot encoded. There are two reasons why integer-encoding does not work well with neural networks. On one hand, integer-encoding introduces an order to the states which does not necessarily reflect the reality. Machine learning models can arrive at incorrect assumptions due to that ordinality. On the other hand, integer-encoded data cannot be processed straight-forward by a neural network. Assume there is a categorical feature f, which can take three values: If those features were integer-encoded, it could, for example, assume the values $f \in \{0, 1, 2\}$. And if it were beneficial for a neuron to be activated when the variable has the value $f = 1$, it would be impossible for that neuron to achieve that, because the activation function

of the neurons in neural networks are by extension threshold functions (see section 3.1.3). Having at least one additional neuron in the same layer and two neurons in a subsequent layer would be necessary to utilise the feature f. When the feature is one-hot encoded, which means, in that case, that it constitutes three input values which can only assume values of 0 and 1, the feature can be directly used by any neuron. The additional trainable weights generally have no negative impact on the model performance as always only one of the one-hot inputs is non-zero at the same time. However, it is important to ensure that each input is activated sufficiently often so that all weights can be trained properly.

Only features that are actually included in one of the presented models are listed here. In the course of the model developments, several other features were tested as well, but in the end were not advantageous or were surpassed by the mentioned features.

4.3 Baseline models

This section discusses the forecast performance of three linear baseline models: a naïve model, the exponential smoothing model, and the linear regression model. All models are trained on all three datasets. The corresponding forecast errors of the models are depicted after discussing the details of the configuration of the model and the training process.

A forecast model does not necessarily have to be extremely complex to generate good forecasts. Simpler models have the advantage of being easily understood, which leads to fewer misinterpretations and errors in the application of the models. Moreover, the computational complexity of training these models is often much lower than with sophisticated machine learning models. Depending on the utilisation and the purpose of the model, it can be sufficient to rely on simple models.

Applying an ARIMA model to the task of predicting the 36 h electricity load is refrained from, as no outstanding results are to be expected. Non-linear models, like tree-based forecaster or neural networks, have by now superseded ARIMA models when generating forecasts of high-resolution. ARIMA models struggle, on the one hand, with recognising periodicities, and it can be expected that a fair amount of effort must be invested into tuning these models, in order to enable them to properly represent the daily average consumption with its two maxima in the morning and evening. On the other hand, the forecast of an ARIMA model converges to the mean load of the input when several time steps are predicted iteratively. This is

due to the auto-regressive nature of the model. As better results can be expected with different methods and developing a proper model is very time-consuming, no ARIMA models have been trained for the comparison of machine learning methods for time series forecasting.

In the following, three simple forecast models for time series forecasting are introduced and eventually applied to the forecasting task at hand. The dataset is divided into a training and a test set, according to section 4.1 above. The evaluation of the forecast is based exclusively on the test data.

4.3.1 Naïve Forecast

The first model that is discussed is the naïve forecast. The model simply assumes that the electricity consumption of the desired day equals the consumption of the previous day:

$$P(t) = P(t - \Delta t) \tag{4.3}$$

with $\Delta t = 24\,\mathrm{h}$.
This is a reasonable assumption if only a rough prediction is required. The daily consumption patterns resemble each other, in particular, when the aggregation level of cumulated time series is sufficiently large. After all, humans are creatures of habit and significant changes in behaviour are necessary to change the consumption, e.g. due to extreme weather or vacation.
Alternatively, the consumption from the prior week can be used as well by applying $\Delta t = 168\,\mathrm{h}$. That takes, on the one hand, deviations between the weekdays into account, but, on the other hand, constitutes a sizeable time difference between the model input and the forecast.

Table 4.1: The table shows the forecast errors achieved with the two naïve models with different Δt for all three datasets. The forecasts are relatively good when taking the simplicity of the model into account.

	MSE		
Δt	IRE15	IRE40	IRE350
$24\,h$	8.73	24.4	565
$168\,h$	8.64	22.6	469

The resulting forecast errors show that the rough predictions made by the naïve model can already be used as an indicator for the consumption of the next day or week. These models can be useful as a fall-back when other models fail. The results also indicate that the consumption behaviour of the different days of the

week varies. This is the only explanation of why the model that takes the more recent historic consumption values as predictions performs worse than the model with $\Delta t = 168\,h$.

4.3.2 Exponential Smoothing

Another model that is regularly used to predict trends in time series is *Exponential Smoothing*. It was already introduced in 3.2.1.1. The forecast, computed by an exponential smoothing model, consists of the weighted sum of historic values with exponentially decaying weights. When forecasting the electricity consumption, which exhibits a periodic behaviour, the historic consumption values the forecast is based on should reflect that periodicity. Therefore, only values that are 24 hours or multiples of 24 hours in the past are utilised as historic values. Thus, the exponential smoothing model is basically a more sophisticated version of the naïve model because it also takes the consumption of more than one previous day into account. A variety of smoothing factors α have been evaluated. The forecast results correspond to the α which generated the best forecasts.

Table 4.2: The table lists the forecast errors that are achieved by an exponential smoothing model. The MSE values represent the most successful model utilising the listed α for the respective dataset.

	IRE15	IRE40	IRE350
MSE	16.4	49.1	2523
α	0.3	0.8	0.9

The resulting forecast errors of the exponential smoothing model suggest that this method is not suitable for higher-resolution time series forecasting, as even the naïve model performs better. That explains why exponential smoothing is mainly used for calculating predictions of directly adjacent values [32, 33].

4.3.3 Ridge Regression

The *Ridge Regression* model is, in comparison to the previous two models, more complex. The forecast is computed similarly to a multiple linear regression with engineered features as input variables or, respectively, explanatory variables. The only difference is the added regularisation term (details in sec. 3.2.1.2). Ridge regression is preferred over linear regression without regularisation and *Lasso Regression*, because the preliminary tests using ridge regression have been the most promising.

The regression model is used in a way that it computes only one output value for each set of explanatory input variables. Hence, in order to create a 36-hour prediction, the model is executed 72 times with differing inputs. The calculation of the used features is specific to the predicted point in time.
Two different versions of the ridge regression model are trained. They only differ in which features are used as input. One model contains historic consumption values and the predicted ambient temperature. This model is comparable to the exponential smoothing model - the main difference being that every historic input value has its own designated, independent regression coefficient. The historic values used as input features are the loads from 48 hours, one week, two weeks, three weeks, and four weeks ago. The second version of the regression model employs more sophisticated features as inputs. In addition to the historic consumption values and the ambient temperature, the input also contains features with the information about the day of the week, the hour of the day, and whether the predicted value is on a weekend. Additionally, more ambient temperature values are included in order to add context and the statistical features of the historic consumption values are included. The statistical features and the format of the features are discussed in detail in 4.2 .

Table 4.3: The table shows the forecast errors achieved with the two different ridge regression models. They only differ in the utilised features.

	MSE		
input features	IRE15	IRE40	IRE350
historic load	5.93	15.9	366
historic load & statistical features	5.86	16.1	372

The results show that the Ridge Regression model is able to compute considerably better forecasts than the naïve approach. The forecast errors also show that the addition of the statistical features does not necessarily improve the forecast quality. Presumably, the information that is passed with the features can only be processed poorly, since the scales and units of the included features differ from that of the output.
Forecasting the energy load with a ridge regression model is, nevertheless, a valuable approach, particularly because the computational load of training the model and generating forecasts is small.

4.4 Fully-Connected Neural Network Models

In this section, how forecasts can be computed using fully-connected neural networks is discussed and the forecast performances of the trained models are evaluated.

It can not generally be assumed that a forecast can be represented as a linear combination of the engineered features. This limits the use of models which utilise linear regression for computing their forecast. Therefore, a neural network model is applied in the following to perform a non-linear regression, in order to compute a load prediction.
The model consists of a fully-connected neural network. The inputs of the model are the same engineered features as used above for the Ridge Regression Model. This means that the features contain some historic consumption values, statistical measures of the historic consumption, the predicted ambient temperature, and information about the point in time for which the prediction is calculated. The used network is the result of testing a variety of network structures with different numbers of layers and varying layer sizes. The architecture that performed the best is applied and consists of four fully-connected layers. The layers have 50, 15, 8, and 1 output neurons respectively. The ReLU function is applied as an activation function in all the neurons except the output neuron which utilises the identity as activation. Figure 4.4 depicts the structure of the neural network. The 36-hour prediction is generated the same way as for the regression model, which means point-wise. That is due to the fact that the features change with the point in time for which the forecast is computed.

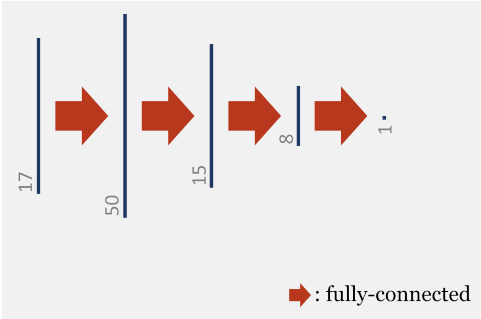


Figure 4.4: This schematic depicts the network structure of the NN regression model with four layers. The vertical lines represent the number of input or ,respectively, output neurons of each layer and the arrows represent the weights. In this case all layers are fully-connected.

Similar to the regression model, several NN models are trained. The models differ only in the number and type of input features; the network structure is identical. The first model receives only the past consumption features and the ambient temperature as input. The second model obtains the temperature values corresponding to the past consumption features and information about whether the forecasted value is for a weekend in addition. The third model, however, can make use of all the features mentioned in section 4.2 .

Table 4.4: The table presents the forecast errors achieved with the fully-connected neural network model. For each configuration, ten models have been trained on each dataset. The listed errors represent the best model of each configuration.

	MSE		
input features	IRE15	IRE40	IRE350
historic loads	5.68	31.4	1528
historic loads & weekend	6.75	18.9	377
historic loads & weekend & statistical features	6.62	17.9	363

Using a fully-connected neural network in order to calculate a non-linear regression of the features generates good forecast results. The more information the NN model receives, generally the better the forecast. However, the improvements in comparison to the ridge regression model in section 4.3.3 are only minor. The non-linear model seems to be more capable of processing the information of the additional features, because in contrast to the ridge regression model, the forecast improves with the inclusion of more features. The NN model is, on the other hand, not able to make as much use of the historic loads as the ridge regression model. The computational load of training the NN model is considerably larger. When deciding between a linear model and a fully-connected NN model for load forecasting, the advantages and disadvantages of each model must be considered.

The results confirm that the most widely used machine learning model for time series forecasting presumably does not exploit the full potential that neural networks can offer. This is one of the main reasons why a novel CNN model for time series forecasting is being developed in the course of this work.
It has additionally also been tested whether a fully-connected neural network model is able to compute an accurate forecast from the complete input series instead from computing it from the engineered features. The forecast performance of the model, however, was unsatisfactory which is why the approach has not been pursued further.

4.5 Recurrent Neural Network Models

This section shortly discusses how time series forecasting applying recurrent neural networks works and why they are not a great tool for generating forecasts with a large forecast horizon.

The previously discussed models have shown that the electricity load can be reasonably well modelled using the bygone loads, because the load patterns of the days resemble each other. It can also be assumed that, from the recent consumption, indications for the next few hours can be derived. It seems therefore natural to compute electricity predictions with recurrent neural networks (RNNs), as this type of neural network is able to store and utilise past states of its units. The input data is sequentially fed into and processed by the RNN. For each step, the state of the recurrent units is updated and an output is generated, if desired. By feeding measured loads into the RNN, the state of the units is updated until the input series ends and a forecast is created. Then, predictions for any forecast horizon can be computed iteratively.
The most recent advances in research applying RNNs has been achieved by LSTM networks, which are neural networks with more complex recurrent units. LSTM stands for *Long Short-Term Memory*. These units are more complex than a regular recurrent neuron, as they possess an internal state, which is not effected by any activation function, and several gates that adjust the internal state based on the input or extract information from the state, using it for calculating the output of the unit [113].

In the course of developing a forecast model for electricity data, the first neural network models that were tested have been RNN models. Several networks consisting of neurons with recurrent connections have been analysed for their forecasting capabilities. After those models did not produce accurate forecasts, different LSTM networks were examined. Even though the networks containing LSTM units generated better forecasts, they were still less accurate than the forecast of the linear models. It was determined that the recurrent networks work well for short forecast horizons, but fall short when longer forecast horizons are demanded, like the desired 36 h forecast. It seems that the network units are not able to utilise input values that have been processed ten or more steps previously. Natural language processing (NLP) is the main application field of RNNs, in particular of LSTM networks. In NLP, the inputs correspond to words and only the last few words are relevant for computing the output. As the meaning of a sentence primarily depends on the previous words of this sentence, or maybe the previous sentence, it is not necessary to access information that has been presented to the network many steps previously. In the studies where LSTMs are applied to time series forecasting, the

forecast horizon is usually very small. It is furthermore computationally extremely expensive to train recurrent networks, in particular LSTM networks. This makes the training of RNNs extremely time-consuming and, thus, makes it difficult to test several different network configurations and structures on their potential for time series forecasting.

The uncertain prospect of whether recurrent networks are even suited to compute forecasts with a large forecast horizon combined with the time-consuming development of the models led to the decision not to further investigate the potential of LSTM networks for electricity load time series forecasting. For that reason, no fully developed forecast model with LSTM units exists for which the forecast results could be included in good conscience in the comparison of the different models regarding their forecast capabilities.

4.6 Tree-Based Models

In the following, the performance of tree-based models for time series forecasting is evaluated. First, for what reason which features are utilised for generating the forecasts is discussed. Thereafter, some hyper-parameters of the model are discussed and an adjusted training algorithm utilising cross-validation is introduced. Finally, forecast results of the tree-based models and the influence of the model configuration choices are analysed.

A well established method for forecasting time series are tree-based methods. Those methods are particularly popular within the industry, as the computational load for training and applying tree-based methods is relatively small. In this work, the algorithm used to train an ensemble of regression trees is XGBoost [108]. It is one of the most effective and successful algorithms for tree-based models.
An advantage of an XGBoost model is that it can handle any type of input - categorical, boolean, string, or float. The input vector can even consist of different types, as long as the variable has the same type for all samples. Furthermore, no normalisation of the data is necessary.

The engineered input features of the model are discussed first. Then, a method utilising cross-validation that enables the training of an XGBoost model on a small dataset without over-fitting is developed. Finally, the model is applied to forecast the three electricity load datasets and the results are discussed.

4.6.1 Feature Selection

Tree-based models over-fit easily when redundant information is presented to them. Therefore, it is not reasonable to use a whole load time series as an input. The neighbouring points of a load time series are usually highly correlated and, therefore, each point contains only a small amount of new information. Moreover, similar load patterns repeat every day, which constitutes additional redundant information in the time series. Hence, the key to generating good forecasts using tree-based models is to extract meaningful features from the data, which are then used as variables for the regression model instead of the complete load time series. In the following, the engineered features that are utilised in the computation of the forecasts with the XGBoost model are introduced.

During the development of the models, several engineered features have been tested. The features that proved to be very effective can be divided into three groups: past consumption features, weather features, and date features.
The *consumption features* contain information extracted from the historic load. They include the electricity load from 48 hours ago and the electricity load from one, two, and three weeks ago. For the forecast task at hand, only the second-to-last day consumption values for all timestamps that can be used for the forecast are available. That is why the load from 48 hours ago is utilised as a feature instead of the one from 24 hours ago. Additionally, it turned out that the forecast quality is improved by the inclusion of statistical measures of the last known day. Hence, the mean value, the 0.25- and 0.75-percentiles, the maximum, and the minimum load of that day are utilised as features. They offer a condensed survey of the recent consumption, which presumably offers insights into the future consumption.
The *weather features* contain information about the weather. In particular, the ambient temperature strongly influences the residential electricity consumption. The features include the predicted temperature for the time of the prediction, and the predicted temperatures three hours prior and thereafter, which enable the model to identify a possible temperature trend. Furthermore, the corresponding temperatures to the past consumption features are added in order to contextualise the consumption with the temperature.
The *date features* offer information that is connected to the date or, respectively, the point in time for which the forecast is computed. They include the hour of the day and the month of the year. Instead of computing an average daily consumption (standard load curve) and passing the corresponding load value on to the model, the hour of the day is given to the model. This enables the model, on the one hand, to learn the dependency between the time of day and the load itself and, on the other hand, enables it to recognise and use complicated relationships between the time of day and other inputs. Additionally, a binary feature that states whether the sun is

up or not can be utilised to take the variability of the length of day in the course of a year into account. Furthermore, the weekend feature is included, because the electricity consumption behaviour changes on weekends.

In summary, when forecasting the electricity load series created from the CER Smart Meter Trial dataset, the below listed features are used as inputs:

- historic-consumption features:
  $load - 48h, load - 1w, load - 2w, load - 3w, mean, max, min,$
  $0.25\text{-}percentile, 0.75\text{-}percentile$
- weather features:
  $temp, temp - 3h, temp + 3h, temp - 48h, temp - 1w, temp - 2w, temp - 3w$
- date features:
  $hour, weekday, weekend, month, sun\ is\ up$

During the development of the XGBoost model, various other features have been tested as well. The above list of engineered features represent those that have proven to be beneficial to the forecast performance of the model. Depending on the dataset, it is possible that additional features which improve the forecast quality exist.

4.6.2 Hyper-Parameter Adjustments

The performance of an XGBoost model is mainly dependent on the hyper-parameters of the model. There are plenty of parameters that can be adjusted. The most influential hyper-parameters of a tree-based model are the following:

- rounds: the maximum number of gradient-boosted trees that are created during the training
- tree-depth: the maximum tree depth of each base learner
- eta: weight shrinkage of the leaves, similar to learning rate
- subsampling-rate: ratio of training samples that is used to grow a tree

The main challenge when training tree-based models is preventing over-fitting. That is particularly a problem when the training dataset is small, because the model keeps adding more trees until the training error does not decrease further. In order to prevent over-fitting, fixed values are assigned to the above mentioned parameters with which the minimal training error is not achieved, but the generalisation ability of the model increases. By limiting the number of trees and their depth, the model is

not capable of memorising the training data and by training each tree only on a sub-set of the training set over-fitting is limited to a certain extent. As a consequence, the model learns to generalise during the training and over-fitting is avoided.
In order to identify the hyper-parameter values that prevent over-fitting but still offer a sufficient amount of trainable parameters, a k-fold cross-validation is the first part of the model training. A k-fold cross-validation means that the training data is split into k complementary sub-sets. The model is then trained on $(k-1)$ of these sub-sets and the remaining sub-set is used for evaluation during the training process. The addition of new trees stops when the error on the remaining sub-set does not decrease any further. This procedure is repeated k times until every sub-set is used for validation exactly once. The total training error is computed as the mean of the different evaluation errors. A grid-search is conducted for the four parameters to find the most effective model. Thus, the model is trained several times with different combinations of hyper-parameters and each time the total training error is computed. The parameter combination with the smallest error is chosen to eventually train the final model on the complete training set. If the model were trained on the complete training data without any restrictions to the hyper-parameters, the result would be a more complex model, which would perform better on the training data but much worse on the test data or any unseen data. For the other parameters, which are less influential than the mentioned ones, the default values are used, which can be found in the XGBoost documentation [114]. The exact training procedure is described in Algorithm 1.

4.6.3 Forecast Evaluation

The evaluation of the generated forecasts shows that the tree-based models are indeed capable of computing satisfying load predictions. The best trained models outperform all previously developed forecast models on all three datasets (see table 4.5). In particular, the forecast error of the IRE350 set improved drastically (by more than 20 %) in comparison to the fully-connected NN model. This proves that a non-linear model, like the random forest model, is able to produce better forecasts by identifying and exploiting non-linear correlations between the model inputs and the predicted value. The performance increase on the IRE350 dataset is probably much larger than on the other two sets, because, due to the less volatile series consumption patterns, relations between features and the output and changes in consumption can be detected more effectively and put to use. This means on the other hand, that there is still plenty of room for improvement on the IRE15 and IRE40 dataset.

Algorithm 1 The steps of training a random forest model applying the XGBoost algorithm with cross-validation are listed below.

▷ Define cross-validation parameters

tree-depths $D = \{1, 2, 3, \ldots, 20\}$
subsampling-rate $S = \{0.1, 0.2, \ldots, 1.0\}$
eta $E = \{0.05, 0.10, 0.15, \ldots, 0.70\}$
rounds $r = 50$
$folds\ k = 4$

▷ Start cross-validation

divide data into k sets

▷ Find best tree-depth

for d in D **do**:
    for each set **do**:
        initialise XGBoost model with $tree\text{-}depth = d$
        train model with remaining $f - 1$ sets
        evaluate model with the set
        save forecast error
    end for
    average error over all k sets
end for
set d_{final} to the value with the smallest forecast error

▷ Find best subsampling-rate

repeat the procedure from above with $s \in S$ utilising d_{final}
set f_{final} to the value with the smallest forecast error

▷ Find best eta

repeat the procedure from above with $e \in E$ utilising $d_{\text{final}}, f_{\text{final}}$
set e_{final} to the value with the smallest forecast error

▷ Check best tree-depth again

repeat the procedure from above with $d \in D$ utilising $f_{\text{final}}, e_{\text{final}}$
set d_{final} to the value with the smallest forecast error

▷ Find best rounds

train XGBoost model with final parameters applying cross-validation
stop adding trees when the forecast error does not further decay $\rightarrow r_{\text{stop}}$
$r_{\text{final}} = \overline{r_{\text{stop}}}$

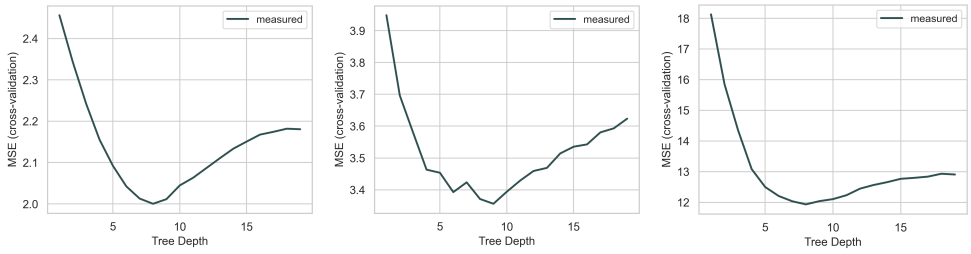
▷ Train final model

train XGBoost model utilising $d_{\text{final}}, f_{\text{final}}, e_{\text{final}}, r_{\text{final}}$ on the complete data
$\rightarrow model_{\text{final}}$

Table 4.5: The mean squared error of the XGBoost regression model trained with 4-fold cross-validation for the three datasets.

	MSE		
model configuration	IRE15	IRE40	IRE350
all features, parameters from cv	5.81	15.8	290
only shifts, parameter from cv	6.26	17.8	360
all features, default parameters	5.91	16.3	347

In addition to the models trained with the previously described features and the developed cross-validation, models of two further configurations have been trained in order to gain a better understanding of the influence of the modelling choices. On the one hand, a model that includes only the shifts and the corresponding temperature values has been trained to have a better comparison to the linear models. On the other hand, a model using the default hyper-parameters, which perform reasonably well in a lot of cases, has been trained to evaluate the effectiveness of the cross-validation.

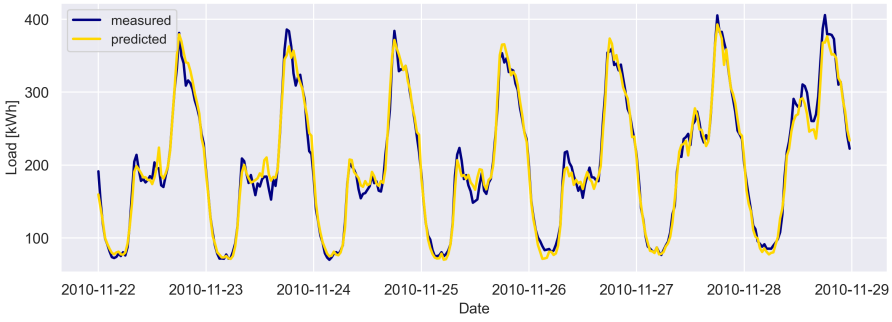


(a) IRE15 **(b)** IRE40 **(c)** IRE350

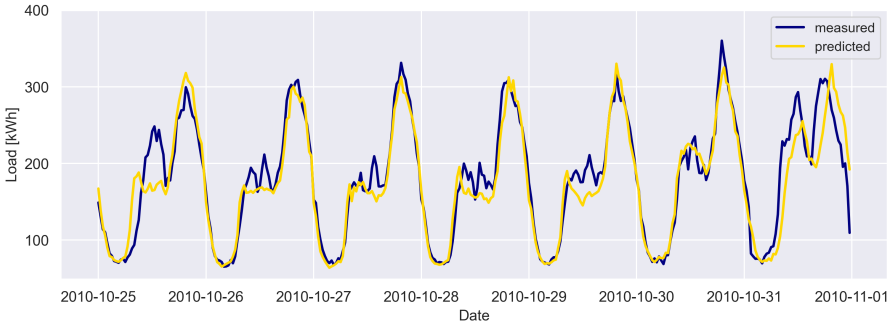
Figure 4.5: Development of the MSE with respect the tree depth during the grid-search. The error is calculated with the training set during the cross-validation process of the training. All curves show a clear minimum that constitutes the optimal tree-depth, which is then used for the final model configuration.

The model that only utilises the shifts and temperatures performed significantly worse on all three datasets of differing aggregation level. This proves, in combination with the results of the NN model and the linear models that the developed features contain information which enable more accurate predictions to be computed. The forecast results from the model that has been trained on the default hyper-parameters without cross-validation are larger for all three datasets, even if only marginally for IRE15 data. This shows that the determination of the hyper-

parameters by means of cross-validation contributes to a more successful forecast model.
To illustrate the influence of the hyper-parameters on the forecast performance of the tree-based model, the error with respect to the tree-depth is exemplarily evaluated during the cross-validation. Figure 4.5 shows the error development for all assumed values. It is noticeable that the curves for the three datasets have roughly the same shape. With decreasing tree-depth, first, the error decreases and then slowly increases again. The minimum of that curve constitutes the optimal tree-depth, which is used for the final model. The learning rate and the sub-sampling ratio are determined analogously.



(a) Measured and predicted load of an accurately forecasted week.



(b) Measured and predicted load of an inaccurately forecasted week. Deviations are, in particular, noticeable during the first half of the days.

Figure 4.6: Exemplary forecasts of the IRE350 electricity load time series with the XGBoost model trained with cross-validation. The blue line represents the measured load and the yellow line the prediction. The forecast in (a) is very accurate, whereas in (b) deviations from the actual load are visible.

In order to illustrate the accuracy of the forecasts, figure 4.6 displays two exemplary weeks of predictions of the IRE350 set computed with the random-forest model. It can be seen in 4.6(a) that very accurate load forecasts are possible with the model, but 4.6(b) shows that, for some part of the test data, there are still obvious discrepancies between the prediction and the measured load. The deviations of the first day of 4.6(b) are particularly prominent. 25 October 2010 was a Monday, but its load curve deviates from that of a typical weekday. The reason is that there was a bank holiday on that day. When the selection of which features to include was made, a feature that indicates bank holidays was tested as well. However, no improvement in forecast accuracy could be achieved because, as an analysis showed, the feature has not been considered by any decision tree, hence it was not used by the model. Apparently, the influence of this feature is too minor to be considered by the training algorithms, probably due to the fact that the training set contains only a few bank holidays. Therefore, the model is not able to accurately predict the electricity loads of bank holidays.

