Household Electricity Load Forecasting for Demand Side Management Applications

Matthijs Ruben Kok

February 27, 2023
Matthijs Ruben Kok

University of Twente

UNIVERSITY
OF TWENTE.

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Computer Architecture for Embedded Systems (CAES) group

Master's Thesis, M-CS

Household Electricity Load Forecasting for Demand Side Management Applications

Matthijs Ruben Kok

Chair Dr.ir. M.E.T. Gerards
EEMCS, CAES
University of Twente

Supervisor Prof.dr. J.L. Hurink
EEMCS, MOR
University of Twente

External Dr. E. Mocanu
EEMCS, DMB
University of Twente

Company Supervisor R. Binnendijk
CGI

February 27, 2023

Matthijs Ruben Kok

Household Electricity Load Forecasting for Demand Side Management Applications

Master's Thesis, M-CS, February 27, 2023

Supervising Committee: Dr.ir. M.E.T. Gerards, Prof.dr. J.L. Hurink, Dr. E. Mocanu, and R. Binnendijk

University of Twente

Computer Architecture for Embedded Systems (CAES) group

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)

Drienerlolaan 5, 7522 NB, Enschede, The Netherlands

Abstract

As part of the energy transition, many residents have installed rooftop solar panels and started using more electrical appliances, such as electric vehicles (EVs) and heat pumps. This decentralized production and consumption puts pressure on the current electricity grid. To flatten load peaks and prevent overloading of grid components, demand side management (DSM) techniques can be employed to match electricity supply and demand. Some DSM techniques require house load predictions. However, there are challenges regarding the application of load predictions, the variation between household loads, and the volatility of a load within a single household. To overcome these challenges, this thesis proposes an online "rolling horizon" forecasting model that is continuously being trained on the smart meter data of a single household. It can therefore autonomously adapt to the load profile behaviour of that household and to structural changes in electricity consumption over time.

Based on two correlation analyses, relevant feature variables are selected as input for the proposed forecasting model. First, the house load series exhibits a clear daily seasonality. Second, several weather variables are shown to highly correlate with the house load, but only in case the house has rooftop solar panels installed. The rolling horizon of the forecasting model is explained with a black-box paradigm, and is simulated using discrete time (15-minute) intervals. The proposed forecasting model uses artificial neural networks (ANNs) to make day ahead predictions of the house load with a 15-minute granularity. The hyperparameters of the ANNs are chosen based on a grid search, and are shown to have an influence on the length of the startup period of the forecasting model, as well as its sensitivity towards feedback. Four different forecasting model architectures are compared (on neighbourhood- and house-level) that apply ANNs in a different way. Out of the discussed architectures, the hybrid architecture performs the best overall. Furthermore, prediction errors differ significantly per house load and the results indicate a linear relationship between the house load standard deviation and prediction error. Over time, the proposed forecasting model error is higher during late spring and early summer suggesting that load predictions are worse when there is a lot of PV production. Finally, the impact of updating predictions is evaluated in two DSM scenarios. This thesis shows that predictions with a specific forecast window property are more suitable to be updated each time step.

Dankwoord

Voor mijn afstudeerscriptie was ik overal in Nederland: Soms thuis in Enschede of bij de Energy Group op de Universiteit in Twente, maar soms ook bij mijn ouders in Winsum of op kantoor bij CGI in Rotterdam of Arnhem. In totaal heb ik denk ik net zo veel uren gewerkt in de trein als achter een bureau. Gelukkig was er niet alleen overal een bureau voor mij beschikbaar, maar ook ontzettend fijne mensen die mij hebben geholpen met mijn onderzoek. Graag wil deze mensen hierbij bedanken.

Marco en Johann, bedankt voor jullie inzet en hulp tijdens het begeleiden van mijn afstudeeronderzoek. Onze twee-wekelijkse meetings waren altijd erg interessant en ik zat na elke meeting weer vol nieuwe ideeën voor mijn onderzoek. Ook al was jullie handschrift niet altijd even goed te lezen, bedankt voor jullie gedetailleerde feedback. Hier heb ik ontzettend veel aan gehad tijdens het schrijven van mijn scriptie. Furthermore I would like to thank Elena, who was only later asked to be my external supervisor, but nevertheless showed a huge interest in me and my project.

Daarnaast bedank ik graag Raymond voor mijn begeleiding en introductie binnen CGI. Onze meetings vond ik altijd gezellig en effectief. In het begin was ik helemaal nieuw en onbekend binnen het bedrijf, maar door jouw hulp heb ik heel snel een hele hoop mensen leren kennen, waaronder Robert, die elke maandag- en woensdagmiddag de stand-ups heeft geleid. Robert, het blijft zeker niet onopgemerkt hoeveel energie je steekt in studenten die afstuderen bij CGI, waarvoor mijn enorme waardering. Daarnaast wil ik Camera bedanken voor de hulp bij mijn vragen over Engelse grammatica. Ook alle andere CGI interns, dankjulliewel voor de gezellige koffiepauzes, lunches, en uitstapjes. Ondanks dat ik niet van jenever hou, houd ik me zeker aanbevolen voor een volgende proeverij!

Verder bedank ik graag iedereen van de Energy Group, en met name van het energie-hok, voor de gezellige pauzes en lunchwalks die we hebben gehad en natuurlijk alle keren dat ik jullie passen mocht lenen voor gratis kopjes koffie. In het bijzonder wil ik Victor graag bedanken voor het beantwoorden van al mijn vragen over de GridFlex data set, Ivo voor het prachtige template waarin dit verslag geschreven is, en Gerwin voor de uitleg van driefasespanning en het asynchronous event-driven profile steering algoritme.

Als laatste wil ik graag papa, mama, en Selinde bedanken voor hun steun tijdens mijn afstuderen. Het was altijd fijn om in het weekend weer naar thuisthuis te gaan om ‘s avonds gezellig te gaan borrelen en de Harry Potter marathon te hervatten. Maar ook inhoudelijk heb ik met jullie vele discussies kunnen voeren over mijn afstudeeronderwerp. Zonder jullie support was dit verslag nooit wat geworden.

Contents

1 Introduction

Chapter Objective: This chapter introduces the thesis and states the research questions.

Chapter Contents

1.1 Energy Transition

The energy transition involves the transition from an energy system based on fossil energy sources to an energy system based on renewable energy sources (RES). With the Paris Agreement in 2015, the global community agreed to try to limit global warming to a temperature rise of $1.5°C$ above pre-industrial levels. With the 2030 Climate Target Plan, the European Commission proposed to reduce greenhouse gas emissions to at least 55% below 1990 levels by 2030 [1], in line with the Paris Agreement and the goal to make the EU climate neutral by 2050. In order to do this, the EU 2030 climate & energy framework includes targets to increase the share of renewable energy to 32% and improve energy efficiency by 32.5% [2]. Climate Action Network (CAN) Europe developed the Paris Agreement Compatible (PAC) scenario, suggesting targets that mean a phase-out of of coal by 2030, fossil gas by 2035, and fossil oil by 2040 [3]. According to these plans, more RES such as solar parks and wind farms are built and connected to the grid to replace existing fossil energy production plants [4], [5]. Also, an increasing number of household residents are installing solar panels on the roof of their house. For example, in houses, districts and neighbourhoods in the Netherlands, there were 960248 installations with a capacity of 3236417kW in 2019 [6] and 1267651 installations with a capacity of 4488623kW in 2020 [7]. This is an increase of 32,0% and 38,7% respectively. Because these 'prosumers' are now locally generating electricity, the energy system

has to deal with a distributed production of energy. However, the electricity grid was originally not designed for this. In line with the supply side, the demand side is undergoing an electrification trend. Consumers are using more and more electricity, partly as a result of the increasing use of electric vehicles (EVs) and heat pumps [8], [9].

Hence, there is an increase in distributed demand and supply of electricity. However, the amount of local energy consumption and production are generally not equal at all times. Figure 1.1 displays a so-called "Duck curve", showing the timing imbalance between peak demand and production. This curve is typical for households that have rooftop solar panels installed.

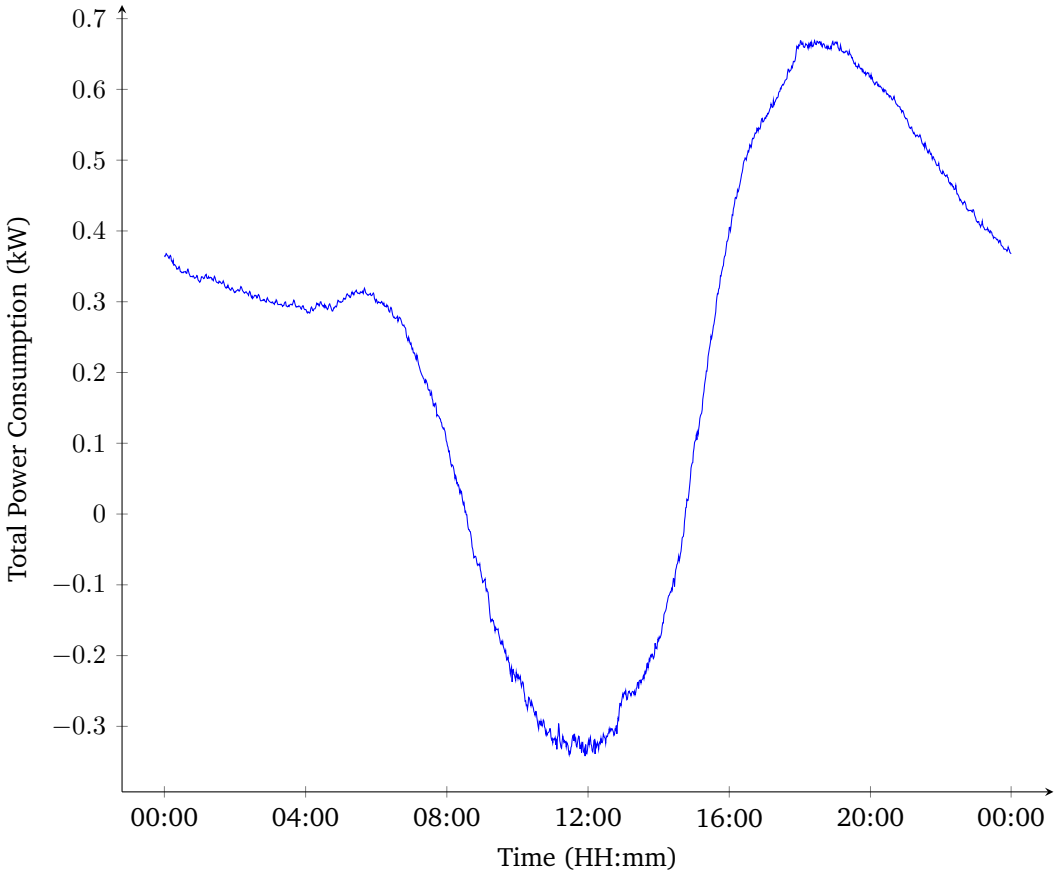


Figure 1.1: Average day load profile for the "average" household (average over all houses in the GridFlex data set [10])

Solar panels produce most electricity when the sun shines, roughly in the middle of the day. Residents consume most electricity in the morning and mostly in the evening. As a result, "peaks" arise on the grid, of both electricity consumption and production. If these peaks are too large and reach the capacity of the grid, congestion can occur, causing the components in the grid to become overloaded. In

many regions in the Netherlands, there is already a shortage in transmission capacity on the electricity grid, causing the need for congestion management [11].

To prevent overloading, these peaks should be addressed. A possible solution may be to upgrade grid components, such as the physical grid itself, intermediate stations, and transformers. However, this is very expensive and takes a very long time. A cheaper and faster solution may be to first "flatten" these peaks as much as possible. In order to flatten peaks, one can change the time and amount of electricity supply and that of electricity demand. Unfortunately, the electricity yield of many RES is weather dependent, and therefore intermittent and hard to control. In order to match supply and demand, more control is required on the demand side of the energy grid: demand side management (DSM). DSM is explained in more detail in Chapter 3.

1.2 Problem Statement And Contribution

Some DSM techniques rely heavily on forecasts. Instead of purely focusing on making accurate predictions by itself, finding out how predictions influence the performance of these DSM techniques and how different DSM techniques currently make use of available predictions, potentially provides insights for making predictions that are more meaningful in this context. However, there are a few challenges regarding household electricity consumption forecasting. Achieving a good forecasting performance in the residential sector proves to be difficult, because there are many aspects that need to be considered when forecasting household electricity consumption, such as seasonality, scale, exogenous variables, and the forecasting horizon. These are all explained in more detail in Section 2.2. To this end, many different forecasting models have been proposed, often applying machine learning. A review of several forecasting models is given in Section 2.3.

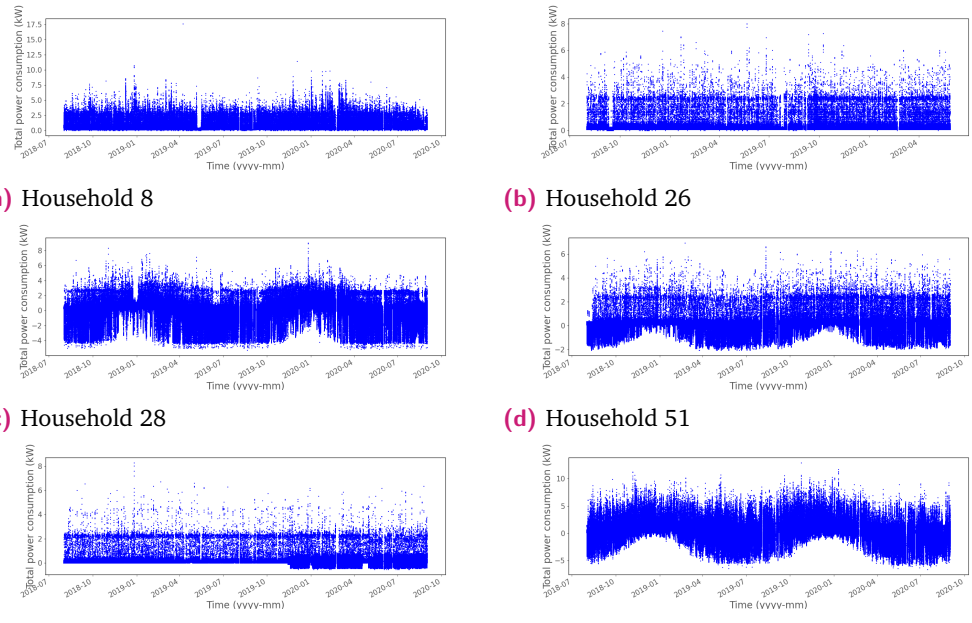
However, there are three main problems with forecasting models for the electrical load of a household. First, many existing forecasting models strive for a high accuracy, but do not take into account the application of the predictions (Section 1.2.1). Furthermore, there are two main challenges inherent to forecasting household power consumption. Namely, (1) the large load variation *between* multiple households (Section 1.2.2), and (2) the large load volatility *within* a single household (Section 1.2.3). To solve these challenges, a novel forecasting model is proposed in Section 1.2.4.

1.2.1 Forecasting For A DSM Algorithm

Although the purpose (e.g. an energy management system for a particular building) of the prediction is often defined, the performance of the applied DSM (scheduling) algorithm is rarely considered when making a prediction [12]–[15]. For example, typically DSM algorithms require short-term horizon (see Section 2.2.4 for more information) load predictions, but many existing forecasting models predict for longer horizons ahead. Furthermore, a lot of research focuses on making the best predictions of electricity consumption. However, since predictions cannot be perfect, prediction errors will always occur. Therefore, it is important to consider how to deal with electricity forecasting errors when the forecasts are used as input for a DSM algorithm. Section 3.2 discusses this topic in more detail.

1.2.2 Variation Between Houses

One of the characteristics of household load profiles is that they are very different for different households. This is shown in both Figure 1.2 and Figure 1.3. Figure 1.2 displays all measured data points over a two years time period from six households that were randomly picked from the GridFlex data set [10], with the total power consumption (kW) on the y-axis and the time on the x-axis. Note that some households have a negative total power consumption, which indicates that the household is supplying electricity back to the grid.



(a) Household 8 **(b)** Household 26

(c) Household 28 **(d)** Household 51

(e) Household 52 **(f)** Household 29

Figure 1.2: Household total power consumption over a two years period: from 2018-08-01 to 2020-08-31, for six households from the GridFlex data set [10]

The houses that were selected are summarized in Table 1.1. Some of the houses have solar panels or an electric battery. Figure 1.2 shows that the load of the households with solar panels have a distinctive seasonal pattern compared to the households without solar panels.

House	Solar	Battery
8	-	-
26	-	-
28	V	-
51	V	-
52	-	V
29	V	V

Table 1.1: Six houses, randomly picked from the GridFlex data set [10]. For each house, the table indicates whether it has solar panels or a battery

Figure 1.3 displays the average daily load profile of these six households. Each profile is displayed as a red line with the total power consumption (kW) on the y-axis and the time on the x-axis. Note that the scale of the y-axes for the graphs in Figure 1.3 differ. In order to compare the graphs, the average daily load profile of the average household (from Figure 1.1) is displayed in the background using a grey line, which is the same for each graph in Figure 1.3. Figure 1.3 shows that the average daily load profile can be very different for different households. What is immediately clear from the graphs, is that the households with solar panels (PV) have a negative peak during the middle of the day (House 28, 51, and 29), which is not apparent for the houses without solar panels (House 8, 26, and 52). Apart from the solar production peak during the middle of the day, there are more differences between the chosen average profiles in Figure 1.3. The visualizations of household profiles from the work by Ziekow et al. [16] confirm that such differences are very common for load profiles of households. These differences may be caused by exogenous variables, which are further explained in Section 2.2.3.

The problem with the large variation in load between different houses, is that developing a forecasting model that makes accurate load predictions for different households becomes very difficult. In order to reduce peaks on the LV grid or at transformer level, sufficient forecasts of household base loads need to be available. Therefore, a forecasting model needs to be able to predict the load for several households in the neighbourhood (or even larger area). However, the performance of such a forecasting model may be negatively impacted by the differences between the load characteristics of every household. When a forecasting model is trained using a training data set, this training data set must accommodate sufficient variety to create a forecasting model that is applicable to many households. A forecasting model might not generalize well to household loads that it has not trained for (outside of training data). In other words, the load forecasting model may not make

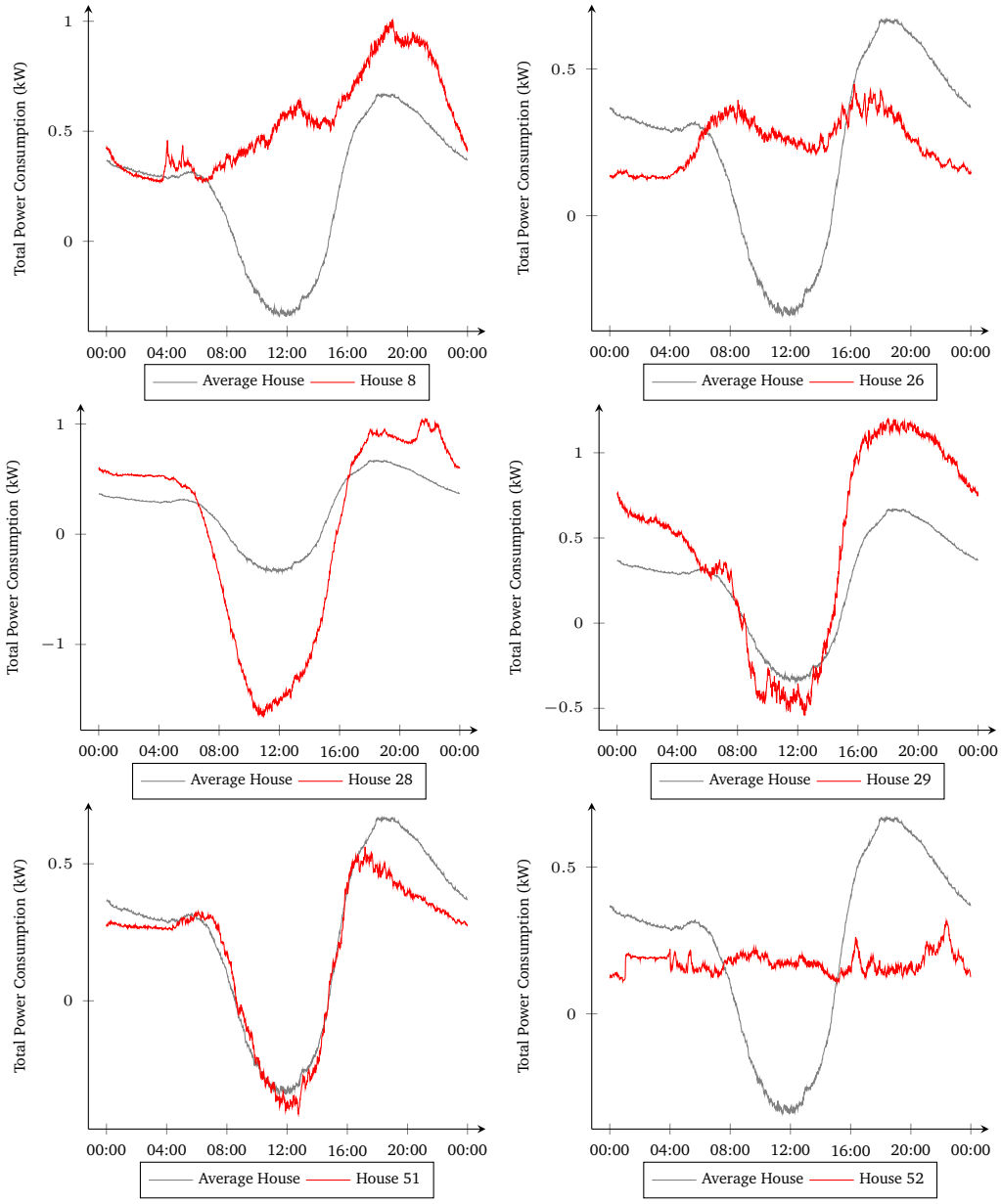


Figure 1.3: Household average daily consumption

accurate load predictions for a new household, if the training data set (that contains the loads of several known households) is not sufficiently representative of the actual variation between houses. For example, a model that was trained using a small data set including only a few buildings, may not work well when it is applied to buildings with different occupant behaviour, physical characteristics, and weather conditions [13]. However, acquiring a very large representative data set might not immediately solve the problem either. A very general forecasting model trained on a huge data set with many different types of households might still not perform well on a specific household, due to the substantial influence of the large number of "common" households. The load profile of a single, possibly typical, household might be very different from the "common" load profiles, causing the predictions

of the forecasting model to always be inaccurate. Some researchers suggest that it is necessary to create forecasting models that take into account the decisions and preferences of the building occupants, which are different for different buildings [14]. Ideally, there would be one general forecasting model that is able to accurately predict the electricity consumption for all households, but unfortunately there exists no such generic forecasting model [13].