4.7 Developing a Convolutional Neural Network Forecasting Model

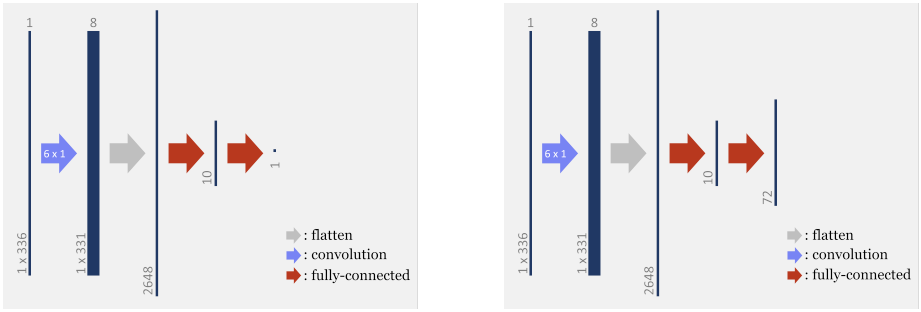
In this section, a novel model for time series forecasting based on convolutional neural networks is developed. The basic structure of the the model is as follows. Several one-dimensional kernels of a convolutional neural network layer operate directly on the input time series. An arbitrary number of further convolutional layers can subsequently process the resulting activation maps. Subsequent, several fully-connected layers process the data of the last convolutional layer and compute the output, which represents the forecast. With the application of a fully-connected layer on an activation map, the temporal context of the data is lost, because the layer treats all its inputs identically. The activation maps of the last convolutional layer are used comparably by the fully-connected part of the suggested neural network model than the input features of a non-linear regression model. The function of the convolutional layer(s) is, therefore, to extract meaningful information from the input series and to provide this information to the fully-connected part of the network similar to features. Hence, the suggested model does not require manually engineered features in order to calculate a forecast.
The performance of the novel model depends on many parameters. In the following sections, the process of determining effective model parameters is presented in detail. After defining the pre-processing and fundamental parameters of the model, the influence of the fully-connected part of the model on the forecast performance is evaluated. Afterwards, different hyper-parameters and the size of the convolutional

part are varied in order to obtain an accurate forecast model. Finally, external features are included in the model and evaluated. Based on the gained understanding of the model a final network structure is developed.

4.7.1 Choosing a Forecast Approach

Before the development of a convolutional neural network model for time series forecasting commences, the fundamental choice whether to calculate the forecast at once or iteratively must be made. In the following, this decision is made based on a preliminary test.

As mentioned in section 3.3.3, there are two approaches to compute a time series forecast using a CNN model. The forecast can be computed directly or iteratively. When computing the forecast iteratively, only one data point is calculated at once. Potentially erroneous data points from earlier predictions are, thus, part of the input. Hence, the errors of the earlier prediction steps might be amplified and corrupt the following prediction steps, which deteriorates the overall forecast performance. In order to identify the most promising approach, a preliminary test has been conducted, which included the training of two basic CNN models that differ only in the configuration of the output layer. The neural network consists of one convolutional layer comprised of eight kernels, each of which consists of six neurons. The resulting feature maps serve as input for two successive fully-connected layers with the temporal order and the distinction between the feature maps being considered. All inputs of a fully-connected layer are treated equally. The first layer contains ten output neurons and the final output layer has either one output neuron or 72, depending on which approach is applied. The network architectures are also depicted in figure 4.7. All other hyper-parameters are identical. The first model only has a single output neuron. The predicted value of this network is appended to the input series after each prediction step and the first value of the series is removed in return, in order to preserve the length of the input series. This procedure is repeated 72 times until a 36 hour forecast has been calculated. The output of the second network consists of 72 output neurons, hence it produces a 36 hour forecast.
The comparison of the forecast performance of the two different models indicates that the iterative approach excels against the direct approach. One possible explanation for this rather unexpected outcome lies in the number of trainable variables of the two models. The number of trainable weights of the model with 72 output neurons is significantly greater than of the model with one output neuron. Each output neuron of the direct model is connected to the ten preceding neurons and the bias. Hence, the last layer contains 792 weights in total. The last layer of the iterative model, on the contrary, contains only eleven weights. The large amount of



(a) CNN used for iterative approach. **(b)** CNN used for direct approach.

Figure 4.7: The figures depict the two networks trained in the preliminary test. They both consist of one convolutional layer, utilising eight kernels of size $k_{\text{size}} = 6$, two fully-connected layers, and differ only in the number of output neurons. The different coloured arrows represent the described layers. The "flatten" layer represents the suspension of temporal and kernel-based order.

trainable weights of the direct model in combination with a rather small number of training samples presumably facilitates over-fitting. This is only of minor concern for the iterative model due to the small number of trainable weights in the output layer.

Due to the preliminary test, the iterative forecasting approach of creating a time series forecast was chosen and is pursued in this work. The direct approach would also possibly generate good forecasts if the number of training samples were sufficiently large. For the research presented in this thesis, however, the training set is limited to one year of electricity consumption data due to the reasons mentioned in section 4.1. Furthermore, the iterative approach seems more intriguing from a research point-of-view, as it differs from the well-studied architecture of fully-connected neural networks, which are commonly applied to time series forecasting as discussed in chapter 2. Applying CNN models on iteratively computing a one-dimensional time series forecasts constitutes a novel approach in the file of time series analysis.

4.7.2 Pre-Processing of the Load Time Series

The pre-processing of data is an important step in creating a successful machine learning model. It can be divided into several steps: assessing the data quality, utilising domain knowledge, and data transformation or, respectively, rescaling. In the following, the electricity consumption data is analysed regarding these

steps.

Firstly, the data quality is assessed and missing and inconsistent values are identified. These data points or segments are then either excluded from the dataset or estimated using an appropriate method. The majority of these processes are not highlighted here because they are standard procedures of every machine learning algorithm and because no unconventional measures had to be taken due to the good quality of the data from the Irish Smart Meter Trial. Another aspect of pre-processing is utilising domain knowledge about the data source and the context of the data. With that knowledge, information and features can be extracted from the data and the data can be better contextualised. Feature extraction has already been discussed in section 4.2. Furthermore, the data can be transformed in such a way that the machine learning models process the data more effectively, e.g. normalisation, de-noising, etc. .

Electricity load data exhibits a clear, seasonal behaviour as already mentioned earlier. The underlying causes for such a changing electricity consumption behaviour during the year are manifold and not explicitly known. The behaviour and changes in the course of a year are dependent on the country and probably also correlate with socio-economic factors, which makes it difficult to exactly identify them. That is why predicting electricity loads is such a challenging task.
During the winter, the daily electricity consumption in Ireland is on average higher than during the summer. There are two apparent explanations for that behaviour. Firstly, the daily sunshine duration is shorter during the winter. Hence, more lighting and, thereby more electricity, is required. Secondly, due to poorer weather conditions and colder temperatures during the winter, the residents spend more time in their houses and, therefore, use more electrical household appliances. In order to validate these assumptions, the electricity load, the temperature, and the daily sunshine duration are plotted in figure 4.3 for the complete IRE350 dataset. Apart from the yearly oscillation that both the load and the sunshine curve exhibit, their shapes clearly differ from each other, which militates against a linear correlation of the two. The shape of the load and the (inverse) temperature curves, however, resemble each other to a great extent, which indicates a strong linear correlation. That assumption is further supported by several coinciding local extrema of the two curves. Three of them are marked in the plot with a vertical, grey, dashed line for accentuation. Based on these findings, it is assumed that a linear dependency between the electricity load and the temperature, or, respectively, the weather, for which the temperature is one indication, exists. Additionally, such a dependency can also be noted for the temperature and the local standard deviation of the load. The standard deviation of the load changes with the seasons, similar to the daily average load (see fig. 4.1). It increases with falling temperatures and vice versa.

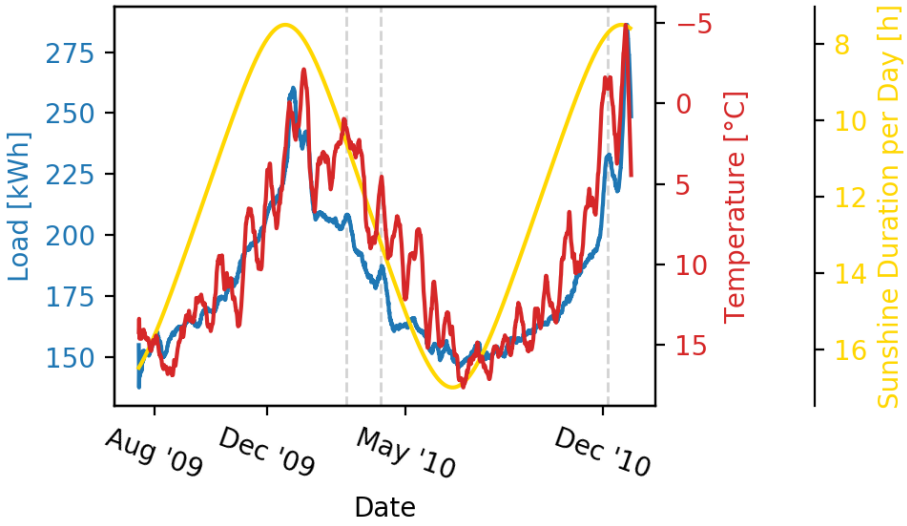


Figure 4.8: The graph depicts the IRE350 load (blue), the temperature (red), and the daily sunshine duration (yellow) of the complete dataset. For a better depiction of trends, the seven-day running average of the load and temperature are displayed. It can be seen that the temperature curve closely follows the load curve in contrast to the curve of the daily sunshine duration. The grey, vertical, dashed lines indicate dates where a strong correlation between the load and temperature is visible.

In order to utilise this obtained domain knowledge and to facilitate the training process of the machine learning model, the data is modified with the use of the temperature time series.
First, a linear regression model is fit using the rolling seven day average temperature as regressor to approximate the rolling seven day average electricity load, the regressand. As a result, the regression model is able to roughly predict the development of future electricity consumption, based on the weather forecast. Subsequently, the approximated load by the regression model is subtracted from the actual load. If the temperature were a perfect approximator, the seven day running mean of the resulting time series would constantly be zero. As that is not the case, only the total mean of the time series is zero. Second, an additional linear regression is fit to estimate the influence of the temperature on the variance of the electricity load. The regressor is, again, the temperature time series and the regressand is the squared seven day running mean. By dividing the squared time series by the estimated squared time series approximated by the regressor, the total variance of the resulting series is eventually zero.

The objective of these two regressions is to reduce the influence of the temperature on the load time series and, thereby, remove one influence factor from the series, which otherwise has to be taken into account by the forecast model. The seven day running average is used to avoid daily and weekly patterns influencing the training of the regression models.

Algorithm 2 Suggested pre-processing procedure for the electricity load data. The pre-processing tries to eliminate the influence of the temperature t on the load l.

1: *temperature* t, 7-*day-running average temperature* $\bar{t}$
2: *load* l, 7-*day-running average load* $\bar{l}$

3: fit linear regression with parameters α_1, β_1
4: $\bar{l} = \alpha_1 \bar{t} + \beta_1 \quad \Rightarrow \quad \alpha_1, \beta_1$
5: convert load: $l_{\text{reg1}} = l - (\alpha_1 t + \beta_1)$

6: fit linear regression with parameters α_2, β_2
7: $\bar{l}^2_{\text{reg1}} = \alpha_2 \bar{t} + \beta_2 \quad \Rightarrow \quad \alpha_2, \beta_2$
8: convert load: $l_{\text{reg2}} = \frac{l_{\text{reg1}}}{\sqrt{\alpha_2 t + \beta_2}}$

9: normalise data to interval $[-1, 1]$

10: use normalised l_{reg2} as input for the CNN model

As the final step of the pre-processing, the data is normalised so that all values are situated in the interval $[-1, 1]$. Most of the common activation functions for neural networks are saturating at large negative and positive values. Hence, the error back-propagation, and therefore the training, is less effective when the input value range is large, because the back-propagated error is multiplied with the derivative of the activation function. A neural network that uses data that is not normalised should eventually converge as well, but the training could take longer.
The scaling and regression parameters are determined using the training set. When unseen data (e.g. live-data, test data) is processed by the fitted models, including the pre-processing, the same scaling parameters as for the training set are applied. Even though this can result in input values of the machine learning model not being in the desired input interval of $[-1, 1]$, the pre-processing is not changed. Otherwise, the same temperature-load-tuple can result in differing input values of the machine learning model during training and application of the algorithm, depending on the range of the presented data. This behaviour is not intended and would result in a decline of the forecast performance.

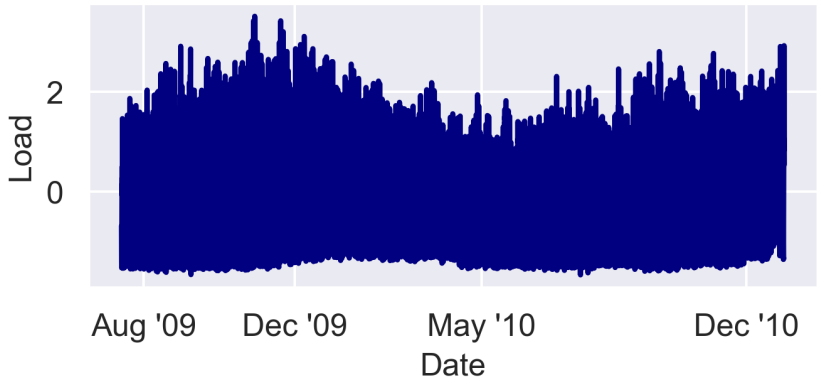


Figure 4.9: IRE350 load time series after being transformed by the two linear regressions with the temperature as regressor. The series has a mean around zero and the magnitude varies noticeably less than before the transformation.

4.7.3 Determining Fundamental Hyper-Parameters

This section addresses the fundamental parameters of a convolutional neural network for time series forecasting which influence the way the model is trained and the predictions are computed. A variation in those hyper-parameter changes the architecture and character of the network drastically, hence they are being called fundamental parameters. A change of those parameters also makes a comparison between different forecast models virtually impossible. In the following, the effects that a variation of these parameters has on the model are discussed and finally their values are set and are not changed for all subsequently presented models. First, fundamental parameters that effect the training process are discussed, followed by fundamental parameters that describe the network structure.

4.7.3.1 Training Parameters

A convolutional neural network has some hyper-parameters that specify the training procedure of the neural network. They are also called training parameters. They have a great influence on the effectiveness and efficiency of the training process. In the following, first the impact of the different parameters on the training process are discussed and, thereafter, effective parameters for electricity load forecasting are determined.
The most influential training parameters of a neural network are the number of epochs, the (mini-)batch-size, the learning rate, and the applied optimiser.

Introduction of the Fundamental Training Parameters

First, the *number of epochs* are discussed. The trainable variables of a neural network, which are usually the weights, are initialised with random values before the training commences. After the complete training set has been presented to the neural network, the model generally does not produce good forecasts and the weights are still trained poorly. Therefore, the training set is presented to the network several times during the training process. Each such repetition is called an epoch. Once the model has been trained with a sufficient number of epochs, a local minimum of the error function is approached. If the network is trained further with more epochs, its generalisation capability will diminish, because the CNN model memorises the data. This means that over-fitting occurs, which results in a worsened forecast quality of new, unseen data that is not in the training set. In order to avoid over-fitting and ensure an efficient training, the number of epochs must be chosen neither too large nor too small. To identify a number of epochs that balances out the effects of insufficient training and over-fitting, the forecast error of the training set and of the validation set is evaluated after each epoch. The training error usually decreases heavily during the first epochs and subsequently flattens out when the neural network model does not improve further. The validation error displays a similar behaviour with the addition that it starts increasing again at some point, which then indicates over-fitting. The optimal amount of training epochs can be deduced from the minimum of the validation error, because it represents the number of epochs that are necessary to achieve the best possible generalisation properties. When trained further, the model adapts only specifically to the training data. It does not learn the general nature and behaviour of the data any more. Therefore, the model becomes less accurate for unseen data and, hence the validation error increases. Typical error curves can be seen in figure 4.11, extracted from the training on the IRE40 dataset.

Another crucial training parameter is the *batch-size*. It describes the number of samples after which the network weights are updated. The training set is typically divided into sub-sets that have the cardinality of the batch-size. The samples of the sub-sets are randomly sampled without replacement from the training set. The sub-sets are presented to the network one after another and the network weights are updated every time all samples of one sub-set have been processed.
There is no general guideline that states an optimal batch-size value. The influence of the batch-size on the training process depends strongly on the data, the used neural network model, and the resulting error landscape. The batch-size is often chosen to be a power of two, in order to allow for better parallel computing.

The *learning rate* of a neural network model determines the adjustment of the model in response to the computed errors during training. It is a measure for how fast the

model adapts to the training samples. The influence of the learning rate on the training in the setup used is small due to the chosen optimiser and is, therefore, not further discussed here.
All networks in this thesis are trained using the NADAM [88] optimiser. The name is derived from *Nesterov Accelerated Adaptive Moment Estimation*. The optimiser utilises an adaptive learning rate, which means that each trainable parameter of the neural network possesses an individual learning rate. These individual learning rates change during the training according to the first and second moment of the gradients.

Determination of the Fundamental Training Parameters

As first step of developing a convolutional neural network for time series forecasting, the effects of the above mentioned training parameters are analysed using the available training set and their values are determined. They are not changed any more during the further development of the CNN model.
Firstly, the differing behaviour when applying different batch-sizes was examined. For this purpose, a basic 1D-convolutional neural network has been trained using varying batch-sizes. The network structure is identical to that in section 4.7.1 (see also fig. 4.7). In order to validate the training progress after each epoch during the training and to monitor the convergence, a validation set is necessary. Due to the limited available data, it has only been feasible to split the data into a training and a test set without producing biased datasets, as described in section 4.1. It is not possible to create an unbiased validation and an unbiased test set with the limited amount of data. Therefore, the test set is used here for validation purposes during the training. That contradicts the idea of a test set, but is justifiable in this instance, because no final model is developed. During the development of the neural network structure in the following section, validating the training progress is, therefore, abstained from.
The experiments with different batch-sizes have led to the conclusion that a batch-size of 128 is best suited for the given task. Identical networks have been trained with different batch-sizes and with a deliberately chosen very large number of epochs. The examined batch-sizes b_{size} and the number of epochs e are:

- $b_{\text{size}} \in \{2, 4, 8, 16, 32, 64, 128, 192, 256, 512\}$
- $e = 200$

The analysis of the differing training processes has shown that the best achievable forecast errors are very similar for all the examined batch-sizes. A comparison of the training duration per epoch, however, has yielded a very large training duration for small batch-sizes (see fig. 4.10). For medium and large batch-sizes, the duration

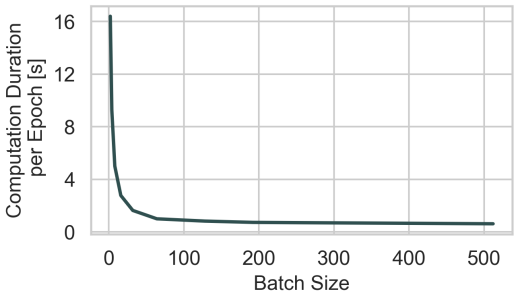
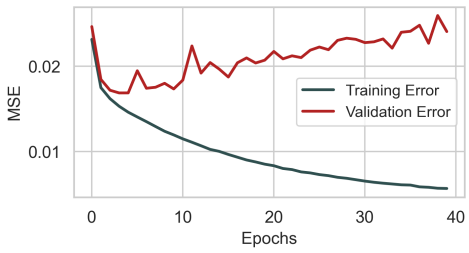


Figure 4.10: The graph shows the computation duration for a network trained on a home PC. The computation takes longer for small batch-sizes, probably because the training processes cannot be distributed on several CPU- or GPU-kernels. Furthermore, computing the weight updates and, in the case of the adaptive learning rates, updating the individual learning rates for all weights is computationally costly as well.

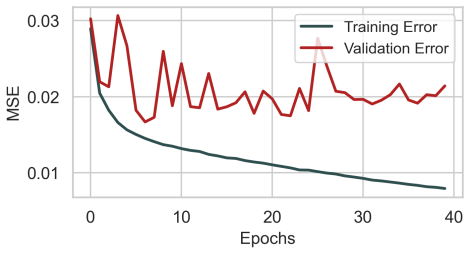
differences are marginal. Moreover, the number of epochs necessary for the neural network to converge increases with the batch-size. When trained with a small batch-size, the networks converge within a few epochs. Unfortunately, over-fitting occurs only a few epochs afterwards (see fig. 4.11(a)). The small range of epochs that promise a properly trained model and the large training duration per epoch discards batch-sizes with $b_{\text{size}} < 32$. Batch-sizes of $b_{\text{size}} \geq 256$ are also not deemed practical, because the training duration per epoch is the same as for smaller batch sizes, but more training epochs are necessary to reach the minimal validation error and, hence, the best possible forecaster. Since the fluctuation of the validation error during training is significantly lower for batch-sizes of $b_{\text{size}} \geq 128$ (see fig. 4.11(b) and (c)) and because the validation error plateaus at the minimum error for several epochs for batch-size $b_{\text{size}} \geq 64$, I decided to employ 128 samples in every batch. The training behaviour is very similar for all three datasets (IRE15, IRE40, and IRE350) and the same batch-size of $b_{\text{size}} = 128$ is applied.

Secondly, an optimal number of epochs that ensures convergence but minimises the risk of over-fitting had to be determined. Therefore, the same basic network that was used to determine the batch-size has been trained using a far-too-large number of epochs, in order to better identify the optimal number of epochs. The typical development of the error during the training process is depicted in figure 4.12. The neural network is trained with a sufficient number of epochs when the validation error reaches a minimum. A further decline of the training error with a simultaneous increase of the validation error is a sign of over-fitting, which reduces the generalisation ability of the neural network model. An epoch size of $e = 40$ appears to be optimal for the CNN model trained with a batch-size of $b_{\text{size}} = 128$. For all three

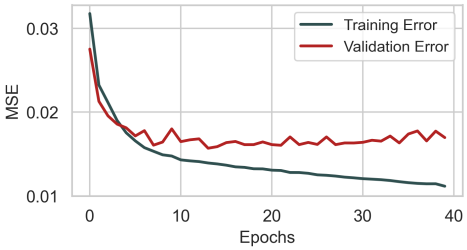
datasets, the validation error typically reaches a minimum earlier than 40 training epochs, but over-fitting commences only afterwards.



(a) batch size of 2



(b) batch size of 64



(c) batch size of 128

Figure 4.11: The three graphs depict the training error (grey) and validation error (red) development of the IRE40 dataset for three different batch-sizes. The error is computed during the training process and represents therefore only the aberration of the one-point forecast, hence the different scaling. The comparison of (b) with (c) emphasises the decrease of validation error fluctuation for larger batch-sizes. (a) illustrates the rapid training process and the subsequent over-fitting of a CNN with a small batch size.

The learning rate is set to $\eta = 0.001$, which is the default value of the NADAM optimiser. The learning rate is not adjusted due to the applied optimiser. As mentioned before, the NADAM optimiser computes individual learning rates for all trainable parameters of the neural network that are adapted during the training. Therefore, the initial learning rate has a smaller influence on the training than for optimisers that use a global learning rate. Additional tests with varying learning rates have confirmed this assumption. They have also shown that 0.001 is in the range of the most efficient learning rates. There are two more parameters, β_1 and β_2, that control the influence of the momentum on the individual learning rates. Those parameters are kept at their default values as well. The applied loss function is the mean-squared error, because it penalises large deviations more severely, which is

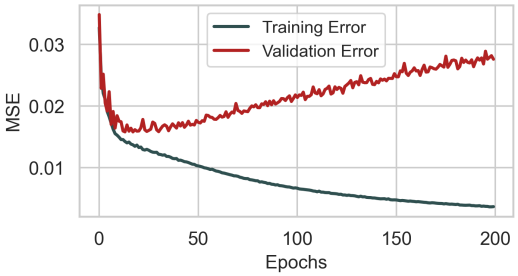


Figure 4.12: This graph shows the training and validation error for the one-point forecast during training with the IRE40 dataset. A batch-size of $b_{\text{size}} = 128$ was applied. The CNN is trained with a large number of epochs. The validation error reaches a minimum in the range of 20 to 50 epochs. The graph is representative for all three datasets.

favourable by the energy system in MAGGIE. Small deviations can be compensated by the heat buffer.

The training parameters determined in this section are used throughout the whole thesis. It is assumed that neural networks with similar architectures to the tested one still work very effectively with these parameters. Only if substantial changes to the network architecture were made, would a reassessment of the training parameters be necessary. The high computational costs of finding optimal training parameters for each tested model architecture are not justified by the possibly minimal benefits. Therefore, the parameters are fixed to the values presented in table 4.33.

Table 4.6: The training parameters determined in this section are listed in the table. They are used throughout this whole work if not stated otherwise.

training parameter	batch size	epochs	learning rate
applied value	128	40	0.001

4.7.3.2 Network Parameters

After having determined the training parameters, a suitable network architecture for load time series forecasting has to be found. There are some fundamental network parameters that are not changed during the model development. In this section, first the fundamental model parameters are outlined. Then, the influence of these parameters are discussed and optimal values are determined in order to acquire an effective forecast model.

There are some parameters of the network that are fundamental for the setup of the whole forecast model. A change of these parameters not only drastically changes the way the forecaster works, but makes it difficult to compare variations in the model configuration. These parameters are specified in the following paragraphs and the parameter choice is justified.
As mentioned before, there are two ways a neural network model can construct forecasts: directly or iteratively. This thesis focuses on creating forecasts iteratively. Preliminary tests have shown that this approach is more promising. Additionally, creating the forecasts iteratively distinguishes the research into time series forecasting with CNNs more distinctly from forecast models utilising solely fully-connected neural networks. As outlined in chapter 2, forecast models using CNNs have hardly been studied yet opposed to fully-connected NNs. Moreover, creating an iterative forecast utilising fully-connected NNs is only possible to a limited extent in contrast to CNNs. The input time series of a fully-connected NN should preferably always contain the same period of time, meaning it always starts with the identical time of the day, if the full potential of the neural network is to be exploited. This limitation does not exist for 1D-CNNs, which makes them more versatile and allows for the iterative approach to work. Therefore, the number of output neurons of the neural network is fixed to one.
Another parameter that is not changed throughout the thesis is the *length of the model input series*, because it strongly influences the training process as a whole. The load input time series for all the CNN models and hence the network is exactly one week, which translates with a sampling rate of 30 minutes to 336 data points. The input length of seven days is chosen because the consumption behaviours and the average consumption of different weekdays differ from each other (see fig. 4.13). With an input length of one week, it is ensured that the model input is not biased. If the input length were shorter, for example, four days, it would consist of the loads of the previous Thursday, Friday, Saturday, and Sunday in order to predict the consumption of a Tuesday. As, on average, more electricity is consumed in a residential household on the weekend than on weekdays, the daily average consumption of the input data would be too large and possibly distort the forecast. Therefore, the length of the input series is set to 336 samples, which corresponds to the repeating weekly pattern the electricity consumption data exhibits. Furthermore, trials with larger input series have shown that no gains could be achieved with an input of two or more weeks. Larger inputs merely inflate the number of trainable weights of the CNN, which makes them more difficult to train. It has been concluded that electricity consumption data from further in the past has no real impact on the forecast.
Another integral component of a neural network and, hence, an influential parameter is the *activation function* of the neurons. Typically, all neurons in one layer have the identical activation function. The most common activation functions have been presented in section 3.1.3. As the CNN forecast model is a regression model, the output

neuron of the last layer possesses the identity as activation function. A regression model should have an unbounded output, which is ensured by an activation function that is also unbounded. The identity, $f(x) = x$, is the most simple function that fulfils this prerequisite and, additionally, has the benefit of being easily calculated. The activation function of the other neurons can be selected without restrictions. The ReLU activation function has been chosen for the other neurons, because it has been proven very effective in highly successful neural network models [115, 116, 117]. Moreover, it allows for efficient training, as it reduces the vanishing gradient problem due to its non-vanishing slope of one for positive values. Additionally, it is cheap to compute, as no calculation of exponential functions is necessary. The non-linearity of the ReLU activation function is provided by its change of slope at zero.

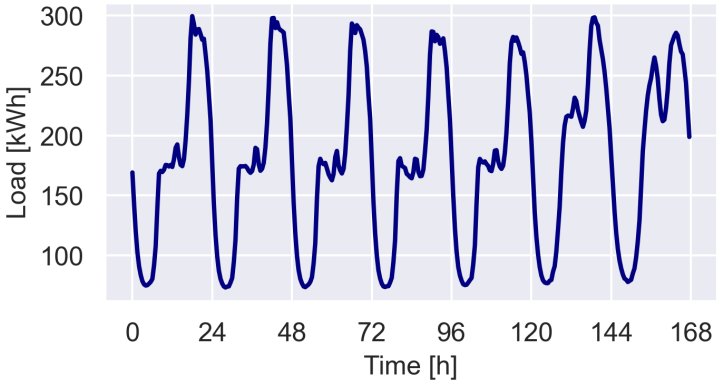


Figure 4.13: The graph shows the average weekly electricity load of the IRE350 dataset. The first day depicted is Monday. It can be seen that the consumption of different days of the week varies. Particularly prominent is the difference between weekdays and the weekend. However, also the weekdays vary slightly among one another.

The basic structure of the above described network remains the same throughout most of this work. There are many crucial network parameters that can and will be modified in order to find the best convolutional neural network for electricity load time series forecasting.

4.7.4 Parameter Variations

In the previous sections, the pre-processing, the training parameters, as well as the fundamental model parameters of a CNN model have been investigated and

specified, and, thus, a prototype of a convolutional neural network for time series forecasting has been created. This prototype forecasting model is used as basis of the subsequent research. In this section, the prototype is developed further by evaluating the influence of several parameters on the forecast performance. The development of the improved CNN model is, from now on, purely data-driven by the load data, in order to assess the potential of the novel forecasting approach using convolutional layers in a NN to extract information or, respectively, features from a time series. That implies that no manually extracted features or external variables are considered during this step of the development process.
There are several hyper-parameters in a convolutional neural network that can be adjusted. The influence of the following parameters on the forecast performance is discussed in this section in detail: the size and amount of the kernels in the convolutional layer, the size of the subsequent fully-connected network, the amount of convolutional layers in the network, the influence of dilated kernels, changes in the stride size, the addition of pooling layers, different variations of dropouts, and an additional parallel convolutional layer.

The prototype CNN model, which is modified in this section, has been developed in the previous section and computes the forecast iteratively. It consists of one convolutional layer, followed by two fully-connected layers, where the output of the second layer consists of the single output neuron. The activation function of the neurons is a ReLU function. The input time series consists of one week of electricity load consumption data, which equals 336 data points. The batch-size is set to $b_{\text{size}} = 128$ and the model is trained with $e = 40$ epochs.
By testing a great range of models with varying hyper-parameters and network architectures of increasing complexity, a CNN model that is optimally adjusted to the task of energy load forecasting is eventually derived. Furthermore, a better understanding of the influence and importance of the different hyper-parameters on the information extraction and forecast performance of the CNN model is gained.

4.7.4.1 CNN with Two Fully-Connected Layers

As the first step of the enhancement of the forecast model, the influence of and the interaction between the kernel size, the amount of kernels, and the number of output neurons of the first fully-connected layer are investigated. Therefore, a large grid-search is conducted in which a wide range of values is assigned to these three hyper-parameters. The variation in the forecast performance with the changes of the parameter values is studied, as is the relation of the parameters to each other. The model structure is depicted in figure 4.14.

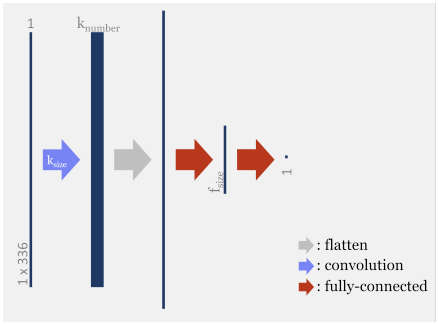


Figure 4.14: The graphic depicts the structure of the CNN model with one convolutional layer and two fully-connected layers. The different coloured arrows symbolise the different layer types that are utilised in the model.

Before the training starts, the hyper-parameter values are set and the CNN is compiled accordingly. The network weights are initialised with random values according to the uniform Glorot initialisation [118] and the biases are initialised with zeros. The initialisation, however, effects where in the multi-dimensional search space the training of the neural networks commences and, hence, has an influence on the resulting trained model. Therefore, in order to make the results of the different parameter combinations more independent from the initial weights and, thus, more comparable, ten networks of each parameter configuration are trained. Each of the networks has different initial weights, hence behaving differently during the training process. The total forecast error for one configuration is calculated as the average of all ten models. Whenever forecast errors are mentioned in the following sections and chapters, it is referring to the average error of the ten iterations.

The hyper-parameters kernel size k_{size}, amount of kernels k_{number}, and the size of the fully-connected layer f_{size} are varied in the following range:

- $k_{\text{size}} \in \{2, 4, 6, 8, 10, 12\}$
- $k_{\text{number}} \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $f_{\text{size}} \in \{1, 2, 4, 6, \ldots, 20, 25, 30, \ldots, 50\}$

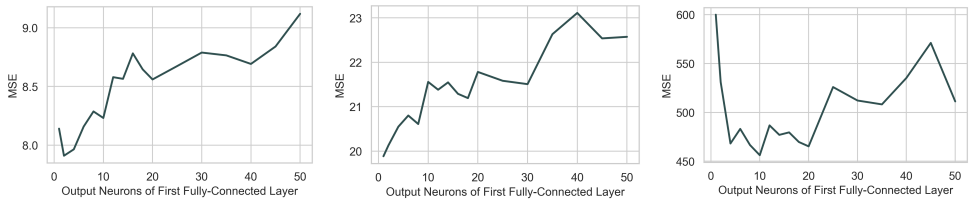
With the ten iterations per configuration being taken into account, 10 200 CNN models are trained in the course of this grid search for each of the three datasets.

The main finding of that grid search is that too large of a fully-connected layer has a detrimental effect on the forecast performance of the CNN model. Figure 4.15 depicts the development of the forecast error in dependency of f_{size}. For all three datasets, the error increases with an increasing number of neurons and, hence, with

Table 4.7: The table presents the forecast error averaged over all configurations and the error of the most successful configuration.

dataset		IRE15	IRE40	IRE350
MSE	average	8.51	21.5	503
	best model	6.59	17.1	287

an increasing number of trainable weights in the NN. It seems that over-fitting occurs when the fully-connected layers are too large. This result implies that smaller fully-connected layers are sufficient to process the information extracted by the convolutional layer. It is also noticeable that the prediction quality of the models for the IRE15 and IRE40 dataset deteriorates already at small f_{size} values, but for IRE350 only at values from $f_{\text{size}} > 20$. For smaller f_{size} (except for $f_{\text{size}} = 1$), the error on the IRE350 set remains basically constant. This is an indicator that the IRE350 dataset contains more information, respectively more information can be extracted from it than from the two other datasets. This is consistent with the findings of section 4.6.3, which concludes that the IRE350 dataset has a lower volatility due to large aggregation of households. Therefore, more information can presumably be extracted because it is easier for the model to notice changes in the consumption behaviour. That, in turn, improves the forecast. The fact that the error for $4 \leq f_{\text{size}} \leq 20$ is more or less the same also indicates that the extra information cannot be processed in meaningful way within the fully-connected part of the NN.



(a) Forecast error of the IRE15 set. **(b)** Forecast error of the IRE40 set. **(c)** Forecast error of the IRE350 set.

Figure 4.15: The plots depict the dependency of the amount of output neurons of the first fully-connected layer on the forecast error for each dataset. The error values represent the average error for all trained networks with the respective number of output neurons independently from k_{size} and k_{number}. It is clearly visible that a large number of neurons deteriorates the forecast error for all datasets.

No clear preferences could be identified for k_{size} and k_{number} in this grid-search. For IRE40 and IRE350, the number of applied kernels does not seem to have an influence

on the forecast performance. For IRE15, a slight preference for fewer kernels can be detected and is presented in figure 4.16.

The kernel size has no obvious influence on the forecast performance for all datasets and is further investigated in the following paragraphs.

The grid-search, which included 1020 different model configurations, has lead mainly to two findings. Firstly, a convolutional NN, which computes time series predictions of the volatile, high-resolution data at hand, does not require an extremely large amount of trainable variables in order to produce proper forecasts. This has been demonstrated by the fact that the forecast quality has gotten worse with a larger fully-connected part of the CNN model. Secondly, the concept of producing forecasts with a NN containing convolutional layers has proven to be very promising. The forecast errors on the test set of the CNNs with the best configurations are in the same range as the errors of the tree-based models and are superior to the baseline models.

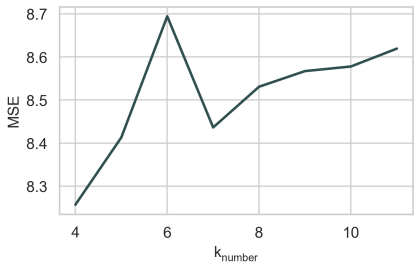


Figure 4.16: The graph depicts the forecast error of the IRE15 dataset with respect to k_{number} and is averaged over all k_{size} and f_{size}. The spike for $k_{\text{number}} = 6$ is caused by a few very poorly trained models that exhibit forecast errors far larger than the average. It can be considered an outlier. Apart from that, the error increases with an increasing k_{number}.

4.7.4.2 CNN with Three Fully-Connected Layers

In the next step of finding a better CNN model for time series forecasting, the fully-connected part of the neural network is expanded in order to be able to represent more complex relationships between the extracted features by the convolutional layer and between the features and the output. A further fully-connected layer is added, which means the fully-connected part of the network now contains three fully-connected layers with the last layer being the output layer with one output neuron (see fig. 4.17). Deeper networks are able to represent more complex

correlations between features as they allow for a greater degree of non-linearity.

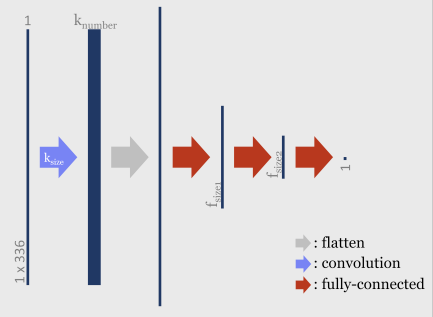


Figure 4.17: The graphic depicts the structure of the CNN model with one convolutional layer and three fully-connected layers.

The insight of the first-grid search - that a large amount of trainable variables is not necessary - is utilised by avoiding too large a number of weights. This is achieved by reducing the number of output neurons of the first fully-connected layer to a maximum of ten.
The hyper parameters for the new grid-search are:

- $k_{\text{size}} \in \{2, 4, 6, 8, 10, 12\}$
- $k_{\text{number}} \in \{2, 3, 4, 5, 6, 7, 8, 9\}$
- $f_{\text{size}} \in \{$[2-2-1], [4-2-1], [6-2-1], [6-4-1], [8-2-1], [8-4-1], [8-6-1], [10-2-1], [10-4-1] [10-6-1], [10-8-1], [12-2-1], [12-4-1], [12-6-1], [12-8-1], [12-10-1]$\}$

The f_{size} is depicted layer-wise from now on. For example, [4-2-1] means the first fully-connected layer has four output neurons, the second two, and the third layer, which is the output layer, one. The size of the respective input is already determined by the outputs of the previous layer. The possible sizes of the additional fully-connected layer are always chosen to be smaller than the size of the prior fully-connected layer. In fully-connected networks with several hidden layers, the size of the hidden layers typically changes gradually from the input shape to the output shape of the network. That principle is applied here as well.

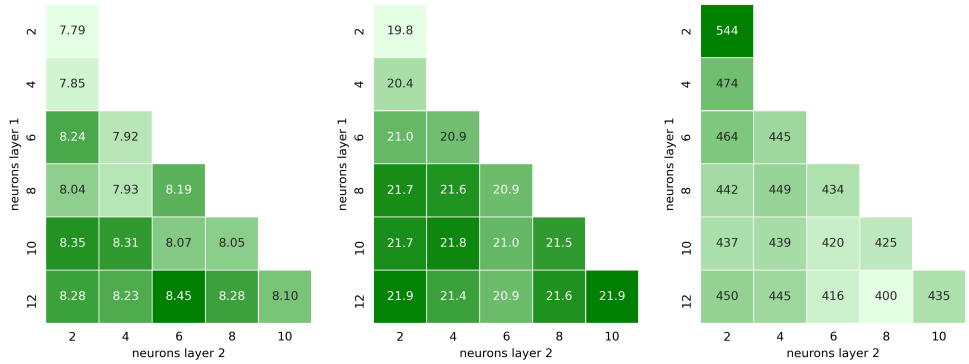
In order to have a better representation of the results, the average MSE of the configurations is depicted as color-coded in the below figures, when the error in dependency of two variables is plotted. Dark colors correspond to large error values. Thereby, patterns and dependencies can be recognised more easily.

The evaluation of the results reveals interesting insights into the capabilities of the proposed CNN regarding the processing of the different aggregation levels of the data.

Table 4.8: The table shows the forecast errors achieved by networks that contain three fully-connected layers.

dataset		IRE15	IRE40	IRE350
MSE	average	8.13	21.2	445
	best model	6.53	17.6	304

For the IRE15 dataset, the best results have been achieved with relatively small fully-connected layers after the convolutional layer. The most promising results are accomplished with fully-connected layers of the size $f_{\text{size}} \in [[2\text{-}2\text{-}1], [4\text{-}2\text{-}1], [6\text{-}4\text{-}1], [8\text{-}2\text{-}1]]$. The errors of larger fully-connected layers essentially increase with the size of trainable parameters.
For the IRE40 dataset, the results are not as distinct. It is, however, noticeable that the errors increase when the first-fully connected layers are large, which indicates that a neural network with fewer trainable weights is favourable for the IRE40 dataset.



(a) Errors on the IRE15 set. Small fully-connected NNs perform the best.

(b) Errors on the IRE40 set. Small fully-connected NNs perform the best.

(c) Errors on the IRE350 set. The error decreases with increasing fully-connected NN size.

Figure 4.18: The heatmaps depict the forecast error with respect to the amount of output neurons of the first and the second fully-connected layer. The values shown represent the average errors for all models with the respective fully-connected configuration, independently from k_{size} and k_{number}.

The error as a dependency of the size of the fully-connected layers develops the other way around for the IRE350 dataset. The CNN models with large fully-connected layers perform the best. The large error difference of more than 20 % between the best and worst performing setup is to be highlighted. The maximal difference for the other two datasets was each below 10 %. This indicates that, with changes in the model configurations, more improvements of the forecast error are possible for the IRE350 dataset than for the other two sets. That and the earlier analysis of the volatility supports the assumption that the information content that can be extracted from the data increases with decreasing volatility.
No conclusive statements can be made about the influence or the best values of the kernel size or the number of kernels, based on the results of the grid-search with two fully-connected layers. A few tendencies, however, are detectable. Concerning the amount of kernels, a smaller number seems favourable for the IRE15 dataset and a larger number for the IRE350 data. Different kernel sizes have no significant influence on the forecast results in the conducted experiments. Only a kernel size of $k_{\text{size}} = 2$ for the IRE350 set exhibits a larger error.

In general, it can be stated that a high volatility makes the extraction of information more difficult. Therefore, the gains that can be achieved by having more trainable weights in the neural network are larger for less volatile data.
However, on average, the additional fully-connected layer improves the forecast quality for all three datasets significantly, which means that by increasing the depth of the NN, more information could be utilised for the forecast calculation. The most improvement is observable for the IRE350 dataset. Whether the training parameters determined in section 4.7.3.1 are still working properly after adding an additional fully-connected layer, which is the case has additionally been investigated.

4.7.4.3 CNN with Four Fully-Connected Layers

The more layers a fully-connected neural network has, the more complex, non-linear correlations between its input and its output can be represented by the network. The earlier applied architecture of three fully-connected layers after the convolution, one of which is the output layer with a linear activation, is still rather shallow and small. Therefore, the next step of finding a good CNN for forecasting has been to add another fully-connected layer to see if allowing for more non-linearity produces better forecast results. The network structure of the model with four fully-connected layers is depicted in figure 4.19

In order to avoid unnecessary computational load and to minimise the resulting computation duration, the maximum number of output neurons of the first fully-

connected layer was set to 10, because the first grid-search suggests that a large, first fully-connected layer is not beneficial to the forecast quality. Moreover, it does not seem practical to test a large number of different k_{number}, due to computational load and the resulting computation duration. Therefore, the architectures to be tested have been narrowed down to the following. The same applies here as for the previous computations - ten iterations have been calculated per setup, in order to minimise effects introduced by the initialisation values and the random order of the training samples.

- $k_{\text{size}} \in \{2, 4, 6, 8, 10, 12\}$
- $k_{\text{number}} \in \{4, 5, 6, 7, 8, 9\}$
- $f_{\text{size}} \in \{[6\text{-}4\text{-}2\text{-}1], [8\text{-}4\text{-}2\text{-}1], [8\text{-}6\text{-}2\text{-}1], [8\text{-}6\text{-}4\text{-}1], [10\text{-}4\text{-}2\text{-}1], [10\text{-}6\text{-}2\text{-}1], [10\text{-}6\text{-}4\text{-}1], [10\text{-}8\text{-}2\text{-}1], [10\text{-}8\text{-}4\text{-}1], [10\text{-}8\text{-}6\text{-}1]\}$

The analysis of the trained models and their forecast errors shows that no significant improvements in comparison to CNNs with three fully-connected layers have been achieved. The forecast errors for all three datasets are consistently worse in comparison to the model with three fully-connected layers. Neither the average error nor the error of the best performing model decreased in this experiment. The forecast performance did, however, not significantly worsen.

Table 4.9: The table shows the forecast errors achieved by networks that contain four fully-connected layers.

dataset		IRE15	IRE40	IRE350
MSE	average	8.17	21.6	449
	best model	6.59	17.9	318

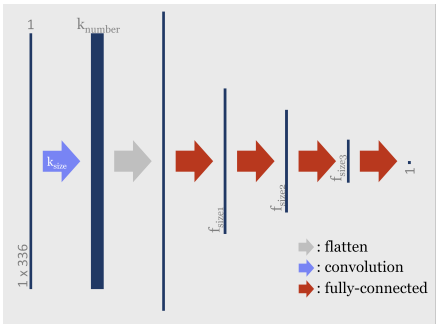


Figure 4.19: The graphic depicts the structure of the CNN model with one convolutional layer and four fully-connected layers.

These results lead to the conclusion that there is no need for the fully-connected part of the neural network model to have more than three layers to be able to compute good forecasts. That deduction is, of course, limited by the used dataset. It is entirely possible that the forecast performance can benefit from a deeper network structure for a substantially larger dataset.
Due to the fact that no improvements of the forecast quality have been made with four fully-connected layers, testing even deeper networks is refrained from. From now on, three fully-connected layers are used in the CNN model, because the performance is at least as good as with four layers. Based on the idea of Occam's Razor, the simpler model is chosen for the time series forecasting model.

4.7.4.4 Adding Dropout to the Neural Network

The next step of improving the load forecast model is to include dropout to the neural network. In the following, first, the concept of dropout is described. Afterwards, different types of dropout are applied and their effect evaluated. The investigated dropout variants include spatial dropout, dropout on the fully-connected part of the NN with fixed and adjusted numbers of neurons, and a combination of the methods.

Dropout was first introduced by Hinton in 2012. They demonstrated that dropout improves the performance of neural networks for many known datasets. [119]
When dropout is applied in neural networks, it means that neurons, together with their connections, are randomly removed - they are dropped temporarily from the network. For each training sample, different neurons are being dropped. Hence, only the weights connected to a fraction of neurons are adjusted during one training step. However, all neurons and weights are used during the test and the application of the NN. The weights are then down-scaled accordingly. [120, 119]
Applying dropout basically equals combining several different, equally weighted models that share the same weights. Averaging several models usually increases the overall performance of machine learning methods. When neurons are dropped from the network, it corresponds to sampling one out of 2^n thinned networks, with n being the total number of neurons in the network. These different NNs are then trained with extensive weight sharing between them, which is a more efficient way of model averaging than training several models independently from each other. Additionally, dropping units reduces complex co-adaptations. Two or more neurons are co-adapted when the output of one neuron can only be utilised in the presence of one or more other neurons. For example, when one unit fixes the error another unit causes. By applying dropout, the neurons must be able to work with and without any other neuron in the NN and, therefore, complex co-adaptations are broken up

and the full capacity of the NN is utilised. This makes the model more robust and decreases the generalisation error. [120, 119]
In a nutshell, both model averaging and breaking up co-adaptations prevent the model from over-fitting and, therefore, enhance the model performance.

In practice, dropout is implemented through a dropout layer. The dropout layer succeeds the layer of which neurons should be dropped and sets the output of a pre-defined fraction p of the total neurons to zero during training. Additionally, the remaining weights are up-scaled by $1/(1-p)$ during training, such that the average sum over all inputs of the subsequent layer stays constant. If neurons of several layers should be subject to dropout, each of those layers must be succeeded by a dropout layer.

Numerous tests of varying variants of dropout with different dropout rates have been conducted as part of the research on time series forecasting with convolutional neural networks. In this section, the different experiments and their results are presented. The kernel size k_{size} and the amount of kernels k_{number} have not been varied for the computations in this section. Instead, they have been fixed to values which have proven to be promising in the previous section. The variation of k_{size}, k_{number}, and f_{size} in the previous section was already computationally expensive. Adding another parameter further multiplies the computational cost and is out of proportion with the gained insights into the behaviour of the CNN under parameter variation. A variation of k_{size}, k_{number}, f_{size}, and the dropout rate d_{rate}, and computing ten iterations of each configuration in order to balance out performance deviations due to the initialisation would take four to five months. Therefore, the kernel size and the amount of kernels are set to:

- $k_{\text{size}} = 6$
- $k_{\text{size}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

The above values have been chosen by considering the findings of computations of CNNs with three fully-connected layers in the section before. The less volatile the data, the better were the results with larger kernel sizes. As the amount of kernels only had a minor influence on the forecast performance, a value in the medium range of the tested parameter is used.
The computations of the following hyper-parameter grid-searches still took up to a week each to complete.

In the neural network model developed thus far, there are two possibilities of where to apply dropout: in the convolutional layer or in the fully-connected layers. In the

fully-connected layers, dropout is applied as described above. In the convolutional layer, however, applying dropout is more tricky, because randomly removing neurons corrupts the filters that are the basis of the convolutional operations. With neurons removed from the filters, they can no longer fulfil their intended task of identifying patterns and changes in patterns over time. Therefore, so-called spatial dropout is applied in convolutional layers. This variation of dropout removes whole filters, respectively feature maps, during training and thereby avoids the mentioned shortcomings.
In order to find the best possible network that works with dropout, the testing of the different dropout types and dropout rates d_{rate} is combined with an additional grid-search regarding the size of the fully-connected layer f_{size}.

Spatial Dropout

First, spatial dropout is tested, which means a portion of the filters are randomly dropped during training. The hyper parameters have been varied in the following range:

- $d_{\text{rate}} \in \{0.1, 0.2, \ldots, 0.9\}$
- $f_{\text{size}} \in \{$[2-2-1], [4-2-1], [6-2-1], [6-4-1], [8-2-1], [8-4-1], [8-6-1], [10-2-1], [10-4-1] [10-6-1], [10-8-1], [12-2-1], [12-4-1], [12-6-1], [12-8-1], [12-10-1]$\}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

Again, ten iterations per setting have been carried out to minimise effects introduced by the initialisation. After conducting preliminary tests, it became clear that a larger number of filters is necessary when spatial dropout is used. This behaviour is not unexpected due to the fact that, for large dropout rates, very few to only one kernel is left to generate the features of which the forecast is composed. Hence, only information about very few or one pattern is extracted from the time series and available for the computation of the forecast. Therefore, k_{number} is adjusted according to the dropout rate:

$$\tilde{k}_{\text{number}} = \frac{k_{\text{number}}}{\lceil 1 - d_{\text{rate}} \rceil} \tag{4.4}$$

with $\lceil\ \rceil$ indicating the ceiling function and $\tilde{k}_{\text{number}}$ the adjusted number of layers based on the applied d_{rate}.

With the adjusted number of kernels, there are always k_{number} kernels trained during each training step independently from the dropout rate. However, the active kernels are chosen randomly for each sample, which avoids co-adaptation.

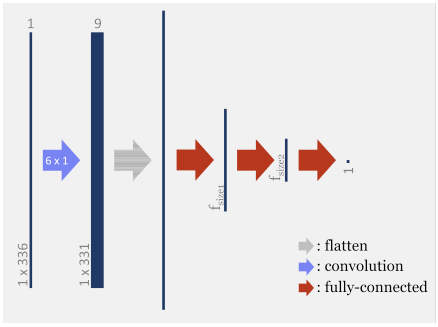


Figure 4.20: The graphic depicts the structure of the CNN model with one convolutional layer and three fully-connected layers. The striped, grey arrow indicates that spatial dropout is applied. The values in the graphic correspond to the model forecasting the IRE350 data.

The analysis of the resulting forecast errors of the different forecast models shows a clear picture. The inclusion of spatial dropout to the CNN model does not improve the forecast quality of the models. The error values displayed in table 4.10 support this conclusion. The MSE of the best trained model is larger for all three datasets than for the training without spatial dropout. The average error values are smaller than in table 4.8 because only the best performing values of k_{size} and k_{number} are included in the grid-search conducted in the context of this paragraph.

Table 4.10: The table shows the forecast errors of CNN models trained with spatial dropout. The forecasts of the best models are less accurate than of models without dropout.

dataset		IRE15	IRE40	IRE350
MSE	average	7.63	22.8	374
	best model	6.95	19.3	338

Dropout on One Fully-Connected Layer

After applying spatial dropout, the influence of dropout on the fully-connected part of the convolutional neural network for time series forecasting is investigated.
For this purpose, the neural network architecture developed in the previous sections

is used (see fig. 4.21. Dropout is, at first, only applied to the first fully-connected layer, because it contains the biggest part of the trainable weights of the whole network. For this purpose, a dropout layer has been added between the feature maps and the output neurons of the first fully-connected layer. The hyper-parameters are varied in the same range as before:

- $d_{\text{rate}} \in \{0.1, 0.2, \ldots, 0.9\}$
- $f_{\text{size}} \in \{[2\text{-}2\text{-}1], [4\text{-}2\text{-}1], [6\text{-}2\text{-}1], [6\text{-}4\text{-}1], [8\text{-}2\text{-}1], [8\text{-}4\text{-}1], [8\text{-}6\text{-}1], [10\text{-}2\text{-}1], [10\text{-}4\text{-}1]$ $[10\text{-}6\text{-}1], [10\text{-}8\text{-}1], [12\text{-}2\text{-}1], [12\text{-}4\text{-}1], [12\text{-}6\text{-}1], [12\text{-}8\text{-}1], [12\text{-}10\text{-}1]\}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

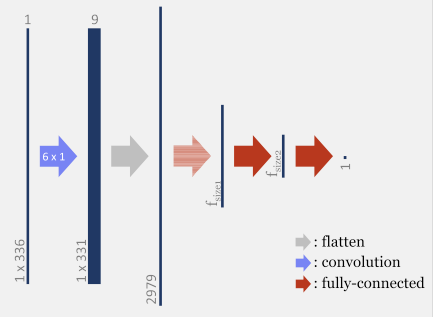


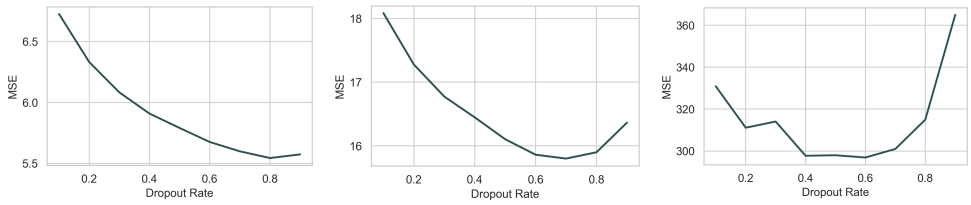
Figure 4.21: The graphic depicts the structure of the CNN model applying dropout in the first fully-connected layer, which is indicated by the striped, red arrow. The values in the graphic correspond to the model forecasting the IRE350 data.