1.2.3 Volatility Within Household Load Profile

The load volatility of a single house load profile is much higher than the volatility of a profile at a larger scale. This is discussed in more detail in Section 2.2.2. Additionally, new smart meter data is continuously generated and new patterns might arise in this data. Due to this, existing energy consumption patterns might become outdated and possibly contradict earlier observed patterns. In many applications, a forecasting model would be trained once and then applied in the real world. For example, Molderink et al. [17] consider that characteristics of a house are static for their prediction model. However, such a model would not be able to take into account future structural changes in electricity consumption or human behaviour. Every time that a structural change in energy consumption arises, a new model would have to be trained, since each structural consumption change causes predictions based on the old model to deviate too much from the actual electricity consumption of the household [18]. Especially in the long run, domestic consumption patterns are likely to change, e.g. a new appliance or EV might be bought, an old highly consuming device is replaced by a more energy efficient one, a new family moves in, a child is born or moves out. In each of the cases, the consumption patterns of the household may change drastically.

1.2.4 Objective

In summary, there are challenges regarding the application of power consumption predictions, the variation between household loads, and the volatility of a load within a single household. First, many forecasting models strive for the best accuracy but do not take prediction errors into account for the application or train based on the performance of the DSM algorithm where the predictions are going to be used. Second, the variation between households causes challenges with finding a representative training data set. Also, creating a general forecasting model that performs well for all types of households is hard, due to the differences between households caused by solar panels, number of residents, geographical factors, etc. Third, the challenges regarding volatility within a household are that a household load profile is not smooth due to its small scale, more irregular than that of a

commercial or industrial building, and that it can change over time due to the purchase of new devices or a change in household occupants.

Therefore, this research proposes an online "rolling horizon" forecasting model that is continuously being trained on the smart meter data of a single household. It can therefore adapt to the load profile behaviour of that household and to structural changes in electricity consumption over time. Such a forecasting model overcomes the variation problem as it adapts its predictions to the load consumption behaviour of a single household, by only training on the data of that household. This also resolves the problem of acquiring a representative data set. Furthermore, the forecasting model takes into account the volatility problem by adapting its predictions to structural changes in electricity consumption over time. For this purpose, artificial neural networks (ANN) are used as they are very well suited for continuous training (see Section 2.3 for a motivation). ANNs are explained in more detail in Section 2.4. We aim for a forecasting model that runs autonomously. Considering the fact that prediction errors will always exist, we propose to dynamically update the predictions that are provided to the DSM algorithm, so that the DSM algorithm can take into account these errors and possibly adapt its schedule.

1.3 Research Questions

This section presents the research questions for this research. For each question, a small description is provided.

RQ 1 What input data correlates the most with the residential electricity consumption that needs to be predicted?
The goal of this question is to define the input to the proposed forecasting model. This includes defining what (exogenous) variables can be used to predict the household electricity consumption, and what subset of the historical house electricity consumption should be used to make a prediction.

RQ 2 How can an online short-term forecasting model be determined to predict household electricity consumption?
The goal of this question is to define exactly how the proposed forecasting model is used to make a prediction. Some variations of the proposed forecasting model can be tested and their prediction errors can be compared to find the best performing version.

RQ 3 How to make meaningful predictions of household electricity consumption that can be used as input to a DSM scheduling algorithm?

The goal of this question is to define a relationship between the predictions of the proposed forecating model and the performance of a DSM scheduling algorithm. Predictions should be made with the application in mind, meaning that predictions and scheduling algorithms should not be treated as separate problems.

1.4 Thesis Structure

For the remaining part of this thesis, Chapter 2 discusses literature that is related to electricity load forecasting in general and methods and techniques that are used to predict household electricity consumption. In this chapter, Section 2.2 discusses characteristics of electricity load forecasting that need to be taken into account. Section 2.3 reviews several forecasting techniques and motivates why ANNs are chosen to be used in the proposed prediction model. Section 2.4 provides a background of ANNs and how they can be used in energy context, and Section 2.5 discusses how they can be evaluated. Chapter 3 discusses literature related to DSM. In this chapter, Section 3.2 introduces DSM through an example. Section 3.3 explains smart grids and Section 3.4 explains energy management systems (EMS). Section 3.5 and Section 3.6 present two different categorizations of DSM techniques. Finally, Section 3.7 and Section 3.8 explain two specific DSM techniques in more detail. Chapter 4 describes input data used for the proposed forecasting model. In this chapter, Section 4.2 describes the GridFlex [10] data set, which is the main data set used in this research. Section 4.3 describes how the data was cleaned and Section 4.4 describes what features are extracted and selected to be used as input to the proposed forecasting model. Chapter 5 explains the proposed forecasting model itself. In this chapter, Section 5.2 explains how the rolling horizon is implemented. Section 5.3 explains and compares the various ANN architectures. Section 5.4 explains the design choices that were made for the ANN architectures. Finally, Section 5.5 discusses how the predictions of the proposed forecasting model can be used as input to a DSM technique, using two scenarios. Chapter 6 shows and discusses the results of applying the proposed forecasting model. In this chapter, Section 6.2 discusses the performance of the different architectures of the proposed forecasting model. This section compares the performance for different households, evaluates the performance convergence and performance over time, and the impact of the forecasting horizon by evaluating the performance of individual components of the forecast window. Furthermore, a simulation where predictions of all forecasting models (corresponding to all houses) are aggregated, is evaluated to give an indication of the performance on neighbourhood level. Section 6.3 discusses the two scenarios (from Section 5.5) where the performance of a specific DSM algorithm is evaluated, when it uses the predictions of the proposed forecasting model. Finally, Chapter 7 answers the research questions from Section 1.3 and concludes this thesis.

2 Forecasting

Chapter Objective: *This chapter presents challenges of forecasting electricity consumption and reviews several forecasting models.*

Chapter Contents

2.1 Introduction

This chapter describes related work that is used to develop the proposed forecasting model that is explained in Chapter 5. First, Section 2.2 discusses some characteristics specific to electricity consumption forecasting. Then, Section 2.3 reviews several time series forecasting models that are used for predicting electricity consumption and provides a motivation for ANNs in particular. Subsequently, Section 2.4 provides a brief background on ANNs and how they can be used in energy context. Finally, Section 2.5 discusses how forecasting models can be evaluated when working with electricity consumption data.

2.2 Electricity Consumption Forecasting

There are a few things specific to forecasting electric load, as opposed to forecasting any other type of time series. These include seasonality and trends, the effects of the scale and aggregation of the electric loads, how different exogenous variables may influence the load, and how forecasts with different forecasting horizons have different applications in practice. When choosing and developing a forecasting model to predict electric load, these things should be taken into account.

2.2.1 Seasonality And Trends

Hippert et al. [19] stated that an electricity load series exhibits seasonality, because it is strongly auto-correlated. In other words, the load at a given hour depends on itself at previous moment in time (a historic load observation). Such historic load observations are also called lagged load observations, or "lags" for short. There is currently no standard way to determine what and how many lags (or, equivalently, what and how much history) are needed to accurately predict the future [20]. However, several methods are suggested to determine what lagged load values should be used as inputs to the forecasting model. When working with hourly data, the current load value especially depends on the load at the previous hour, on the load on the previous day at exactly the same hour, and also on the load on the same day of the previous week at the same hour [19]. To formalize this, Yuan et al. [21] showed using a scatterplot that the load on the next day is roughly linearly correlated with the load on the current day (regardless of week- or weekend-days). Therefore, they decided to select lagged load values based on a simple correlation measure. Some other authors [19], [20] used the Box-Jenkins [22] method and selected the most useful lags using the autocorrelation function (ACF) and the partial autocorrelation function (PACF) [23]. The downside of these methods is that they only focus on the *linear* correlation between the lag and the current load. Hence, using this method, lags that are only *nonlinearly* correlated to the current load will not be selected. Therefore, some studies use nonlinear tests [19], [20]. Next to seasonality, trends are present in an electricity load series. In order to capture those trends, Humeau et al. [24] added the first and second derivatives of the historical load as input to their forecasting model, to give an indication of the evolution of the electricity consumption.

2.2.2 Scale And Aggregation

When making predictions of the electricity load, the scale (amount of aggregation) of the load is important. The lowest level of scale describes the load of a single device (device-level). The load at a household level, is simply the aggregation of all appliances and devices within a household. Hierarchically, one can increase the load scale. The load at transformer level or neighbourhood level, is the aggregation of all the households connected to the transformer or within a neighbourhood, respectively. Continuing this way, the scale can be increased and smaller loads can be aggregated to ultimately arrive at the load at regional or national level. However, the load profile at a different scale has a different shape. As already briefly mentioned in Section 1.2.3, the load shape is often more predictable at larger scale than at a small scale. The reason for this is that the power consumption at a large scale (e.g. a neighbourhood) is less capricious and more "smooth" compared to

the power consumption at a small scale (e.g. a household). This is also shown in Figure 2.1, where the aggregated load of all households is compared to the load of four single (randomly selected) households, of the GridFlex data set [10] at a randomly selected day. The aggregated load profile is essentially the load at

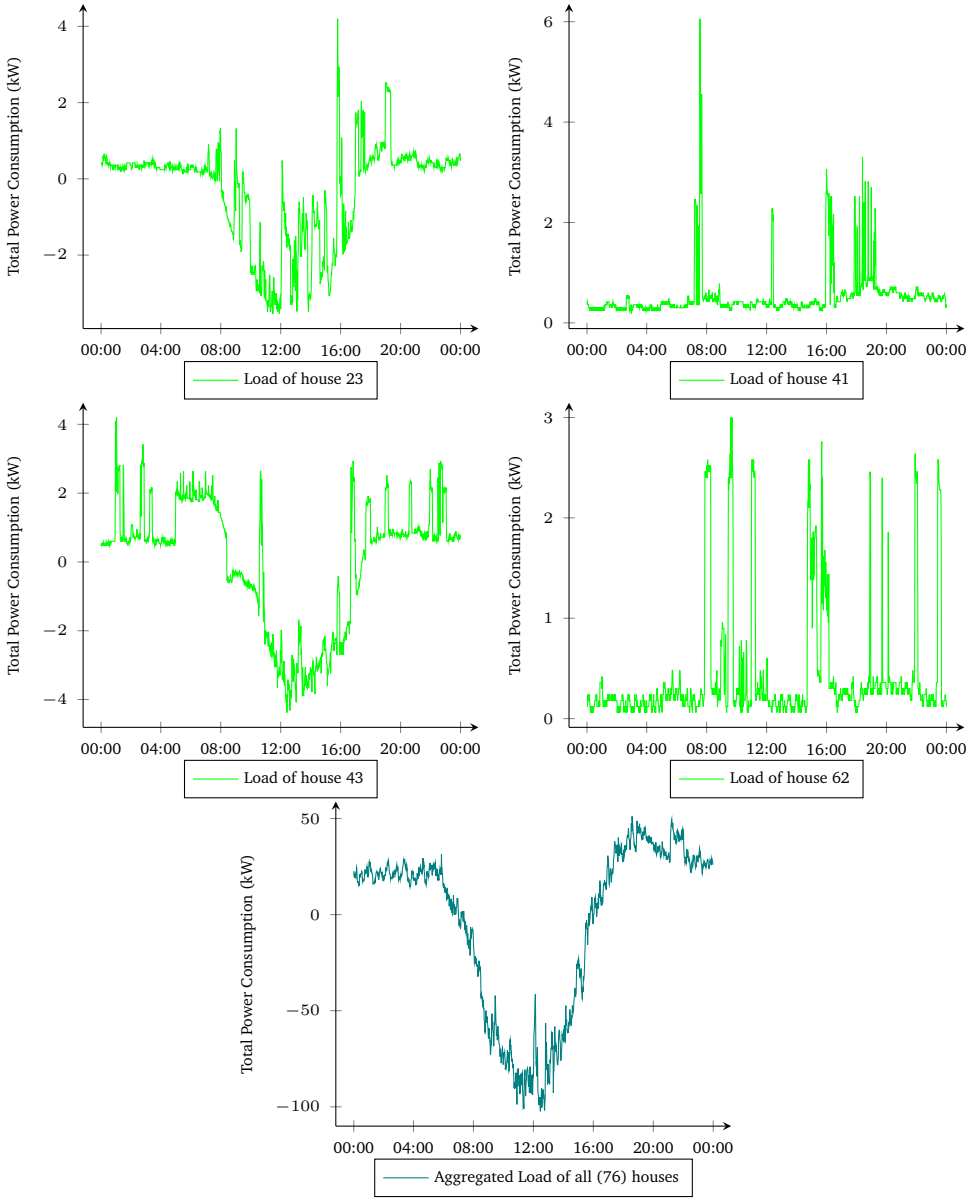


Figure 2.1: Load profile of a randomly selected day (2020-04-10) of the aggregation of all households versus four randomly selected single households (23, 41, 43, 62), of the GridFlex data set [10]

transformer (or neighbourhood) level, so the law of large numbers applies [25] as extremes of individual loads are canceled out. Furthermore, the variance within individual household load profiles is generally larger than the load variance in profiles of commercial or industrial buildings [26]. One of the reasons for this is that there is more uncertainty in the energy demand of households compared to the energy demand of commercial or industrial buildings. Household occupants could

be at home any time of the day, and their behaviour (which appliances they use at what times) is very unpredictable, resulting in an unpredictable power consumption too. Commercial buildings tend to have opening and closing times, and employees often have more predictable and periodic working hours. Industrial buildings or factories often use power for specific repeatedly used processes. Another reason for the high variance of a household, is that a household has a relatively small base load, compared to a larger building [27]. Yildiz et al. [26] showed a positive correlation between the performance error of several forecasting models and the standard deviation of household load profiles. Their research indicates that more volatile load profiles are more difficult to predict for the forecasting models that were tested. Javed et al. [28] found that the volatility of a single house load is two orders of magnitude larger than the volatility of a regional load. This can make forecasting at a small scale much more difficult than forecasting at a larger scale. Sevlian et al. [29] introduce a scaling law that describes the effect of aggregation on load forecasting accuracy, and found that the aggregation of loads can reduce this volatility and make the load more smooth and predictable up to a certain point. In other words, aggregated loads are less random, fluctuating, and chaotic than a household load curve. Often regular patterns in a load profile are more easily found in aggregated load profiles. Tidemann et al. [30] also

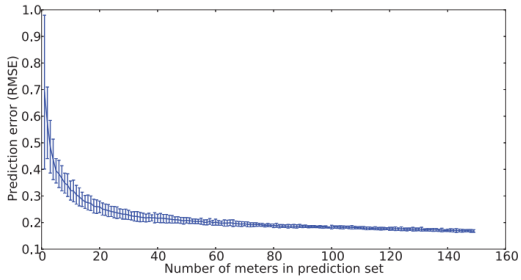


Figure 2.2: Prediction Error (RMSE) for different levels of aggregation, from [30]

investigated the effects of scale on load prediction algorithms and plotted how the prediction error changes for a different number of aggregated meters, in Figure 2.2. They found that different algorithms are suitable at different scales. Humeau et al. [24] try to improve consumption forecasting by using the statistical relation between household electricity consumption series. They also try to determine which forecasting technique, out of support vector regression (SVR) and linear regression, is best adapted to which scale. When the loads of more than 32 houses were aggregated (district scale), SVR outperforms linear regression. At a household scale (32 houses or less), they found that linear regression is the better alternative.

In order to predict electricity consumption at neighbourhood level for reducing peaks on the grid, different strategies can be used. Wijaya et al. [31] explained three strategies:

1. A *top-down* approach, where the load profile of all households is summed into one single aggregate load profile. Subsequently, this aggregate load profile can be forecasted.
2. A *bottom-up* approach, where the load profile of each household is forecasted separately. Then the forecasts are aggregated.
3. A *hybrid* approach, where the households are segmented into k clusters and in each cluster the household energy consumption is aggregated. Then the aggregate consumption of each cluster is forecasted separately. Finally, the k forecasts are aggregated into one aggregate forecast.

A potential risk of the bottom-up approach is that household prediction errors could accumulate if the forecasts are aggregated [32], which happens if the household prediction errors correlate. However, the top-down approach will not have any forecast information about the individual households, and predictions of individual house loads, as opposed to the aggregated load of all houses on neighbourhood level, are required for many (decentralized) DSM techniques (e.g. [17], [33]). Predictions of the individual house load profiles are used by decentralized DSM to ensure local flatness of load profiles. Keeping individual house profiles as flat as possible has several advantages [32]. Firstly, self-consumption is increased. Self-consumption means consuming the energy that was produced on-site. For example, a household with a rooftop solar PV system may try to maximize self-consumption by scheduling appliances at times when there is PV generation. This is very beneficial for the utility as it lowers the load on the local grid. However, Gerards et al. [34] mention that there are conflicting objectives between a network operator (DSO) and the households, regarding self-consumption. The second reason for local flat individual house profiles is that it improves power quality and helps keeping voltage levels within bounds, which is important because the NEN-normatives require that the voltage on the low voltage (LV) grid in the Netherlands does not deviate more than 10% from the standard 230V [35]. Finally, local flat individual house profiles decrease losses and avoids overloading [32].

2.2.3 Exogenous Variables

Different exogenous variables affect the load profile, depending on the scale of the load profile. The load of an appliance depends on device specific properties and can be turned on and off by a user. The load of a household depends on the properties of the house, the number of household occupants, when and how much time they spend at home, the number an type of owned appliances, and the type of house [28]. The load in a neighbourhood depends on geographical location of the neighbourhood and the type of buildings located there. The load profiles of households displayed

in Figure 1.3 are all from the village Heeten, in the Netherlands [10]. The load on national level may even depend on national electricity price changes [36].

The most used exogenous variables are weather and social variables [19]. An often used weather variable is the air temperature which is nonlinearly correlated to the electricity load of a household, as electricity demand increases on either very hot or very cold days, through the use of (electric) air conditioning and heating appliances. Other often used variables are the relative humidity and wind speed. These influence how a person perceives the weather, which also results in using previously mentioned devices. Lastly, most studies make use of the current weather variables instead of weather forecast variables, because forecast errors may be propagated in the performance of the ANN [19]. When determining the most relevant variables for prediction models, some researchers choose the ones with the highest correlation with the forecasted electricity consumption [14], [24], [26], [28], [31], [37], [38]. Furthermore, data can be transformed before using it as input for the prediction model [13], [30], [38], [39].

Javed et al. [28] found that the use of anthropological aspects and structural variants as variables, has a positive effect on the performance of their forecasting model. Here anthropological aspects are characteristics of the house occupant, such as the number of occupants, the age of the occupants, the owned devices, the level of education, and other demographic information. Structural variants refer to physical characteristics of the house such as the type of house, how many windows it has, the type of walls, ventilation, how well the house is insulated, and the size of living space. In order to collect data, they distributed a survey with questions to the household occupants. Examples of information that was asked in the survey were the average indoor temperature, a list of household appliances, building year of the property, etc. Their initial results, however, showed that the correlation between the house load and these variables is weak. They found that the strongest influence on demand is weather. They created one forecasting model that is able to differentiate the output (the predictions) for each household, based on the input variables. They found that using this information can greatly enhance forecasting of an individual household load, when the forecasting model has to differentiate between households. The downside of this approach is that a survey including a large amount of privacy-sensitive data has to be completed by the household occupants of each house, before the forecasting model can be used. Furthermore, some of this information such as building information is fixed or static information. Bakker et al. [40], [41] argue against using fixed or static inputs to their forecasting model (ANN), because they expect that their forecasting model will learn the characteristics of the home.

2.2.4 Forecasting Horizon

The forecasting horizon for energy consumption forecasting is traditionally classified into Long-term (LTLF), Medium-term (MTLF), and Short-term load forecasting (STLF). Sometimes, a subcategory of STLF is considered, called Very short-term load forecasting (VSTLF). The categories are summarized in Table 2.1 [12]–[15], [18], [42].

Name	Time Horizon	Applications
LTLF	A year to 10-20 years ahead.	Applied in territorial or national power utilization whereby demand is often influenced by economic factors. Applications include strategic planning decisions involving energy supply or the installation and improvement of storage systems, capacity expansion planning, investment decisions, and purchasing new energy generating infrastructure.
MTLF	Few weeks to few months (or quarterly) ahead.	Applied in system maintenance, security, energy purchasing policies, price settlement, and ensuring balance between supply and demand.
STLF	Sub-hourly, hourly, to one day ahead or a week. The range from seconds to minutes to several hours is often called Very STLF (VSTLF), a subcategory of STLF.	Weather variables have a significant influence on STLF. STLF aims at improving load dispatching, DSM, and (operational) control strategies, such as scheduling and allocation.

Table 2.1: Forecasting Horizons and their applications

This thesis focuses on forecasting models with a short-term horizon as these are also most often used for DSM techniques [42] (see also Chapter 3).

2.3 Time Series Forecasting Models

This section provides a review of different (V)STLF techniques that are used in literature. In particular, a comparison between several regression techniques and artificial neural networks (ANNs) is discussed as well as their respective advantages and disadvantages. Fumo et al. [43] review several types of regression techniques and indicate that, compared to other methods, linear regression has a reasonable accuracy, needs little computing power, and has a relatively simple implementation. They also advocate to apply linear regression to forecast the load of individual households, as an increasing number of people owns a smart meter. Some authors consider regression techniques useful to explain the relationship between input- and

output variables [12], [42], [44], but not useful enough to apply in a forecasting model in practice [42]. There exist many variants of regression techniques. An example is the autoregressive integrated moving average (ARIMA) model. Barua et al. [42] argue that an ARIMA model makes reliable predictions, but is not good at LTLF. In contrast, Wei et al. found that regression techniques are most often applied on a long-term horizon and on a relatively large scale. Gajowniczek et al. [37] agree with this and argue that regression techniques such as ARIMA are not suitable for highly volatile data and are therefore not useful for STLF of an individual household load. For example, Kapoor et al. [45] found that ARIMA is least accurate when comparing ARIMA, Multiple Linear Regression (MLR), Recursive Partitioning and Regression Trees (RPART), Conditional Inference Trees (CTREE) with Bootstrap Aggregating (BAGGING), and Random Forest (RF) models. Because regression techniques like ARIMA are not suitable for very volatile data, Gajowniczek et al. [37] prefer machine learning techniques such as support vector machines (SVM) or ANNs for this type of application, because they can make better predictions with noisy data. Note however that if there exist only linear relationships within the data, an ANN may not be better than a linear method [20].

Hippert et al. [19] have done an extensive review of ANNs used in STLF applications. According to them, a large number of forecasting models that make use of ANNs are successful, but there still exists skepticism because there is not much hard evidence that ANNs are better than other forecasting methods in general. For example, Marinescu et al. [39] did not find substantial differences in performance when comparing autoregressive (AR), ARMA, and ARIMA models with fuzzy logic (FL), ANNs, and wavelet neural networks (WNN). WNNs combine wavelet functions with ANNs such that frequency, phase, space and time are also taken into account [20]. In their comparison, the regression approaches (AR, ARMA, and ARIMA) even had the overall best performance on average. On the other hand, ANNs, WNNs, and FL were better at predicting peaks during the morning and evening. Marinescu et al. [39] suggest to use a combination that uses the advantages of each method they reviewed.