The analysis of the forecast errors when applying dropout shows a major increase in the forecast performance. The inclusion of dropout improves the generated forecasts drastically. The average error of all trained architectures decreases up to 30 % compared to the average of the computations without dropout. Even the forecast performance of the best configuration improves significantly (see table 4.11). The improvement of the best models is not as strong as for the average per dataset. That indicates that adding dropout to the fully-connected layer particularly improves the performance of the worst performing configurations. The great improvements also show that the neural networks did not previously exhaust their full capacity. Apparently, some co-adaptations were developed that are prevented from being formed when dropout is used.

Table 4.11: The table contains the forecast errors of the models trained with dropout applied to the first fully-connected layer. The percentages represent the improvement to the errors achieved with the models without dropout.

dataset		IRE15	IRE40	IRE350
MSE	average	5.91 (27 %)	16.5 (22 %)	314 (30 %)
	best model	5.43 (17 %)	15.5 (7 %)	267 (13 %)

A more detailed analysis of the forecast errors comparing the different settings and the different behaviour of the three datasets illuminates a few findings:
Firstly, concerning the variation in the dropout rate. For the IRE15 and IRE40 dataset, large dropout rates yield the best forecast results. That result is astonishing, because training with a large dropout rate implies that only a very small number of output neurons is active and thus trained during each training step. This shows that even smaller CNNs are capable of providing satisfying forecasts, when they are trained well. Furthermore, it proves that dropout is very successful on noisy data or, respectively, highly volatile data. Both datasets exhibit a high variance. Such data is more prone to over-fitting, as patterns and information are harder to extract. Hence, neural networks are likely to memorise training data instead of extracting meaningful information. Over-fitting becomes less probable with a large dropout rate, since only a limited amount of trainable weights are available. In summary, it is assumed that the advantages of large dropout rates outweigh the disadvantages for highly volatile data. Hence, large dropout rates produce the more accurate forecasts. The forecast error of the less volatile IRE350 dataset reacts slightly different to a varying dropout rate (see fig. 4.22). With an increasing rate, the error first decreases and, after reaching an minimum for $d_{\text{rate}} = 0.5$, increases again. Especially for $d_{\text{rate}} = 0.9$, the forecast error is significantly larger. This indicates that the network is not able to process the extracted information properly with a strongly thinned-out network. In that case, the advantages of avoiding over-fitting and co-adaptation disappear. The poor forecast results when applying very large dropout rates suggest that a NN that is severely thinned out is no longer able to represent, analyse, and process the more complex relations and dependencies between the extracted features from the IRE350 set sufficiently. This also indicates that the CNN is able to extract more information from the time series when it is less volatile.
Secondly, the influence of the size of the first fully-connected layer is analysed with respect to the dropout rate. Figure 4.23 shows not only the forecast error at different dropout rates, but also the errors of a varying number of output neurons of the first fully-connected layer. The size of the first fully-connected layer is of interest because the dropout is applied to this layer. No distinct optimum of the layer sizes can be determined for the IRE15 and IRE40 dataset, however, large layer sizes seem to be slightly favoured. The number of output neurons of the first fully-connected



(a) Error on the IRE15 set. The best performances are achieved with large rates.

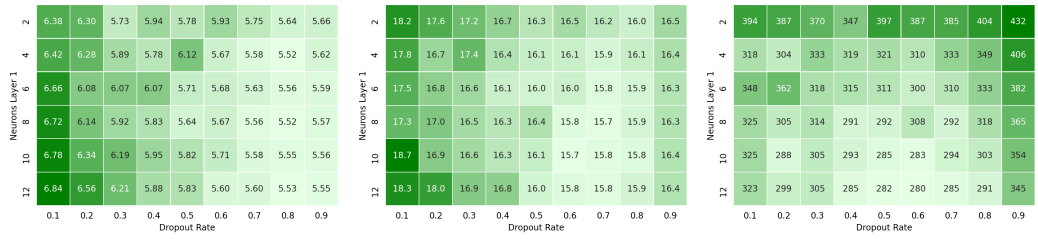
(b) Error on the IRE40 set. The error increases again for $d = 0.9$.

(c) Error on the IRE350 set. The best performances are achieved with medium rates.

Figure 4.22: The plots depict the average forecast error of a CNN with three fully-connected layers when dropout is applied in the first of these layers with respect to the dropout rate.

layer, however, has a discernible influence on the results when predicting the IRE350 dataset. A CNN with only two output neurons in the first layer performs significantly worse than larger networks, regardless of the dropout rate. Hence, very small neural networks do not possess enough variables to process the extracted information properly or to be able to represent the correlations between the features properly. This again supports the assumption that more information can be derived from less volatile time series.
Thirdly, the influence of the size of the entire fully-connected part of the CNN on the forecast performance is examined. The forecast errors shown in figure 4.23 show that the best results for both the IRE15 and the IRE40 dataset have been achieved with the medium sized layers. The forecast quality of the CNN models when forecasting the IRE40 dataset worsens when the CNN contains a rather small or a large fully-connected section. When forecasting the IRE350 dataset, the larger the fully-connected part, the better the forecast performance. Apparently, many trainable weights are necessary to fully describe the correlations between the extracted features and the output of the NN. It seems that an even larger network performs even better. It is, however, refrained from training larger networks without knowing whether the CNN model for the IRE350 dataset requires more trainable parameters in order to perform better or whether the model needs more active neurons during training in order to represent the complex correlations in the data. The decrease in the forecast quality for large dropout rates can be an indication of the second assertion. This question is explored in the next paragraph.

In summary, applying dropout to the output neurons of the first fully-connected layer enhances the forecast performance of convolutional neural network models drastically. This implies that the NNs previously trained have been inhibited by over-fitting and co-adaptation and could, therefore, not develop their full capacity.



(a) Errors of the IRE15 set. **(b)** Errors of the IRE40 set. **(c)** Errors of the IRE350 set.

Figure 4.23: The heatmaps represent the forecast errors with respect to the dropout rate and the number of neurons in the first fully-connected layer.

Medium to large dropout rates have shown the lowest forecast errors. Very large dropout rates perform slightly worse, which possibly indicates that not enough neurons were active during training to ensure an effective training. The most effective rates are very large in comparison to the commonly applied dropout rates, which range between 0.1 and 0.5. However, time series, for which there is no sign in the literature that dropout has hitherto been applied, contain a lot of redundant information in contrast to images. The negative influence of the redundancy is presumably reduced by applying dropout. The best forecasts for the IRE15 and IRE40 set are achieved with CNN models with medium-sized fully-connected sections and for the IRE350 set with large fully-connected sections.

Dropout on One Fully-Connected Layer with Adjusted Layer Size

The size of the fully-connected layers that promise the best forecast results has shifted to larger layers with the addition of dropout. Therefore, in this paragraph, the size of the fully-connected network part is increased according to the dropout rate, to further enhance the forecast performance of the CNN model. The number of output neurons of the first fully-connected layer is adjusted according to the chosen dropout rate. The number of neurons is changed similarly to the earlier paragraph:

$$\tilde{f}_{\text{number\_l1}} = \frac{f_{\text{number\_l1}}}{\lceil 1 - d_{\text{rate}} \rceil} \tag{4.5}$$

with $f_{\text{number\_l1}}$ being the number of output neurons of the first fully-connected layer and $\tilde{f}_{\text{number\_l1}}$ the adjusted number. Thereby, the same number of neurons is always active during training for a fixed $f_{\text{number\_l1}}$, regardless of the applied dropout rate. This also allows for meaningful assertions about whether a change in forecast error is caused by the variation in dropout rate or in the size of the fully-connected section. At the same time, a larger number of neurons, and thereby

trainable weights, are available during training, which has already proven promising in the previous experiments. Additionally, the possibility that only one or very few output neurons of the first layer are active is avoided when using this approach.
The same hyper-parameter range as before has been used for the application of dropout with the adjusted first fully-connected layer:

- $d_{\text{rate}} \in \{0.1, 0.2, \ldots, 0.9\}$
- $f_{\text{size}} \in \{$[2-2-1], [4-2-1], [6-2-1], [6-4-1], [8-2-1], [8-4-1], [8-6-1], [10-2-1], [10-4-1] [10-6-1], [10-8-1], [12-2-1], [12-4-1], [12-6-1], [12-8-1], [12-10-1]$\}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

The evaluation of the trained models and their forecast errors on the test set shows a substantially increased forecast performance in comparison to CNN models without any dropout (see table 4.12). In comparison to the results from the previous paragraph, the new models perform, on average, slightly better than before. The best trained models improved as well, in particular for the IRE40 dataset. That underlines the previous assumption that applying dropout without adjusting the size of the fully-connected layer, on one hand, avoids over-fitting, but, on the other hand, limits the potential of the neural network. The better performance of the best models is presumably the result of compensating this downside of applying dropout by increasing the size of the fully-connected layer. A certain number of active neurons is apparently necessary to extract and process all the information.

Table 4.12: The table contains the forecast errors of the models trained with an adjusted layer size according to the dropout rate. The percentages represent the improvement to the errors achieved without dropout. The errors are consistently lower than for the models with dropout but without adjustment of the layer size.

dataset		IRE15	IRE40	IRE350
MSE	average	5.94 (27 %)	16.4 (23 %)	298 (33 %)
	best model	5.43 (17 %)	15.4 (13 %)	260 (14 %)

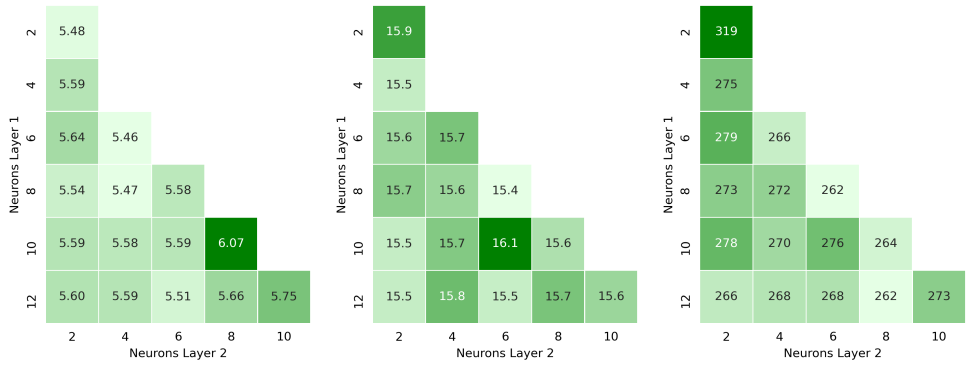
An examination of the influence of the dropout rates on the forecast error reveals that the errors clearly change depending on the dropout rate (see table 4.13). In general, small dropout rates $d_{\text{rate}} < 0.5$ perform worse than larger rates. For the more volatile datasets of IRE15 and IRE40, the error decreases further for even larger rates, but the errors of the IRE350 dataset show a different behaviour. For the less

volatile data, medium dropout rates promise the best performance. In particular, the large forecast error when applying $d_{\text{rate}} = 0.9$ indicates that very large dropout rates do not work as well for less volatile time series.

Table 4.13: The table shows the average MSE with respect to the dropout rate. The error values represent the average of all fully-connected configurations computed with the same dropout rate. The size of the first fully-connected layer has been adjusted according to the dropout rate.
CNN models utilising large dropout rates perform the best.

dataset		IRE15	IRE40	IRE350
	0.1	6.76	18.3	338
	0.2	6.37	17.3	306
	0.3	6.13	16.6	308
	0.4	5.98	16.3	292
dropout rate	0.5	5.85	16.0	279
	0.6	5.72	15.8	273
	0.7	5.61	15.7	275
	0.8	5.51	15.7	281
	0.9	5.55	16.3	326

In order to find the fully-connected configuration that works the best for each dataset, a more in-depth analysis of the forecast errors is done. The configuration that has the lowest MSE on average does not necessarily produce the best forecast model. The MSEs of less successful dropout rates can distort the results. Therefore, an analysis of the error distribution with respect to the dropout rate is necessary.
The best forecasts for the IRE15 set have been made with very large dropout rates, $d_{\text{rate}} \in \{0.7, 0.8, 0.9\}$. The analysis has shown that a dropout rate of $d_{\text{rate}} = 0.7$ produces the consistently best forecasts. $d_{\text{rate}} = 0.8$ displays the lowest error and the best single result is achieved with that rate, however, the training with $d_{\text{rate}} = 0.8$ seems to be less stable, as the variation in the forecast errors is larger than for $d_{\text{rate}} = 0.7$ and configurations that perform worse are seemingly randomly distributed over all tested fully-connected configurations. The training process of $d_{\text{rate}} = 0.9$ appears to be more stable than for $d_{\text{rate}} = 0.8$ but less than for $d_{\text{rate}} = 0.7$. In addition, no single result with MSE < 5.5 has been achieved for $d_{\text{rate}} = 0.9$, which is the case for the two other rates. Therefore, the dropout rate for the IRE15 dataset is chosen to be $d_{\text{rate}} = 0.7$. The forecast errors of the different fully-connected configurations are shown in figure 4.24(a). The best configuration is $f_{\text{size}} = [6\text{-}4\text{-}1]$ with a forecast error of MSE = 5.46.
The changes in the forecast error depending on the variation of the dropout rate are more similar for the IRE40 dataset than for the IRE15 set. CNNs trained with small



(a) Errors of the IRE15 set for $d = 0.7$.

(b) Errors of the IRE40 set for $d = 0.7$.

(c) Errors of the IRE350 set for $d = 0.6$.

Figure 4.24: The heatmaps display the forecast performance of the models with dropout and an adjusted layer size with respect to the number of output neurons of the first and second fully-connected layer. The number of neurons of the first layer represent the amount before the adjustment according to the dropout rate or, respectively, the number of active neurons during training.

dropout rates perform worse than those trained with medium and large dropout rates. The best results are achieved when a dropout rate of $d_{\text{rate}} \in \{0.6, 0.7, 0.8\}$ has been applied. CNNs with a dropout rate of $d_{\text{rate}} = 0.9$, however, are not able to produce forecasts of the same quality. Since there are no discrepancies between the variations of the forecast errors, $d_{\text{rate}} = 0.7$ is chosen as dropout for the IRE40 dataset, because CNNs utilising $d_{\text{rate}} = 0.7$ display the best forecast performance on average. The best fully-connected configuration with that dropout rate according to the forecast error is $f_{\text{size}} = [8\text{-}6\text{-}1]$ (see fig. 4.24(b)), which exhibits a forecast error of MSE = 15.4.

The behaviour of the forecast performance with changing dropout rates is a bit different for the IRE350 dataset. The CNNs trained with small dropout rates perform the worst for the IRE350 set, however, very large dropout rates do not produce the most accurate forecasts either. The neural network models performed the best when medium size dropout rates were applied, $d_{\text{rate}} \in \{0.5, 0.6, 0.7\}$. When examining the forecast errors of the CNNs with the best performing dropout rate of $d_{\text{rate}} = 0.6$, however, it can be noticed that the best results are produced by the models with the largest fully-connected layers (see fig. 4.24(c)). The results show a trend of a decreasing forecast error for an increasing fully-connected part of the CNN. Therefore, additional CNN forecast models with larger f_{size} are trained in order to expand the grid-search and identify the f_{size} that works the best for the IRE350 dataset. However, applying small dropout rates is restrained from, as

models which utilise small dropout rates consistently perform the worst. The hyper-parameter range for determining the best CNN configuration for the IRE350 dataset is adapted to:

- $d_{\text{rate}} \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
- $f_{\text{size}} \in \{$[2-2-1], [4-2-1], [6-2-1], [6-4-1], [8-2-1], [8-4-1], [8-6-1], [10-2-1], [10-4-1] [10-6-1], [10-8-1], [12-2-1], ..., [12-10-1], [14-2-1], ..., [14-12-1], [16-2-1], ..., [16-14-1], [18-2-1], ..., [18-16-1]$\}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = 9$

The evaluation of the forecast errors of the new grid-search reveals that the parameter range was sufficient. Table 4.14 shows that there is, on average, no substantial error differences for the CNNs with 10, 12, 14, 16, and 18 output neurons in the first layer, which indicates that no further performance increases are expected for larger networks. Hence, the performed grid-search has been sufficient in finding a good parameter configuration for the IRE350 CNN model. The most successful dropout rates are again $d_{\text{rate}} \in \{0.5, 0.6, 0.7\}$, with the difference being that the error is now the smallest for $d_{\text{rate}} = 0.7$. Additionally, the analysis of the model performance analysis of these dropout rates exhibits that $d_{\text{rate}} = 0.7$ not only shows the best results on average, but also includes the configuration that produces the most accurate forecast. The fully-connected configuration that performed the best is $f_{\text{size}} =$ [16-6-1], with an error of MSE = 260.

Table 4.14: The table depicts forecast errors with respect to the applied dropout rate (left) and with respect to the number of output neurons of the first fully-connected layer (right).

dropout rate	MSE	output neurons	MSE
		2	304
		4	286
		6	270
0.4	287	8	280
0.5	277	10	268
0.6	275	12	270
0.7	272	14	271
0.8	281	16	268
0.9	327	18	272

Table 4.15: The table shows the forecast errors of the extended grid-search for the IRE350 dataset. In contrast to the earlier conducted grid-search, only models with dropout rates $d \geq 0.4$ have been trained and, hence, are included in the average MSE below.

dataset		IRE15	IRE40	IRE350
MSE	average	5.94	16.4	286
	best model	5.43	15.4	258

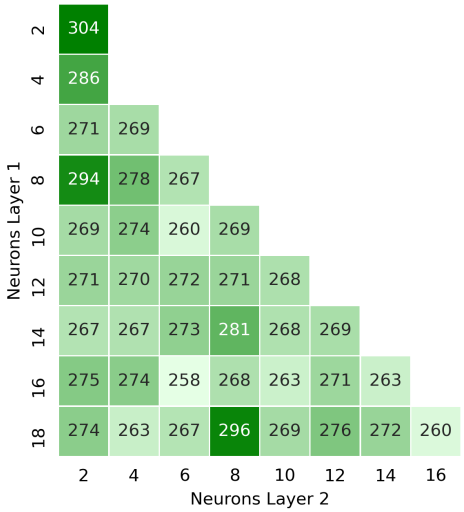


Figure 4.25: The heatmap shows the errors of the model with respect to the number of output neurons in the first and second layer for $d = 0.7$ after the extended grid-search.

Dropout in All Fully-Connected Layers

As applying droput to the first fully-connected layer works well, in this paragraph, it is consequently applied to all layers. The dropout rate of the first layer is fixed to $d_{\text{drop1}} = 0.7$, because it worked the best in the previous experiments. The best f_{size} of the earlier experiments are used as well. The dropout rate of the second fully-connected layer d_{drop2} has been varied similarly to the earlier experiments. The number of ouput neurons for both layers has been adjusted with respect to the individual dropout rate:

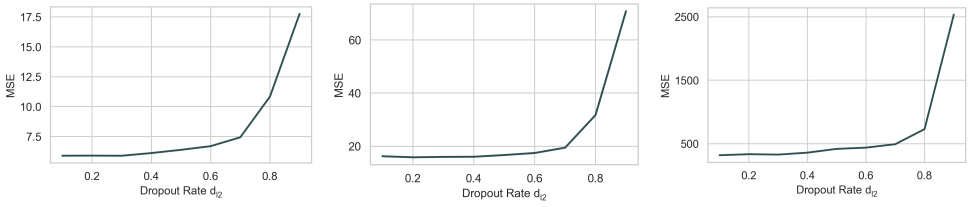
- $d_{\text{rate1}} = 0.7$
- $d_{\text{rate2}} \in \{0.1, 0.2, \ldots 0.9\}$

- $f_{\text{size}} = \begin{cases} \text{[6-4-1] for IRE15} \\ \text{[8-6-1] for IRE40} \\ \text{[16-6-1] for IRE350} \end{cases}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

The forecast errors of the different CNN models with dropout applied in the first and second fully-connected layer show that there are no benefits in applying dropout to more than the first layer of the neural network. None of the models is able to perform better than the forecast models, with dropout in only one layer (see table 4.16). In addition, it can be seen in figure 4.26 that the forecast error increases continuously with an increasing dropout rate. This suggests that the most accurate forecasts can be achieved with no dropout in the second fully-connected layer.

Table 4.16: The table describes the forecast errors achieved with models when dropout is applied to all fully-connected layers. The large average forecast errors are the result of the larger errors for large dropout rates.

dataset		IRE15	IRE40	IRE350
MSE	average	8.11	24.5	661
	best model	5.90	15.8	318



(a) Forecast error of the IRE15 dataset. **(b)** Forecast error of the IRE40 dataset. **(c)** Forecast error of the IRE40 dataset.

Figure 4.26: The plots depict the forecast error with respect to the dropout rate applied in the second fully-connected layer. The error increases drastically with an increasing dropout rate for all datasets.

Why the application of dropout in the whole fully-connected part of the CNN deteriorates the forecast quality can only be speculated. One potential explanation would be that applying dropout rather eliminates critical information instead of avoiding co-adaptations in the network. The first fully-connected layer already condenses the extracted information, so that, probably, hardly any redundancies in the output of the layer exist any more. Thus, when applying dropout in the second layer, the information the layer receives is incomplete and therefore the training is less effective.

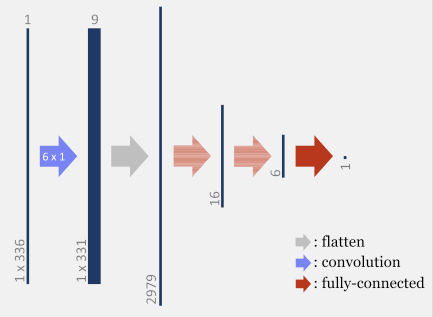


Figure 4.27: The graphic depicts the structure of the CNN model applying dropout in all suitable fully-connected layers, which is indicated by the striped, red arrows. The values in the graphic correspond to the model forecasting the IRE350 data.

As the forecast performance of the models is worse than in the previous paragraph, the concept of applying dropout to all fully-connected layers in the network is discarded.

Combining Spatial Dropout with Dropout in the Fully-Connected Layer

Even though earlier experiments have shown that the inclusion of spatial dropout in the CNN models does not necessarily improve the forecast performance of the models, more tests are conducted in order to verify whether that is still the case in combination with dropout in the fully-connected layer. The parameters of the best performing CNN models using dropout in the first fully-connected layer are utilised. In addition, spatial dropout is applied to the convolutional layer. The best performing dropout rate from the earlier experiments with spatial dropout is chosen as the spatial dropout rate in these computations. k_{number} is adapted accordingly to the rate, as done previously. No parameter search is conducted before the effectiveness of combining the two dropout types has been confirmed.

- $d_{\text{rate}} = 0.7$

- $d_{\text{rate\_spatial}} = \begin{cases} 0.9 \text{ for IRE15} \\ 0.8 \text{ for IRE40} \\ 0.5 \text{ for IRE350} \end{cases}$
- $f_{\text{size}} = \begin{cases} [6\text{-}4\text{-}1] \text{ for IRE15} \\ [8\text{-}6\text{-}1] \text{ for IRE40} \\ [16\text{-}6\text{-}1] \text{ for IRE350} \end{cases}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

The results (see table 4.17) show that, in combination with dropout in the fully-connected layer, the addition of spatial dropout to the CNN model does not improve its ability to create accurate forecasts. This indicates that adding spatial dropout is not beneficial for time series forecasting or the type of data that is used in this thesis.

Table 4.17: The forecast error of the models that use spatial dropout and dropout in the first fully-connected layer. The errors are larger than for models without spatial dropout.

dataset	IRE15	IRE40	IRE350
MSE	6.04	16.1	327

Conclusion

The experiments conducted in this section demonstrate that time series forecasting with convolutional neural networks of data with a large sampling rate strongly benefits from the inclusion of dropout into the network. The experiments have shown that, on one hand, dropout in the fully-connected part of the CNN is highly effective. Adding spatial dropout to the CNN, on the other hand, has only a marginal effect on the forecast perfomance of the CNN model.
Dropout in the fully-connected part of the NN is successfully applied to the first layer. The differences in the forecast results between CNN models with and without an adjusted size of the layer to which dropout is applied indicates that thinning out the network by applying dropout slightly reduces the ability to learn the correlations of the features, which are extracted by the convolutional layer. Hence, increasing the size of the network when applying dropout is crucial for achieving the best results

possible.
Another finding of the experiments is that neither combining spatial dropout with dropout in the first fully-connected layer nor applying dropout to the entire fully-connected part of the CNN model increases the forecast quality. On the contrary, the forecast error increases when more dropout has been introduced to the neural network. Therefore, dropout is only applied to the first fully-connected layer with an adjusted number of output neurons, according to the dropout rate.

The best configurations for each dataset have been determined according to the mean squared error of the 36 hour forecast created. These configurations, which are used from now on as the basis of further research, if not mentioned otherwise, are summarised in table 4.18.

Table 4.18: The table summarises the network configuration for each dataset that generated the most accurate forecasts so far.

dataset	MSE	configuration
IRE15	5.46	$k_{\text{size}} = 6$, $k_{\text{number}} = 5$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [6\text{-}4\text{-}1]$ $\tilde{f}_{\text{size}} = [20\text{-}4\text{-}1]$
IRE40	15.4	$k_{\text{size}} = 6$, $k_{\text{number}} = 7$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [8\text{-}6\text{-}1]$ $\tilde{f}_{\text{size}} = [27\text{-}6\text{-}1]$
IRE350	260	$k_{\text{size}} = 6$, $k_{\text{number}} = 9$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [16\text{-}6\text{-}1]$ $\tilde{f}_{\text{size}} = [54\text{-}6\text{-}1]$

The best network configurations and the performances of different dropout rates indicate that CNN models applied to data with a large sampling rate benefit the most when the dropout rate is rather large. The lowest forecast errors in the experiments were achieved with large rates of $d_{\text{rate}} \in \{0.6, 0.7, 0.8\}$. In established networks and most of the publications that include experiments with dropout, the chosen rate is mostly smaller ($d_{\text{rate}} \in [0.1, 0.5]$). It seems dropout is crucial for good results when working with time series that have a large sampling rate. A possible explanation is that the time series include a lot of redundant information, due to the large sampling rate. The values of adjacent data points often barely differ from each other, which means, in return, that they contain similar information. By applying a large dropout rate, in addition to avoiding co-adaptation the probability of processing redundant information in a training step is severely reduced. This enhances the forecast quality, because less over-fitting takes place. When data with redundant information is presented to a NN, the NN learns the noise of the data or, respectively, memorises

the training data instead of learning meaningful correlations. This happens because the network has too many trainable weights at its disposal, due to the fact that the input neurons of the layer, which represents potentially redundant features, are connected to each output neuron of the first layer. That is presumably the reason why large dropout rates work well for the used datasets. However, very large dropout rates $d_{\text{rate}} > 0.8$ have an opposing effect on the forecast quality. This is probably because correlations between the input are hardly learnt, since it is very unlikely that the same neurons are regularly active together. Another reason is the possible large temporal distance between two active input neurons, which can result in the trend of the data not being properly represented. For similar data with a higher temporal resolution, larger dropout rates would probably work better. The best results were eventually achieved with a dropout rate of $d_{\text{rate}} = 0.7$ for all datasets.

When examining the best configuration for each dataset (see table 4.18), it is noticeable that the size of the fully-connected part of the CNN model increases with a decreasing volatility of the datasets. The necessity of a larger f_{size} indicates that more useful information can be extracted and, hence, must be processed by the fully-connected layers. This coincides with earlier findings that the neural network is able to derive more information from a less volatile time series. In general, it can be assumed that predicting a load time series of large aggregation level is easier than of smaller aggregation level, because the volatility is smaller; which, on the one hand, allows more information to be extracted and, on the other hand, reduces the influence of single households on the time series. This means, in return, that the time series is less susceptible to interferences caused by one household.

The configurations determined in this section are used for every CNN model from now on, if not mentioned otherwise. Naturally, the question of if the number of training epochs e is still sufficient for the training with dropout was also examined. Those tests concluded that training the chosen neural network models with $e = 40$ epochs is still adequate.

4.7.4.5 Adding Pooling Layers

A common component in established convolutional neural networks that process two-dimensional data are pooling layers [95, 121]. They serve two purposes - to reduce the dimensionality of the data and to make the CNN model less sensitive to small translations of patterns. A pooling layer typically consists of a pooling operation that transforms several input values to one output value. The most common pooling operations are max-pooling, where the output is the maximum of the

input values, and average-pooling, which computes the average of the input values.

That CNN models with large dropout rates have produced the best forecasting results in the previous paragraph suggests that the data contains redundant information. A detailed discussion can be found in the previous paragraph. In this paragraph, whether that redundant information can also be removed by the addition of a pooling layer to the CNN model and whether, thereby, the forecast accuracy increases is investigated. The assumption is that, by reducing the output dimensionality of the convolutional layer, less over-fitting occurs and, thus, the forecast performance of the model increases.
The pooling layer is placed subsequent to the convolutional layer and is, therefore, applied to the extracted feature maps. Depending on the configuration of the pooling layer, the size of the resulting, newly computed feature maps decreases and, therefore, the input dimensionality of the fully-connected part of the network is reduced. In addition to the type of the pooling operation, pooling layers can vary in the size of the pooling window and in stride size or increment. The stride size specifies the step size with which the pooling window samples the feature map of the convolutional layer. It should not be larger than the size of the pooling window itself, because, otherwise, some data points would be neglected and not be processed at all. Executing a pooling operation and reducing the dimensionality also involves the risk that information will get lost during the process. The intention of the following experiment is to ascertain whether the benefits or the drawbacks of adding a pooling layer to a CNN model for time forecasting dominate.

In order to determine the effectiveness of a pooling layer in a CNN model, a grid-search that varies the size of the pooling window p_{size} and the stride size p_{stride} is conducted. Additionally, two types of pooling operations are assessed: average pooling and max pooling. The setup of the fully-connected layer is chosen to be the best performing setup from the previous paragraph, that analysed dropout, but dropout will not be applied in this experiment. The assumption is that these setups have proven to produce good results for the used datasets. If the pooling operations are able to minimise the amount of redundant information, the forecast error should be similar to models with dropout. If that is the case, the size of the fully-connected part will be increased in a following step, to check if pooling offers even more potential. Additionally, varying f_{size} would result in a unreasonable computational load as well. The neural networks do not include any dropout and, therefore, the number of output neurons of the first fully-connected layer is not changed. Dropout is not applied, as the experiment is intended to show whether pooling operations have a similar effect to dropout layers.
The parameter variations of the grid-search are the following:

- $p_{\text{size}} \in \{2, 4, 6, 8\}$
- $p_{\text{stride}} \in \{1, 2, 4, 6, 8\}$ if $p_{\text{size}} \geq p_{\text{stride}}$
- $p_{\text{type}} \in \{$ max pooling, average pooling $\}$
- $f_{\text{size}} = \begin{cases} \text{[6-4-1] for IRE15} \\ \text{[8-6-1] for IRE40} \\ \text{[16-6-1] for IRE350} \end{cases}$
- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$

with p_{type} being the type of pooling operation applied.

The forecast errors of the CNN models trained with a pooling layer following the fully-connected layer (see table 4.19) are larger than those of the networks trained with dropout. That indicates that adding an average or max pooling layer subsequent to the convolutional layer does not have the desired effect of reducing the impact of the existing information redundancy in the data. The forecast errors are consistently larger than from models utilising dropout. A comparison with the earlier experiment of computing CNNs with three fully-connected layers and no dropout even suggests that adding the pooling layers has no positive influence on the forecast performance of the CNNs at all. The average errors and the errors of the best models are both as large or larger than the comparable errors produced by networks with no dropout and no pooling (see table 4.8 for comparison). Consequently, the approach of adding a pooling layer between the convolutional layer and the fully-connected layer of the convolutional neural network is not pursued further.

Table 4.19: The table presents the forecasting errors achieved with CNN models that apply a pooling operation subsequent to the convolution. The top part represents the errors when average pooling is applied, the bottom when max pooling is applied.

dataset		IRE15	IRE40	IRE350
MSE	average	7.30	20.3	399
	best model	6.08	16.8	341
MSE	average	7.94	19.3	409
	best model	7.06	17.4	339

4.7.4.6 Variation in Stride Sizes

The previous paragraph about pooling layers mentions the stride size as a hyper-parameter of pooling layers. The possibility of changing the stride size is not exclusive to pooling layers. Convolutional layers and pooling layers both process their input data incrementally. Therefore, it is also possible to adjust the stride size of a convolutional layer. It is common for the deep convolutional layers within a CNN to have a stride size $k_{\text{stride}} \neq 1$. The stride size of the input layer, however, is typically set to $k_{\text{stride}} = 1$. Nevertheless, an experiment is conducted in order to investigate whether changing the stride size to larger values influences the forecast performance of the CNN model. The remaining hyper-parameters of the convolutional layer as well as of the fully-connected layers are kept the same as in the most successful network.

- $k_{\text{size}} = 6$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$
- $k_{\text{stride}} \in \{2, 4, 6\}$
- $f_{\text{size}} = \begin{cases} \text{[6-4-1] for IRE15} \\ \text{[8-6-1] for IRE40} \\ \text{[16-6-1] for IRE350} \end{cases}$

No k_{stride} values larger than six are investigated, as the stride size would then be larger than the kernel size. With larger stride sizes, some input values of the time series would be omitted.

The analysis of the results shows that applying larger stride sizes to the input layer of the CNN does not result in more accurate forecasts in comparison to the most successful forecast models. The results also show that the forecast quality deteriorates with an increasing stride size, which is a further indication that a stride size of $k_{\text{stride}} = 1$ is favourable in the first convolutional layer. Therefore, the stride size of the first layer is always set to one in the following experiments.

4.7.4.7 Adding more Convolutional Layers

In this section, additional convolutional layers are added subsequent to the current layers in place. First, one layer and, afterwards, two layers are added, and the

Table 4.20: The forecast errors of CNN models with stride sizes of the convolutional layer larger than one.

dataset		IRE15	IRE40	IRE350
MSE	average	5.92	15.9	367
	best model	5.81	15.6	314

influence on the forecast performance is examined.

The experiments discussed in the previous sections were mostly focussed on optimising the fully-connected part of the convolutional neural network and its input. That is due to the early observation that variations in the kernel size as well as in the amount of kernels have a smaller influence on the forecast performance on the models. The changes in the forecast error and, thus, the potential for improvement have been larger when the configuration of the fully-connected layers is varied. Therefore, the configuration of the fully-connected part has been optimised first.

Another possibility of potentially improving the forecast performance of the CNN model is adding more convolutional layers to the neural network, hence creating a deeper network. A network with more convolutional layers is capable of recognising more complex and larger patterns. With every layer, the values of the feature maps, which represent the presence of a pattern, can be combined to more sophisticated patterns. This property of CNNs is the reason for their success in image and pattern recognition. Furthermore, the receptive field is enlarged by adding more convolutional layers. The receptive field indicates how many input values contribute to a value in an activation map (see fig. 4.28). With a larger receptive field, larger patterns and correlations can be represented in this activation map. In order to analyse whether additional convolutional layers enhance the forecast performance of the CNN models for time series forecasting, first, one convolutional layer is added to the CNN model established in the previous section (see fig. 4.30(a)). The kernel size and the amount of kernels of the additional layer are varied.
The best results with CNN models are usually achieved with networks where the number of filters successively increases with the depth of the network. Most of the successful CNN models are structured in that way [115, 122]. Therefore, the amount of kernels of the second layer k_{number2} is chosen to be at least equal to the first layer k_{number1}. For a better comparability between the three datasets and for a good coverage of the search space, k_{number2} is chosen to be a multiple of k_{number1}. Additionally, a stride size $k_{\text{stride2}} \geq 1$ is also tested for the kernels in the second convolutional layer. Even though changing the stride size in the first convolutional layer has not improved the forecast quality (see section 4.7.4.6), it is tested for the

second convolutional layer again. Increasing the stride size reduces the amount of additionally added trainable weights, thus reducing the possibility of over-fitting due to an abundant number of weights in the neural network.

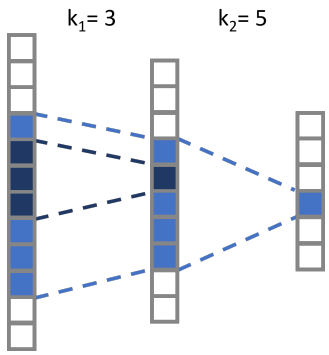


Figure 4.28: This graphic illustrates the receptive fields of two values in consecutive activation maps. The values (grey boxes) on the left depict the input series and the k_i the kernel size of the layers. The dark blue value in the first feature map is computed of the dark blue values of the input series. Its receptive field corresponds to its kernel size. The light blue value in the 3rd layer is computed of the light blue values of the 2nd layer. Its receptive field corresponds to the blue value in the 1st layer, hence is larger than its kernel size.

The configuration of the fully-connected layers is retained from the earlier experiments. If the forecast errors with an additional convolutional layer are the same or better than with one layer, the best convolutional configuration will be kept and a further variation of the fully-connected layers will be conducted. Varying both parts of the network at the same time would otherwise result in a very large computational load. The grid-search conducted with the parameter ranges below already includes the training of more than a thousand neural networks.

- $k_{\text{size1}} = 6$
- $k_{\text{number1}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$
- $k_{\text{size2}} \in \{4, 6, 8, 10, 12\}$

- $k_{\text{number2}} \in \begin{cases} \{5, 10, 15, 20\} \text{ for IRE15} \\ \{7, 14, 21, 28\} \text{ for IRE40} \\ \{9, 18, 27, 36\} \text{ for IRE350} \end{cases}$
- $k_{\text{stride2}} \in \{1, 2, 4\}$
- $f_{\text{size}} = \begin{cases} \text{[6-4-1] for IRE15} \\ \text{[8-6-1] for IRE40} \\ \text{[16-6-1] for IRE350} \end{cases}$

The resulting forecast errors of the best trained CNN models with two convolutional layers show that a deeper convolutional network slightly enhances the information extraction from the time series. The forecast errors (see table 4.21) of IRE15 and IRE350 dataset are similar to the errors when one convolutional layer is used (compare with table 4.15). The forecast error of the best model for the IRE40 dataset shows a slight decrease of about 4 %.

Table 4.21: The forecast errors of the models containing two consecutive convolutional layers are displayed.

dataset		IRE15	IRE40	IRE350
MSE	average	5.58	15.4	265
	best model	5.41	14.9	251

The influence of the stride size on the forecast performance decreases with the aggregation level. For the IRE15 datasets, models with $k_{\text{stride2}} > 1$ produce significantly better forecasts than $k_{\text{stride2}} = 1$. The differences in the forecast error diminish for the IRE40 dataset and are no longer existent for the less volatile IRE350 dataset. The differences in the amount of trainable weights probably cause this behaviour. A doubling of the stride size results in half the values in the final feature maps of the convolutional part of the network, which constitute the inputs for the first fully-connected layer. A reduction in the number of inputs, therefore, results in a reduction in the number of weights by a multitude. Fewer weights means a smaller probability of over-fitting. As less meaningful information can probably be extracted from the more volatile time series, the CNN model over-fits more easily because the abundant weights are then utilised to memorise the dataset, which decreases the generalisation ability of the model; hence, the forecasts of the test set are less accurate. For all three datasets, the forecast errors have been smaller in the average and the respective best configuration with a stride size $k_{\text{stride2}} > 1$.
It seems that for a stride size of $k_{\text{stride2}} = 4$, another effect occurs. For small filter sizes k_{size2}, the forecast errors are larger than for larger filter sizes. The errors indicate that the information gets lost when the chosen size of the convolutional layer is

too small. The errors are then larger than with only one convolutional layer. Hence, there exists a fine line between choosing a size of the second convolutional layer that is too small or too large, which must be taken into account when this CNN model is adapted to another dataset.
The choice of the kernel size in the second convolutional layer, however, seems to have only a minor effect on the model performance. It was expected that a larger kernel size, which results in a large receptive field, would be able to provide more context in the second convolutional layer. However, the results do not confirm this assumption.

Table 4.22: The table shows the configurations of the models that computed the most accurate forecasts with the corresponding forecast error.

dataset	MSE	configuration
IRE15	5.41	$k_{\text{size1}} = 6$, $k_{\text{number1}} = 9$, $k_{\text{strides1}} = 1$, $k_{\text{size2}} = 8$, $k_{\text{number2}} = 15$, $k_{\text{strides2}} = 4$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [6\text{-}4\text{-}1]$, $\tilde{f}_{\text{size}} = [20\text{-}4\text{-}1]$
IRE40	14.9	$k_{\text{size}} = 6$, $k_{\text{number}} = 9$, $k_{\text{strides1}} = 1$, $k_{\text{size2}} = 12$, $k_{\text{number2}} = 21$, $k_{\text{strides2}} = 2$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [8\text{-}6\text{-}1]$, $\tilde{f}_{\text{size}} = [27\text{-}6\text{-}1]$
IRE350	251	$k_{\text{size}} = 6$, $k_{\text{number}} = 9$, $k_{\text{strides1}} = 1$, $k_{\text{size2}} = 12$, $k_{\text{number2}} = 36$, $k_{\text{strides2}} = 2$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [16\text{-}6\text{-}1]$, $\tilde{f}_{\text{size}} = [54\text{-}6\text{-}1]$

More than one consecutive convolutional layer allows the addition of pooling layers between the convolutional layers. Preliminary tests, however, have not shown promising results. In all tests, the forecast error deteriorates when pooling layers are added between the two convolutional layers. Therefore, investigating the influence of pooling layers any further has been refrained from.

As the forecast error could be decreased by adding a second convolutional layer to the neural network model, naturally, it is also investigated whether the forecast performance can be further enhanced by using even deeper CNNs for the computation of the predictions. In the next experiment, a third convolutional layer is added to the model (see fig. 4.30(b)). The amount of filters in the third layer k_{number3} is, again, a multiple of the amount in the previous layer. Different kernel sizes k_{size3} and strides k_{stride3} are tested. Only strides larger than one are used in an effort not to increase the amount of data points in the feature map too much. The addition of one more input value to the layer creates connections to all the output neurons, hence adding several trainable weights and, therefore, increases the amount of trainable parameters, which in return increases the chance of over-fitting.

(a) Forecast errors of the IRE15 dataset.

(b) Forecast errors of the IRE40 dataset.

(c) Forecast errors of the IRE350 dataset.

Figure 4.29: The heatmaps depict the forecast errors with respect to the stride size and the kernel size of the new, additional convolutional layer. The error values of the best networks cannot be found, because the depicted values represent the average of all configurations that have the respective parameters.

The hyper-parameters of the conducted grid search are the following:

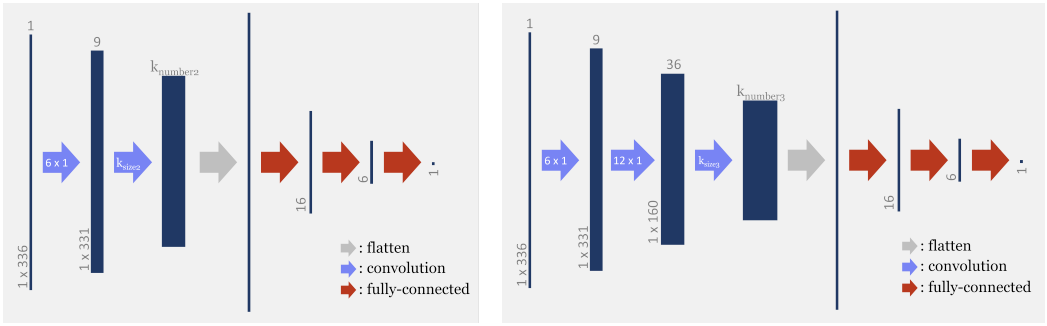
- $k_{\text{size1}} = 6$ $\quad k_{\text{number1}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$ $\quad k_{\text{stride1}} = 1$
- $k_{\text{size2}} = \begin{cases} 8 \text{ for IRE15} \\ 12 \text{ for IRE40} \\ 12 \text{ for IRE350} \end{cases}$ $\quad k_{\text{number2}} = \begin{cases} 15 \text{ for IRE15} \\ 21 \text{ for IRE40} \\ 36 \text{ for IRE350} \end{cases}$ $\quad k_{\text{stride2}} = \begin{cases} 4 \text{ for IRE15} \\ 2 \text{ for IRE40} \\ 2 \text{ for IRE350} \end{cases}$
- $k_{\text{size3}} \in \{4, 8, 12, 16\}$ $\quad k_{\text{number3}} \in \begin{cases} \{15, 30\} \text{ for IRE15} \\ \{21, 42\} \text{ for IRE40} \\ \{36, 72\} \text{ for IRE350} \end{cases}$ $\quad k_{\text{stride3}} \in \{2, 4\}$
- $f_{\text{size}} = \begin{cases} \text{[6-4-1] for IRE15} \\ \text{[8-6-1] for IRE40} \\ \text{[16-6-1] for IRE350} \end{cases}$

The resulting forecast errors of this experiment suggest that the addition of a third convolutional layer is not beneficial to the forecasting abilities of the CNN model. The errors (see table 4.23) decreased or are similar in comparison to the errors achieved with a model consisting of two convolutional layers. Therefore, no deeper models have been tested as it is assumed that the addition of more layers only worsens the forecast performance of the models.

Table 4.23: The forecast errors of the CNN model containing three consecutive convolutional layers.

dataset		IRE15	IRE40	IRE350
MSE	average	5.94	15.3	267
	best model	5.79	15.0	257

For the best network configurations of both model architectures with two and three convolutional layers, it has additionally been tested whether a variation in the size of the fully-connected network part has the potential to improve the forecast performance of the changed CNN. The results, however, show that the configuration of the fully-connected part derived in 4.7.4.4 still provides the most accurate forecasts, also for a deeper network.



(a) The CNN model with two consecutive convolutional layers.

(b) The CNN model with three consecutive convolutional layers.

Figure 4.30: The graphics depict the structures of CNN models with a varying number of convolutional layers. The values in the graphic correspond to the model forecasting the IRE350 data.

In conclusion, it is noticeable that one additional convolutional layer, and hence a deeper network, does not deteriorate the forecast quality produced by the CNN model. The forecast error improvements seen for the IRE40 dataset indicate, on the contrary, that a deeper network offers the potential to extract more information. It also became obvious, however, that choosing an appropriate model configuration is more critical to the forecast performance of deeper networks, at least for datasets of a size comparable to the Irish Smart Meter Trial.
Adding even more convolutional layers, however, does not improve the forecast quality further. Therefore, the models with two convolutional layers are the most promising models to create good electricity load forecasts.
Table 4.22 presents the network configurations that produced the best result.

4.7.4.8 Influence of Dilated Kernels

Another possibility to adjust the convolutional layer in order to improve the forecast performance of the CNN model is to use dilated kernels. Usually, the kernels of a convolutional layer process k_{size} neighbouring inputs. The computed outputs from each convolutional step compose the feature map. In the feature maps, the spatial and temporal order of the input is still preserved. A kernel of a dilated convolutional layer, in contrast to a normal kernel, does not process neighbouring inputs. Instead, the values the kernel uses for the computation are uniformly spaced inputs with a fixed distance between each other. The distance to the next used input value is called dil_{rate} as skipping inputs dilates the time span the kernels operate on. The time span is enlarged by the factor of the dilation rate. The formula of the one-dimensional convolutional operation (see sec. 3.3.3) changes with dilation to:

$$(F * w)(n) = \sum_{s+d \cdot t = n} F(s)w(t) \tag{4.6}$$

with the dilation rate shortened to $d = dil_{\text{rate}}$.

Dilated convolutional layers were first introduced by Fisher et.al in 2016 [123] for image segmentation. When using several layers of dilated convolutions, the receptive field enlarges and more context of the neighbouring areas is available. The term receptive field is derived from the visual system and describes the size of the model input the computation of a feature is based on. The term is only used for convolutional layers, as fully-connected layers have the maximum receptive field by definition and their input is not ordered. Naturally, the receptive field enlarges with each convolutional layer in the neural network. It extends with an increasing kernel size and an increasing stride size as well. The stride size of dilated kernels, however, is fixed to $k_{\text{stride}} = 1$, because otherwise some input values are potentially not processed, depending on the combination of dilation rate and stride size.
The purpose of using dilated convolutions in CNN models is, on the one hand, to increase the receptive field and, on the other hand, to decrease the amount of redundant information one kernel receives. As discussed earlier, it can often be assumed that temporal neighbouring values of the input time series contain redundant information. That is particularly the case for time series with a large sampling rate and, thus, a high resolution. If kernels would not perceive neighbouring data points and use them for the computation of their output, less redundant information would be processed by the kernels. This might increase the performance of the CNN forecast models.

Several CNN models of varying configurations are trained in order to study the influence of dilated kernels in the convolutional layer on the forecast performance.

As a first step, CNN models with only one convolutional layer are used to assess the changes in forecast performance when applying dilated kernels. The best networks from section 4.7.4.4 (see also table 4.18) are used as the basis for the networks with dilations. The dilation rate dil_{rate} and the kernel size k_{size} are varied in the range described below. The kernel size is not fixed to the previously determined value because, with the change of the dilation rate, a kernel processes a different time span and $k_{\text{size}} = 6$ might not be optimal.

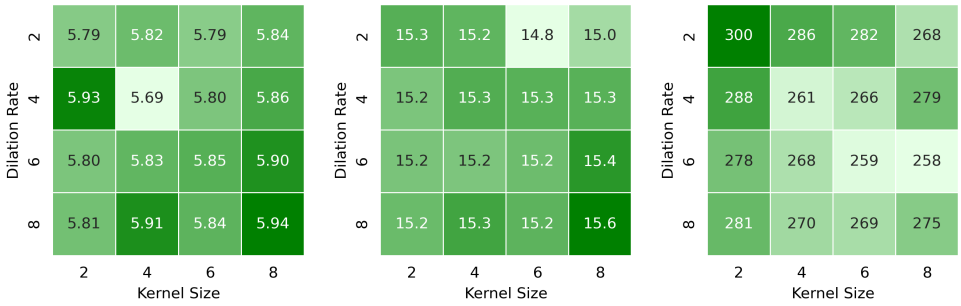
- $k_{\text{size}} \in \{2, 4, 6, 8\}$
- $k_{\text{number}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$
- $dil_{\text{rate}} \in \{2, 4, 6, 8\}$
- $f_{\text{size}} = \begin{cases} [6\text{-}4\text{-}1] \text{ for IRE15} \\ [8\text{-}6\text{-}1] \text{ for IRE40} \\ [16\text{-}6\text{-}1] \text{ for IRE350} \end{cases}$

The results of this experiment are mixed. While the best forecast errors for the IRE15 and IRE350 dataset remain similar to the best errors achieved without dilations, the forecast error of the IRE40 dataset has improved. This indicates that more information can be extracted with a larger receptive field of the kernels. This observation is in line with the results from the previous paragraph, in which networks with two convolutional layers (without dilation) were examined. Forecasts of the IRE40 dataset exhibited the largest improvement when a second convolutional layer was added and, thus, the receptive field is enlarged. This means a large receptive field is desirable and can improve the forecast quality.

Table 4.24: The errors of CNN models with dilated kernels used in their convolutional layer.

dataset		IRE15	IRE40	IRE350
MSE	average	5.84	15.2	274
	best model	5.70	14.8	258

An analysis of the results shows that the best results are achieved with medium kernel sizes of $k_{\text{size}} \in \{4, 6\}$. With increasing kernel size, the forecast performance decreases, in particular for the IRE15 and IRE40 data and independently from the dilation rate. No strong influence of the dilation rates can be recognised in the results. That, combined with the fact that models without dilation perform the same or slightly better, leads to the conclusion that the inclusion of dilations does not



(a) IRE15 **(b)** IRE40 **(c)** IRE350

Figure 4.31: The heatmap shows the forecast errors of the models with respect to the kernel size and the applied dilation rate. No clear preference for a dilation rate is recognisable.

improve the forecast performance of the CNN models on electricity load time series. The enhanced performance on the IRE40 dataset is limited to exactly one configuration (see fig. 4.31) and can hardly be explained.

The previous section has shown that CNN models that contain two convolutional layers are better suited for electricity load forecasting, probably due to the increased receptive field. The next experiment examines whether a further enlargement of the receptive field by using dilated kernels in the second convolutional layer further decreases the forecast errors.
For this purpose, several CNN models are trained which differ in kernel size k_{size2} and dilation rate dil_{rate2} of the second convolutional layer. The amount of kernels k_{number2} in the second layer are kept at the optimal value determined in section 4.7.4.7, assuming that the amount of extracted information does not drastically increase, which would require more kernels. The range of the hyper-parameter values assumed in this experiment is listed below:

- $k_{\text{size1}} = 6$
- $k_{\text{number1}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$
- $k_{\text{size2}} \in \{6, 8, 10, 12, 14\}$
- $k_{\text{number2}} = \begin{cases} 15 \text{ for IRE15} \\ 21 \text{ for IRE40} \\ 36 \text{ for IRE350} \end{cases}$

- $dil_{\text{rate2}} \in \{2, 4, 6, 8\}$
- $f_{\text{size}} = \begin{cases} \text{[6-4-1] for IRE15} \\ \text{[8-6-1] for IRE40} \\ \text{[16-6-1] for IRE350} \end{cases}$

The analysis of the forecast errors obtained during the experiment indicates that dilated kernels in the second convolutional layer have no positive influence on the forecast performance of the CNN models. Throughout all three datasets, the forecast performance of the best model decreased between 7 % and 3 % in comparison to the models with two convolutional layers, but without dilation.

Table 4.25: The table presents the forecast errors achieved with CNN models that contain two convolutional layers with the second layer consisting of dilated kernels.

dataset		IRE15	IRE40	IRE350
MSE	average	5.96	16.0	267
	best model	5.83	15.5	259

Both experiments - using dilated kernels in a CNN model with one and with two convolutional layers - resulted in a worsened forecast performance in comparison to the respective models without dilation. This indicates that using dilated kernels in time series forecasting is not beneficial. It does not seem to be the right tool to address the problem of redundant information in the data. Therefore, dilated kernels are no longer applied.

4.7.4.9 Adding a Parallel Convolutional Layer

In the previous paragraphs, the influence of several hyper-parameters of the CNN model on the forecast quality have been examined and the parameter values have been tuned in order to obtain an accurate forecast. However, the fundamental model architecture has been kept identical for all experiments; only the depth of the model has been varied. Section 4.7.4.7 has shown that the CNN models perform best with a shallow convolutional part of the network. This may partly be caused by the limited dataset size. With data of several years available, it is possible that models with more convolutional layers perform better. The shallow convolutional part of the NN combined with the small kernel sizes and no dilation makes the receptive field of the convolutional layers very small. That means each value of the final feature map is computed based only on a small part of the input time series. As a result, the features cannot represent large-scale interrelationships within the input load time

series. A model with a larger receptive field should more easily be able to detect those relationships.
In the previous section, attempts have already been made to compensate for the disadvantages of small kernel sizes by using dilated kernels. However, no forecast improvements have been achieved. The drawback of the small receptive field of the convolutional layer is addressed in the following by again adding another set of larger kernels to the neural network, which operate independently on the input time series.