The downsides of ANNs are that they require a lot of historical data [44], that they are hard to interpret (hard to explain relationship between input- and output variables) [20], and that they suffer from overfitting because of the large number of parameters (complexity) and are thus hard to generalise [20], [42]. Deeper neural networks such as convolutional neural networks (CNN) and long short-term memory (LSTM) networks are even more complex. Yan et al. [46] proposed to combine an ensemble of LSTMs with the Stationary Wavelet Transform (SWT) technique. Some authors propose to combine CNNs and LSTMs [47], [48]. But others argue that CNNs are more suitable for images or that LSTMs are unsuitable for short-term predictions [42]. Because CNNs and LSTMs are more complex than standard

ANNs, they are computationally slower. However, a forecasting model should not be computationally heavy or slow, especially when applied on a simple local house controller in an online setting.

ANNs have many parameters and therefore may be computationally slower than simple regression techniques, but they are often computationally faster than deeper networks such as CNN and LSTMs [42]. Furthermore, their prediction accuracy is often better than regression techniques such as ARIMA [13] due to their ability to deal with nonlinear complex problems [12], [38], [47] and to approximate any (nonlinear) continuous function numerically [19], [20]. Therefore, they are often applied on a short-term horizon and on a relatively small scale [12]. Alfares et al. [49] compared nine different forecasting techniques and found that the most active research area, at the time of writing, was forecasting based on ANNs. Liu et al. [50] made a comparison of three VSTLF techniques: FL, ANN, and AR models. Before evaluating their performances, they used a low-pass filter to remove high-frequency noise components from the data. Their study shows that it is even feasible to develop a simple ANN or FL model to predict very short-term load in an online approach. Especially when applying STLF at a local controller on a small scale such as the electricity load of an individual household, the benefits of a standard ANN outweigh the benefits of deeper networks (e.g. CNNs or LSTMs) or other regression techniques (e.g. ARIMA). Therefore, in this research we focus on using ANNs when forecasting the load of households.

2.4 Artificial Neural Networks (ANN)

2.4.1 Introduction

Because ANNs are used in the forecasting models proposed in this research, a small background on ANNs is provided in Section 2.4.2, including a definition of an ANN (Section 2.4.2.1) and a brief discussion on hyperparameters (Section 2.4.2.2). ANNs can be used for many applications, but in this research they are used for household energy consumption prediction. Therefore, different ANN architectures used for energy consumption are considered in Section 2.4.3, especially their input configuration (Section 2.4.3.1) and output configuration (Section 2.4.3.2).

2.4.2 Background

An Artificial Neural Network (ANN) is a computational model, originally inspired by a biological neural network of the brain, that is typically used for (supervised) machine learning applications, such as classification and regression. An ANN can be described as a structured collection of nodes connected by edges. Each node represents a neuron and each edge represents the connection between two neurons. The neurons are ordered in layers. The input neurons are located in the first (input-) layer, the output neurons in the last (output-) layer, and in between these layers exist the hidden layer(s). The input to the ANN is a vector of numerical values, of which each value corresponds to an input neuron. The output of the network is also a vector of numerical values. An example of an ANNs is shown in Figure 2.3. The

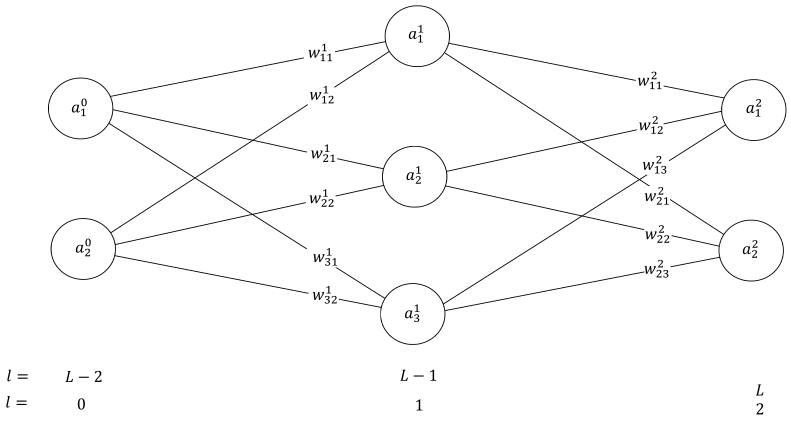


Figure 2.3: Example feed-forward ANN with two neurons in the input layer, three neurons in the hidden layer, and two neurons in the output layer, from [51]

output of a single neuron, called the *activation* of that neuron, is (except for the neurons in the input layer) a function of the sum of all the inputs (output activations of neurons in a previous layer), weighted by the *weights* of the connections to these inputs. Typically, a constant term, a bias, is added to this sum. This complete *weighted sum* is passed to an *activation function* and forms the output activation of the neuron. This output can then, in turn, be used as input to a neuron in the next layer. The neuron computes its output activation as follows (from [51])

$$a_j^l = \sigma\left[\sum_i (a_i^{l-1} w_{ji}^l) + b_j^l\right]. \tag{2.1}$$

Here, a_j^l and b_j^l denote the activation and bias, resp., of neuron j in layer l. w_{ji}^l denotes the weight corresponding to the connection between neuron j in layer l and neuron i in layer $l-1$. σ denotes the activation function, which is typically a bounded, monotonically increasing, nonlinear, continuous, and differentiable function. Examples of activation functions are the sigmoid (logistic) function, the

hyperbolic tangent (tanh) function, the sine or cosine function. As example, the sigmoid activation function is given by

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.2}$$

When the input layer of the the network is provided with an input vector, all neurons in subsequent layers can compute an output in a layer by layer fashion, until the last layer has computed a final output. This is called a feed-forward pass or prediction of the network. When the network has made a prediction, its output vector can be compared to a target output. Typically, this is done by using a cost function that computes an error metric indicating how close the output of the network was to the target output. An example of a cost function (the sum of squares) is

$$C(\mathbf{w}, \mathbf{b}) = \sum_j (a_j^L - y_j)^2. \tag{2.3}$$

Here, a_j^L is the activation of neuron j in the output layer L and y_j is the target value for neuron j. The vector $\mathbf{y}$ is also called the target or label. Note that the cost function $C(\mathbf{w}, \mathbf{b})$ is parameterized by all weights $\mathbf{w}$ and biases $\mathbf{b}$ in the network, since the activation of the output layer $\mathbf{a^L}$ is a composite function of these parameters. A typical approach to train the network consists of updating the values of the weights and biases using gradient descent, to find a minimum of the cost function. The amount that each value is changed, depends on the step of the update. The update step of a weight is the partial derivative of that weight with respect to the cost function, multiplied by a learning rate. The learning rate is a hyperparameter of the ANN that can be chosen to change the size of the update step. The process of updating all weights and biases in the network (estimating the ANNs parameters), is called training the ANN [51]–[53].

2.4.2.2 Hyperparameters

Apart from the number of neurons in the input and output layers of an ANN, there are hyperparameters that can be modified that have an effect on the training process of the ANN. Different from weights and biases, hyperparameters are parameters that cannot be learned from training an ANN. They are used to control the training process. Examples of hyperparameters are the learning rate, the number of hidden layers, and the number of neurons in the hidden layer(s). There are also choices that can be made regarding the type of activation function, stopping criteria, the type of optimization, and regularization. Regularization techniques are techniques that help prevent overfitting.

Determining the number of hidden layers and neurons in each hidden layer is less straightforward than for the input layer and output layer of the ANN. One hidden

layer is sufficient to approximate any (continuous) function, but most studies use either one or two hidden layers [19], [28]. There is no generally accepted method that can be applied to determine the correct number of neurons in a hidden layer, but the effects of small variations are often not significant. However, if the number of hidden neurons is insufficient, the ANN is not flexible enough and will not be able to make good predictions. Too many hidden neurons makes the model very complex, computationally slower, and might cause overfitting. Most studies choose the number of hidden neurons by trial and error [19].

Another choice that affects the behaviour of the ANN is the activation function. Most researchers use the sigmoid or the hyperbolic tangent function, but there is no general agreement on the "best" activation function [19], [20]. Also the type of optimization has an effect on the ANN. Typically, an ANN is trained using gradient descent with back-propagation, but there are several extensions to this method. For example, gradient descent with momentum can reduce oscillation effects in the cost space during training [20], [52].

Another way to deal with this is dropout [52]. Dropout is a regularization technique where some hidden neurons are randomly removed during the the training process. By ignoring different sets of hidden neurons during training, overfitting can be reduced. The effect of dropout is similar to that of ensemble learning. A related concept is sparsity, but most ANNs are fully connected when used for prediction [20]. Zhang et al. [20] even argue that adding extra direct connections between input and output neurons help model the linear relationships and therefore improve the accuracy of the ANN.

2.4.3 ANN Architectures In Energy Context

When using an ANN to predict the electricity load of a household, various ANN architectures may be used. Especially the input and the output of the network is important for the application. There are many ways to apply neural networks for short-term load forecasting. Especially, there are different ways to select the number of neurons in the input and output layer of an ANN when predicting electricity load.

2.4.3.1 Input Configuration

Zhang et al. [20] found that many studies create a new input neuron for each lag (see Section 2.2.1). In other words, the number of lags corresponds to the number of neurons in the input layer of an ANN. However, this would result in a very large input layer of the ANN. Only creating neurons for lags with a strong auto-correlation

(see Section 2.2.1) would alleviate this problem. Zhang et al. [20] argue that a small number of essential input neurons is required, enough to discover the underlying pattern. Either too many or too few input neurons could negatively impact the prediction or training effectiveness of the ANN. However in their review, they found that most studies use arbitrary experiments or intuitive ideas to select the number of input neurons.

Another approach is to classify input (load and weather) data, before load forecasting [19]. For example, it is often argued that week day profiles are not similar to weekend day profiles. Hence, the input data would need to be classified into two classes: week days and weekend days. Another classification is one class for each day of the week, resulting in seven classes (this is done in [40], [41]). This way, one can come up with several classifications. Combining multiple classes might be difficult. For example, some researchers classify holidays as weekend days [19]. There are three different ways that such classifications can be handled when designing the architecture of an ANN. The first way is to create a binary variable for the class (holiday or no holiday), and create a new input neuron for each class. This means that as many input neurons should be added to the input layer of the ANN as there are classes. Alternatively, a classification can be numbered (e.g. 0-6 for Monday-Sunday) and creating a new input neuron for each classification. The difference is that now only one input neuron is needed to model all classes (0-6) of the classification. The last way is to build a separate ANN for each class (e.g. a separate ANN for each day of the week). The downside of this approach is that there may not be enough data for each separate ANN to be trained sufficiently well.

2.4.3.2 Output Configuration

Most studies do not predict a load value at a single value (snapshot) in the future, but rather forecast a load profile: a window that includes e.g. the load of all values of the next day. If the data has an hourly granularity, the load profile of the next day consists of 24 values. For this case, Hippert et al. [19] classified three types of ANN architectures:

1. **Iterative forecasting (one-step)**
   This ANN architecture has a single neuron in its output layer. The ANN makes a prediction of the load of one hour in the future, and then uses this prediction as input for the forecast of the load of the hour after that. This process is repeated iteratively until 24 forecasts (the complete load profile of the next day) is predicted. This can be seen as a recursive process, as the output of the network is used as input for the next iteration. Each prediction of the ANN represents a prediction of a single time step (hour) in the future (one-step), which is repeated 24 times.

2. **Multivariate forecasting (multi-step)**
   This ANN architecture has 24 neurons in its output layer, making it a bit larger and more complex. The ANN makes one prediction of the load of all the 24 hours in the future (the complete load profile of the next day). Thus, each prediction of the ANN represents a prediction of all 24 time steps (hours) in the future at once (multi-step). This architecture was used by most of the researchers [19].
3. **Multi-model forecasting**
   In this architecture, 24 ANNs are used in parallel. Each single ANN has only one neuron in its output layer. To predict the complete load profile of the next day (24 values), each single ANN predicts the load of a particular hour of the day. The advantage of this architecture is that the individual ANNs are relatively small and only receive a part of the data, so they are less likely to be overfitted. The disadvantage is that each individual ANN may not be trained sufficiently, as each ANN receives only part of the data

Zhang et al. [20] report that there is no consensus in literature about which method is preferred, but their opinion is that the multivariate forecasting architecture may be better.

2.5 Evaluation Metrics

Error metrics measure the difference between the predicted value (model output) and the actual value (real observation), and are used to measure the performance of forecasting models [13], [38]. Javed et al. [28] evaluate the performance of their forecasting model according to three measures: *precision*, *accuracy*, and *stability*. Here, precision is represented by such an error metric indicating how close the forecast is to the actual load. Accuracy represents the number of "correct" forecasts that the forecasting model makes. Lastly, stability means how consistent the size of the errors of the forecasting model are, which is represented by the variance in the error.

To measure precision, there are several widely used performance metrics for time series forecasting [12], [13], [38], [42]. Most error metrics can be categorized as either scale-dependent errors, percentage based errors, or relative errors [54]. An often used (percentage-based) error in time series forecasting is the the Mean Absolute Percentage Error (MAPE):

$$\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right|\times 100\%. \tag{2.4}$$

Here, n is the number of datapoints, y_i is the real observation at point i, $\hat{y}_i$ is the predicted/forecasted model output at point i. Although MAPE is very often used in time series forecasting, a big problem with MAPE is that it can heavily penalize relatively small errors when the actual load y_i is very small, and even become infinite when $y_i = 0$. Additionally, MAPE suffers from symmetry problems: the MAPE value is larger for positive errors (over-forecasts) compared to negative errors (under-forecasts) [54], [55]. This is why many authors point out its inconvenience [24], [38], [54], [55], and some argue that squared errors are preferred because they do not suffer from these issues and penalize large errors more [19], which is useful for training an ANN. An example is the Root Mean Squared Error (RMSE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{2.5}$$

Hyndman et al. [54] recommend to use (scale-based) errors, such as Equation (2.5), when comparing different methods on a single data set.

Unfortunately, many error metrics can attain contradictory values, especially for volatile and irregular data such as household loads [27]. For example, Haben et al. [56] explain the "double penalty effect" of point-wise metrics, such as MAPE and RMSE, when a peakload is predicted slightly before or after an actual peak occurs. When exact timing is less important, their proposed adjusted error, may be used. Furthermore, tailored error metrics are devised that try to improve overall well-performing metrics that fail on load spikes [26], [28] and try to treat forecasting and control as a combined problem [55].

Another issue with error metrics is their inconsistency among different studies. Hyndman et al. [54] argue that the Mean Absolute Error (MAE)

$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2.6}$$

is preferred when all data is on the same scale, because the MAE is simple to explain. However, the performance of a forecasting model using an error metric such as the MAE, may be completely different for two sets of households with a different load profile variability. For example, an error metric often scores worse for a set of load profiles with a higher load variance than for a set of load profiles with a small load variance. In order to make different studies more comparable, their error metrics should first be normalized according to the volatility (e.g. dividing by the standard deviation) of the data [24], [54]. As the load changes over time, the distribution of errors over time could also provide interesting insights [19]. Finally, when evaluating a model, some researchers argue that its performance should be

compared to that of a naïve or well-accepted method, regardless of the error metric that was used.

Demand Side Management

3

Chapter Objective: This chapter introduces demand side management (DSM) and reviews several DSM techniques.

Chapter Contents

3.1 Introduction

As stated in Section 1.2, some DSM techniques rely heavily on predictions and finding out how predictions influence the performance of these DSM techniques and how different DSM techniques currently make use of available predictions, can potentially provide insights for making predictions that are more meaningful and useful in this context. Section 3.2 therefore introduces the concept of Demand Side Management (DSM) and explains how DSM leverages flexible devices. To see how DSM is implemented in practice on a neighbourhood level, Section 3.3 looks into smart grids. A smart grid makes use of one or more energy management systems, which are explained in Section 3.4. Before, looking into how predictions are used by DSM techniques, it is important to have a good overview of different types of DSM technologies. Therefore, DSM techniques are categorized by the optimization strategy that is used (Section 3.5) and a distinction between control-based and planning-based DSM is discussed (Section 3.6). Next, two specific DSM methods are discussed in more detail: the EV charging problem is presented in Section 3.7 and the profile steering algorithm is explained in Section 3.8. Finally, it is explained how

the predictions of the forecasting model proposed in this document could be used as input for these DSM techniques, such that the schedule of appliances and their energy consumption can be optimized in order to reduce peaks on the electricity grid.

3.2 Introduction To DSM

The goal of DSM is to incentivise consumers (household residents) to decrease the electricity consumption during peak hours or to shift the electricity consumption to a time where there is less overall energy demand. DSM is typically applied in a residential area, such as in a house or a neighbourhood. DSM techniques may provide very promising solutions to flatten load peaks. DSM techniques leverage flexible devices. Flexible devices are devices that have a load that can be easily shifted in time, without affecting user comfort too much. Typically, these devices have a relatively large load, so that scheduling or controlling them has an actual impact on the grid. Examples are an electric heat pump (with heat storage) and an EV. Devices with a relatively smaller load can also provide flexibility, such as a washing machine. Many DSM techniques schedule or control such flexible load.

3.2.1 EV Example

Instead of charging a car (with a fuel engine) at a gas station, residents with an EV can often charge at home, using a home charging point. An EV charged at home consumes roughly 7.2 kW [57], but this can vary depending on the type of EV, the charge rate and type of charger. This is a large amount of power compared to the average household consumption. Note that the average household consumption varies depending on the type of house and the number of household occupants. Table 3.1 shows the average electricity consumption in kWh for a house in the Netherlands [58]. Although houses get more appliances, note that the average power consumption in the Netherlands is decreasing. This may suggest appliances are becoming more efficient, but the main reason for the decrease in recent years is the result of rooftop solar panels [59]. A prognosis of the overall average electricity consumption of a Dutch household is 2479 kWh [59], which is roughly 0.3 kW on average at all times (much less than 7.2kW). Of course this number varies very frequently over time in practice, between approximately 17.25kW and -17.25kW for newer households with a $3 \times 25A$ connection (negative number indicates that the house can supply electricity to the grid, e.g. when it has solar panels) [60]–[62]. Since EV users typically arrive at home in the evening, EVs in a neighbourhood are connected to their home charging points at roughly the same time. In a neighbourhood with many EV users, this may cause a huge peak of consumption in the evening when

Type of house	Built in (year)	Size (m^2)	Number of residents	2018 (kWh)	2019 (kWh)	2020 (kWh)
Apartment	≤ 1992	< 100	1	1580	1550	1550
Apartment	> 1992	< 100	1	1640	1610	1580
Terraced*	≤ 1992	< 100	1	1730	1690	1680
Terraced*	≤ 1992	$100-150$	1	2050	2000	1990
Apartment	≤ 1992	< 100	>1	2280	2220	2270
Terraced*	≤ 1992	< 100	>1	2860	2780	2790
Terraced*	≤ 1992	$100-150$	>1	3290	3190	3200
Terraced*	> 1992	$100-150$	>1	3340	3260	3260
Terraced*	≤ 1992	≥ 150	>1	3950	3860	3900
Detached	≤ 1992	≥ 150	>1	4490	4450	4530

Table 3.1: Data from CBS [58]. Average power consumption per year (kWh).
*A terraced house also includes corner houses or semi-detached houses. Also, only houses with a gas connection are considered

electricity demand is already high [32]. However, EVs are commonly connected to the home charging point for a longer period of time than is needed to fully charge the EV. For example, an EV user may arrive at home in the evening, connect the EV to the charging point, and leave the EV to charge over night. In this case, the EV may be fully charged much sooner than the end of the night. Therefore, a smarter approach would be to schedule the EVs in the neighbourhood to charge at different times, instead of charging them simultaneously in the evening. For example, when the charging windows of the EVs are distributed over the night, a peak on the grid in the evening may be prevented. From the point of view of an EV user, it does not matter when the EV is charged, as long as it is charged enough at the time that the user wants to leave. By taking into account the arrival and end time between which each EV needs to be charged and how much each EV still needs to be charged, the charging windows and rate of charge of these EVs can be adjusted without affecting the user too much.

3.2.2 Vehicle To Grid (V2G)

When a battery of an EV is used to supply electricity back to the grid, this is called vehicle-to-grid (V2G). V2G can provide even more flexibility as the battery of the EV can also discharge energy when there is a lot of demand. However, there currently only exist pilots using DC V2G with cars such as the Nissan Leaf, but most cars do not yet support this technology [63]. In practice, V2G is not applied at home charging points yet, because most houses have AC home charging points [63]. In contrast, home electric batteries are already used in practice. A home electric battery can be scheduled to be charged at times when there is most electricity production (to flatten production peaks) and discharge when there is a lot of demand. For example, when charging the battery at a time when there is a lot of solar supply in the afternoon, and discharging the battery when there is a lot of demand in the evening, the size of peaks (in supply and demand) could ultimately be reduced. The advantage of a home battery over V2G, is that can be used at all times whereas the battery an EV is unavailable when the EV is used.

3.2.3 Different Flexible Devices

Next to EVs or batteries, there are other appliances that can provide flexibility. Molderink et al. [64] categorize different types of appliances such as a consuming, producing, or buffering devices. For example, an electric heat pump with heat storage may be scheduled to consume energy when overall demand on the grid is low, similar to an EV. The heat that is produced by the electricity consumption can be stored and used later to heat the house at a time that the heat is actually needed. A washing machine or tumble dryer can also be shifted in time. However different from an EV, these devices prove to be less flexible, because once started they should not be interrupted (e.g. a user wants their washing program to run completely).

3.2.4 Forecasting

By adjusting the loads of several flexible devices to each other, the peaks of a total household power profile can be reduced. Applying a DSM technique to multiple households could significantly reduce the peaks on the LV grid. However, some DSM techniques rely heavily on forecasts, especially of loads that cannot be controlled. In order to create a good schedule for flexible devices, inflexible loads (loads that cannot be scheduled in time) are often predicted beforehand. Examples of such inflexible loads are the power used by the lights in a house, a television, or a fridge. The user either wants to use these appliances immediately or they should be powered constantly, and therefore they cannot be scheduled or adjusted in a flexible way. At transformer level, the total household electricity consumption is often called a base load and consists of the aggregation of the inflexible appliance loads within the house. These household base loads cannot be adjusted flexibly, but should be considered for scheduling or control in a DSM technique, as they ultimately do have an influence on the peaks on the grid. Having accurate predictions of household base loads is therefore really important for a DSM technique to work effectively.