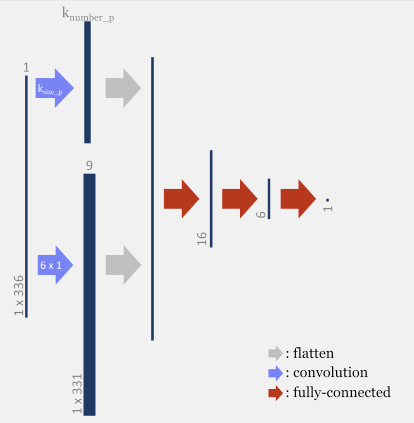


Figure 4.32: The graphic depicts the structure of the CNN model with two parallel convolutional layers, which both operate independently on the input series. The values in the graphic correspond to the model forecasting the IRE350 data.

The new, secondary convolutional layer, therefore, operates parallel to the existing layer (see fig. 4.32). The idea is that the parallel layer extracts information about trends and long time relations between the inputs. This has not been possible with the previously applied network architecture. In order to enable the layer to fulfil this function, it requires a large receptive field. This can be achieved by composing the layer of large kernels. However, in order to not raise the number of weights too much, dilated kernels are applied in the parallel convolutional layer. Furthermore, only a small amount of kernels is used in the new layer. The resulting feature maps are treated equally to the feature maps of the other convolutional layer. Hence, the values of all feature maps constitute the input of the first fully-connected layer and are, therefore, equally subject to the dropout applied in that layer. The dropout rate is kept at $d_{\text{rate}} = 0.7$. Additionally, the size and the amount of kernels is varied, as is the dilation rate. The different configurations have a receptive field between six hours (with $k_{\text{size\_p}} = 12$ combined with $dil_{\text{rate\_p}} = 1$) and 94.5 hours (with

$k_{\text{size\_p}} = 48$ combined with $dil_{\text{rate\_p}} = 4$). The receptive field of the primary layer, on the other hand, is only three hours.
The parameters of the grid search are listed below. The remaining network is configured according to the best network derived in paragraph 4.7.4.4. It was decided not to use the CNN model with two consecutive convolutional layers, even though they have a better forecast performance, for the simple reason that the second layer in those networks fulfils the same function of enlarging the receptive field as the newly added parallel layer. Therefore, the simpler model architecture derived in section 4.7.4.4 is resorted to. This allows a comparison of which method works more effectively.

- $k_{\text{size1}} = 6$
- $k_{\text{number1}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$
- $f_{\text{size}} = \begin{cases} [6\text{-}4\text{-}1] \text{ for IRE15} \\ [8\text{-}6\text{-}1] \text{ for IRE40} \\ [16\text{-}6\text{-}1] \text{ for IRE350} \end{cases}$
- $k_{\text{size\_p}} \in \{12, 24, 48\}$
- $k_{\text{number\_p}} \in \{2, 4, 6, 8\}$
- $dil_{\text{rate\_p}} \in \{1, 2, 4\}$

$k_{\text{size\_p}}$, $k_{\text{number\_p}}$, and $dil_{\text{rate\_p}}$ describe the parameters kernel size, amount of kernels and dilation rate of the parallel convolutional layer.

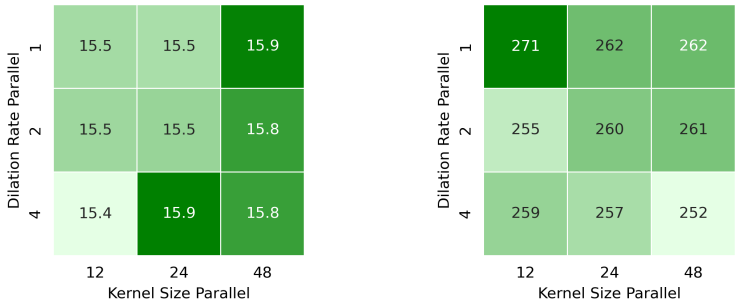
The analysis of the computed forecasts by the CNN model with two parallel convolutional layers shows positive results. In general, it is ascertained that the parallel layer fulfils its purpose of improving the forecasts by adding long-range information extracted by large kernels.
The forecast error on the IRE15 dataset of the best tested configuration is a little bit worse, but in a similar range, than for the CNN model with only one CNN layer. The fact that only minor changes in the forecast error between the models with one convolutional layer, two subsequent convolutional layers, three convolutional layers, and two parallel layers can be observed, indicates that purely data-driven improvements are hardly possible any more for the IRE15 dataset. The data driven information extraction seems to be exhausted, at least with a CNN model. The slightly larger forecast error is then probably caused by over-fitting, as the addition of the parallel layer increases the amount of weights in the first fully-connected layer

significantly.
For the IRE40 and IRE350 dataset, the newly added parallel convolutional layer improves the forecast quality of the CNN model with respect to the model with only one convolutional layer. For the IRE350 set, the error is even smaller than for the model with two consecutive convolutional layers. Increasing the receptive field improves the forecast performance, in particular for the IRE350 dataset (see fig. 4.33). Using even larger dilation rates or kernel sizes is, however, not expedient, since the receptive field then enlarges to almost the size of the model input series, which is one week. This would reduce the idea of the convolutional operations to absurdity. For the IRE40 and also the IRE15 set, models with a small receptive field seem to perform the best. This might be due to the volatility of the series, as the large-range patterns interfere with the spikes due to the volatility.

Table 4.26: The table shows the configurations of the best performing models containing a secondary convolutional layer and their respective forecast error.

dataset	MSE	configuration
IRE15	5.54	$k_{\text{size}} = 6$, $k_{\text{number}} = 9$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [6\text{-}4\text{-}1]$ $k_{\text{size\_p}} = 24$, $k_{\text{number\_p}} = 2$, $dil_{\text{rate\_p}} = 2$
IRE40	15.2	$k_{\text{size}} = 6$, $k_{\text{number}} = 9$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [8\text{-}6\text{-}1]$ $k_{\text{size\_p}} = 12$, $k_{\text{number\_p}} = 4$, $dil_{\text{rate\_p}} = 2$
IRE350	245	$k_{\text{size}} = 6$, $k_{\text{number}} = 9$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [16\text{-}6\text{-}1]$ $k_{\text{size\_p}} = 24$, $k_{\text{number\_p}} = 6$, $dil_{\text{rate\_p}} = 4$

In conclusion, the approach of extracting large-scale information from the input series with a secondary convolutional layer that consists of large kernels seems to work, as the forecast errors of the IRE40 and IRE350 improved.



(a) Errors of the IRE40 dataset. Medium size receptive fields perform the best.

(b) Errors of the IRE350 dataset. Large receptive fields perform the best.
df

Figure 4.33: The heatmaps show the forecast error of CNN models containing a secondary convolutional layer with respect to the dilation rate and the kernel size of the parallel layer. The two parameters determine the size of the receptive field. The larger the parameter values, the larger the receptive field.

4.7.4.10 Creating a More Complex Network

The experiments of the two previous sections have shown that by enlarging the receptive field, there is still potential for improvement in the purely data-driven CNN model development. The forecast computation benefits from the extraction of trends and the recognition of larger patterns that have not been extracted before. The next logical step is to combine the two network structures, thereby creating a more complex network, and examine whether the forecasts can improve even more. For this purpose, the neural network consists of two consecutive convolutional layers with small kernels, as derived in section 4.7.4.7. Parallel to those layers, a second set of larger kernels operates on the input series and extracts large-scale information. The configuration of this convolutional layer is taken from section 4.7.4.9. The structure of the neural network is also depicted in figure 4.34.

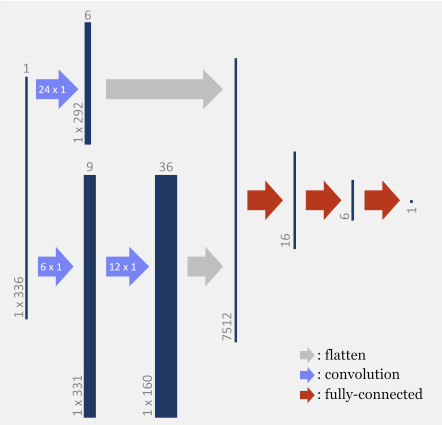


Figure 4.34: The graphic depicts the structure of the CNN model which contains two consecutive convolutional layers and a parallel convolutional layer that operates on the input series as well. The values in the graphic correspond to the model forecasting the IRE350 data.

For a first examination of the new model, the configurations of the kernels and the fully-connected layers are chosen to be identical to the configurations that resulted in the most precise forecasts in the earlier experiments:

- $k_{\text{size1}} = 6$ $\quad k_{\text{number1}} = \begin{cases} 5 \text{ for IRE15} \\ 7 \text{ for IRE40} \\ 9 \text{ for IRE350} \end{cases}$ $\quad k_{\text{stride1}} = 1$
- $k_{\text{size2}} = \begin{cases} 8 \text{ for IRE15} \\ 12 \text{ for IRE40} \\ 12 \text{ for IRE350} \end{cases}$ $\quad k_{\text{number2}} = \begin{cases} 15 \text{ for IRE15} \\ 21 \text{ for IRE40} \\ 36 \text{ for IRE350} \end{cases}$ $\quad k_{\text{stride2}} = \begin{cases} 4 \text{ for IRE15} \\ 2 \text{ for IRE40} \\ 2 \text{ for IRE350} \end{cases}$
- $k_{\text{size\_p}} = \begin{cases} 24 \text{ for IRE15} \\ 12 \text{ for IRE40} \\ 24 \text{ for IRE350} \end{cases}$ $\quad k_{\text{number\_p}} = \begin{cases} 2 \text{ for IRE15} \\ 4 \text{ for IRE40} \\ 6 \text{ for IRE350} \end{cases}$ $\quad k_{\text{dilation\_p}} = \begin{cases} 2 \text{ for IRE15} \\ 2 \text{ for IRE40} \\ 4 \text{ for IRE350} \end{cases}$
- $f_{\text{size}} = \begin{cases} [6\text{-}4\text{-}1] \text{ for IRE15} \\ [8\text{-}6\text{-}1] \text{ for IRE40} \\ [16\text{-}6\text{-}1] \text{ for IRE350} \end{cases}$

The resulting forecast errors, averaged over ten iterations, are presented in table 4.27. The forecast quality for all three datasets deteriorated with respect to the two network structures that have been combined: the network with two consecutive

convolutional layers and the one with two parallel convolutional layers. To ensure that the root of the worsened results is not an unfinished trained model, the training progress has been additionally evaluated with the test data as validation set. The validation loss, however, is stable for the last ten episodes of training, which indicates that the $e = 40$ epochs are still sufficient, even though the number of trainable weights in the models is drastically increased. As an incompletely trained model is not the cause for the unsatisfactory results, there are two further possible explanations. Firstly, the fully-connected part of the neural network is too small to process the increased amount of extracted features, hence the information cannot be properly put to use. Secondly, the two parallel convolutional components of the network extract similar information, which results in redundant information being available to the fully-connected part, thus offering great potential for over-fitting due to the large number of weights with which the two components are each connected to the output neurons of the first fully-connected layer. Both explanations imply that the configurations of the different parts of the network are not aligned with each other, which is a possibility.

Table 4.27: Forecast errors of the CNN model with two consecutive and a parallel convolutional layer. The errors are larger than for less complex models.

dataset	IRE15	IRE40	IRE350
MSE	5.54	15.3	276

In addition, the new, more complex models are less stable, by which it is meant that the final state of the networks are more sensitive to the initial network parameters. This is recognisable by the large variation of the forecast errors for the same configuration. The error values of the different iterations, which vary only in the initial weights, differ by up to 30% for the models trained on the IRE350 dataset. That is a significantly larger variation than observed before with simpler network structures. Hence, a more complex model is less likely to end up in a state in which it produces the best possible outputs. The larger variation also means that the average error over all iterations looses significance as a measure of the forecast quality. However, even the models with the smallest error of all iterations are still outperformed by the best models of the earlier experiments with only either two consecutive or two parallel convolutional layers.

In order to ensure that the fully-connected part of the neural network is sufficient for processing the increased number of extracted features, the size of the fully-connected layer is enlarged. Models with the following, larger fully-connected layers are trained and their forecast performance is compared with the previously trained model. The other hyper-parameters of the network are not changed and, therefore, they are not

listed again:

- $f_{\text{size}} \in \begin{cases} \{[8\text{-}6\text{-}1], [10\text{-}8\text{-}1], [12\text{-}8\text{-}1], [14\text{-}8\text{-}1], [14\text{-}10\text{-}1]\} & \text{for IRE15} \\ \{[10\text{-}8\text{-}1], [12\text{-}10\text{-}1], [14\text{-}10\text{-}1], [16\text{-}10\text{-}1], [16\text{-}12\text{-}1]\} & \text{for IRE40} \\ \{[16\text{-}8\text{-}1], [16\text{-}10\text{-}1], [16\text{-}12\text{-}1], [18\text{-}10\text{-}1], [18\text{-}12\text{-}1], [20\text{-}10\text{-}1], [20\text{-}14\text{-}1]\} & \text{for IRE350} \end{cases}$

Enlarging the fully-connected layers of the CNN model improves the forecast performance slightly and the forecast errors approach the previously achieved errors of the networks with only two convolutional layers.

Table 4.28: The table lists the configurations of the fully-connected network part of the best performing CNN models within the conducted grid-search with the corresponding averaged forecast errors.

dataset	MSE	configuration
IRE15	5.52	$f_{\text{size}} = [10\text{-}8\text{-}1]$
IRE40	15.2	$f_{\text{size}} = [10\text{-}8\text{-}1]$
IRE350	248	$f_{\text{size}} = [18\text{-}10\text{-}1]$

However, the averaged error of the best configuration and the error of the best single model are still larger than of the previously trained models with only two convolutional layers (see tables 4.22 & 4.26). Therefore, the second possible reason for the deterioration of the results is now addressed - redundant information. Assuming the two consecutive layers and the parallel layer extract redundant information, the dimensionality of the extracted features should be reduced to avoid over-fitting. A way of achieving that is pooling, which has already been described in section 4.7.4.5. Here, the pooling layer is located after the secondary convolutional layer containing the larger kernels, in order to decrease the dimensionality of the extracted features. Optimally, only information that is not yet available is extracted and passed on to the next layer. Average pooling is applied, which means the kernel computes the average of all its inputs as output. Several kernel sizes of the pooling layer $k_{\text{size\_pool}}$ are tested for their impact on the forecast performance. The largest size of the pooling kernel $k_{\text{size\_pool}} = 208$ corresponds to the dimension of the feature map. Thus, when $k_{\text{size\_pool}} = 208$ is applied, each feature map is reduced to exactly one output value. Half the size of the pooling window corresponds to two outputs, and so forth. The size of the fully-connected layers is changed to the optimised size previously derived. The other hyper-parameters are not changed with respect to the previous experiment and are not listed for clarity. This experiment is, however, only performed for the models that predict the load of the IRE40 and IRE350 dataset. The forecast errors of the models predicting the IRE15 datasets have essentially been unaffected by any of the previous experiments. It seems that a more complex model

is not able to increase the forecast performance of the IRE15 models due to the high volatility of the dataset.

- $k_{\text{size\_pool}} \in \{208, 104, 52, 26, 13, 5\}$
- $f_{\text{size}} = \begin{cases} [10\text{-}8\text{-}1] & \text{for IRE40} \\ [18\text{-}10\text{-}1] & \text{for IRE350} \end{cases}$

Some of the errors have slightly decreased in comparison to the network without pooling. There are, however, hardly any performance changes for the different kernel sizes of the pooling layer. The fact that $k_{\text{size\_pool}} = 5$ produces the best result indicates that the addition the pooling layers was not as effective as anticipated. The models are, again, producing slightly worse forecasts than the best models with a simpler network structure. In a short, subsequent experiment, it was also tested whether an increase in the amount of parallel kernels improves the performance, but that was not the case.

Table 4.29: The model configuration with an average pooling layer that performed the best and the corresponding errors. $k_{\text{size\_pool}} = 5$ is the smallest pooling size that was tested in the experiment.

dataset	MSE	configuration
IRE40	14.9	$f_{\text{size}} = [10\text{-}8\text{-}1]$, $k_{\text{size\_pool}} = 5$
IRE350	253	$f_{\text{size}} = [18\text{-}10\text{-}1]$, $k_{\text{size\_pool}} = 5$

It must, therefore, be concluded that increasing the complexity of the convolutional network does not further improve the obtained forecasts. The exact reason for the worsened results with the more complex model could not be determined. It is likely that with the training data available, the model is not able to utilise the possibilities a more complex network architecture offers, and instead begins to over-fit. Either the limited amount of training data of only one year is the reason for that or there is not more information in the data that can be extracted and used for forecasting by a convolutional neural network.
Therefore, it is concluded that the potential of a purely data-driven development of the neural network model has been fully exploited.

4.7.4.11 Conclusion of Data-Driven Model Development

After exploiting all possibilities of adjusting the network structure and the hyper-parameters of the network, a short conclusion of this chapter is drawn.
In this chapter, the depth and the size of the convolutional and the fully-connected

part of the CNN have been determined. Furthermore, the influence of different stride sizes, dilations, and pooling operations have been examined. The biggest advance in forecast performance, however, has been achieved with the addition of dropout to the neural network, which reduces the risk of over-fitting by reducing the amount of redundant information the fully-connected layers process during training

The model configurations that produced the most accurate forecast for each dataset are summarised in the table below (table 4.30), including the corresponding average forecast error of the iterations and the forecast error of the single best model of that configuration.
The best 36-hour prediction of the IRE15 and the IRE40 datasets are generated with the models containing two consecutive convolutional layers and three fully-connected layers. The best forecasts for the IRE350 set have been achieved with a CNN model containing two parallel convolutional layers and also three fully-connected layers. All three models employ dropout in their first fully-connected layer.
In the course of the data-driven model development, reductions of 20 % for IRE15, 16 % for IRE40, and 22 % for IRE350 in forecast error with respect to the prototype model, which was the starting point of the development, have been achieved.

Table 4.30: The table sums up the findings of this chapter. The best model configuration for each dataset is presented together with the average forecast error of that configuration and the error achieved with the best iteration. Note the different network structures for IRE15 / IRE40 and IRE350.

dataset	MSE		configuration
	average	best	
IRE15	5.41	5.29	$k_{\text{size1}} = 6$, $k_{\text{number1}} = 5$, $k_{\text{strides1}} = 1$, $k_{\text{size2}} = 8$, $k_{\text{number2}} = 15$, $k_{\text{strides2}} = 4$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [6\text{-}4\text{-}1]$, $\tilde{f}_{\text{size}} = [20\text{-}4\text{-}1]$
IRE40	14.9	14.4	$k_{\text{size}} = 6$, $k_{\text{number}} = 7$, $k_{\text{strides1}} = 1$, $k_{\text{size2}} = 12$, $k_{\text{number2}} = 21$, $k_{\text{strides2}} = 2$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [8\text{-}6\text{-}1]$, $\tilde{f}_{\text{size}} = [27\text{-}6\text{-}1]$
IRE350	245	225	$k_{\text{size1}} = 6$, $k_{\text{number1}} = 9$, $k_{\text{strides1}} = 1$, $k_{\text{size\_p}} = 24$, $k_{\text{number\_p}} = 6$, $dil_{\text{rate\_p}} = 4$, $d_{\text{rate}} = 0.7$, $f_{\text{size}} = [16\text{-}6\text{-}1]$, $\tilde{f}_{\text{size}} = [54\text{-}6\text{-}1]$

4.7.5 Influence of Externally Added Features

In the previous section, the computations of the load forecasts with the novel CNN model are based exclusively on the pre-processed energy load time series. The model extracts information from its input time series and utilises this information to calculate the forecast. All tested modifications of the neural network have been focused on extracting the maximum meaningful information from the input time series and, thereby, improving the forecast performance of the CNN model. However, the origin of the data and knowledge about the setting from which the data stems offer information as well. That information is called domain-knowledge. Utilising domain-knowledge is a crucial part of the development of every machine learning model. Until now, the development of the CNN forecast model has been purely data-driven, which means no domain-knowledge is used to compute the forecast.
Based on the knowledge about the underlying processes that create the data, features are designed in order to add further information to the neural network and, thus, increase the forecast performance of the model.
In the following, several different, external features are included in the CNN model, in order to evaluate their influence on the forecast performance. First, the tested features and the reasoning to include them are discussed. Then, single features and, afterwards, combinations of features are tested.

The time series data presents aggregated electricity loads of residential households. The electricity consumption naturally exhibits a daily repeating pattern due to the day and night cycle and the consequentially resulting human behaviour. Additionally, as discussed earlier, there is a weekly repeating pattern because the consumption on the different days of the week differ from each other. In particular the consumption of working days differs from that of a Saturday or Sunday. All that knowledge can be used to create features that complement the information extracted by the convolutional part of the neural network.
All features that are mentioned in the following were already introduced earlier. For more details, see section 4.2.

First, the different tested features are shortly introduced.
A criterion that obviously strongly influences the behaviour of the residential electricity consumption is the ambient temperature. The warmer it is, the more time people spend outside and, thus, the less electricity is used. During the pre-processing of the load time series, the influence of the ambient temperature on the consumption is already utilised to transform the time series. It is used to eliminate the influence of the temperature on the time series (see section 4.7.2). Even though this process reduces the impact of the temperature on the pre-processed time series, there are still noticeable shifts which coincide with changes in the temperature. Therefore,

the ambient temperature is tested as a feature.
Similarly to the temperature, the cloudage and the precipitation influence the electricity load consumption as well. Both environmental variables vary greatly, depending on the location. Unfortunately, the exact location of the homes included in the time series are not known for data privacy reasons. They can be located anywhere in the Republic of Ireland. Thus, the locations of the homes can differ by hundreds of kilometres, which means no reliable weather information can be obtained. Hence, the cloudage and precipitation are not included as features. If the location of the consumers were not as wide-spread as in the Ireland dataset and the location were (roughly) known, the inclusion of the two variables as features could well improve the forecast quality.
In addition to features associated with environmental influences, there have been extracted features related to the re-occurring cycles in the electricity consumption. The included features connected to those cycles are the *day of the week*, the *weekend*, the *month of the year*, and the *day of the year*. All these features have been tested one-hot encoded and integer-encoded, whereby the models using the one-hot encoded features where more successful. Thus, only those models are mentioned in the analysis. The knowledge about the weekly cycle was also put to use when the length of the input series was determined to be exactly seven days.
Even after pre-processing, the average daily consumption varies in the course of the year. In the winter months, more electricity is consumed; in the summer months, less. To reflect those differences, a new feature has been developed. The months are divided into four categories, depending on their daily average consumption. The analysis of the electricity load shows that the most electricity is consumed in the months January, February, and December, followed by the months March, April, and November. Once again, consumption is lower in the months May, September, and October. As expected, the least amount of electric energy is consumed in the summer months of June, July, and August. This gained insight into the consumption changes during the year is abstracted into a one-hot-encoded feature consisting of four categories. Even though, the analysis of the consumption data further shows that the average electricity load on bank holidays differs from the load of a workday, the holiday feature is not included because the training data contains very few bank holidays. An improvement in the quality by including this feature is essentially impossible. This finding was already obtained during the previous analysis of forecasting time series with the XGBoost model (see section 4.6.3).

In addition to domain-knowledge, features regularly include information about properties of the data, for example statistical key performance indicators. Particularly for a fully-connected neural network processing time series, the statistical properties of the data are regularly used as features to represent the series instead of feeding the series directly to the model. The advantage of forecasting a time series with

a convolutional neural network is that the network performs the task of extracting meaningful features from the time series itself. Therefore, no features are included in the CNN model that is derived directly from the data, because the objective in the CNN model development is for the CNN to extract this information itself.
An examination of the computed forecasts, however, reveals that the forecasts sometimes underestimate the maximum of the daily consumption, which occurs in the early evening. Particularly the forecasts of the winter months show this kind of deviation from the actual load. The consumption in the winter time is generally larger and the daily maximum shifts to a large power value. The reason for underestimating the maximum in the winter might be that for a majority of the days the maximal consumption is smaller. In order to better predict the maximum, a feature is introduced that includes the maximum and the minimum of the last completely known day. These two values essentially describe the load range of the respective day. The load range of adjacent days stays relatively constant over the course of a few days. Therefore, the load of the second-to-last day is used to calculate this feature. Including this feature should allow the model to more accurately predict the maximal consumption of a day.

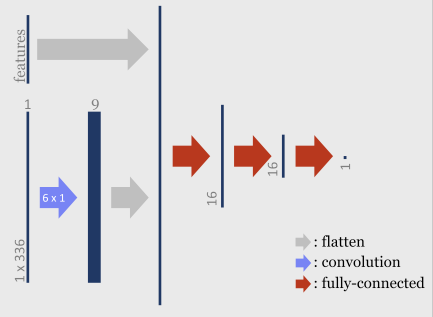


Figure 4.35: The graphic depicts the structure of the CNN model that is used for testing the influence of the features on the forecast quality. The features are added as input to the first fully-connected layer. The values in the graphic correspond to the model forecasting the IRE350 data.

The CNN model with which the effectiveness of the features is tested is the model with only one convolutional layer and three fully-connected layers, which was investigated in section 4.7.4.4. The network structure is also depicted in figure 4.35. The best derived configuration for each dataset (see table 4.18) is utilised in the CNN model that includes features. The features are incorporated into the network after the convolutional layer. They are equal to the features extracted by that layer.

As the first step, the influence of each newly developed feature on the forecast performance is investigated. For this purpose, several models are trained that each include only one of the external features and the resulting forecast error is examined. For each feature configuration and dataset, twenty models are trained that only differ in their initialisation. The number of iterations is increased from the earlier experiments, as preliminary tests have shown that the model performance is more dependent on the initialisation when features are included.
The average forecast errors are presented in the following table:

Table 4.31: The table displays the forecast errors that have been achieved with models utilising one externally calculated feature for the three datasets. As a reference, the forecast errors of the very same model without features are mentioned as well. Each of the values represents the average over twenty iterations. Only the *weekend* and the *day of the week* feature improve the forecast quality.

included feature	MSE		
	IRE15	IRE40	IRE350
None	5.43	15.4	255
weekend	5.11	14.2	242
day of the week	5.18	14.2	250
month category	5.57	15.6	259
value range two days ago	5.55	15.5	265
sun is up	5.67	15.7	269
sunshine duration	5.66	15.7	283
day of the year	6.43	18.6	472
hour	5.63	15.6	259
temperature	5.70	16.0	276

It is striking that only the *weekend* and the *day-of-the-week* feature reduce the forecast error in comparison to the corresponding model without any features. With the inclusion of the other features the forecast error either increases or stays the same as without the features.
There are two possible explanations for why none of the other features improves the model performance. Either the features contain no information that is beneficial for the forecast computation. Hence, they can be excluded. Or the deviation between the predicted and the actual value is mainly caused by a factor which is not represented in the feature, so that the potential improvement from including the feature is many times smaller. The addition of the feature then rather offers potential for over-fitting, instead of improvement. This may explain the deteriorated error values for some of the other features.

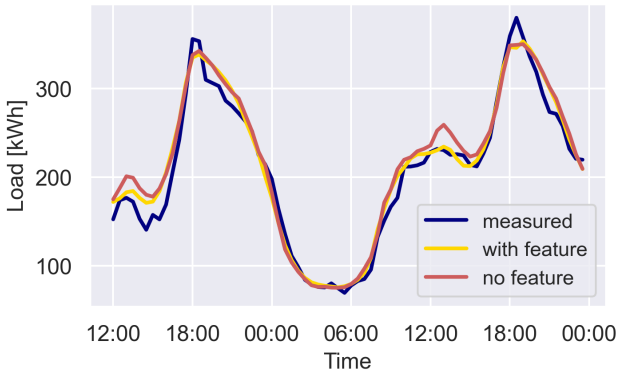


Figure 4.36: The graph depicts the predictions with and without the weekend feature for a Saturday in November. Thus, the first predicted value shown is for Friday noon and the last for Saturday midnight. Around Saturday noon, the prediction computed with features is clearly more accurate. This can be observed for most Saturdays and shows the effectiveness of the included feature.

The inclusion of some of the features into the model deteriorates the forecast error drastically. This is especially the case for the feature that describes the *day-of-the-year*, but also for the *temperature* feature. The hope when including the day-of-the-year feature was that the model may use this information to reflect the changes in the consumption patterns over the course of the year. That is apparently not working and, therefore, the feature is not used any further. That also applies to the temperature feature. Even though there are still seasonal changes in the data after the pre-processing, the temperature seems not to be an adequate parameter to represent those fluctuations. Possibly due to the unknown location of the consumers that make up the dataset, the temperature is not accurate enough to explain the consumption changes. The features that describe the temperature development perform similarly or worse and are, therefore, not used.

The other features are tested in combination with the successful *weekend* feature in a subsequent experiment. The results show whether the features are able to add valuable information to the model, now that the deviations caused by the different consumption behaviour on the weekend and on working days are eliminated. Hence, the possibly smaller influence of the other features can be studied when combing the weekend feature with one other feature at a time. The *day-of-the-week* feature, however, is not combined in the same way with all other features, as it is presumed that the same information was extracted earlier from this feature.

Table 4.32: The forecast errors achieved by models including the *weekend* feature and one additional feature. For reference, the errors of two simpler models are mentioned as well.

included feature	MSE		
	IRE15	IRE40	IRE350
None	5.43	15.4	255
weekend	5.11	14.2	242
weekend & month category	5.16	14.2	232
weekend & value range two days ago	5.08	14.2	233
weekend & sun is up	5.18	14.2	243
weekend & sunshine duration	5.32	14.4	245
weekend & hour	5.17	14.3	234
weekend & day of the week	5.18	14.2	239

The forecast errors of the models including two features are shown in the table above. The combination of the *weekend* feature with the remaining promising features has partly resulted in an improved forecast performance. Only the daily *sunshine duration* does not improve the forecast for any dataset and is, therefore, no longer used. The inclusion of the information about whether the *sun is up* at the predicted time only improved the forecast quality of the model for the IRE40 dataset. For the IRE15 dataset, only the *value range two days ago* improved the forecast further. For the IRE350 dataset, on the other hand, all of the other features improved the forecast. This shows, again, that it is more difficult to compute accurate forecasts for highly volatile time series.

In the next computations, three and more external features are included in the models to further improve the forecast quality. In order to limit the number of possible combinations and, therefore, the computational load, only models that include the *weekend* and the *month category* feature are evaluated. The *month category* feature seems the most promising for the IRE350 dataset, where the most gains can still be expected, and seems to reflect the forecast deviations over the course of a year the best. All tested feature combinations and the resulting forecast errors are listed in table 4.33.

The tests of the different feature combinations show that small improvements to the forecast quality can still be achieved by including more features. The best forecast results were obtained by the combination of the *weekend*, the *month category*, the *value range two days ago*, and the *day of the week* feature for all three datasets. When even more features are combined together, the forecast quality either does not further improve or deteriorates. Therefore, the conducted computations of models

Table 4.33: The forecast errors of models including several features are presented. For reference, three simpler models are mentioned as well.

included feature	MSE		
	IRE15	IRE40	IRE350
None	5.43	15.4	255
weekend	5.11	14.2	242
weekend & month category	5.16	14.2	232
weekend & month category & value range	5.09	13.9	234
weekend & month category & hour	5.24	14.5	236
weekend & month category & day of the week	5.11	14.2	233
weekend & month category & sun is up	5.33	14.1	233
weekend & month category & value range & hour	5.18	14.5	230
weekend & month category & value range & sun is up	5.28	14.0	228
weekend & month category & day of week & hour	5.13	14.3	227
weekend & month category & day of week & sun is up	5.28	14.0	230
weekend & month category & day of week & value range	5.07	13.8	225

including even more features are not listed in table 4.33.

In comparison to the CNN models without any features, the inclusion of features decreases the forecast error of the CNN model, regardless of the dataset the models work with. The biggest improvement has been achieved for the models for the IRE350 dataset. The error of the forecast decreased by 12 %. It decreased by 10 % for the IRE40 models and 7 % for the IRE15 models.
The inclusion of one or two of the most influential features, however, improves the forecast quality significantly. It can, therefore, be concluded that a big part of the forecast deviations were caused by the different consumption behaviour on the weekend, which the model has apparently not been able to represent and forecast properly. If more data were available, another option to deal with those consumption differences would be to train two independent models: one for the weekends and one for the work days.

In contrast to the features utilised when computing the forecast with the XGBoost model (see sec. 4.6), which are mostly directly created from the time series, the features used in the creation of the forecasts with the CNN model only describe external influences or domain-knowledge. There are two reasons for this. Firstly, the anticipated advantage and the objective of the developed CNN model for time series forecasting is that no manual feature extraction from the time series should be necessary any more, because meaningful features are extracted from the data by the convo-

lutional layer of the neural network. Particularly the shift-features, which describe the consumption at the same time of the day on previous days, are not necessary for the CNN model. The XGBoost model, however, relies heavily on those features. If the same features as for the XGBoost model were included, there is a chance that the forecast would be mainly a regression of the added features and the information extracted by the convolutional layer(s) would be neglected. Secondly, naturally, it has also been tested whether features containing statistical information about the input, e.g. the variance, improve the forecast accuracy. This was not the case, which affirms that information is successfully extracted by the model itself. The sole exception is the feature that describes the value range of two days ago with respect to the predicted value. This feature was included after ascertaining that the maximal values of the 36h-forecast are regularly under-estimated.

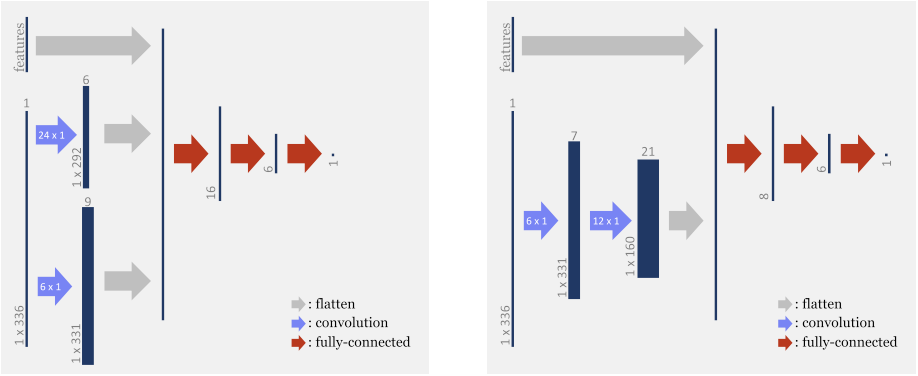
In conclusion, it was shown that the addition of externally calculated features based on domain-knowledge and weather data improves the forecast performance of CNN models. The largest deviation between the computed forecasts and the actual load was caused by the difference in consumption behaviour between weekends and weekdays, as the inclusion of the *weekend* feature improved the forecast quality the most. The feature combination which resulted in the best forecast model is: *weekend*, *month category*, *value range two days ago*, and *day-of-the-week*.

4.7.6 Utilising Features for Best Models

In section 4.7.4 and its subsections, the network architecture and the associated hyper-parameters which promise to generate the best possible results without any externally engineered inputs were identified. The final configurations are summarised in table 4.30 on page 126. Furthermore, in section 4.2, the engineered features that have the greatest positive influence on the forecast performance when included in a more basic CNN model than the resulting model of section 4.7.4 were determined. They are listed in table 4.33.

In order to achieve the best forecast performance possible, the models examined in this section combine the most promising model architectures (see fig. 4.37) and hyper-parameters with the engineered features. For each dataset, 20 identical models are trained to better analyse the potential these models offer.

The results show that the forecast quality could, again, be improved by combining the more complex model architecture with the engineered features.
These models constitute the final model of the novel CNN approach for time series forecasting. The best achieved forecast errors are:



(a) Model structure with two parallel convolutional layers that is applied to the IRE350 dataset.

(b) Model structure with two consecutive convolutional layers that is applied to the IRE40 and the IRE15 datasets.

Figure 4.37: The graphic depicts the structure of the models that produce the most accurate forecasts. These models are the final result of the development of the novel forecast model based on convolutional neural networks.

Table 4.34: The table shows the forecast errors of the CNN models combining the best developed network architecture with the most promising features. The average errors over all iterations and the errors of the single best model are presented.

dataset		IRE15	IRE40	IRE350
MSE	average	4.94	14.1	225
	best model	4.89	13.7	211

Best forecast errors for each dataset

- IRE15: MSE = 4.89
- IRE40: MSE = 13.7
- IRE350: MSE = 211

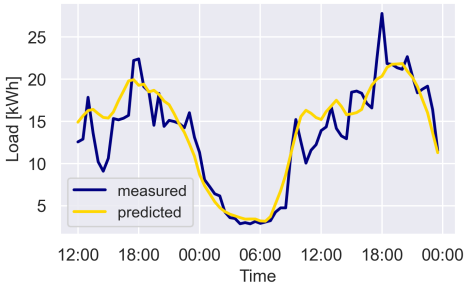


Figure 4.38: The graphic shows an exemplary 36-h forecast for a November day of the IRE15 dataset computed with the final model. It can be seen that the model is able to compute a prediction that captures the trend of the measured data well, but is not able to follow each fluctuation.

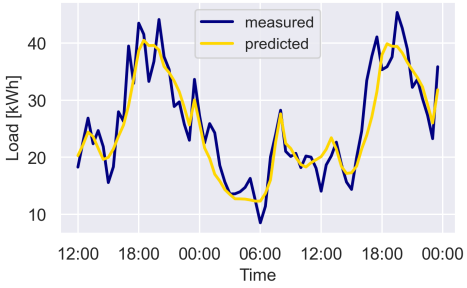


Figure 4.39: The graphic shows an exemplary 36-h forecast for a November day of the IRE40 dataset computed with the final model. The model is able to produce a forecast that captures the trend and most of the fluctuations in the data well.

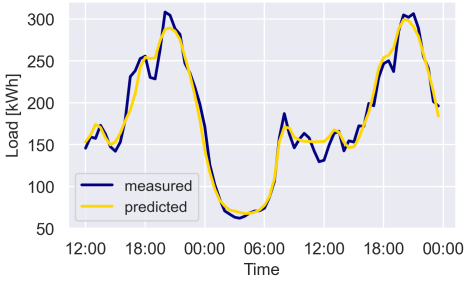


Figure 4.40: The graphic shows an exemplary 36-h forecast for a November day of the IRE40 dataset computed with the final model. The model is able to compute a quite accurate forecast, which even captures small variations in the load.

4.7.7 Analysis of the Models

This section offers a deeper analysis of the obtained CNN models. The transferability of the models to other datasets is addressed and the kernels of the models are analysed to better understand their functionality. The models referred to in this section are taken from the previous section (sec. 4.7.6) and include external features. For each dataset, the single model exhibiting the lowest prediction error on the complete test set is chosen to be analysed. The models differ in their configuration, as the best configuration for each dataset is chosen (see table 4.30).

4.7.7.1 Transferability

The hyper-parameters of the models have been adjusted very specifically to the explicitly available datasets. In the following, how the models perform on datasets that are made up of different households of the Irish Smart Meter Trial is examined. The amount of households in the datasets is, in the first step, kept the same as in the datasets with which the models have been trained. That means the new datasets contain the aggregated electricity consumption of either 15, 40, or 350 households. In the second step, how the models perform when they work with datasets of a differing aggregation level is tested.

For the first analysis, ten consumption time series of the same number of households are created for each of the three models by randomly sampling households from the Irish Smart Meter Trial dataset without replacement. The CER data contains the consumption of 3273 residential households.

Table 4.35: The table shows the forecast errors when the model of a specified aggregation level is applied to a new dataset constructed from the same amount of households, in which the households are randomly sampled from the CER data. The predictions are calculated once with the original pre-processing parameters and once with adjusted parameters. The error values represent the average error of all ten new datasets.

aggregation level	MSE on original data	MSE on new data	
		original pre-processing	new pre-processing
15	4.89	6.01	5.68
40	13.7	16.4	16.4
350	211	224	270

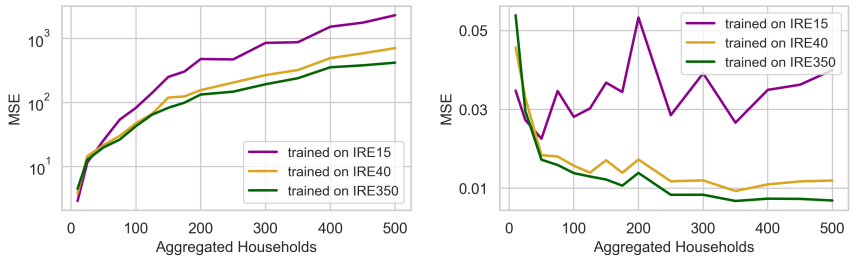
Table 4.36: The table shows the forecast errors similar to the previous table. The only difference is that the error values represent the forecast error before the predictions are rescaled to their actual size. Thereby whether the forecast errors of table 4.35 originate from the forecast itself or the rescaling can be evaluated.

aggregation level	MSE (no rescaling) on original data	MSE (no rescaling) on new data	
		original pre-processing	new pre-processing
15	0.028	0.035	0.035
40	0.021	0.024	0.023
350	0.006	0.006	0.008

The errors of the new time series are computed based on the test data. The new datasets are split the same way as the original datasets into training and test sets. Only the predictions of the test set are used for calculating the error. The new training data is discarded and not put to use at all. This makes the errors comparable with each other. The errors are displayed in table 4.35 . The pre-processing, i.e. the regression with the temperature data and the normalisation, are done with the same scaling parameters extracted and used for the original dataset with which the model was trained. For each time series, the pre-processing is also performed utilising newly extracted scaling parameters, which are computed individually for each time series, in order to examine if a possibly non-optimal pre-processing has an influence on the computed predictions. The exact pre-processing procedure is described in section 4.7.2. Additionally, for comparative reasons, the prediction error before re-normalising and reversing the regression is computed as well (see table 4.36).
The forecast errors generated with the new data are considerably larger than with the data the models are trained with for the IRE15 and IRE40 models. The model trained on the IRE350 dataset, however, is capable of producing good forecasts for unknown datasets. The smallest error achieved with one of the new datasets is $\mathrm{MSE}_{\mathrm{best}} = 208$ and, thus, even smaller than of the original data. This might be caused by slightly smaller amplitude or less volatile data.
The forecast errors of the IRE350 model are more sensitive to a change in the pre-processing parameters. The forecast errors for the IRE15 and the IRE40 model with the newly derived pre-processing parameters are similar to the models with the original parameters. The reason for that is that the forecasts generated by the IRE15 and the IRE40 model are, in comparison, less accurate than by the IRE350 model, which is indicated by the errors computed before re-scaling which are shown in table 4.36. It is apparently essential for the IRE350 model that the input values of the neural network always correspond to the same consumption values that they had when the model was trained. Moreover, the deviations between the original datasets

and the newly created datasets are smaller, the larger the aggregation level is. The influence of the consumption behaviour of a single household on the aggregated consumption time series is way larger when the time series consists of 15 or 40 households than of 350 households. The characteristics of the volatile time series can differ drastically between the ten newly created series.

Secondly, whether the trained CNN models are capable of processing data of a differing aggregation level is tested. For this purpose, 15 new consumption time series of varying aggregation level have been created by sampling residential households from the Irish Smart Meter Trial data. The time series of the lowest aggregation level represents the total electricity consumption of ten households, the one of the highest level represents 500 households. In this analysis, the scaling parameters used in the pre-processing are re-calculated for each of the 15 time series. In order to better compare the forecast performance of the different time series, the error prior to the re-scaling is additionally calculated.



(a) The forecast error is calculated after re-scaling the predictions. Hence, the error naturally increases with an increasing aggregation level or, respectively, load amplitude.

(b) The forecast errors presented in this plot are calculated before the predictions are re-scaled. It shows that the IRE15 model is not able to generate more accurate forecasts with an increasing aggregation level.

Figure 4.41: The plots show the forecast error of the new datasets with respect to their aggregation level. The IRE350 model outperforms the other two models on almost all aggregation levels.

The forecast results of the three models with the fifteen time series of varying aggregation level are displayed in figure 4.41. The errors before re-scaling show, as expected, that the models perform the best in the vicinity of the aggregation level they have been trained with. The plot further shows that the IRE350 model outperforms the other two models, with the exception of the lowest aggregation level. The reason is that the models, which are trained on more volatile data, are probably

not able to analyse small changes in the consumption behaviour of the less volatile time series, as detailed as the IRE350 model. They are, additionally, not able to extract and process as much information from the time series as they contain less kernels and a smaller fully-connected network in order to perform optimally on their corresponding training data.

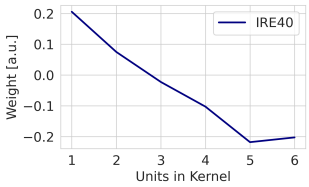
It can be concluded that the models perform the best on the datasets with which they have been trained. Utilising trained models for forecasting comparable datasets can only be advised when the data the model was trained on is not very volatile. For volatile time series, the characteristics of the series differ too much for the same model to produce an accurate forecast. Similar findings regarding applying a trained model to data that exhibits a different magnitude have been obtained. It only works when the time series is not very volatile. This, again, shows that the prediction of volatile time series is very demanding.

4.7.7.2 Analysis of Kernels

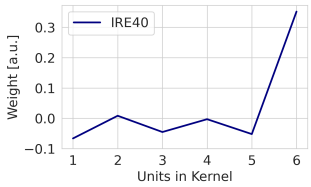
In the attempt to better understand the information extraction process of the convolutional layers, the kernels of the resulting models are examined. For this purpose, the kernels of the CNN models with one convolutional layer that provide the best results were first examined. Afterwards, the models with two consecutive and two parallel layers are analysed. Only the best configurations of each network structure are a part of this examination.

The kernel size of the first layer is fixed to $k_{\text{size1}} = 6$ for all models, regardless of which dataset they are applied to. Figure 4.42 shows a selection of kernels taken from the models with only one convolutional layer and the task of the kernels in the convolutional neural network is discussed in the corresponding captions. The selection of kernels is limited to those that are humanly interpretable. There are many kernels in the networks whose function is not clear. If the function of all kernels were obvious, however, there would be no need to train machine learning models, as one could manually write the models instead. For some kernels, a portion of their weights assume values close to zero. That suggests that only the part of the kernel with the non-zero weights extracts information and it indicates that, for this particular extraction, a smaller kernel size would have been sufficient as well.

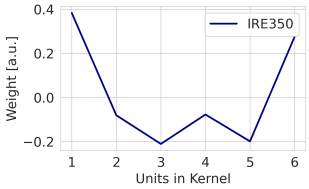
The kernels exhibit different shapes and structures in order to extract the characteristics, e.g. the location of the extrema, or the gradient. However, there exist also a few kernels which only have one weight that is considerably different from zero. They basically generate a representation of the input series as their feature



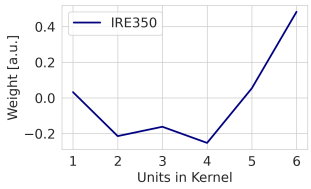
(a) This kernel is activated when its input exhibits a non-vanishing gradient. Thus, it indicates the location of a changing load in its feature map.



(b) The last input value of this kernel has the major influence on the activation of the kernel. Hence, it basically reproduces the input series as its feature map.



(c) This kernel is active when its input represents an extremum. Thus, the resulting feature map shows where in the data the extrema are located.



(d) This kernel mainly detects slopes within the second half of its input. Its activation is increased when the first half of its input assumes small values. Its function might be to identify the ending of the nightly load minimum.

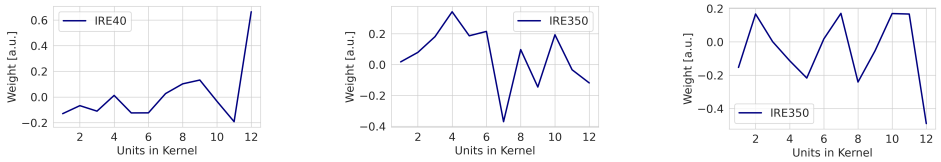
Figure 4.42: The different plots all represent one kernel from networks with only one convolutional layer. In the legend of each plot, the dataset on which the model, which the kernel is a part of, has been trained is listed.

maps and, hence, fulfil a similar function to a skip connection residual NNs, namely propagating the input series to the subsequent layers.
In figure 4.44, the kernels of the second layer of the networks containing two consecutive convolutional layers are depicted. They differ in size depending on the dataset the model is trained on. Additionally, a stride size $k_{\text{stride2}} > 1$ is applied, which means the kernels of the second layer have less overlap when processing their input. It must be kept in mind that those kernels operate on the feature maps of the first convolutional layer, which is not necessarily similar to the input series. This makes interpreting the kernels' purpose more challenging.

The kernels in figure 4.43 show the kernels of networks which contain two parallel convolutional layers. The interpretation here is extraordinarily hard because the kernels are large and have a dilation rate $dil_{\text{rate\_p}} > 1$, which results in a large receptive field. For most of the kernels, it is not clear what function they fulfil in the convolutional layer.

The analysis of the kernels unfortunately does not provide much insight into the

process of the forecast computation and, therefore, does not explain why the CNN model performs better than the other machine learning models. On the other hand, it was to be expected that the kernels only represent elementary structures, as the load series does not exhibit very complex shapes itself. The function of the first layers of convolutional networks is to extract basic shapes from the input and that is exactly what the presented kernels do.



(a) The activation of this kernel is, again, mainly influenced by the last few input values.

(b) This kernel seems to identify an extrema within the first half of its input. The second part of the kernel has a minor influence, as the weights are close to zero and noisy.

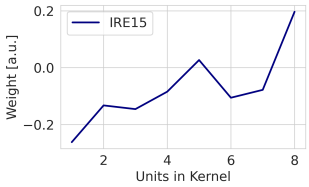
(c) This kernel is representative for many kernels in the parallel layer as it is not really interpretable.

Figure 4.43: The plots represent the kernels of the second parallel convolutional layer, which also operates directly on the input data. The networks contain two parallel convolutional layers, with the first layer consisting of a larger number of rather small kernels in comparison to the depicted kernels.

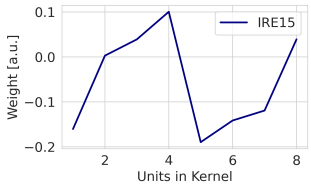
4.8 Comparison of the Machine Learning Models

This section offers a comparison between the different tested models and summarises the main findings for each forecast approach.

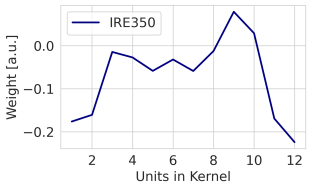
In this chapter, several differing methods for time series forecasting have been applied to forecasting electricity loads of three different aggregation levels derived from the CER Smart Meter Trial. Naïve models and different linear and non-linear regression models have been used and a novel forecast model based on one-dimensional convolutional neural networks was eventually developed.
Table 4.37 shows the forecast errors for the different models. The newly developed CNN model clearly performs the best on all datasets. The results, however, show that simpler models are also able to provide good forecasts. Only the exponential smoothing model performs poorly when applied to forecasting high-resolution time series. In particular the performance of the linear ridge regression model is impressive. The forecasts generated by the model for the IRE15 and the IRE40 dataset are quite accurate and are only beaten marginally by the established non-linear models.



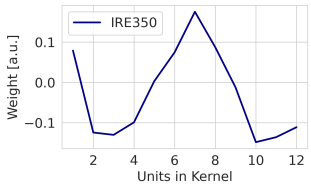
(a) This kernel extracts the gradient of its input. The noisy gradient probably originates from the volatile IRE15 input series.



(b) This kernel is highly point-symmetrical. That means when the input is axis-symmetric, its activation is zero. This allows it, for example, to extract the degree of skewness of an extremum.



(c) This kernel identifies wide extrema. It distantly resembles the shape of the typical load during the later day.



(d) This kernel identifies sharper extrema in comparison to (c).

Figure 4.44: The plots represent the kernels of the second convolutional layer from networks with two consecutive convolutional layers. In the legend of each plot, the dataset is listed on which the model, which the kernel is part of, has been trained.

This proves that linear models are valuable forecast tools as well, in particular for volatile data. The advantages of the non-linear models only appear when processing less volatile data, because they are better able to detect and utilise subtle variations and changes in the time series, which are concealed in the fluctuations of more volatile series. Hence, it can be sufficient to apply a linear forecast model when volatile time series are predicted. They generate comparably accurate forecasts, but are less computationally costly.
Of the established models, the tree-based model generally performed the best. It was especially able to employ its strengths with the less volatile IRE350 series, but performed very well on the other two datasets as well. The computational load is, however, significantly larger than for the ridge regression model when training the model with cross-validation. That handicap must be taken into account when choosing between the models.

In addition to the completely developed CNN model, the table also includes the forecast errors of the CNN model without externally engineered features. Only the temperature forecast was utilised in order to pre-process the data accordingly to

Table 4.37: The table compares the forecast performance of all the tested models. The mean squared error and the mean absolute percentage error are provided for each dataset. The forecast errors correspond to the model that performed the best, if several variations were evaluated.

	IRE15		IRE40		IRE350	
	MSE	MAPE	MSE	MAPE	MSE	MAPE
exponential smoothing	16.4	270 %	49.1	23 %	2523	15 %
naïve forecast	8.64	29 %	22.6	17 %	469	8 %
ridge regression	5.86	25 %	15.9	14 %	366	7 %
fully-connected NN	5.68	26 %	17.9	15 %	363	8 %
tree-based model	5.81	24 %	15.8	14 %	290	7 %
CNN model (no external features)	5.29	24 %	14.4	14 %	225	6 %
CNN model (with external features)	4.89	23 %	13.7	13 %	211	6 %

the procedure introduced in section 4.7.2. The forecast errors achieved with this simplified CNN model are larger than those of the complete CNN model but still smaller than the errors of all established methods. This confirms, on the one hand, that the extraction of useful information with convolutional layers does work very well. The biggest part of the improvement in forecast performance with respect to the established models stems from the novel way of extracting features from the series. On the other hand, it shows that the addition of the engineered features is beneficial to the forecast, but only adds the last bit of information to the model in order to compute an accurate forecast.

In order to compare the forecast accuracy of the models between the three different datasets, the mean absolute percentage error (MAPE) has also been calculated for the best forecast models according to the MSE. In contrast to the MSE, large deviations from the actual load are not particularly penalised. This can induce a deviating assessment of the forecast performance.
The MAPE values clearly show that computing time series forecasts for volatile data series is more difficult for all tested forecast models. The MAPE values generally differ around 20% between the IRE15 and the IRE350 set. The main reason for the less accurate forecasts is, as stated before, that extracting information is way more demanding due to rapidly changing consumption and due to the large influence of single households on the aggregated load, which results in more unforeseeable consumption changes. However, even for the highly volatile IRE15 set, which consists of the load of only fifteen households, the novel CNN model clearly outperforms the established models which again shows the effectiveness of this new approach.

4.9 Comparison to Similar Research

The topic of electricity load forecasting just recently gained the attention of the research community with the advancing digitisation of the energy sector. Hence, the number of studies in this research area is still limited and the studies focus on a wide variety of aspects. In the following, a few studies that have a focus similar to this thesis are exemplarily presented and their findings are briefly summarised.