3.3 Smart Grids

As the number of flexible devices (such as EVs) is expected to grow, it is important that DSM techniques are designed to be scalable. To achieve scalability, a smart grid is required. A smart grid is an ICT system that monitors electricity transmission, consumption and/or production of connections to a part of the physical electricity grid (typically a neighbourhood), and can make use of information and coordinate electricity demand by using communication between one or more controllers or flexible devices [64], [65]. In a smart grid, there is often a global central controller,

also called an energy management system (EMS), that optimizes an objective and sends resulting steering signals to the houses connected to the smart grid. The house residents can act upon these steering signals by adapting their electricity consumption manually, or install a local (house) controller that does this for them. Such a local controller is also called a home energy management system (HEMS) [32], [66].

3.4 Energy Management Systems (EMS)

A (H)EMS is a system that can be installed in a smart grid, building or household and that helps to manage its electricity demand. The goal of the EMS is typically to optimize (maximize or minimize) an objective, subject to a set of constraints. A HEMS commonly considers a few objectives (e.g. maximize user comfort, minimize peak demand) simultaneously, introducing trade-offs between objectives [38]. This is called "multi-objective optimization" [67] and is most commonly implemented by a weighted sum of the objectives. In order to do that, a HEMS may use household load data, appliance data, forecasts, occupancy preferences, and/or electricity price/tariff information [38]. The output of a HEMS could either be an optimal overall load profile, or an optimal appliance schedule, including heating, ventilation, and air-conditioning (HVAC), electric vehicles (EV), and batteries [67]. However, note that most devices cannot be controlled without limits. User preferences and device constraints form restrictions on the flexibility of the device. These restrictions are called *flexibility constraints* [33], and should be taken into account in the HEMS. In order to shave peaks on the complete house load (the the aggregation of all appliances), a HEMS may regard some non-flexible appliances (or base loads) as fixed input when scheduling flexible appliance loads [68].

3.5 Optimization Strategies

Most DSM methodologies use a form of optimization with a function that takes into account the flexibility constraints. Such a function is often called a utility function, bidding function, or cost function. In practice, an energy price can be determined by solving a cost optimization problem that is subject to some constraints. The cost function together with the energy price results in a dispatch for each device. Molderink et al. [64] split the optimization strategies into three groups:

- *Auction based optimization*
  When an auction based optimization strategy is used, both electricity consumers and producers can place bids for the energy price. Together, they form

cost functions that help to determine the market clearing price, see Figure 3.1 [64].

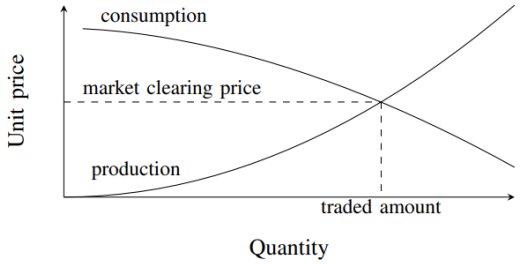


Figure 3.1: Determining the market clearing price, from [64]

- *Game Theory based optimization*
  Mohsenian-Rad et al. [69] formulate an energy consumption scheduling game. The users represent the players of the game and the daily load schedules of the devices in their houses represent the players' strategies. The assumption is that by changing the electricity price, user energy consumption can be changed. They found that the total amount of electricity costs is minimized at the Nash equilibrium of the formulated scheduling game. This means that each player simply applies it best response strategy to the current electricity price. Their results show that load peaks are reduced, as well as individual and total electricity costs of the players. Tang et al. [70] optimized dynamic pricing and daily building electricity demand using the Stackelberg game.
- *Other mathematical optimization strategies*
  Van der Klauw et al. [71] state that the problem of finding optimal load schedules based on steering signals received from centralized controllers, can be categorized as a resource allocation problem (RAP). There exist many efficient methods that can solve this problem (see [64] for an overview), but most of them use a form of mathematical programming.

Comparing different DSM approaches is difficult, because differences in results depend partly on the definition of the cost or objective functions. For a good comparison between two methodologies, identical cost or objective functions should be used [64].

3.6 Control Versus Planning

Some of the DSM techniques use operational control, focusing on the optimization at the current moment. Other DSM techniques optimize for a longer period, based on a planning. This sections discusses a comparison between control-based and planning-based techniques. *Control-based DSM* looks ahead one interval and risks

using available flexibility too soon while a large problem may occur later. On the other hand, *planning-based DSM* looks further ahead to make trade-offs, most commonly to one or two days (24h-48h) using a 15 minute data resolution. However, the accuracy of forecasts may significantly affect the quality of the planning-based optimization techniques. For example, when the predictions are poor, planning-based DSM approaches might establish a low quality (or invalid) planning [32].

In practice, it appears to be more effective to use planning-based DSM, as it shown by Molderink et al. [64]. They compared two operational control methodologies: one based on auction and one based on Integer Linear Programming (ILP). Their simulation shows that making decisions based on purely historical data leads to worse results, but combining these methodologies (auction and ILP) with a planning leads to better results. Here, predictions are made on device level and the overall planning is made on global level. They found that the auction method is better at dealing with prediction errors. Therefore, they argue that the best solution is a good planning algorithm combined with auction as operational control, in order to reach a desired consumption profile for a large group of houses.

3.7 The EV Charging Problem

Since EVs offer great flexibility potential for DSM [66], this section describes the EV charging problem using the notation and ideas of Schoot Uiterkamp et al. and Gerards et al. [32], [72]. When an EV charges, it has a charging window $t \in \{1, ..., T\}$ of T time intervals. Let x_t, p_t, and d_t denote the electricity charged by the EV, the uncontrollable house power consumption (base load), and the desired total (house load and EV load together) load, respectively, during time interval t. Then, the *charging profile*, the *base load profile*, and the *desired profile* are denoted by $\vec{x} = [x_1, ..., x_T]$, $\vec{p} = [p_1, ..., p_T]$, $\vec{d} = [d_1, ..., d_T]$. Furthermore, the total amount of energy that needs to be charged in the charging window is denoted by C and the maximum charging power for each time interval is denoted by $\bar{x}$. The goal is to find a charging profile $\vec{x}$ such that the sum of the base load profile and charging profile $\vec{p} + \vec{x}$ is closest to the desired profile $\vec{d}$. This results in optimization problem (3.1):

$$\begin{aligned} \min_{\vec{x}} \quad & \sqrt{\sum_{t=1}^{T} ((p_t + x_t) - d_t)^2} \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t = C, \\ & 0 \leq x_t \leq \bar{x}, \quad t = 1, ..., T \end{aligned} \tag{3.1}$$

The objective is to obtain a profile which is as flat as possible. This can be done by choosing $\vec{d}$ to be a constant vector (e.g. the zero vector $\vec{d} = \vec{0}$). Van der Klauw et al. [66] extend Problem (3.1) and show that they can calculate an optimal schedule using an efficient algorithm.

3.7.1 Valley-Filling Using A Fill-Level

An often used DSM technique to solve the EV charging problem, is called *valley-filling*. Valley-filling is a technique that uses EV charging (positive load peaks) to "fill" the valleys (e.g. negative peaks created by PV electricity production) of a load profile, in order to realize a flat load profile. A parameter of this method is the *fill-level* Z, which is the level up to which the valley of the load profile must be filled. In a perfect situation where all information is known beforehand, an optimal value of the fill-level can be chosen (depending on $\vec{p}$ and C [32]). Schoot Uiterkamp et al. [72] present a method to predict this optimal value of Z. Using an empirically created distribution of the optimal fill-level Z, they can sample an estimate $\hat{Z}$. For this, they use a risk parameter α that represents a preference for either an under- or over-prediction of this optimal value Z. They found that over-predicting is generally preferred, and show that the larger the amount that needs to be charged within the charging window C, the less significant is the base load profile $\vec{p}$. Additionally, given this prediction of the optimal fill-level Z, they can determine the required (optimal) charge of the EV in each time interval x_t using a decision rule $x_t = \max(0, \min(Z - p_t, \bar{x}))$. Provided with an estimate of the optimal fill-level $\hat{Z}$, they do not need a prediction of the base load profile $\vec{p}$ at multiple intervals in the future, but only of the base load of the upcoming time interval p_t and the sum of complete total energy consumption (over all later intervals) [34], to make a schedule for an EV that is close to optimal. For instance, when using a battery, it is more important to know how much of the battery capacity needs to be reserved to absorb peaks that might occur later, than to know when exactly these peaks will occur [34]. To make up for earlier made errors and since the prediction of the upcoming interval will be more accurate than the prediction of later intervals, Gerards et al. [32] chose to make a decision (of the amount of charging to be done) only one interval in advance. In other words, both the size and timing of load peaks do not have a significant effect on the method of Schoot Uiterkamp et al [72].

3.7.2 Fleet Planning

Solving the EV problem for a fleet of EVs is often done with the objective of the grid operator in mind [66]. On a neighbourhood level (as opposed to a single EV), Schoot Uiterkamp et al. [72] argue that under-predicting Z for a few of the EVs can

be beneficial for the aggregated power profile. Gerards et al. [32] explain how the approach above can be applied to multiple EVs (fleet planning). They ask the EVs to decrease or increase their total load for the upcoming time interval when the power of a group of houses is above or below a given threshold [32].

3.8 Profile Steering

Pricing techniques provide a common incentive for residential electricity users to avoid peak electricity consumption. However, many pricing techniques introduce new problems. Uniform dynamic pricing encourages household occupants to collectively move their energy consumption from different time intervals all to the same time interval, effectively shifting the peak (not shaving it). Furthermore, differentiated dynamic pricing may motivate extreme behaviour as individual households get a price incentive to increase or decrease their load at specific time intervals. Although this method properly shaves peaks at transformer level, it introduces local peaks at household level. Nonlinear pricing could be a solution but quickly becomes very complex. Instead of auction or bidding methods based on pricing, Gerards et al. [33] propose to avoid using pricing techniques altogether and introduce a *profile steering* algorithm as an alternative.

3.8.1 The Profile Steering Algorithm

The goal of the profile steering algorithm of Gerards et al. [33] is to minimize the distance between the planning or *obtained profile* $\vec{x} = [x_1, ..., x_N]^T$ (N is the number of intervals, e.g. 96 when using 15 minute data covering a period of a day) and a *desired profile* $\vec{p} = [p_1, ..., p_N]^T$, e.g. a flat or zero $\vec{p} = [0_1, ..., 0_N]^T$ profile. The *obtained profile* $\vec{x}$ that is stored at a global controller is considered to be the aggregated profile of all individual device profiles $\vec{x} = \sum_{m=1}^{M} \vec{x}_m$, where $\vec{x}_m$ is the planning of device $m \in \{1, ..., M\}$. In an iterative manner, the global controller sends out the difference profile $\vec{d} = \vec{x} - \vec{p}$ and each device responds with how much it can reduce the (euclidean) distance $||\vec{d}||_2$ between the *obtained profile* and the *desired profile* by replacing its current planning $\vec{x}_m$ by a new candidate planning $\vec{\hat{x}}_m$. The global controller then selects the device with the largest improvement. Finally, the device m updates its planing locally $\vec{x}_m := \vec{\hat{x}}_m$ and the global controller updates the *obtained profile* $\vec{x}$. This process is repeated as long as there is sufficient progress. The result of the profile steering algorithm is a planning $\vec{x_m}$ for each device [33].

3.8.2 Asynchronous Event-Driven Profile Steering

Hoogsteen et al. [25] built upon the profile steering algorithm of Gerards et al. [33] by considering asynchronous events. They consider a two-phase approach. The first phase constitutes a synchronous planning phase, resulting in a *planned profile*. The second phase uses an event-driven approach that schedules devices asynchronously to realize the *planned profile* that was created in the first phase. After the first phase, the resulting *planned profile* $\vec{q}$ is a schedule stored at the global controller equal to the aggregated profile $\vec{x}$ that resulted from the profile steering algorithm [33], thus $\vec{q} := \vec{x}$. Then all plannings ($\vec{x}_m$ for a subset of $\{1, ..., M\}$) from uncertain devices (e.g. EVs) are discarded. Predictions of uncontrollable loads (e.g. household baseloads) are kept and stored as a *committed profile* $\vec{r}_m := \vec{x}_m$, of which the global controller stores an aggregated *committed profile* $\vec{r} = \sum_{m \in M} \vec{r}_m$. In the second phase, the global controller updates the *committed profile* $\vec{r}$ when an event is triggered. An event is triggered when there is enough confidence that the planning of an uncertain device will not change (significantly). For example, the moment that an EV arrives at a home charging point, the requested charge, start time, and end time are known and unlikely to change. Thus, an event is triggered at this moment. In this event, the profile steering algorithm [33] is executed once. Recall that the objective of the global controller is to realize the *planned profile* $\vec{q}$. Similar to the profile steering algorithm, the global controller sends out the difference profile $\vec{d} = \vec{q} - \vec{r}$ to this device m, called the *desired profile* as it represents the gap between the *planned profile* and *committed profile* that needs to be filled. In case there are no other (remaining parts of) flexible profiles of earlier arrived EVs left to schedule, the global controller immediately adopts the profile $\vec{\hat{r}}_m$ that is reported by the EV (there is no choice between largest improvements as the event describes a single device). Hence, when the global controller obtains a new *committed profile*: $\vec{\hat{r}} = \vec{r} + \vec{\hat{r}}_m$, it can immediately update it $\vec{r} := \vec{\hat{r}}$. In case there are other flexible profiles left to schedule, the profile steering algorithm is runs until an optimal profile is found for all flexible devices [25].

4 Model Input

Chapter Objective: This chapter provides an analysis of the input data for the proposed forecasting model.

Chapter Contents

4.1 Introduction

The objective of this research is to develop an online "rolling horizon" forecasting model that is continuously being trained on the smart meter data of a single household. It can therefore adapt to the load profile behaviour of that household and to structural changes in electricity consumption over time. With this forecasting model, we hope to overcome challenges regarding variation between house load profiles and the volatility within house load profiles (see Section 1.2). The proposed forecasting model requires input data. This chapter presents the data that was used, how this data is cleaned, and how relevant features are selected from this data. Section 4.2 presents the GridFlex data set [10], which contains household electricity consumption data. Section 4.3 explains how this data was cleaned and transformed. Section 4.4 discusses all the features that are used as input to the proposed forecasting model. This includes features that were extracted from the GridFlex data set, but also features extracted from weather data.

4.2 GridFlex Data

The forecasting model uses electricity data as input, which was collected during the GridFlex Heeten project and was made available publicly (although non-

commercially) for research purposes. The data set is in CSV format and contains 61.49 GB of data. The data in this data set is subdivided as a value per house, per appliance-group, per measurement type, and per time interval. The data was collected between August 2018 and August 2020 in 77 households all situated in Heeten (the Netherlands) and consist of electricity consumption data and gas usage per minute, per household. All participating households allowed that their anonymous data could be used in further research. The data of this project was collected in accordance with a privacy-by-design approach [10]. It was collected by installing an energy management system (EMS) in each household that was connected to the P4 port on the smart meter and read out the consumption power (in kW and kWh) once per minute. Furthermore, if a battery was present, the battery management system was separately connected to the EMS. Also, if a PV system was present, a pulse meter was installed and connected to the EMS to separately measure its output. All this data was then transmitted via Wi-Fi to a cloud.

4.3 Data Cleaning

Forecasting models, and specifically neural networks, are sensitive to missing values, NaN values, and outliers [19]. Therefore, some preprocessing is needed to clean the data of the GridFlex [10] project. After that, the data can be used to predict the electricity consumption.

4.3.1 Missing Values

Some data points (minutes) were missing from the total time span of the GridFlex data set [10]. When the GridFlex data set was created, smart meter data was collected every minute and sent to a central server. Either, during this process or when data was retrieved from this central server, transmission errors may have occurred causing missing (minute) values in the timeline [10]. The percentage (%) of missing data points (minutes) is displayed in the bar plot in Figure 4.1, and is quite high for some households.

These data points (minutes) have been added to the data set to get a complete timeline that includes all minutes of the total time period that the GridFlex data set [10] covers. For each added data point, a NaN (Not a Number) value is stored for each of the measured variables as their values are unknown.

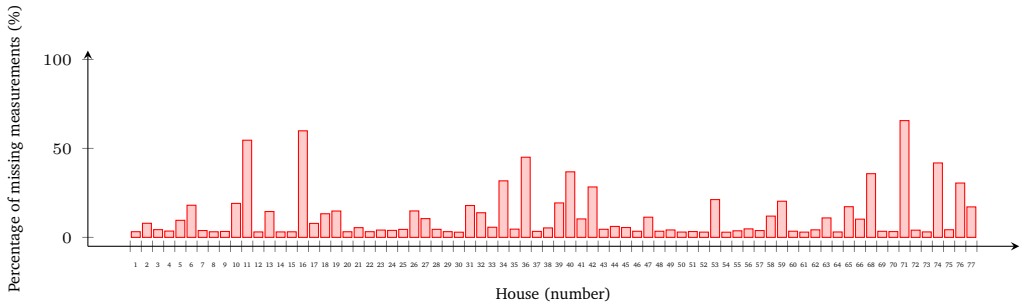


Figure 4.1: The percentage (%) of missing data points (minute-values) for each of the households in the GridFlex data set [10]

4.3.2 NaN Values

Next to missing values, some values in the GridFlex data set [10] are NaN. During some minutes and for some variables, either nothing was measured, data was lost because of transmission errors, or a measurement error occurred causing extreme values. These errors were filtered from the data, leaving some values empty. In these cases, a NaN value has been recorded. The number of NaN values of the data set is relatively large, which may be problematic. The bar plot in Figure 4.2 shows the percentage of NaN values for each variable that was measured, for each household. The variables depicted in the legend of Figure 4.2 are explained in Table 4.1. Note that not all houses have solar panels or a battery.

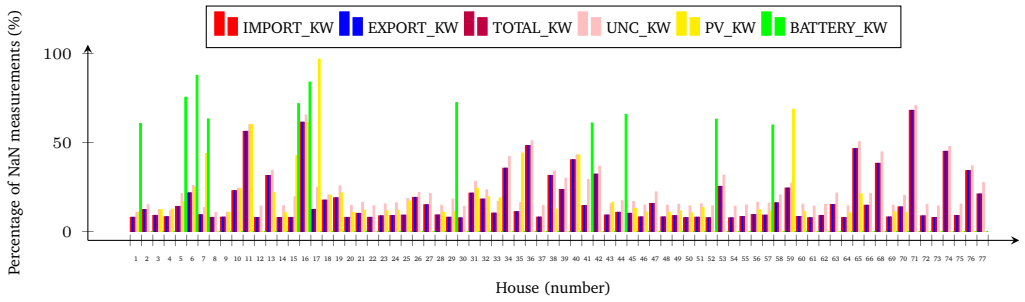


Figure 4.2: The percentage (%) of NaN values for each variable and for each of the households in the GridFlex data set [10]

Unfortunately, a variable where more than 50% of the values is NaN is not uncommon. Figure 4.3 shows how the NaN values are distributed over time, grouped by day. The total number of values that are measured each day is 1440, as there are 1440 minutes in 24 hours.

In this figure, only the NaN values that overlap for every household are shown. This shows that from approximately June 2019 significantly less NaN values are stored. There are also some days where all values of the day (1440 minutes) are NaN,

Variable	Description
IMPORT_KW	Average power input of the household (in kW) in the last minute. It is the difference of consecutive measurements of the total energy the household has imported (in kWh) since the household was connected to the smart meter.
EXPORT_KW	Average power output of the household (in kW) in the last minute. It is the difference of consecutive measurements of the total energy the household has exported (in kWh) since the household was connected to the smart meter.
TOTAL_KW	Average power usage of the household (in kW) in the last minute, where a negative value means exporting power. It is the difference of consecutive measurements of the total energy the household has used (in kWh) since the household was connected to the smart meter (a negative value means exported energy). In other words, it is the difference of export and import.
UNC_KW	Average power usage of the household excluding PV and battery (in kW) in the last minute.
PV_KW	Average power production of the PV system (in kW) in the last minute. It is the difference of consecutive measurements of the total energy produced by the PV system (in kWh) since the PV system was connected to the pulse meter.
BATTERY_KW	Average power output of the battery (in kW) in the last minute (negative means charging).

Table 4.1: Variable descriptions of a selection of variables from the GridFlex data set [10]

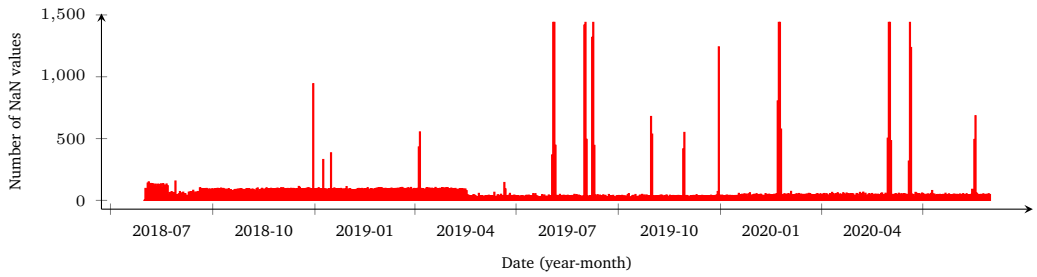


Figure 4.3: The number of NaN values distributed over time and grouped by day, that occur in all the households (intersection) in the GridFlex data set [10]

possibly due to transmission errors that occurred the whole day. A possible solution to this method is to fill small gaps of NaN values with imputed values using a simple imputation strategy. Note, however, that some households have significantly more NaN values than displayed in Figure 4.3. To illustrate this, the distribution of the percentage of NaN values over time, grouped by month, is displayed in Figure 4.4 for some extreme cases. These houses show gaps of NaN values of approximately 6 to 12 months. Such enormously large gaps should not simply be filled with imputed values, because this would result in too much fake data that is not representative of

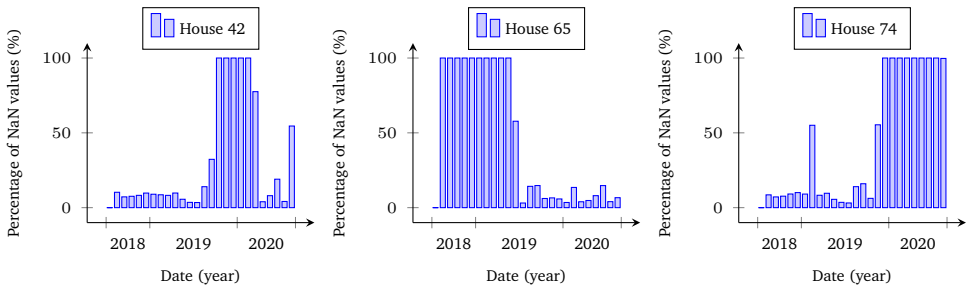


Figure 4.4: The percentage of NaN values (%) distributed over time and grouped by month, that occur in some households in the GridFlex data set [10]

reality. Hence, the proposed forecasting model that is explained in Chapter 5 must somehow deal with these gaps.

4.3.3 Imputation Strategy

To fill gaps of NaN data, an imputation strategy was used that uses a simple linear interpolation method, because the curve of a load profile is generally not a smooth function and a cubic, spline or polynomial interpolation method would interpolate values in a way that creates a smooth curve.

To make sure the gaps that are imputed by the linear interpolation are not too large, a limit was set to one hour (60 minute values) away from an existing (not-NaN) data point. Note that, NaN values are also extrapolated, meaning that values that do not lie between two existing data points (so values that lie to the left or right of the leftmost or rightmost value respectively) will be imputed too. Since a distance of 60 values away from an existing data point was chosen as a limit, at most two hours (120 data points) is the largest gap that can be filled. If more data points would be filled, the new data set might be unrealistic. Therefore, in those cases NaN values are left as they are for the forecasting model to deal with, which is described in more detail in Chapter 5.

4.3.4 Time Frequency Conversion

The GridFlex data set [10] contains smart meter measurements of every minute for a set of houses. When the actual load of a household is considered to be a continuous signal, the measurements of the GridFlex data set [10] can be considered a discrete representation of this continuous signal, with a sampling rate of one minute. Many DSM algorithms, such as the profile steering algorithm [34], use a coarser granularity than one minute intervals. Therefore, the data of the GridFlex data set [10] was resampled. The data was downsampled from one minute intervals to 15 minute

intervals, because a 15 minute granularity is very common in practice. The value of each new data point is the average over the 15 minutes that was sampled from. In the remainder of this thesis, the use of the word “interval” will generally concern a 15-minute interval unless specified otherwise.

4.3.5 Detecting And Handling Outliers

Outliers in the data result from faulty measurements, which could distort the learning process of the proposed forecasting model that is discussed in Chapter 5. A possible way to handle outliers is to remove them from the data set by replacing them by NaN values. However, this is not desired as creates extra gaps in the data. This might even remove correct data, as some electricity consumption measurements might be approximately correct but exceptionally high. In these cases, these measurements might be wrongfully recognized as an outlier, and as a result precious data is lost. To solve both of these issues, the household electricity consumption data of the GridFlex data set [10] could be winsorized beforehand. Winsorization is a method that tries to limit the effect of outliers by clipping them to a maximum or minimum value. A common strategy is to choose these values based on percentiles of the data. For example, a 90%-winsorization method would set all data below the 5th percentile to the 5th percentile and all data above the 95th percentile to the 95th percentile, essentially clipping the data within a certain range. This makes sure that the effect of outliers on the data is reduced without introducing gaps in the data or completely removing incorrectly classified outliers.

Instead of fixed percentiles (e.g. 5th and 95th) that can be used for all households, we decided to choose the percentiles based on the Inter Quartile Range (IQR) rule. The motivation for this choice is that the number of outliers differs per household, as can be seen in Figure 4.5. For this IQR rule, $IQR = Q3 - Q1$, where $Q1$ and $Q3$ are the 25th and 75th percentile respectively. All values falling outside the range (LB, UB) $= (Q1 - p \times IQR, Q3 + p \times IQR)$ are considered outliers. Here p is typically chosen to be 1.5, but can be changed in order to change the tolerance bounds of the outlier detection.

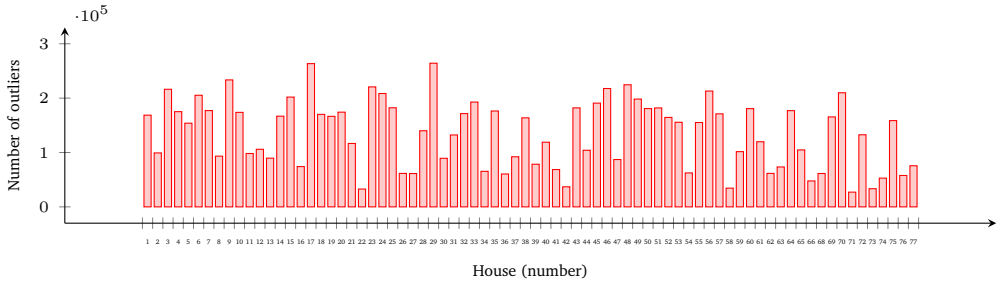


Figure 4.5: Number of outliers per household of the GridFlex data set [10]

The percentiles for the winsorization are then chosen to be LB and UB determined by the $p \times IQR$ rule. In this way, the percentiles are chosen dynamically based on the data of the household. In Figure 4.6, the original data of house 29 is plotted using a red line, and the winsorized data is plotted on top, using a blue line. However, even though outliers are winsorized using $p = 6$ (considered extremely tolerant towards outliers) and all 0's are removed from the data (an electricity consumption of 0 is very common, hence removing 0's would make the winsorization even more tolerant towards outliers), the winsorization bounds seem too strict. This again confirms the huge volatility in electricity consumption within a household. In this case, only

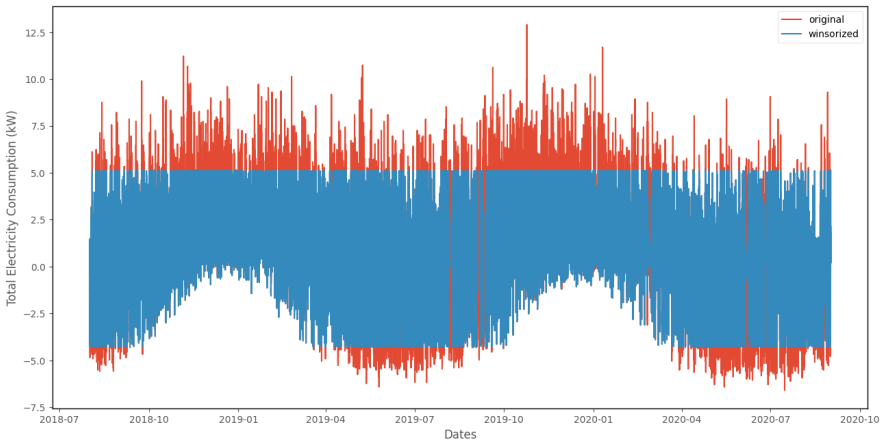


Figure 4.6: Winsorization method according to $6 * IQR$ rule, for household 29.

1.928% of the data was considered an outlier, but from visual observation of the resulting data it can be concluded that the winsorization does not have the desired effect. It seems the winsorization method wrongfully categorized (and clipped) outliers, making the resulting data set unrealistic. Figure 4.6 also triggers the idea that outliers should not be determined on the complete data set but rather on a small time window. For example, the electricity consumption in the summer (more production by solar panels, thus more negative values) is different from that in the winter. This makes an the characterization of an outlier dependent on a time window too. This is also shown when the distribution of outliers is plotted over time, as done for household 25 in Figure 4.7. In general, this seems to be the case for all households in the GridFlex data set [10].

Figure 4.8 shows the results of applying the winsorization method when outliers are determined based on a small time window of a month. Here, a different subset of the data is categorized as outlier because the winsorization method now takes into account the different characteristics of the data for different time periods. However, after visual inspection some data points are still wrongfully categorized as outlier and some outliers are not categorized as such. Therefore, we chose to let the forecasting

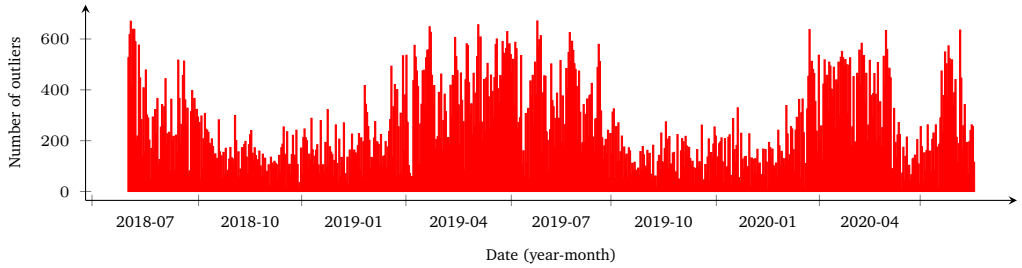


Figure 4.7: Number of outliers over time for household 25, grouped by day.

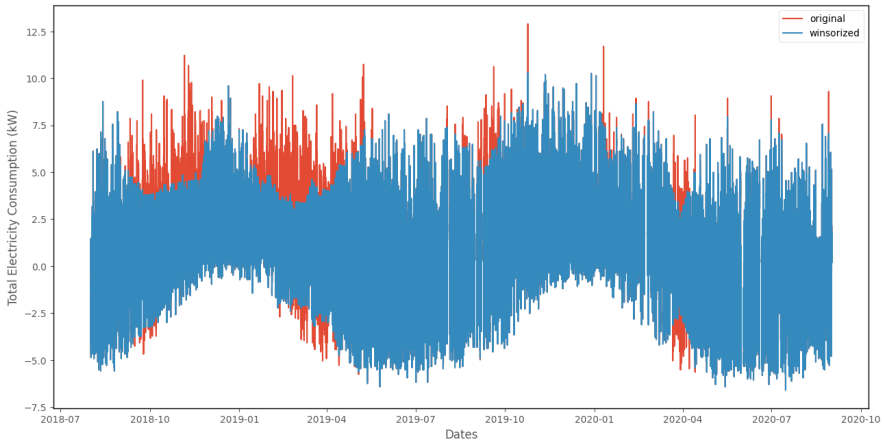


Figure 4.8: Winsorization method according to $6 * IQR$ rule based on the previous month, for household 29.

model handle outliers automatically by adjusting the learning rate, as described in Section 5.4.2.

4.4 Feature Engineering

To increase the performance of the predictions, the proposed forecasting model can use different types of information. Section 2.2.3 provides an overview of different exogenous variables that could be used as input variables. Although human information and building information have a significant influence on the electricity consumption of a household [28], the proposed forecasting model will only use weather data, (historic) electricity data and time data as input. Since the forecasting model should work autonomously (as discussed in Section 1.2), no extra (human or building) information is requested from the resident. Manually collecting data by means of a questionnaire or manually communicating changes about appliances and household occupants for every house, would also be very impractical. Additionally, storing such privacy-sensitive data introduces new risks. Even if this information could be collected automatically and stored locally in a HEMS, most human and

building information is redundant to the proposed forecasting model because it is (relatively) static data and a forecasting model can learn (or adapt to) static information using electricity consumption data [40], [41]. Thus, the proposed forecasting model only uses the (historic) electricity consumption of the household (as that is what the model is also trying to predict) and publicly available information such as weather and time as input.

4.4.1 Weather Data Correlation Analysis

The KNMI ('Koninklijk Nederlands Meteorologisch Instituut') is a scientific institute of the Dutch government specialized in meteorology, climate science, and seismology. The KNMI provides access to weather information on their open data API end point [73] and via downloadable data sets [74]. The KNMI has an extensive observation network of automatic weather stations in the Netherlands, including 48 automatic weather stations that are at most 30 kilometers apart. The automatic weather stations perform synoptic observations, meaning that observations are made at the same time and in the same way all over the world [75]. For this thesis, seven weather data sets were acquired from the KNMI: weather and air pressure, soil temperatures, precipitation data, sunshine duration and radiation, humidity and temperature, wind data, and cloud data. These data sets contain the measured weather data for the time period 2018-08-01 00:10:00+00:00 to 2020-09-01 00:00:00+00:00, which is the same time period in which the electricity consumption data of the GridFlex data set [10] was measured. All houses from the GridFlex data set are located in Heeten [10] and the weather station closest to Heeten is located in Heino. However, since this weather station is not able to measure all variables (from all seven selected data sets), a different (closest possible) weather station was selected in those cases. Each data set, including the weather station where its data was measured, its variables, and their descriptions and units are listed in Table 4.2.

The weather data from these data sets has a ten minute granularity. The weather data of the KNMI [74] needs to be matched with the electricity consumption data of the households in the GridFlex data set [10], which has a 15 minute granularity after the resampling choice made in Section 4.3.4. Table 4.3 shows that the energy data is comprised of a mean over 15 minutes, each 15 minute interval and that the weather data is a single measurement each tenth minute. A drawback of this format is that the weather data at a ten minute frequency does not correspond one-to-one with the energy data at a 15 minute frequency. For example, in some 15 minute intervals (e.g. interval 30-44) two weather data points exist, while in other intervals only one weather data point exists. A possible approach would be to upsample the weather data to a one minute frequency using linear interpolation, and then a downsample using a mean to the 15 minute frequency. However, this is not allowed because the

Data Set	Location	Variable	Description	Unit
Weather and air pressure	Deelen waarneemterrein	P_NAP_MSL_10	air pressure converted to msl or nap 10'	hPa
		P_STN_LEVEL_10	air pressure converted to altitude	hPa
		P_SENSOR_10	air pressure sensor height	hPa
		VV_10	visibility average	m
		WW_IND_CURR_10	weather indicator past weather	code
		WW_IND_PAST_10_10	weather indicator past weather previous 10'	code
		WW_CURR_10	weather code present weather sensor	code
		WW_PAST_10	weather code previous 10'	code
		AH_10	visibility brightness background	Candela
		MOR_10	visbility meteorological daysight	m
Soil temperatures	Marknesse	T_DRYB_MIN5CM_10	soil temperature at -0.05 m	$°C$
		T_DRYB_MIN10CM_10	soil temperature at -0.10 m	$°C$
		T_DRYB_MIN20CM_10	soil temperature at -0.20 m	$°C$
		T_DRYB_MIN50CM_10	soil temperature at -0.50 m	$°C$
		T_DRYB_MIN100CM_10	soil temperature at -1.00 m	$°C$
Cloud data	Deelen locatie A	H_LAYER_10	cloud height first layer of vertical visibility	m
		H1_LAYER_10	cloud height first layer (ceilometer)	m
		H2_LAYER_10	cloud height second layer (ceilometer)	m
		H3_LAYER_10	cloud height third layer (ceilometer)	m
		H_CEILOM_10	cloud height first layer	m
		H1_CEILOM_10	cloud height first layer (ceilometer)	m
		H2_CEILOM_10	cloud height second layer (ceilometer)	m
		H3_CEILOM_10	cloud height third layer (ceilometer)	m
		N_LAYER_10	clouds total degree of cover (ceilometer)	octa
		N1_LAYER_10	clouds degree of cover first layer (ceilometer)	octa
		N2_LAYER_10	clouds degree of cover second layer (ceilometer)	octa
		N3_LAYER_10	clouds degree of cover third layer (ceilometer)	octa
		N_CEILOM_10	clouds total degree of cover (ceilometer)	octa
		N1_CEILOM_10	clouds degree of cover first layer (ceilometer)	octa
		N2_CEILOM_10	clouds degree of cover second layer (ceilometer)	octa
		N3_CEILOM_10	clouds degree of cover third layer (ceilometer)	octa
		NH_CEILOM_10	clouds degree of cover low and middle layer	octa
Precipitation data	Heino	DR_PWS_10	precipitation duration present weather sensor	s
		DR_REGENM_10	precipitation duration electric rain gauge	s
		WW_COR_10	weather corr. code present weather sensor	code
		RI_PWS_10	precipiation intensity present weather sensor	mm/h
		RI_REGENM_10	precipiation intensity electric rain gauge	mm/h
Sunshine duration and radiation	Heino	Q_GLOB_10	radiation global average	W/m^2
		QN_GLOB_10	radiation global minimum	W/m^2
		QX_GLOB_10	radiation global maximum	W/m^2
		SQ_10	sunshine duration derived from radiation	min
Humidity and temperature	Heino	U_BOOL_10	humidity air code boolean	code
		T_DRYB_10	temperature air	$°C$
		TN_10CM_PAST_6H_10	temperature air minimum 0.1 m	$°C$
		T_DEWP_10	temperature air converted to dew point	$°C$
		T_DEWP_SEA_10	temperature air converted to dew point sea	$°C$
		T_DRYB_SEA_10	temperature air height oil rig	$°C$
		TN_DRYB_10	temperature air minimum	$°C$
		T_WETB_10	temperature air converted to wet bulb	$°C$
		TX_DRYB_10	temperature air maximum	$°C$
		U_10	humidity air relative	%
		U_SEA_10	humidity air relative sea	%
Wind data	Heino	FF_10M_10	wind speed average sea converted. 10m land height sensor	m/s
		DD_10	wind direction average height sensor	degrees
		DDN_10	wind direction minimum average height sensor	degrees
		DD_STD_10	wind direction standard deviation height sensor	degrees
		DDX_10	wind direction maximum average height sensor	degrees
		FF_SENSOR_10	wind speed average height sensor	m/s
		FF_10M_STD_10	wind speed standard deviation converted to 10m	m/s
		FX_10M_10	wind speed actual maximum sea converted. 10m land height sensor	m/s
		FX_10M_MD_10	wind speed sea : actual maximum land : marked discontinuity maximum	m/s
		FX_SENSOR_10	wind speed actual maximum height sensor	m/s
		FX_SENSOR_MD_10	wind speed aviation maximum height sensor	m/s
		SQUALL_10	squall indicator	boolean

Table 4.2: Weather Variable descriptions from seven data sets of the KNMI [74]

Minutes	0 - 14	15 - 29	30 - 44	45 - 59	60 - 74	...
Energy data	avg(0-14)	avg(15-29)	avg(30-44)	avg(45-59)	avg(60-74)	...
Weather data	0, 10	20	30, 40	50	60, 70	...

Table 4.3: Minutes at which the energy data of the GridFlex data set [10] and weather data of KNMI [74] are sampled.

interpolated values would then carry information of a measurement that is done in the future, and an information leakage from the future into the past must be prevented. A fair approach would be to map the weather data to the energy data, by taking the average over both points (e.g. average over weather data at minute 30 and minute 40) in case two weather data points exist in an interval (e.g. the interval

30-44), and simply take the corresponding single weather data point (e.g. weather data at minute 50) in case one weather data point exists in an interval (e.g. interval 45-59).

Figure 4.9 shows the results of a correlation analysis including the correlation of each weather variable with the total electricity consumption (TOTAL_KW) of Household 8, 26, 28, 29, 51, and 52, to determine the relevance of the weather variable for predicting household electricity consumption and to decide which input variables the forecasting model uses. Each subfigure of Figure 4.9 corresponds to a weather data set. Table 4.2 describes the variable names on the x-axes of each subfigure in more detail. The correlation between the weather variables and the electricity consumption differs per household. Specifically, weather variables correlate more with the household electricity consumption, if the house has solar panels (see Table 1.1). This suggests that the weather mostly has an effect on the electricity yield of solar panels, but barely on electricity import alone. Since many houses have solar panels, and this number is expected to rise in the Netherlands [6], [7], we decided that the proposed forecasting model will use some weather variables as inputs. However, some weather variables are not used as input to the forecasting model, because they have a small (either positive or negative) correlation with the total electricity consumption of a household. Furthermore, they might be inter-correlated (dependent) and including such inter-correlated variables would be redundant and only increases the complexity of the forecasting model. Table 4.2 displays weather variables that were selected as input to the proposed forecasting model, coloured in light cyan. All other weather variables have been omitted.

4.4.2 Historic Data Correlation Analysis

Historical household electricity consumption is often considered to be indicative for the household electricity consumption in the future (see Section 2.2.1), because appliances in a house do not change often and occupants use many devices in a repetitive manner. These cause repeating patterns at specific time intervals in the data. Therefore, one of the characteristics of electricity consumption data is that there exists seasonality (see Section 2.2.1). However, it is not always clear at which time intervals there is seasonality. For example, one could take into account the electricity consumption at exactly the same time on the previous day. However, repetitive behaviour may also occur over longer time periods, such as the seasons of the year. In the summer there is more solar production than in the winter, causing the total electricity consumption of households with solar panels to decrease during summer. Some types of seasonal behaviour affect large groups of households (e.g. all households with solar panels) and some affect only single households (e.g. an occupant with its individual week schedule). This section looks into three different

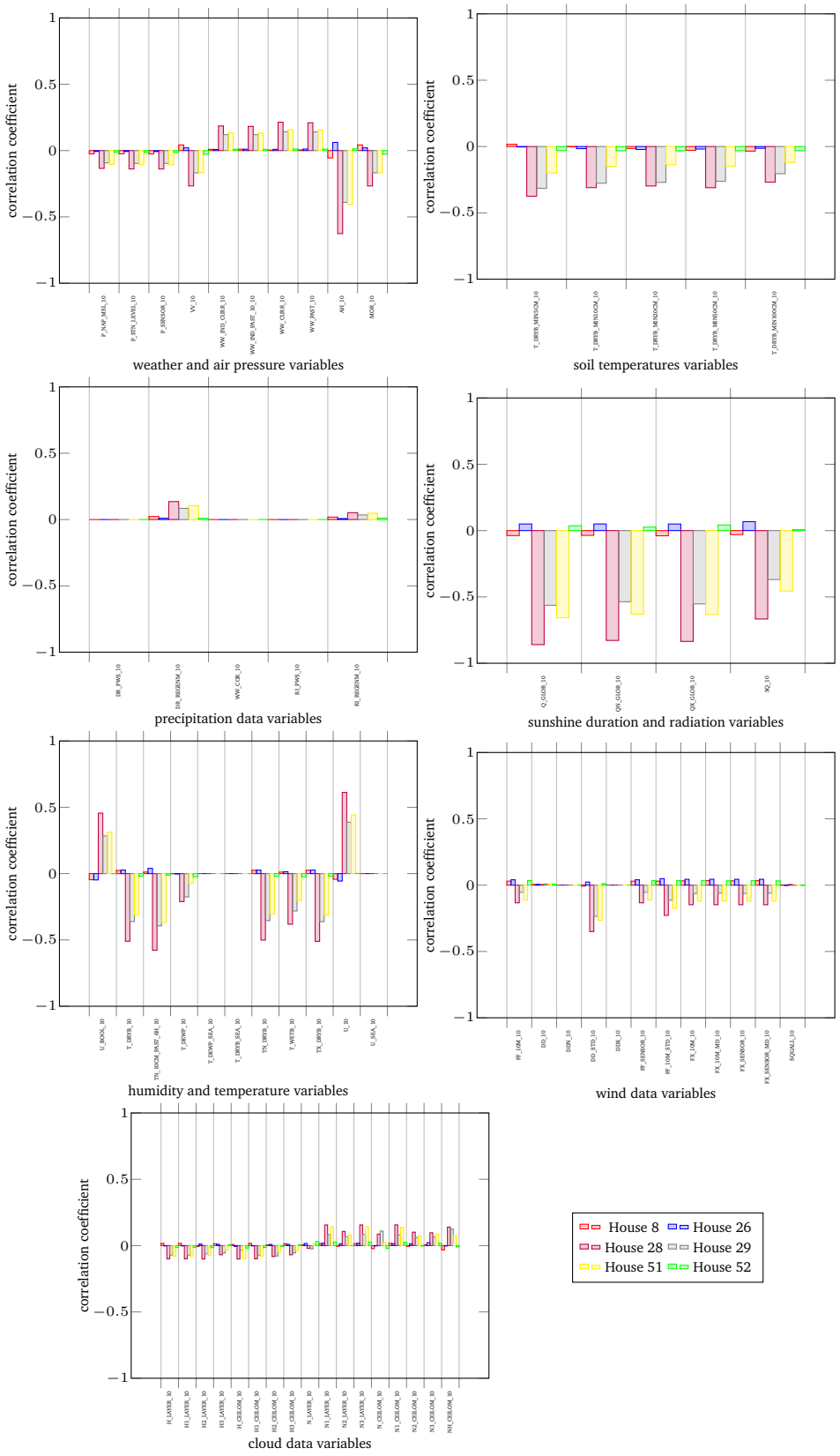


Figure 4.9: Weather Correlations

types of variables that capture historic electricity consumption that the proposed forecasting model can use as input: lag variables, window variables, and trend variables.