There are a few publications that also use the CER Smart Meter Data for load forecasting. One of the first studies using this very dataset is by Wijaya et.al [124] and was published in 2015. They apply linear regression and seasonal ARIMA (SARIMA), amongst other models, to forecasting the aggregated load of roughly 800 households. The focus of the work, however, is on the influence of clustering the households that make up the load series and predicting the load for the clusters individually. The authors evaluated their models on a one-point forecast only and, therefore, the forecast performance cannot be compared with this work. Their results have, however, shown that the SARIMA model performs considerably worse than the other models, which confirms the decision to not include them into this thesis.
[125] works with the CER data as well. Just like the authors of the previous study, Arora et.al. predict one data-point at a time and evaluate their forecasters, which are utilising kernel density estimations, on a one-point prediction. Additionally, they only utilise seven months of data for training and one month for the evaluation, which is unusual and makes a comparison with the results of this thesis meaningless.

In the last years, some publications that utilised a combination of a convolutional and a LSTM network appeared.
Alhussein et.al. [126] published a study in 2020 that includes some interesting findings, even though they only compute a three-hour prediction. A load series of small aggregation level (69 residential buildings) from Australia is used. The authors develop a hybrid model consisting of a CNN and a subsequent LSTM network. They successfully apply the CNN to extracting information from the time series, similar to what is done in this thesis. Additionally, they recognise the benefit of dropout, even if they are applying a smaller dropout rate. Due to the recurrent units included in the model, the computational cost is presumably many times larger than for a solely CNN-based model.
In 2018, Tian et.al. published a study [127] in which they developed a CNN-LSTM hybrid model as well. A comparison with a CNN model has shown that the hybrid model performs only slightly better but is presumably far more costly to train. This

reassures the choice of the convolutional neural network model for computing the forecasts. The electricity load series used in the study is, however, of a far larger aggregation level. It includes the consumption of a whole region in northern Italy. In 2019, Gasparin et.al. [128] compared several forecast models on a load series of very large aggregation level as well. Additionally, they evaluated whether computing a 24-hour forecast at once or iteratively, as done in this work, results in better forecasts and they concluded that the iterative approach is more promising. This confirms that the iterative forecast approach of this work is promising, even if the approach of computing the forecasts recursively has not yet become very common. Furthermore, they determined that exogenous variables can significantly improve the forecast accuracy, which could also be seen during the development of the CNN model.

The mentioned studies show that electricity load forecasting is an active research field, in which no prevalent forecast model has yet emerged. While CNN models were barely mentioned in the context of load forecasting a few years back, in the very recent past, CNN models started appearing in some publications. The studies affirm some findings made in this thesis. None of the found studies, however, were similar enough to the conducted work so that the forecast errors could be directly compared. This shows, furthermore, that the work done in the course of this thesis is a valuable contribution to research field. This thesis thus joins the series of publications that will eventually increase the accuracy of electricity forecasts. The detailed description of the development process of the CNN model offered in this thesis might be particularly valuable for researchers, as none of the publications mentioned that a hyper-parameter search was conducted. It is, therefore, unclear whether the network structures used are based on any empirical findings.

5 Conclusion

This concluding chapter offers, first, an overview of the topics covered in this thesis. Afterwards, the key findings of the research are presented and the significance of this work in the field of machine learning is portrayed. The chapter closes with a short discussion of the limitations of the drawn conclusions and offers suggestions on how to follow up on the work that has been initiated.

After introducing the project MAGGIE, in which the findings of this work will be implemented, in 1.2 and the used dataset in 1.3, first, related works in the research field were presented in 2. Afterwards, in 3.1, neural networks were introduced in detail. Additionally, established machine learning approaches for time series forecasting were presented in 3.2, namely regression models, recurrent neural networks, and tree-based models. Subsequently, in 3.3, how the machine learning methods are applied to time series forecasting was discussed. In the following, the models were applied to forecasting electricity loads. The forecast performance of the established models was evaluated in 4.3, 4.4, and 4.6. Thereupon, a novel forecast model based on convolutional neural networks was developed in 4.7. The final model was introduced in 4.7.6. The results of the different models were eventually compared in 4.8.

5.1 Summary of the Results

The research conducted in the course of my doctoral thesis was focused on evaluating established machine learning models on the task of time series forecasting and on developing a novel model for time series forecasting based on convolutional neural networks. The models were employed to predict aggregated residential electricity loads of different aggregation levels. Computing an accurate electricity forecast is an important element in planning the machine schedules in energy systems.

The naïve approach of assuming that the electricity consumption behaviour of the residents does not change allows for a rough estimate of the consumption. The linear ridge regression model, which bases its forecasts on engineered features and selected

historic load values, already generated more accurate forecasts and, furthermore, is computationally not very demanding. While a non-linear regression of the features utilising a fully-connected neural network did not improve the forecasts significantly, developing a random forest model for forecasting resulted in further improved forecasts. The tree-based models produced the most accurate forecasts of all established models. Thus, applying a random forest utilising engineered features to compute an electricity load prediction is the most successful approach for time series forecasting of the established methods.
The newly developed convolutional neural network model extracts information directly from the input load time series, yet is also able to process external features to further improve its forecasts. The predictions computed by the novel model were much more accurate than of any other model. Even without including any external features, it was able to provide better forecasts than the tree-based model with external features. The major improvement to the established forecast models is, therefore, the ability of the novel model to extract useful information from the data itself. It does not necessarily require manually engineered and selected features. Along with the non-linear data-processing that neural networks offer, this results in a highly effective forecast model.

A key finding in the model development was that randomly dropping out a large rate of the values in the activation maps improves the forecast performance of the CNN model drastically. This illustrates that the amount of redundant information present in time series with large sampling rates should be reduced, in order to obtain good forecast results.
Furthermore, the experiments revealed that predicting the aggregated energy load of a small number of residential households accurately is way more challenging than of a large number of households. Time series of smaller aggregation level exhibit a more volatile behaviour that is very difficult to accurately predict. The volatility for smaller aggregation levels is larger because instances of spontaneous and random human consumption behaviour influence the aggregated load time series more strongly. The forecast error of all tested forecast models decreased with an increasing aggregation level. Non-linear machine learning models stand out particularly with their accurate forecasts for larger aggregation levels. Trying to predict electricity loads of single households seems not to be worthwhile in the light of this fact.

The developed CNN models do not exhibit very complex or very deep network structures and can, therefore, still be trained with relatively small computational cost. The conducted experiments showed that for computing time series forecasts of one-dimensional data, rather simple models perform the best. Large neural network structures should be avoided, because they only offer more potential for over-fitting and training them is more elaborate and costly. The conclusion that highly complex

models are not necessary to generate accurate forecasts is backed by the fact that a linear regression already offers relatively accurate forecasts.
In the light of this finding, it is astonishing that in some recent publications highly complex models are suggested for time series forecasting. This is not consistent with the findings of this thesis. The author rather suggests striving for simplicity and questioning whether an added degree of complexity in the model really improves its capabilities when developing a time series forecasting model.
In addition, the computational cost of training a forecast model increases with the complexity of the chosen model. The more modifiable parameters a model possesses, the longer the training process takes and the more data is necessary to obtain a sufficiently trained model. Therefore, before developing or training a forecast model, what the requirements on the model are, in particular which forecast accuracy one anticipates, how much data is available for training the model, and what computational capacities can be used for the training process should always be determined. Based on these factors, the model selection should be made. There are instances when it could be beneficial to use a simpler model instead of the novel CNN model.

In a nutshell, the research of this thesis has shown that the novel approach of applying a one-dimensional neural network to the task of electricity forecasting works successfully. The convolutional neural network model outperformed all established forecast models. When choosing an appropriate forecast model, however, the data availability and the nature of the data should be taken into account. The most complex and computationally costly model is not necessarily the best choice for the task at hand - simpler models can possibly already fulfil the requirements set with less computational cost.

5.2 Contribution to the Scientific Discourse

The research conducted within the scope of this thesis was focused on time series predictions, specifically on computing short-term electricity load forecasts on a sub-hourly time grid. The focus of the machine learning community has so far mainly been on processing two-dimensional data. The topic of time series forecasting is just gaining popularity as, on the one hand, the progress in image-processing is slowing down and, on the other hand, more time series data is recorded and becomes available due to the advancing digitisation. This increases the need for machine learning methods that process one-dimensional data.
This thesis offers a comparative study of different machine learning models for time series forecasting and highlights the advantages and disadvantages of the discussed models. It emphasises that applying the most complex, available forecast model is

no guarantee for success - simple models also have their raison d'être.
Furthermore, a novel forecasting model based on convolutional neural networks was introduced and its development stages were described in detail. The novel model allows for very accurate time series forecasts. It was optimised to be applied to short-term load forecasting (STLF), but could potentially be utilised for a broad spectrum of forecasting issues. The detailed description of the development process allows other researchers to follow the reasoning for the chosen model structure. It illustrates why the convolutional neural network consists of relatively few layers compared to other neural network models used in forecasting.

5.3 Limitations

There are a few limitations to the information value of the presented results that are discussed in the following.
The evaluation of the different machine learning models regarding their forecast performance is only based on the data from the CER Smart Meter Trial [16]. The availability of datasets that cover more than one year of electricity consumption and have a sub-hourly sampling rate is very limited. During the last few years, the data availability increased slightly, but was still very scarce when the research for this thesis started. The fact that the novel model was developed based on only one dataset confines the use of the CNN model to similar data. Presumably, some adjustments of the configuration or even the network structure are necessary when adapting the CNN model to other datasets and new objectives.
Even though the dataset includes a multitude of single residential household load series, it unfortunately only covers one and a half years of consumption in total. As a result, only one year of electricity loads could be used as training data. This made it impossible for the model to identify annually recurring consumption patterns and utilise them for improving the forecasts. Electricity consumption data that covers several years would be preferable. Another disadvantage of the CER Smart Meter dataset has been that the location of the households is unknown. They could be scattered all over Ireland. This meant it was not possible to include detailed weather information in the features and also introduced an uncertainty to the accuracy of the used temperature time series. With better knowledge of the consumers' location, detailed weather information could be used to enhance the forecast quality.

5.4 Future Works

The limited data availability is one of the issues that needs to be addressed in future research on the topic of time series forecasting. As discussed above, there is still potential for improving the forecast quality of the developed CNN model when larger datasets that cover a longer time period are available for training the model. The CNN model should be evaluated with different datasets in order to get a more detailed image of its capabilities and to identify shortcomings of the model.
Furthermore, it is essential to incorporate detailed weather information to the forecast model. This could improve the forecast accuracy further, since human behaviour is strongly influenced by the weather. This requires knowledge about the geo-location of the consumers.
The next logical step in computing more accurate forecasts would be, in my opinion, to develop an ensemble model which combines the novel CNN model with one or more other machine learning models and utilises the assets of each model to test whether this allows for more accurate predictions - always with the assumption that adequate data is available. In order to train an ensemble model, a larger dataset than the one used here is necessary and, additionally, the fact that the computational load of training such a model is increased needs to be taken into account.

In general, the digitisation of the energy market, in particular on the consumer side, needs to progress further. An extensively digitised energy market offers, on the one hand, more data for machine learning applications and, on the other hand, opens up opportunities to deploy those applications in a large scale. This, in turn, allows for the transformation of the energy market towards carbon neutrality by integrating electricity from renewable energy sources and replacing conventional power plants. Thereby, the emission of greenhouse gases will reduce significantly which would be a big step into slowing down the human-induced climate change.

List of Figures

List of Tables

Bibliography

[1] R. Pachauri and A. Reisinger, eds., *Climate Change 2007: Synthesis Report. Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, 2007.

[2] R. Pachauri and L. Meyer, eds., *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, 2014.

[3] S. C. Herring, N. Christidis, A. Hoell, M. P. Hoerling, and P. A. Stott, "Explaining extreme events of 2018 from a climate perspective," *Bulletin of the American Meteorological Society*, vol. 101, no. 1, pp. 1–140, 01 Jan. 2020.

[4] J. Fourier, "Mémoire sur les températures du globe terrestre et des espaces planétaires," *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, vol. 7, pp. 570–604, 1827.

[5] J. Tyndall, *Heat considered as a mode of motion*. D. Appleton and Company, 1875.

[6] S. Arrhenius, "Xxxi. on the influence of carbonic acid in the air upon the temperature of the ground," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 251, pp. 237–276, 1896.

[7] F. Holl, "Alexander von humboldt und der klimawandel: Mythen und fakten," *HiN: Alexander von Humboldt im Netz; international review for Humboldtian studies*, vol. 19, no. 37, pp. 37–56, 2019.

[8] A. Von Humboldt, *Central-Asien: Untersuchungen über die Gebirgsketten und die vergleichende Klimatologie*, vol. 1. CJ Klemann, 1844.

[9] V. Masson-Delmotte, P. Zhai, H.-O. Pörtner, D. Roberts, J. Skea, P. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, S. Connors, J. Matthews, Y. Chen, X. Zhou, M. Gomis, E. Lonnoy, T. Maycock, M. Tignor, and T. Waterfield, eds., *2018: Summary for Policymakers. In: Global Warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the*

context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty. IPCC, 2018.

[10] O. Steffens, "Gesamtvorhabenbeschreibung maggie: Energetische modernisierung des genossenschaftlichen wohnquartiers margaretenau in regensburg." Forschungsantrag, 2017.

[11] B. für Wirtschaft und Energie, "Das Erneuerbare-Energien-Gesetz." https://www.erneuerbare-energien.de/EE/Redaktion/DE/Dossier/eeg.html, 2020. Accessed: 2020-04-21.

[12] M. Sterner and I. Stadler, *Energiespeicher-Bedarf, Technologien, Integration.* Springer-Verlag, 2014.

[13] C. Lang, F. Steinborn, O. Steffens, and E. W. Lang, "Electricity load forecasting - an evaluation of simple 1d-cnn network structures," in *ITISE 2019-International Conference on Time Series and Forecasting, 25-27 September 2019 Granada (Spain)*, vol. 2, pp. 797–806, 2019.

[14] Pressemitteilung Tennet TSO GmbH, "Aus david wird goliath: Dezentrale kleinanlagen stabilisieren das energiesystem." https://www.tennet.eu/de/news/news/aus-david-wird-goliath-dezentrale-kleinanlagen-stabilisieren-das-energiesystem/, 06 2019. Accessed: 2020-04-21.

[15] Pressemitteilung Tennet TSO GmbH, "Intelligenztest für das stromnetz der zukunft."

[16] Commission for Energy Regulation (CER), "CER smart metering project - electricity customer behaviour trial, 2009-2010." Via the Irish Social Science Data Archive: www.ucd.ie/issda/CER-electricity, 2012. Accessed: 2017-12-01.

[17] X. Dong, L. Qian, and L. Huang, "Short-term load forecasting in smart grid: A combined cnn and k-means clustering approach," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 119–125, IEEE, 2017.

[18] T. Hong, *Short Term Electric Load Forecasting.* PhD thesis, North Carolina State University, 2010.

[19] E. M. de Oliveira and F. L. C. Oliveira, "Forecasting mid-long term electric energy consumption through bagging arima and exponential smoothing methods," *Energy*, vol. 144, pp. 776–788, 2018.

[20] A. El Desouky and M. El Kateb, "Hybrid adaptive techniques for electric-load forecast using ann and arima," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 147, no. 4, pp. 213–217, 2000.

[21] L. Mao and Y. Jiang, "Medium-and long-term load forecasting based on partial least squares regression analysis," *Power System Technology*, vol. 32, no. 19, pp. 71–77, 2008.

[22] D. G. Lee, B. W. Lee, and S. H. Chang, "Genetic programming model for long-term forecasting of electric power demand," *Electric power systems research*, vol. 40, no. 1, pp. 17–22, 1997.

[23] E. Feilat, D. Al-Sha'abi, and M. Momani, "Long-term load forecasting using neural network approach for jordan's power system," *Engineering Press*, vol. 1, pp. 43–50, 12 2017.

[24] R. Achnata, "Long term electric load forecasting using neural networks and support vector machines," *IJCST*, vol. 3, no. 1, 2012.

[25] M. T. Hagan and S. M. Behr, "The time series approach to short term load forecasting," *IEEE transactions on power systems*, vol. 2, no. 3, pp. 785–791, 1987.

[26] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via arma model identification including non-gaussian process considerations," *IEEE Transactions on power systems*, vol. 18, no. 2, pp. 673–679, 2003.

[27] M. Cho, J. Hwang, and C. Chen, "Customer short term load forecasting by using arima transfer function model," in *Proceedings 1995 International Conference on Energy Management and Power Delivery EMPD'95*, vol. 1, pp. 317–322, IEEE, 1995.

[28] N. Mohamed, M. H. Ahmad, Z. Ismail, *et al.*, "Short term load forecasting using double seasonal arima model," in *Proceedings of the regional conference on statistical sciences*, vol. 10, pp. 57–73, 2010.

[29] C. Kuster, Y. Rezgui, and M. Mourshed, "Electrical load forecasting models: A critical systematic review," *Sustainable cities and society*, vol. 35, pp. 257–270, 2017.

[30] A. D. Papalexopoulos and T. C. Hesterberg, "A regression-based approach to short-term system load forecasting," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1535–1547, 1990.

[31] O. Hyde and P. Hodnett, "An adaptable automated procedure for short-term electricity load forecasting," *IEEE Transactions on Power Systems*, vol. 12, no. 1, pp. 84–94, 1997.

[32] P. Ji, D. Xiong, P. Wang, and J. Chen, "A study on exponential smoothing model for load forecasting," in *2012 Asia-Pacific Power and Energy Engineering Conference*, pp. 1–4, IEEE, 2012.

[33] 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.

[34] D. C. Park, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," *IEEE transactions on Power Systems*, vol. 6, no. 2, pp. 442–449, 1991.

[35] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam, "Annstlf-artificial neural network short-term load forecaster-generation three," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1413–1422, 1998.

[36] J. W. Taylor and R. Buizza, "Neural network load forecasting with weather ensemble predictions," *IEEE Transactions on Power systems*, vol. 17, no. 3, pp. 626–632, 2002.

[37] A. Almalaq and G. Edwards, "A review of deep learning methods applied on load forecasting," in *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pp. 511–516, IEEE, 2017.

[38] J. Massana, C. Pous, L. Burgas, J. Melendez, and J. Colomer, "Short-term load forecasting in a non-residential building contrasting models and attributes," *Energy and Buildings*, vol. 92, pp. 322–330, 2015.

[39] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2531–2539, 2010.

[40] A. Kavousi-Fard, H. Samet, and F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting," *Expert systems with applications*, vol. 41, no. 13, pp. 6047–6056, 2014.

[41] J. Moon, Y. Kim, M. Son, and E. Hwang, "Hybrid short-term load forecasting scheme using random forest and multilayer perceptron," *Energies*, vol. 11, no. 12, p. 3283, 2018.

[42] A. Lahouar and J. B. H. Slama, "Day-ahead load forecast using random forest and expert input selection," *Energy Conversion and Management*, vol. 103, pp. 1040–1051, 2015.

[43] E. Busseti, I. Osband, and S. Wong, "Deep learning for time series modeling," *Technical report, Stanford University*, pp. 1–5, 2012.

[44] J. Vermaak and E. Botha, "Recurrent neural networks for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 13, no. 1, pp. 126–132, 1998.

[45] 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*, vol. 10, no. 1, pp. 841–851, 2017.

[46] G.-C. Liao and T.-P. Tsao, "Application of a fuzzy neural network combined with a chaos genetic algorithm and simulated annealing to short-term load forecasting," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 330–340, 2006.

[47] H. Nie, G. Liu, X. Liu, and Y. Wang, "Hybrid of arima and svms for short-term load forecasting," *Energy Procedia*, vol. 16, pp. 1455–1460, 2012.

[48] A. K. Fard and M.-R. Akbari-Zadeh, "A hybrid method based on wavelet, ann and arima model for short-term load forecasting," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 26, no. 2, pp. 167–182, 2014.

[49] T. Hong, P. Wang, and H. L. Willis, "A naïve multiple linear regression benchmark for short term load forecasting," in *2011 IEEE Power and Energy Society General Meeting*, pp. 1–6, IEEE, 2011.

[50] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3642–3649, IEEE, 2012.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[53] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585, IEEE, 2017.

[54] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, 2018.

[55] S. Hosein and P. Hosein, "Load forecasting using deep neural networks," in *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, IEEE, 2017.

[56] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pp. 1483–1488, IEEE, 2017.

[57] W. He, "Load forecasting via deep neural networks," *Procedia Computer Science*, vol. 122, pp. 308–314, 2017.

[58] L. Li, K. Ota, and M. Dong, "Everything is image: Cnn-based short-term electrical load forecasting for smart grid," in *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*, pp. 344–351, IEEE, 2017.

[59] M. Zahid, F. Ahmed, N. Javaid, R. A. Abbasi, H. S. Zainab Kazmi, A. Javaid, M. Bilal, M. Akbar, and M. Ilahi, "Electricity price and load forecasting using enhanced convolutional neural network and enhanced support vector regression in smart grids," *Electronics*, vol. 8, no. 2, p. 122, 2019.

[60] Z. Deng, B. Wang, Y. Xu, T. Xu, C. Liu, and Z. Zhu, "Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting," *IEEE Access*, vol. 7, pp. 88058–88071, 2019.

[61] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied Energy*, vol. 236, pp. 1078–1088, 2019.

[62] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using cnn-lstm neural networks," *Energy*, vol. 182, pp. 72–81, 2019.

[63] P.-H. Kuo and C.-J. Huang, "A high precision artificial neural networks model for short-term energy load forecasting," *Energies*, vol. 11, no. 1, p. 213, 2018.

[64] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[65] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," in *2015 IEEE international conference on image processing (ICIP)*, pp. 452–456, IEEE, 2015.

[66] K. Bimbraw, "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology," in *2015 12th international conference on informatics in control, automation and robotics (ICINCO)*, vol. 1, pp. 191–198, IEEE, 2015.

[67] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[68] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, *Computational intelligence: a methodological introduction*. Springer, 2016.

[69] J. R. Anderson, *Cognitive psychology and its implications*. Worth publishers, 2000.

[70] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[71] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[72] D. O. Hebb, *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[73] B. Widrow and M. E. Hoff, "Adaptive switching circuits," tech. rep., Stanford Univ Ca Stanford Electronics Labs, 1960.

[74] F. Rosenblatt, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. 1961.

[75] M. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[76] H. Niemann, *Klassifikation von Mustern*. springer-Verlag, 7 ed., 2013.

[77] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[78] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[79] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012.

[80] M. Tschoepe, "Beyond sgd: Recent improvements of gradient descent methods," 07 2019.

[81] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[82] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate o (1/kˆ 2)," in *Dokl. akad. nauk Sssr*, vol. 269, pp. 543–547, 1983.

[83] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[84] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of machine learning research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[85] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[86] G. Hinton and T. Tieleman, "Lecture notes in coursera: Neural networks for machine learning," 2012.

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

[88] T. Dozat, "Incorporating nesterov momentum into adam," *Workshop track - ICLR 2016*, 2016.

[89] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[90] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[91] S. Muroga, "Threshold logic and its applications," 1971.

[92] K. Hornik, M. Stinchcombe, H. White, *et al.*, "Multilayer feedforward networks are universal approximators.," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[93] J. Makhoul, A. El-Jaroudi, and R. Schwartz, "Formation of disconnected decision regions with a single hidden layer," in *Proceedings of the International Joint Conference on Neural Networks I*, vol. 455, p. 460, 1989.

[94] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[95] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0588–0592, IEEE, 2017.

[96] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv e-prints*, mar 2016.

[97] "Press release. nobelprize.org. nobel media ab 2020." https://www.nobelprize.org/prizes/medicine/1981/press-release/. Accessed: 2020-04-10.

[98] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[99] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[100] R. J. Rossi, *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

[101] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

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

[103] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[104] R. Couronné, P. Probst, and A.-L. Boulesteix, "Random forest versus logistic regression: a large-scale benchmark experiment," *BMC bioinformatics*, vol. 19, no. 1, p. 270, 2018.

[105] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[106] A. Lahouar and J. B. H. Slama, "Day-ahead load forecast using random forest and expert input selection," *Energy Conversion and Management*, vol. 103, pp. 1040–1051, 2015.

[107] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[108] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[109] D. Nielsen, "Tree boosting with xgboost-why does xgboost win" every" machine learning competition?," Master's thesis, NTNU, 2016.

[110] D. D. M. L. Community, "Awesome xgboost - curated list of xgboost kaggle usercases." https://github.com/dmlc/xgboost/tree/master/demo, 22015. Accessed: 2021-03-27.

[111] V. Sandulescu and M. Chiru, "Predicting the future relevance of research institutions-the winning solution of the kdd cup 2016," *arXiv preprint arXiv:1609.02728*, 2016.

[112] "Met Éireann, historical weather data." https://www.met.ie/climate/available-data/historical-data. Accessed: 2020-10-07.

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

[114] xgboost developers, "Xgboost documentation." https://xgboost.readthedocs.io/en/latest/. Accessed: 2020-10-14.

[115] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[116] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[117] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[118] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[119] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[120] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[121] D. Mishkina, N. Sergievskiyb, and J. Matasa, "Systematic evaluation of cnn advances on the imagenet," *Center for Machine Perception, Faculty of Electrical Engineering*, 2016.

[122] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[123] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[124] T. K. Wijaya, M. Vasirani, S. Humeau, and K. Aberer, "Cluster-based aggregate forecasting for residential electricity demand using smart meter data," in *2015 IEEE international conference on Big data (Big data)*, pp. 879–887, IEEE, 2015.

[125] S. Arora and J. W. Taylor, "Forecasting electricity smart meter data using conditional kernel density estimation," *Omega*, vol. 59, pp. 47–59, 2016.

[126] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid cnn-lstm model for short-term individual household load forecasting," *IEEE Access*, vol. 8, pp. 180544–180557, 2020.

[127] C. Tian, J. Ma, C. Zhang, and P. Zhan, "A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network," *Energies*, vol. 11, no. 12, p. 3493, 2018.

[128] A. Gasparin, S. Lukovic, and C. Alippi, "Deep learning for time series forecasting: The electric load case," *arXiv preprint arXiv:1907.09207*, 2019.

Acknowledgements

I would like to thank the following people, without whom this thesis would not look the way it does and without whom my time at the University of Regensburg would not have been the same.

Many thanks to Elmar Lang for offering me the possibility to do a doctorate, for his friendly guidance during the research, and, above all, for the informal and uncomplicated atmosphere he creates in his group. I further would like to thank Flo and Marinus for the discussions and good times we had in our office. Many thanks also to Simon for his technical support and the effort he put into enabling remote computing for me.
A big thank you to Haley for proof reading this thesis, which I do not take for granted.
Furthermore, I would like to thank everyone who has turned the last few years into what they have been - a genuinely great time that made me the person I am today. Thank you Wacki, Chrissy, Tommy, Thea, Josef, Simon and so many more. A special shout-out goes to Kathl for keeping me motivated during the last months. I also would like to thank my friends from Hof - Martin, Sophia, Timo, Moni, Lise, Anka, Verena, Konstantin, and Jule - who shared the excitement of studying and doing the doctorate with me.
Finally, a big thanks to my parents and my brother for all the support and that I can always rely on you.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe des Literaturzitats gekennzeichnet.

Regensburg, den 27.04.2021

Christian Lang

Publications

- Christian Lang, Florian Steinborn, Oliver Steffens, Elmar W. Lang, *Electricity Forecasting - An Evaluation of Simple 1D-CNN Network Structures*, Proceedings ITISE 2019 - International Conference on Time Series Forecasting, pp. 797-806 (2019)
- Christian Lang, Florian Steinborn, Oliver Steffens, Elmar W. Lang, *Applying a 1D-CNN Network to Electricity Load Forecasting*, Theory and Applications of Time Series Analysis, Springer International Publishing, pp. 205-218 (2020)