4.4.2.1 Lag Variables

When creating a forecasting model, not only the current electricity consumption can be used as input, but also the electricity consumption from the past. However, not all historic electricity consumption may be equally useful for predicting the electricity consumption. In order to take historic electricity consumption into account, lag variables can be used. A lag variable is an input variable that has the value of a previous time interval. For example, a 45 minute lag variable is the electricity consumption of 45 minutes ago. With multiple different lag variables as input to the forecasting model, different parts of the historic electricity consumption can be taken into account when making a prediction. However, the forecasting model would become very large if all lags would be used as input to the forecasting model. Hence, a choice has to be made what parts of the historic electricity consumption should be used as input to the proposed forecasting model. Plotting the correlation between the electricity consumption now and the electricity consumption at a specific lag helps to make this choice. Therefore, Figure 4.10 shows the autocorrelation of the *average* household (average over all houses of the GridFlex data set [10]) for different periods of time. The first hour is shown in the top left graph of Figure 4.10. The first (approximately) 20 lags have a correlation that decreases almost exponentially. After that, the decrease takes on a more linear shape. This suggests that the lags of less than 20 minutes are the most important for predicting the current electricity consumption. A correlation peak occurs each past day at exactly the same time, in the top right graph. This suggests that the electricity consumption at exactly the same time a few days ago may be useful for predicting the current electricity consumption. Additionally, the correlation with the electricity consumption exactly one day ago is higher than the correlation with the electricity consumption half a day ago, implying that the electricity consumption at a more recent moment is not generally more indicative of the current electricity consumption. These findings are in line with those reviewed by Hippert et al. [19]. Additionally, Figure 4.10 shows periods of one month and three months in the graphs on the middle row. It shows a clear declining trend. The bottom left graph of Figure 4.10 clearly shows yearly seasonality, represented by correlations that are much higher at the same time of year than those at other periods of the year. The bottom right graph of Figure 4.10 shows that also a declining trend can be observed over the years.

Figure 4.11 focuses on the period of a month. While one may expect to observe seasonality for a day of the week, this is hard to see due to the daily seasonality. When plotting an envelope over the peaks of the graph, daily seasonality can be disregarded.

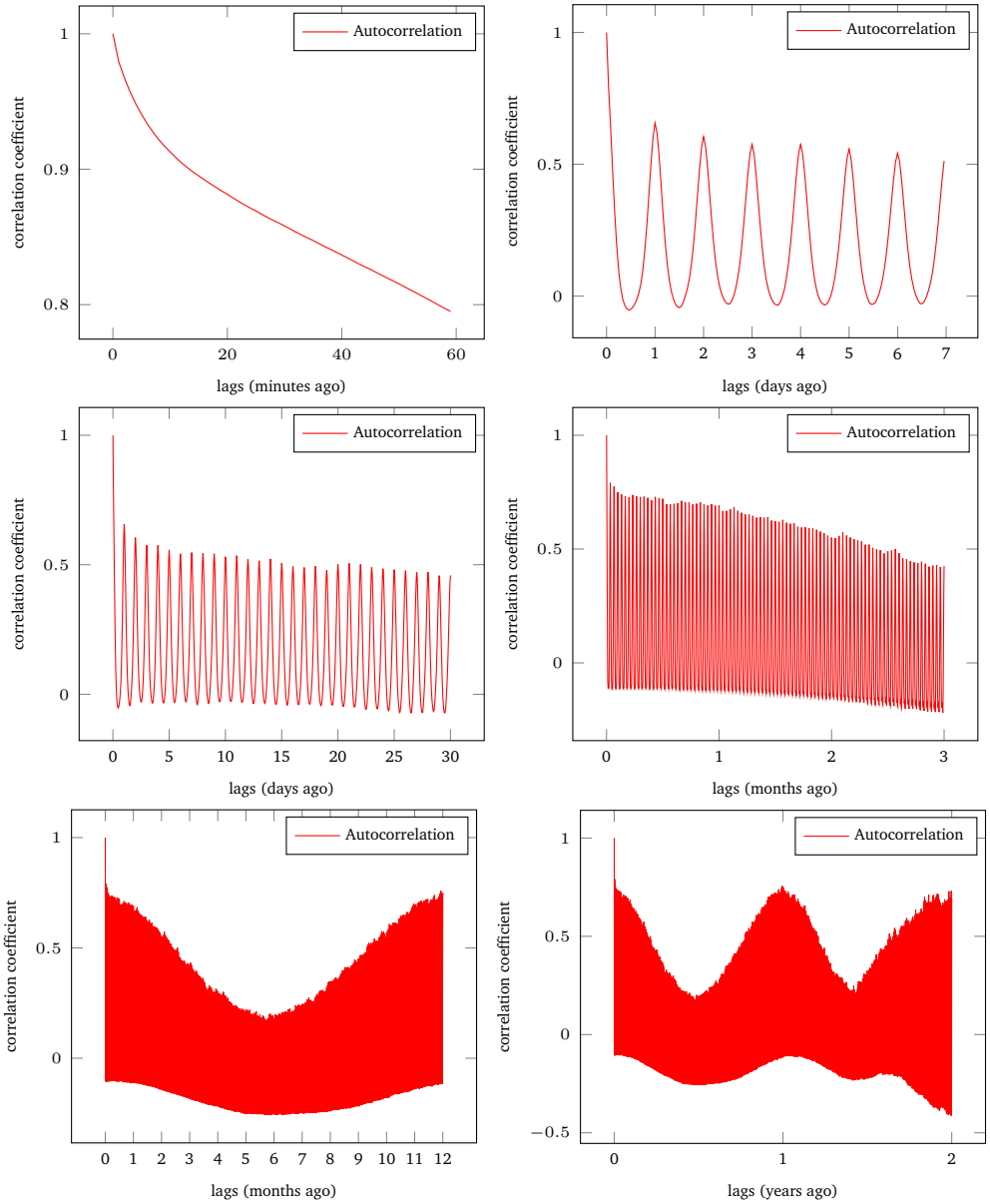


Figure 4.10: Autocorrelation of the electricity consumption of the *average* house (average over all houses from the Gridflex data set [10]) for different lags. Each graph zooms in on a different period of time (one hour, one week, one month, three months, one year, two years, respectively)

The right graph of Figure 4.11 makes it easier to observe weekly seasonality as the envelope is plotted separately. Although really small local correlation peaks appear for each seventh day, they are insignificant compared to peaks of daily seasonality. This might be due to the use of the *average* household (average over all houses from the Gridflex data set [10]) for this correlation analysis. As discussed above, weekly repetitive behaviour might be very personal and thus different for each house. Using the *average* household may mask weekly seasonality that exists for specific households.

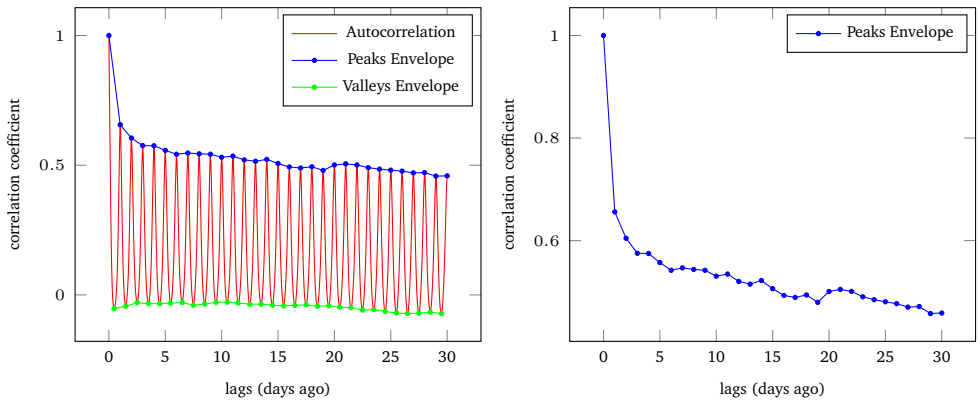


Figure 4.11: Autocorrelation of the electricity consumption of the *average* house (average over all houses from the Gridflex data set [10]) for different lags in a period of a month. The blue and green lines display the peak and valley (resp.) envelopes. The blue envelope is shown separately in the graph on the right.

It is important to note that all graphs in Figure 4.10 and 4.11 display correlations of the electricity consumption at a specific lag with the electricity consumption at some time t. However, we are interested in the correlation between the lag and the *predicted* electricity consumption at some time $t + h$. To find out what lags are relevant for the prediction of the electricity consumption at time $t + h$, the plots of Figure 4.10 could be "shifted" to find out what lags have the highest correlation with the electricity consumption at time $t + h$. In practice, a prediction is a window in the future that consists of multiple values (not a single value). For example, a prediction for the upcoming day consists of 96 values/points when the data has 15 minute granularity: $t+1, t+2, ..., t+96$. For each of those values, different lags will be relevant. Figure 4.12 shows an example of how the plots of Figure 4.10 could be "shifted" to find out what lags have the highest correlation with the electricity consumption at time $t + h$. For $h = 96$, the current electricity consumption at time t has the highest correlation, at time $t + 48$, the lag at $t - 48$ is the highest, and $t + 24$, the lag at time $t - 72$ has the highest correlation. Thus, there are multiple relevant lags (96 for each past day) for a prediction window. Nine lag variables

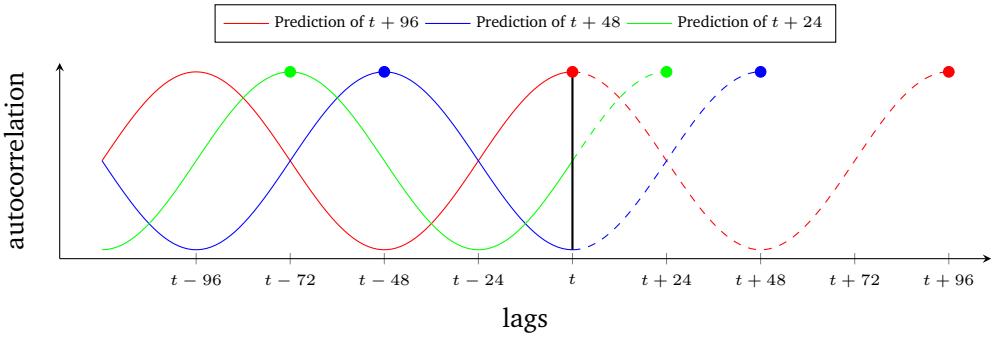


Figure 4.12: Lag relevance for prediction. "Shifting" the autocorrelation plot for each prediction.

are selected (covering the last two hours) as input for the proposed model and presented in Table 4.4, because the autocorrelations of the lags of the last two hours are significantly higher than the autocorrelations of later lags. Together, these lags capture only the recent historic (not more than two hours ago) energy consumption. The window variables are used to capture older energy consumption.

Lag	Period
0	current energy consumption
1	energy consumption 15 minutes ago
2	energy consumption 30 minutes ago
3	energy consumption 45 minutes ago
4	energy consumption 60 minutes ago
5	energy consumption 75 minutes ago
6	energy consumption 90 minutes ago
7	energy consumption 105 minutes ago
8	energy consumption 120 minutes ago

Table 4.4: Lag variables

4.4.2.2 Window Variables

When purely using lag variables to capture seasonality and historic electricity consumption, many input variables may be needed, especially when predicting a window of values in the future. A new set of lag variables is needed then for each historic electricity consumption value or seasonality that may be relevant for the prediction. Furthermore, one lag may have an exceptional value (i.e. is an outlier) so multiple lags are needed. Each of these window variables needs to be calculated and serves as extra input to the forecasting model, making the model more complex. To address this, window variables [76] can be used. Window variables are a statistic over multiple lags. In other words, they calculate a value over a window of past electricity consumption values. For example, a window variable may be the standard deviation of electricity consumption over all electricity consumption values in the past three hours. Another example of a window variable is the average electricity consumption value of all Mondays at exactly the same time. The chosen window variables are listed in Table 4.5. In total, there are 203 window variables.

Count	Statistic	over	Covering a period of	Window size	weighted
1	mean	all intervals	last three hours	12	by correlations of Fig 4.10
1	st.dev.	all intervals	last three hours	12	by correlations of Fig 4.10
1	mean	all intervals	last three hours	12	no
1	st.dev.	all intervals	last three hours	12	no
1	mean	all intervals	last day	96	by correlations of Fig 4.10
1	st.dev.	all intervals	last day	96	by correlations of Fig 4.10
1	mean	all intervals	last day	96	no
1	st.dev.	all intervals	last day	96	no
1	mean	all days (at the exact same 15-minute interval of the day)	last week	7	by correlations of Fig 4.10
1	mean	all days (at the exact same 15-minute interval of the day)	last week	7	no
1	mean	all seventh days (at the exact same 15-minute interval of the day)	last month	4	no
96	mean	ith unique 15-minute interval of the day	last month	30	no
96	mean	ith unique 15-minute interval of the day	last week	7	no

Table 4.5: Window variables used to incorporate historic total electricity consumption

4.4.2.3 Trend Variables

The trend or evolution of the total electricity consumption may be captured using the first and second derivatives over several intervals, as features. To numerically approximate the derivatives, they were calculated using a finite backward difference quotient. An approximation of the first derivative was calculated using

$$x_i' = \frac{x_i - x_{i-p}}{p} \tag{4.1}$$

and an approximation of the second derivative was calculated using

$$x_i'' = \frac{x_i - 2x_{i-p} + x_{i-2p}}{p^2}. \tag{4.2}$$

Here, p indicates the period over which the finite backward difference quotients were calculated. The values $p \in [1, 2, 3, 4, 96, 7 \times 96, 30 \times 96]$ were taken to represent the derivatives over the last hour (1-4), exactly the last day (96), the last week (7×96), and the last month (30×96). Table 4.6 also presents these variables.

Variable	Period p	Trend over	Equation
first derivative	1	last 15-minute interval	(4.1)
second derivative	1	last 15-minute interval	(4.2)
first derivative	2	last half hour	(4.1)
second derivative	2	last half hour	(4.2)
first derivative	3	last three 15-minute intervals	(4.1)
second derivative	3	last three 15-minute intervals	(4.2)
first derivative	4	last hour	(4.1)
second derivative	4	last hour	(4.2)
first derivative	96	last day	(4.1)
second derivative	96	last day	(4.2)
first derivative	7×96	last week	(4.1)
second derivative	7×96	last week	(4.2)
first derivative	30×96	last month	(4.1)
second derivative	30×96	last month	(4.2)

Table 4.6: Trend variables

4.4.3 Time Data

To explicitly incorporate time information into the prediction model, a set of time and holiday variables is used. These variables are very simple and are presented in Table 4.7. Time variables include the hour of the day, day of the week, month, quarter, and year, whether a day is a work- or weekend-day, public holiday, and public break. Note that all of the public holidays of Table 4.7 are official Dutch holidays, except Memorial Day, Sinterklaas, and New Year's Eve. The dates of some of the public holidays may depend on the year. For example, the dates of Good Friday, Easter Sunday, Easter Monday, Ascension Day, Whit Sunday, and Whit Monday differ per year. Public holidays may be useful, because people spend time (and thus electricity) differently on a holiday than on any other day. Public vacation or breaks are based on Dutch nation school break periods, which differ per year and per region of the country (north, middle, south). The break periods in the table are the union

Description	Value	Meaning
Hour	0-23	corresponds to the hour of the day
Day of week	0-6	0=Monday, ..., 6=Sunday
Month	1-12	1=January, ..., 12=December
Quarter	1-4	corresponds to the quarter of the year
Year	e.g. 2019	simply the year
Weekend day	0-1 Binary	1 if Saturday or Sunday, 0 otherwise
Work day	0-1 Binary	0 if Saturday or Sunday, 1 otherwise
Public Holiday	0-1. Binary	1 if date is New Year's Day, Good Friday, Easter Sunday, Easter Monday, King's Day, Memorial Day, Liberation Day, Ascension Day, Whit Sunday, Whit Monday, Sinterklaas, Christmas Day, Boxing Day, or New Year's Eve. 0 otherwise.
Public Vacation	0-1. Binary	1 if date in either spring break, may break, summer break, autumn break, Christmas break. 0 otherwise.

Table 4.7: Time variables

of all regions for a particular year, to make the forecasting model more independent of the geographical location of a household. Many Dutch citizens plan vacations (away from home) or are at home during these periods. Therefore, electricity may be used differently during these periods compared to regular periods.

4.4.4 Overview Of Forecasting Model Input Variables

Section 4.4.1, Section 4.4.2, and Section 4.4.3 discuss weather, historic electricity consumption, and time variables that may be used as input to the forecasting model. This section provides a small overview of all the input variables for the proposed forecasting model. Table 4.2 presents the 16 weather variables (coloured in light cyan) that are chosen as input for the forecasting model, based on the analysis in Section 4.4.1. In Section 4.4.2, Table 4.4 presents nine lag variables that try to capture recent historic energy consumption, Table 4.5 presents 203 window variables that try to capture the seasonality and Table 4.6 presents 14 trend variables that try to capture the evolution of the energy consumption over time. Together, they form the 226 electricity (lag & window & trend) variables that are chosen as input for the forecasting model. Table 4.7 presents the nine time variables that are chosen as input for the forecasting model. Thus, there are $16 + 226 + 9 = 251$ (weather, energy, time) input variables in total that the proposed forecasting model will use. Note that the weather and energy variables are either measured or calculated values, but the time variables are either binary of integer values. Since the forecasting model is based on ANNs, a new input neuron is created for the input layer of the ANN (see Chapter 5) for each variable. Figure 4.13 presents this idea.

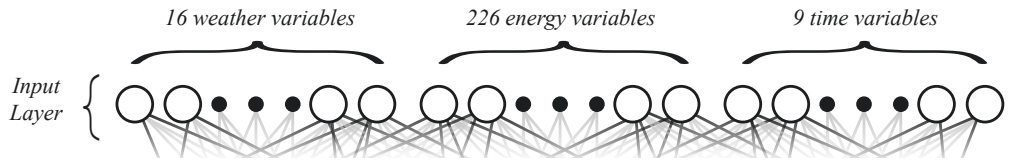


Figure 4.13: ANN input layer

5 Prediction Model Structure

Chapter Objective: *This chapter explains the proposed forecasting model.*

Chapter Contents

5.1 Introduction

Recall that the objective of this research is to develop an online "rolling horizon" forecasting model that is continuously being trained on the smart meter data of a single household and can therefore adapt to the load profile behaviour of that household and to structural changes in electricity consumption over time. Hence, a single untrained basic forecasting model is proposed that can adapt to a specific household when it is trained on the data of that household. In order to predict the energy consumption of multiple households, a copy of the untrained model is applied to each single household. After a reasonable time frame, each copy of the forecasting model is supposed to "learn" how to predict the load profile of the household it has been training on, and reach a reasonable prediction accuracy. After training for some time, each model has adapted to its corresponding household and in the end, each model makes very different predictions (based on the household), meaning that the models are very different for the different households even though they started as copies from the same basic untrained model. Note that this is different from one single pre-trained machine learning model that is the same for every household.

This chapter explains how exactly the forecasting model learns and makes predictions over time. So far throughout this thesis, the forecasting model is considered a black box with an input (Chapter 4) and an output. Section 5.2 explains how the rolling

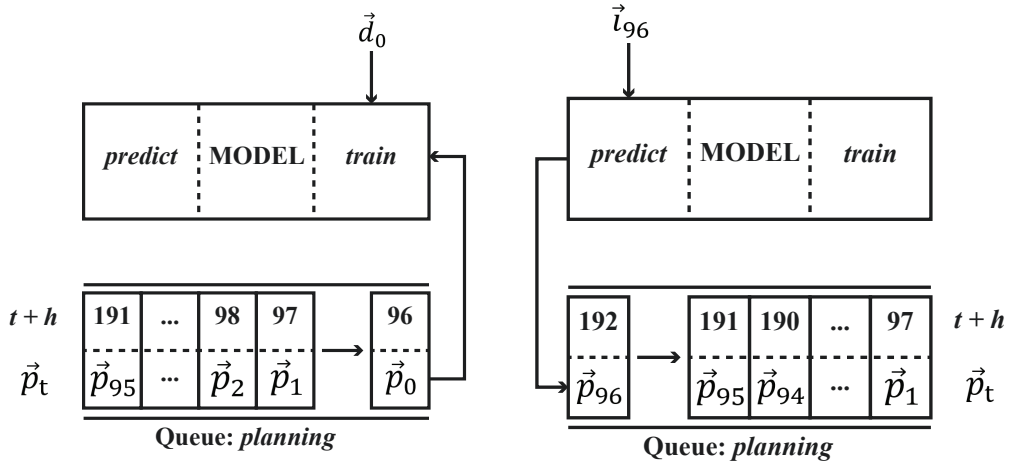
horizon is implemented, while still considering the forecasting model as a black-box. Using discrete time-steps, it is explained how the forecasting model makes predictions in time and plans training updates to be able to adapt to the load profile of a specific house over time. Section 5.3 breaks away from this black-box paradigm and explains what is inside the black-box: an ANN architecture. In this section, different ANN architectures used to make predictions in the forecasting model are illustrated and compared. Also a connection is made with the black-box paradigm used in Section 5.2 to give an understanding of how the forecasts made with the ANN architecture fit together in the discrete time-step rolling horizon. Then, Section 5.4 provides an overview and analysis of the design and implementation choices regarding the ANN architectures discussed in Section 5.3. Finally, Section 5.5 explains how the predictions of the proposed forecasting model can be used as input for a DSM algorithm in two different scenarios. Both scenarios provide the option to use only initial predictions or to dynamically update the predictions that are provided to the DSM algorithm, so that the algorithm can take this into account possibly adapt its schedule.

5.2 Rolling Horizon

The proposed forecasting model can be considered a black box. Using the features described in Section 4.4, the forecasting model has an input of in total 251 values. The predictions that are most commonly used in practice as input for DSM techniques, can be used to determine the output of the forecasting model. For many DSM applications on the distribution grid, a forecast horizon of a day is used with a granularity of 15 minutes (e.g. [25], [33]). Therefore, the forecasting model needs to make 96 load predictions (there are 96 15-minute intervals in a day) in every forecast. This *forecast window* consisting of 96 values corresponds to the next day and is also called the *predicted load profile* of the next day.

To formalize the above, let MODEL denote the forecasting model. Let $\vec{e}$ be the list of measured load values of the smart meter of a household from the GridFlex data set [10] over the approximately two years of data. Let the function *len* denote the length of a list. Then, *len*($\vec{e}$)$= 73144$, corresponding to the 73144 15-minute intervals in $\vec{e}$. Every (integer) time step t, the MODEL has to make a prediction and possibly train based on a prediction that was done in the past. Each time step t corresponds to a 15-minute interval of the data. The list of input values $\vec{i}_t$ is recalculated each time step t as described in Section 4.4, using a function *calculate_input*. One of the functions of the MODEL is *predict*, which takes input values $\vec{i}_t$ as input and produces a prediction $\vec{p}_t$. $\vec{p}_t$ is the predicted load profile of the upcoming day, consisting of 96 values. When a prediction $\vec{p}_t$ is made, it is appended to a list of predicted load

profiles `predictions` using the function *append*. For example, if a prediction $\vec{p}_4$ was made at $t = 4$, then $\vec{p}_4$ contains the predicted values corresponding to $t = 5$ until $t = 100$ and `predictions` is appended with $\vec{p}_4$.



(a) Training at time $t = 96$ (#TRAINING in Algorithm 1)

(b) Prediction at time $t = 96$ (#PREDICTION in Algorithm 1)

Figure 5.1: At time $t = 96$, the MODEL is first trained (shown in 5.1a) using the scheduled prediction in the queue `planning` and then makes a new prediction which is added to the queue `planning` (shown in 5.1b)

The second function of MODEL is *train* and trains the model. It takes as input two arguments: a prediction $\vec{p}_t$ and a desired output $\vec{d}_t$. $\vec{d}_t$ is a subsection of the actual load $\vec{e}$. Continuing the example, $\vec{d}_4 = [e_5, ..., e_{100}]$ corresponds to $t = 5$ until $t = 100$. In an online setting, the model cannot be trained on information from the future. To prevent information leakage from the future, the MODEL may not be trained before it is allowed. When a prediction is made at t, a training update of the MODEL is scheduled at $t + h$, where the constant $h = 96$ corresponds to the forecasting horizon. For example, when a prediction is made at time t, the prediction can be validated only at time $t + h$ because only after h time steps the actual electricity consumption $\vec{d}_t$ is known. Training earlier (e.g. already at time t) would take information ($\vec{d}_t$) from the future (at $t + h$) to update the MODEL, and this would not be possible in an online scenario. The scheduling is done using a queue, called `planning`. The queue `planning` contains tuples with the time of the scheduled training update $t + h$ and the prediction $\vec{p}_t$. Every time a training is completed, the corresponding scheduled tuple is removed from `planning` using *pop(0)*. Scheduling a training update using the queue `planning` is illustrated in Figure 5.1.

The algorithm is displayed in Algorithm 1. Note that Algorithm 1 also handles *NaN* values. Even though most of these are already imputed, some *NaN* values may still remain as described in Section 4.3.2. In an online scenario, the forecasting model

should also be able to handle these issues. Note that `predictions` is a list containing

Algorithm 1 Rolling Horizon

Require: MODEL
Require: $\vec{e}$
  predictions ← [ ]
  planning ← [ ]
  for $t \leftarrow 0$ **to** *len($\vec{e}$)* **do**
    # INPUT
    $\vec{i_t} \leftarrow$ *calculate_input()*

    # TRAINING
    if *len(*planning*)* > 0 **and** t is planning[0][0] **then**
      planning.*pop(0)*
      $\vec{d_t} \leftarrow \vec{e}[\ t - h + 1 : t + 1\ ]$
      if $\vec{e}$ **is** not *NaN* **then**
        MODEL.*train*(planning[0][1], $\vec{d_t}$)
      end if
    end if

    # PREDICTION
    if $\vec{i_t}$ is not *NaN* **then**
      $\vec{p_t} \leftarrow$ MODEL.*predict($\vec{i_t}$)*
      predictions.*append($\vec{p_t}$)*
      planning.*append(* $(t + h, \vec{p_t})$ *)*
    else
      $\vec{p_t}$*.append(NaN)*
    end if
  end for
  return predictions

sub-lists. The tth element (or sub-list) of `predictions` is a predicted load profile $\vec{p_t}$ for the upcoming day, made at time t. The predicted load profile $\vec{p_t}$ contains the predicted values corresponding to $t + 1$ until $t + 96$. Even though the forecast window slides by only one time step t, a completely new load profile is predicted. For example, the forecast window at $\vec{p_0}$ corresponds to $t = 1$ until $t = 96$, and the forecast window $\vec{p_1}$ corresponds to $t = 2$ until $t = 97$. Hence, while the time window to which the forecast window $\vec{p_1}$ corresponds differs only by removing $t = 1$ and adding $t = 97$ to the forecast window of $\vec{p_0}$, its predictions for these time step may be completely different. This example is illustrated in Figure 5.2. Here $\vec{p_t}$ corresponds to the forecast window predicted at time t.

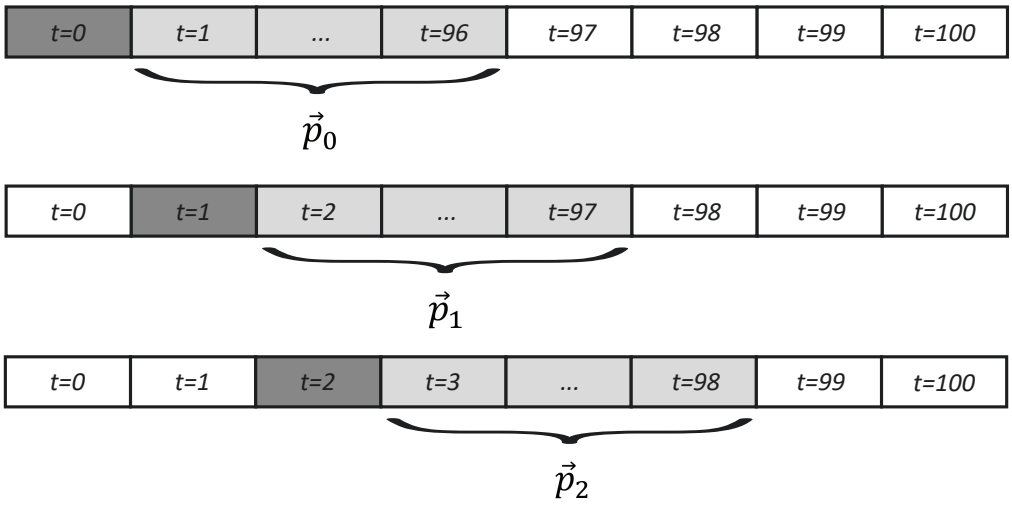


Figure 5.2: Rolling horizon

5.3 Forecasting Model Architecture

So far, the forecasting model is regarded as a black box. This section breaks away from the black-box paradigm and describes what is inside this black box: the architecture of the MODEL (see Section 5.2). The forecasting model could be any model, but in in this work we chose to focus on ANNs (see Section 2.3 for a motivation). Therefore, the *train*($\vec{p}_t$, $\vec{d}_t$) function of the MODEL mentioned in Section 5.2 uses the prediction $\vec{p}_t$ and target $\vec{d}_t$ to calculate the cost function and update all weights and biases in the ANN using gradient descent, as described in Section 2.4.2. The input layer of the ANN consists of $\vec{i}_t$, where each element corresponds to a single input neuron. Thus, the input layer has 251 input neurons. In Section 2.4.3.2, three types of ANN architectures were classified: iterative forecasting (one-step), multivariate forecasting (multi-step), and multi-model forecasting [19]. These three ANN architectures plus a fourth hybrid architecture, are implemented and compared. In this section, their differences are discussed.

Multivariate forecasting (multi-step)

In this type of architecture, the output layer of the ANN consists of $h = 96$ output neurons. Each neuron in the output layer corresponds to exactly one value of the forecasting window $\vec{p}_t$. Hence, only a single feed-forward pass (see Section 2.4.2) is needed to get a prediction of the load profile for the upcoming day $\vec{p}_t$. Then, $\vec{p}_t$ forms the output to the function *predict*($\vec{i}_t$) discussed in Section 5.2. To train the MODEL using this architecture, *train*($\vec{p}_t, \vec{d}_t$) is called once at time $t + h$. Each of the components of the desired or actual load $\vec{d}_t$ correspond to a target value (denoted

by y_j in Section 2.4.2) in the cost function, and are used to update the parameters of the ANN. This type of architecture is shown in Figure 5.3.

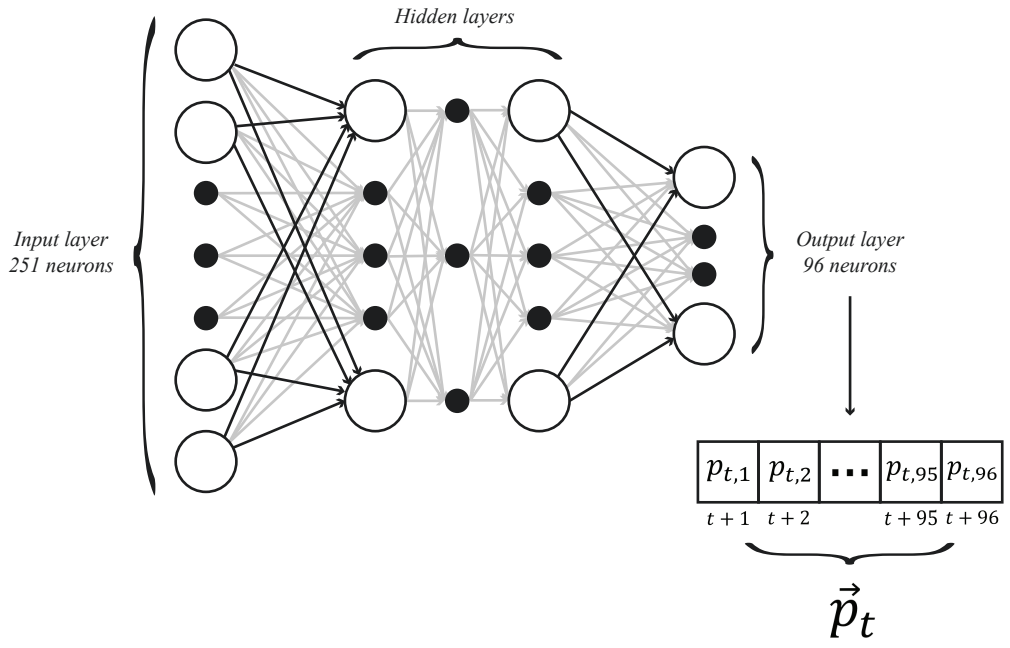


Figure 5.3: Multivariate forecasting (multi-step) ANN architecture

Iterative forecasting (one-step)

In this type of architecture, the output layer of the ANN consists of one single output neuron. This neuron obtains a value after one feed-forward pass of the neural network. This value is the first value (corresponding to $t+1$) of the forecasting window $\vec{p_t}$. This first value ($p_{t,1}$) is used to update the input $\vec{i}_t$. In this update, the energy variables are recalculated, but the variables relating to weather and time are not recalculated (see Section 4.4.4 for an overview of weather, time, and energy variables). The updated vector $\vec{i}_t$ is then used as input to a subsequent feed-forward pass of the ANN. The output of this new feed-forward pass is the second value (corresponding to $t+2$) of the forecasting window $\vec{p_t}$. This second value ($p_{t,2}$) is used to update the input for the next feed-forward pass. When continuing this process, each output value iteratively forms the input for a new feed-forward pass. Each iteration, only the energy variables are recalculated based on the value of the output neuron in the previous iteration (calculation is described in Section 4.4.2). The weather and time variables remain the same across all iterations. This process is repeated iteratively until all $h = 96$ values (corresponding to $t+1, ..., t+h$) of the forecasting window $\vec{p_t}$ are determined. Again, $\vec{p_t}$ forms the output to the function *predict*($\vec{i}$). To train the MODEL using this architecture, the *train* function is called $h = 96$ time successively in a row at time $t+h$, to learn from each iterative feed-forward pass made by the network. This architecture is shown in Figure 5.4.

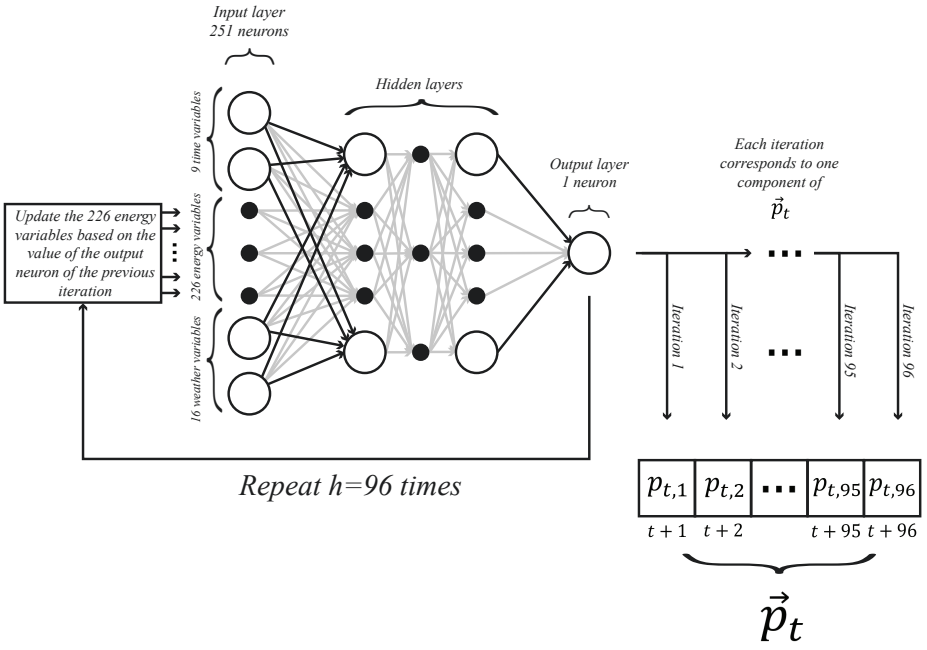


Figure 5.4: Iterative forecasting (one-step) ANN architecture

Multi-model forecasting

In this type of architecture, a single ANN is used for each of the $h = 96$ values of the forecasting window $\vec{p_t}$. This means that there are $h = 96$ ANNs in total, one corresponding to each of the $t + 1, ..., t + h$ values. Each ANN has an input layer of 251 neurons and an output layer of one single neuron. Hence, to calculate the output window $\vec{p_t}$ which forms the output to the function *predict*($\vec{i_t}$), each of the $h = 96$ ANN will need to do a single feed-forward pass. To train the MODEL using this architecture, the *train* function is called once for each ANN. Each *train* function uses only (the corresponding) one component of the target vector $\vec{d_t}$, to update the parameters of the corresponding ANN. This architecture is shown in Figure 5.5.

Hybrid multi-model multivariate forecasting

In this type of architecture, a single ANN is used for each of the 24 hours in a day, since the forecasting window $\vec{p_t}$ covers a period of a day. Each of the 24 ANN has a multivariate architecture with an output layer consisting of four neurons (since there exist four 15-minute intervals in an hour). Thus, this type of architecture combines of multi-model forecasting and multivariate forecasting. In this architecture, the first ANN predicts the first four 15-minute intervals $t + 1, t + 2, t + 3, t + 4$ of the forecasting window $\vec{p_t}$. The second ANN predicts the second four 15-minute intervals $t + 5, t + 6, t + 7, t + 8$. Following this approach, all 24 ANNs together predict the $t + 1, ..., t + h$ values of the forecasting window $\vec{p_t}$ and form the output to *predict*. This architecture is trained by calling the *train* function once for each ANN, where the first four components of $\vec{d_t}$ are used to train the first ANN, the second four

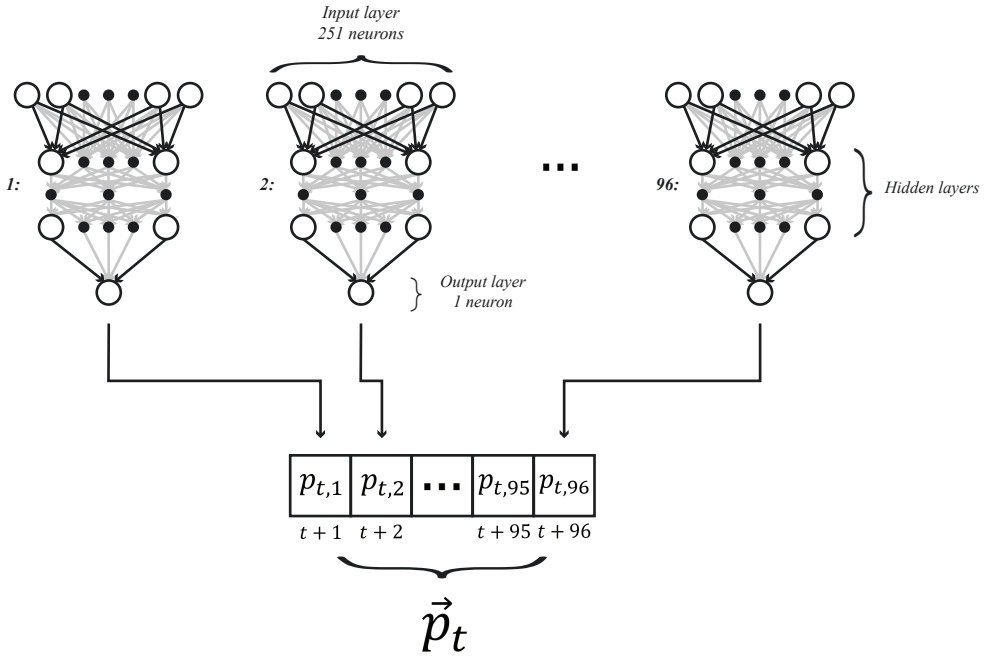


Figure 5.5: Multi-model forecasting ANN architecture

components of $\vec{d}$ are used to train the second ANN, and so on. This architecture is shown in Figure 5.6.

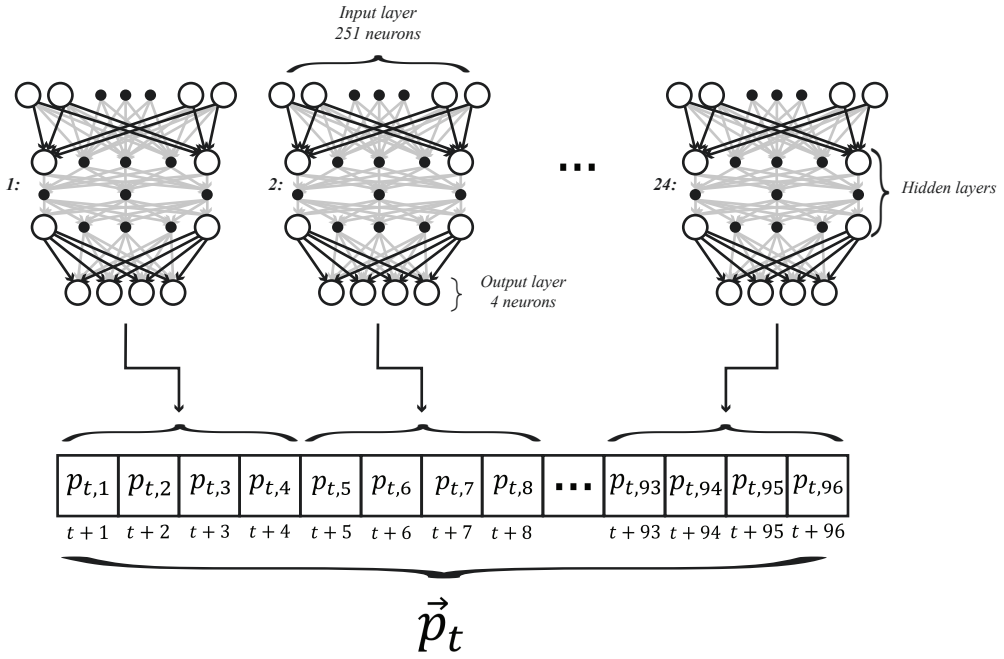
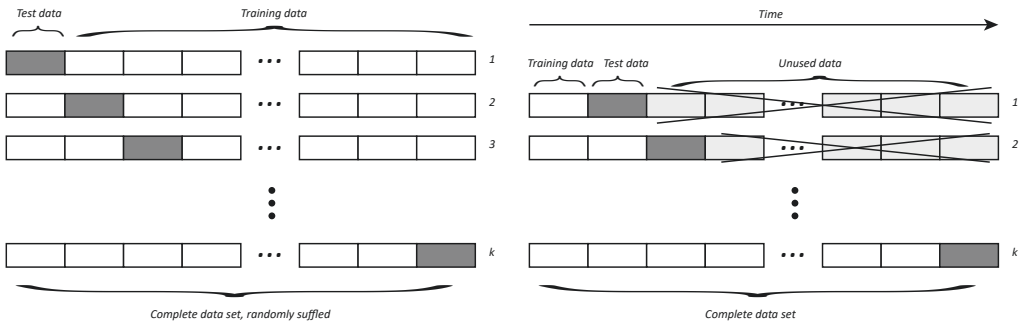


Figure 5.6: Hybrid multi-model multivariate forecasting ANN architecture

5.4 ANN Design Choices

5.4.1 Cross Validation

To evaluate a machine learning method, the data is commonly split randomly between a test set and a training set. An example of this is k-fold cross-validation, shown in Figure 5.7a. Here, the complete data set is first shuffled and then divided into k subsets (where k is smaller than or equal to the total number of data points). In the first iteration, one of the subsets (shaded gray in Figure 5.7a) is chosen to serve as test set and the other subset together serve as training set. The next iteration, the next subset is chosen as the test set. In total there are k iterations, and for each iteration the test set is used to evaluate the performance of the machine learning model. In the end, the performance of the machine learning model is evaluated as the average performance over all iterations. However, when using this approach with time series data, data points from the future might end up in the training set and data points from the past into the test set. Then for the evaluation, the machine learning method would use data from the future to predict the past which is not allowed. For time series problems, the data can therefore not be randomly split. Data points in the time series are not independent as there is a time ordering to the data [76].



(a) Normal k-fold cross validation

(b) k-fold cross validation with time ordering

Figure 5.7: k-fold cross validation

Instead, the data needs to be split by time for a time series problem. In other words, the data is not shuffled, and the time ordering of the training set always precedes the test set. The data that corresponds to the time after the test set, is not used for that iteration. This is shown in Figure 5.7b. Finally, it is important to realise that the forecasting model proposed in Section 1.2 is supposed to train indefinitely. The forecasting model continuously trains and updates its predictions based on new data. Therefore, the forecasting model is never "finished" with training. One could argue

that the training and test set in this case are the same. However, the ideas of k-fold cross validation with time ordering, shown in Figure 5.7b, can still be applied. To evaluate the performances of the different architectures described in Section 5.3, k is set equal to the number of data points. The test set consists of the next 96 time data points, and the training data set consists of the previous month. Different from what is shown in Figure 5.7b (this is called the "rolling forecast origin" technique [77]), not all data from the start ($t = 0$) is used as a training set. Only data about the previous month is used as input (this is called the "sliding window" technique [77]). To evaluate the performance of the forecasting model, $k = t$ iterations are used, similar to a simulation of discrete time steps. In this way, the forecasting model that is trained continuously can still be tested and evaluated over time.

5.4.2 Hyperparameter Choices

During the first period that the forecasting model is being applied, it will give arbitrary predictions. The performance of the model is not good at this stage, because the ANN(s) used by the forecasting model are still untrained. After training for a few iterations, the model starts to perform better. In other words, after each gradient descent step, weights and biases of the network are updated and a step is taken towards a local minimum of the cost function used by the ANN. The learning rate has an influence on the size of this step (see Section 2.4.2). Generally, when choosing the learning rate too large, a local minimum may be approached really quickly but there is a risk of overshooting it. When choosing the learning rate too small, the risk of overshooting is much smaller, but a local minimum is approached in really small steps and hence takes a long time. Since the forecasting model trains continuously and indefinitely, it keeps trying to approach a local minimum after every training step and never finishes trying to approach this minimum. Therefore, the learning rate has an effect on the behaviour of the forecasting model.

In general, forecasting forecasting model is very fast at converging to a reasonable error. Figure 5.8 shows the behaviour of the forecasting model in this startup stage stage. The x-axis represents the time, and the y-axis the actual (black) or predicted (coloured) load at that time. Note that the forecasting model starts training after exactly one day. It cannot start earlier, because only after one day the predictions of the first forecast window (spanning one day) can be validated (actual load compared with predicted load). The learning rate has an effect on how fast the model converges to a reasonable accuracy. Figure 5.8 shows the predictions of the multivariate forecasting model trained with traditional gradient descent using different learning rates. Choosing a relatively large learning rate ($\eta = 0.1$) causes the forecasting model to quickly adjust its predictions to the actual load. In other words, the model converges quickly to a reasonable accuracy and startup effects

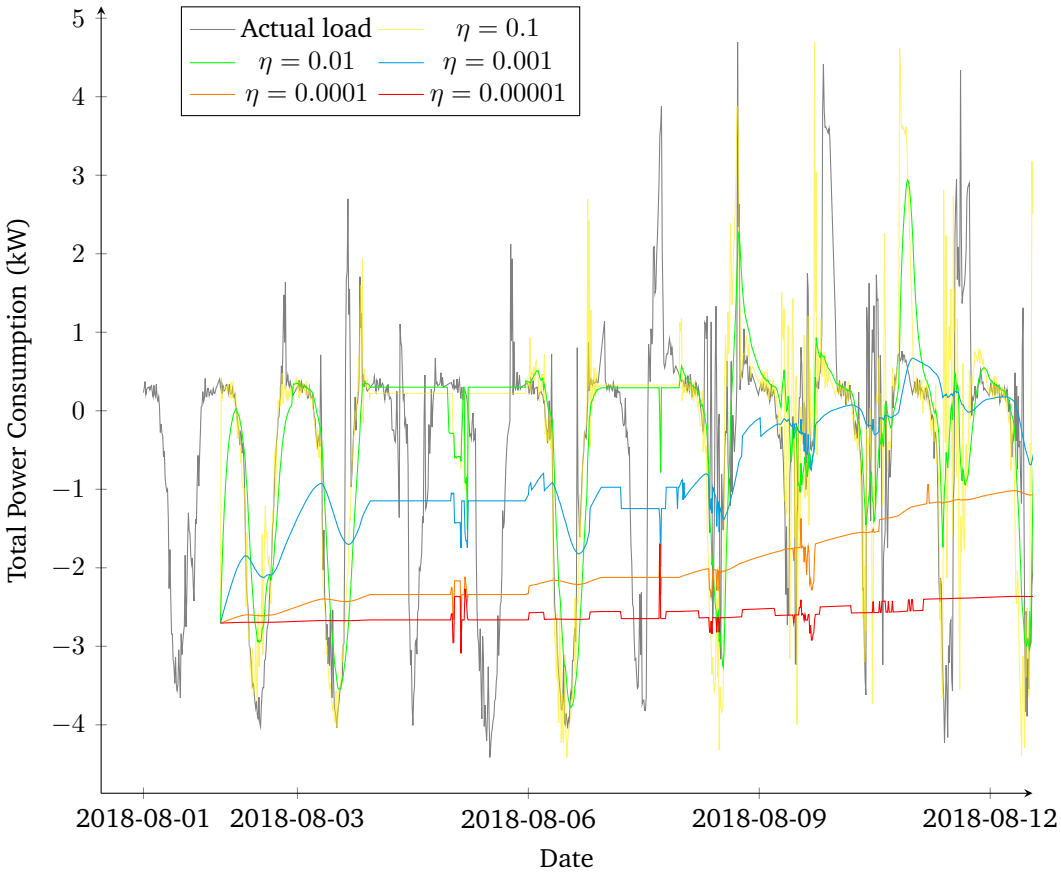


Figure 5.8: Start up effects for predictions with different learning rates (House 29)

are barely notable. When choosing a relatively small learning rate ($\eta = 0.00001$), the forecasting model requires a lot of time to adjust its predictions to the actual load and converges very slowly towards a reasonable accuracy. The startup effects span a period of approximately half a year in this case (this is not completely shown in Figure 5.8). Purely based on this analysis, a large learning rate seems more beneficial as it decreases the startup adjustment time.

However, the learning rate also has an influence on the sensitivity of the forecasting model towards feedback. The sensitivity of the forecasting model towards feedback differs per learning rate. As explained above and in Section 2.4.2, the learning rate has an effect on the training behaviour of the ANN underlying the forecasting model and on the convergence (startup effects) of the forecasting model. The learning rate also has an impact on the sensitivity of feedback. Recall that a large learning rate implies taking large steps towards a minimum of the cost function. Making an erroneous forecast thus results in a large cost. In order to improve, the ANN needs to change its parameters to make better predictions in the future. With a large learning rate, the large cost in combination with the large step size, results in a big change of the parameters of the ANN. In other words, the ANN changes adapts its future

predictions more based on the most recent single error. Thus, the model is more sensitive towards errors or feedback.

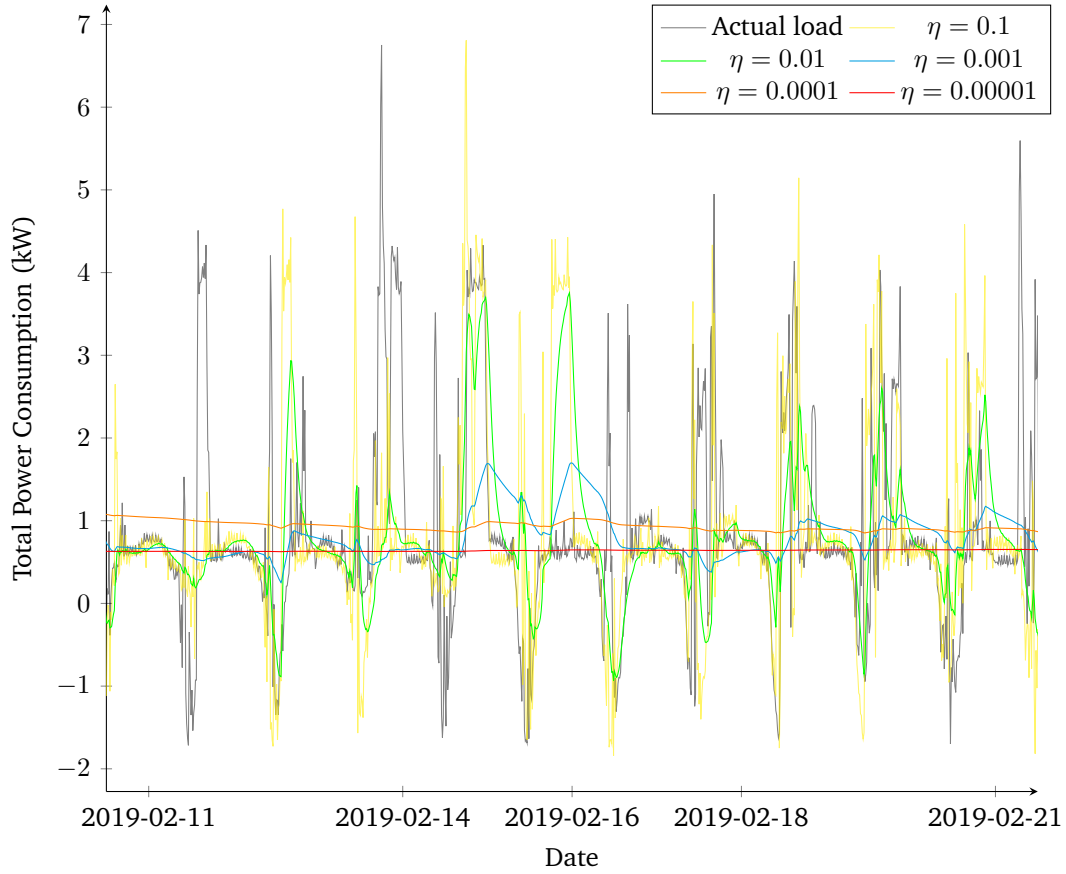


Figure 5.9: Feedback sensitivity for predictions with different learning rates (House 29)

This is shown in Figure 5.9, where an arbitrary period of ten days is shown (between 2019-02-11 and 2019-02-21). The large learning rate ($\eta = 0.1$) in Figure 5.9 shows that the prediction pattern resembles the actual load pattern almost exactly, but shifted a day. This can be explained by the error that the model made a day ago for which it tries to (over-)compensate the next day. The other extreme is when a really small learning rate is chosen ($\eta = 0.00001$). In this case, the forecasting model is less sensitive towards errors and therefore does not adapt its predictions a lot. The result is a smooth line that seems to represent the average over all the load values in a very large period. Choosing a learning rate somewhat in between these two values (e.g. $\eta = 0.001$) the predictions to be very similar across different days even though the days are very different from each other. The challenge is to find the correct balance between a forecasting model that does not overcompensate for individual errors (an individual observation may be an outlier, see Section 4.3.5) but also not completely disregards errors. In other words, the challenge is to find the appropriate sensitivity to feedback, or, the appropriate learning rate.

Different from the traditional gradient descent optimization algorithm, the ANN(s) used by the proposed forecasting model are trained by the adaptive moment estimation optimization algorithm (Adam) [78], [79]. This optimizer is a combination of root mean squared propagation (RMSProp) [80] with momentum [52], [81], but is ultimately still based on the idea of gradient descent. When training an ANN using standard gradient descent at time $t = T$, each parameter θ is updated using the function

$$\theta_T \leftarrow \theta_{T-1} - \eta \nabla C_T. \tag{5.1}$$

When training an ANN using Adam at time $t = T$, the weights and biases of the ANN, represented by θ, are updated using the functions in Equations (5.2), (5.3), and (5.4). Both Equation (5.2) and Equation (5.3) are forms of an exponential moving average of earlier gradients, resulting in a training that takes into account previous errors.

$$\sum_{t=0}^{T} \nabla C_t = \mu \sum_{t=0}^{T-1} \nabla C_t + (1-\mu)\nabla C_T \tag{5.2}$$

$$\sum_{t=0}^{T} \nabla C_t^2 = \beta \sum_{t=0}^{T-1} \nabla C_t^2 + (1-\beta)\nabla C_T^2 \tag{5.3}$$

$$\theta_T \leftarrow \theta_{T-1} - \eta \frac{\sum_{t=0}^{T} \nabla C_t}{\sqrt{\sum_{t=0}^{T} \nabla C_t^2}} \tag{5.4}$$

Adam is generally preferred over other algorithms (e.g. gradient descent (GD), GD with momentum, NAG, AdaGrad, RMSProp) [78]–[81]. This optimizer introduces two new hyper parameters: one for momentum (μ) and one for RMSprop (β).

The sigmoid function is used as activation function in all layers except for the last (output) layer. The output layer has a linear activation, because the output should be real value (even though the household electricity load typically ranges between +3kW and -3kW, it could be any real value) and not a value between 0 and 1 (which is the output of a sigmoid function).

Next to the optimizer and activation function, the weights of the connections are not initialized as random variables from a standard normal distribution, but in a way to prevent neuron saturation [52]. This is done by drawing the weights from a normal distribution with a standard deviation of $\frac{1}{\sqrt{n}}$, where n equals the number of neurons in the layer prior to the connection. This especially helps at dealing with startup-effects and increases the learning speed. Since the forecasting model is trained continuously, weights may become very large after a certain amount of time. Generally, large weights are assumed to cause the function learned by a model to become too complex. Hence, a regularization technique, called L2 regularization,

is used. When using L2 regularization, a regularization term that penalizes large weights is added to the cost function, see Equation (5.5).

$$C = C_{original} + \frac{\lambda}{2}\sum_{w} w^2 \tag{5.5}$$

$$w \leftarrow w - \eta\frac{\partial C}{\partial w} = (1 - \eta\lambda)w - \eta\frac{\partial C_{original}}{\partial w} \tag{5.6}$$

The result of L2 regularization is shown in the update rule in Equation (5.6), which is essentially rescaling a weight before updating it. Therefore L2 regularization is also called weight decay. It introduces a compromise between minimizing the original cost function and finding small weights [52] (note that only weights and no biases are regularized, hence the notation of w is used instead of θ). The compromise depends on the regularization parameter λ. Weight decay is a regularization technique that could reduce overfitting and is especially useful in the proposed forecasting model that trains continuously (indefinitely), because in this case weights could saturate or attain very large values.

Hyperparameters such as the learning rate (η), regularization parameter (λ), optimization parameters (μ and β), the number of hidden layers in the ANN, and the number of neurons in each hidden layer can be determined by empirical experimentation (see Section 2.4.2.2). After testing ANN architectures with different numbers of hidden layers and hidden neurons, ANN architectures with one or two hidden layers and a small but decreasing (per layer) number of hidden neurons show the lowest error. To select the other hyperparameters, three grid searches were done evaluating different combinations of discrete sets of values for each hyperparameter, shown in Table 5.1. The hyperparameters can range between the following values: $\eta \in \mathbb{R}$ (but typically $\eta \leq 1$), $\lambda \in \mathbb{R}$, and $0 \leq \mu, \beta \leq 1$. The discrete set of values chosen for each parameter in the first grid search were chosen with these constraints in mind. In the second and third grid search, the values were chosen based on the values with best performance in the previous grid search. Based on the grid searches,

Hyperparameter	**grid search** 1	**grid search** 2	**grid search** 3
η	$[0.000001, 0.0001, 0.01, 1]$	$[0.0001, 0.001, 0.01]$	$[0.005, 0.009, 0.01, 0.05]$
λ	$[0.001, 0.1, 10, 1000]$	$[0.001, 0.01, 0.1, 10, 100]$	$[5, 10, 50, 100]$
μ	$[0, 0.1, 0.5, 0.9, 1]$	$[0, 0.1, 0.5, 0.9]$	$[0.3, 0.5, 0.8, 0.9]$
β	$[0, 0.1, 0.5, 0.9, 1]$	$[0.1, 0.5, 0.9]$	$[0.5, 0.7, 0.9, 0.95]$

Table 5.1: Discrete set of hyperparameter values that were evaluated exhaustively in each of the three grid searches

a combination of hyperparameters was selected for each architecture discussed in Section 5.3. For the forecasting model using the iterative and multivariate architecture, the best performing combination is $\eta = 0.005$, $\lambda = 50$, $\mu = 0.5$, $\beta = 0.7$. For the forecasting model with the hybrid architecture, the best performing combination

is $\eta = 0.005$, $\lambda = 100$, $\mu = 0.5$, $\beta = 0.7$ and for the forecasting model with the multimodel architecture the best performing combination is $\eta = 0.01$, $\lambda = 10$, $\mu = 0.9$, $\beta = 0.9$.

5.5 Predictions For Profile Steering

This section explains how the predictions of the proposed forecasting model are used to dynamically update the input of a specific DSM technique: profile steering [33] (discussed in Section 3.8). Section 5.5.1 discusses a theoretical neighbourhood scenario with a single large neighbourhood battery to evaluate the impact of the predictions on the effectiveness of the profile steering algorithm [33]. Section 5.5.2 discusses a more realistic neighbourhood scenario that can be simulated to evaluate how the predictions of the proposed forecasting model impact the performance of the asynchronous event-driven profile steering algorithm [25] when multiple EVs start charging (e.g. arrive at home) at random times during the day.

5.5.1 Battery Scenario

This section explains a theoretical scenario that can be used to evaluate the impact of using predictions of the proposed forecasting model as input for the profile steering algorithm [33]. As Gerards et al. [33] explain, the profile steering algorithm can be used on multiple levels in the grid hierarchy. For example, $\vec{x}_m$ (using notation of Section 3.8.1) can represent the load profile of a device m that is used within a household. However, this scenario considers a "neighbourhood" that consists of all 76 houses from the Grid Flex data set [10]. Therefore, $\vec{x}_m$ represents the base load of a house m that is located in this neighbourhood. A house base load profile $\vec{x}_m$ is not known beforehand and must be predicted. The proposed forecasting model could be used to predict this profile. For a single household m, the prediction of the base load profile $\vec{x}_m$ at time t is $\vec{p}_t$ (from Section 5.2), assuming that both have the same number of intervals, e.g. 96 15-minute intervals. To use predictions of the load of all houses, all houses have a local controller with its own copy of the proposed forecasting model. After some time, each copy of the forecasting model has trained and adapted to the load of its corresponding household. At some time step t, the forecasting model of every household makes a prediction $\vec{p}_t$ (unique to every household). For each house $m \in \{1, ..., M\}$, the prediction of its forecasting model is denoted by $\vec{x}_m$ and communicated to a central controller. Then, the central controller knows the aggregated predicted base load profile of all houses, denoted by $\vec{x} = \sum_{m=1}^{M} \vec{x}_m$.

To prevent ambiguity about the results of the profile steering algorithm, a simple scenario is considered. Analyzing a simple scenario gives a better idea of the interaction between the predictions of the forecasting models and the optimization by the profile steering algorithm. This simple scenario includes a single neighbourhood battery that can provide flexibility on the local grid of the neighbourhood. When the aggregated predicted load of all houses in the neighbourhood forms a large peak, the battery could charge to shave this peak at neighbourhood level. When the aggregated predicted load of all houses in the neighbourhood forms a deep valley, the battery could discharge to fill the valley at neighbourhood level. The load profile of this battery is represented by $\vec{x}_B$. The total aggregated neighbourhood profile (all houses plus battery) is then $\sum_{m=1}^{M} \vec{x}_m + \vec{x}_B$ or $\vec{x} + \vec{x}_B$. When the capacity of the battery is chosen too small, there is not enough flexibility to shave all peaks and fill all valleys. It is then harder to evaluate the impact of predictions on the performance of the profile steering algorithm, because part of a possible undesired outcome of the profile steering algorithm can be attributed to flexibility constraints of the battery. To evaluate the impact of purely the predictions on the performance of the profile steering algorithm, the battery capacity is made ideal by making it extremely (unrealistically) large. To concretely implement this, we set the capacity to 1000kWh and use a maximal (dis)charge power of 500kW. The initial state of charge (SoC) of the battery equals 50%. The resulting battery optimization problem is $\min_{\vec{x}_B} \sqrt{\sum_{i=1}^{96} x_i + x_{B,i}}$ subject to $-500 \leq x_{B,i} \leq 500, i = 1, ..., 96$.

At the first time step $t = k$, each copy of the forecasting model makes a prediction ($\vec{p}_k = [p_{k,1}, ..., p_{k,96}]$) of the load $\vec{x}_m = [x_{m,1}, ..., x_{m,96}]$ of its corresponding household m for $m \in \{1, ..., M\}$. After solving the battery optimization problem at this time step using the profile steering algorithm, an optimal battery profile $\vec{x}_B$ is found, given the predictions of the house loads. Then, as time progresses to time step $t = k + 1$, the SoC of the battery is updated because the first interval of $\vec{x}_B$ at time step $t = k$ is now in the past and cannot be changed anymore. In other words, the first interval of the scheduled battery load profile at the previous time step has been executed. At time step $t = k + 1$, we arrive at a new situation. There are now two options:

- The first option is that the predictions made in the beginning ($\vec{p}_k$) are kept fixed for the rest of the simulation. In this case, $[p_{k,2}, ..., p_{k,96}, p_{k+1,96}]$ is assigned to $\vec{x}_m$ for house m at $t = k+1$. This means that the initial prediction is kept fixed and will not be updated, but the last component ($p_{k+1,96}$) of $\vec{p}_{k+1}$ is appended to get a total of 96 intervals again.
- The second option is that predictions of the house base loads are updated. In this case, a new prediction $\vec{p}_{k+1} = [p_{k+1,1}, ..., p_{k+1,96}]$ is assigned to $\vec{x}_m$ for house m at $t = k + 1$.

Regardless of the chosen option at time step $t = k + 1$, the profile steering algorithm is executed again based on the house base loads $\vec{x}_m$ for $m \in \{1, ..., M\}$ to find the optimal battery profile $\vec{x}_B$. Time progresses to time step $t = k + 2$ and the SoC of the battery is updated again. This process is an iterative process that runs until some final time step $t = T$.

5.5.2 Realistic Neighbourhood Scenario

In contrast to Section 5.5.1, this section discusses a more complicated scenario of a neighbourhood that can be simulated to evaluate how the predictions of the proposed forecasting model impact the performance of the profile steering algorithm in a more realistic scenario. In this scenario, the neighbourhood again consists of all the 76 houses from the Grid Flex data set [10], but this time the scenario assumes that (instead of a single large neighbourhood battery) 50% of the households own an EV. This means that together, 38 EVs can provide flexibility to fill valleys of the aggregated house base load. The minimum charging power of each EV is 0kW (no V2G, see Section 3.2.2) and the maximum charging power is 7kW (see Section 3.2.1). In reality, residents have EVs with different capacities, so this scenario models 15 EVs with a battery capacity of 20kWh, 10 EVs with a capacity of 40kWh, 10 EVs with a capacity of 50kWh, and 3 EVs with a capacity of 70kWh. The initial SoC upon arrival is random but always less than 25%. The target capacity (e.g. set by the resident for leaving) is 80% for every EV. In practice, the battery of an EV (opposed to a static battery) is not always available (connected to the home charging point). Therefore, the 38 EVs in this scenario start charging (e.g. arrive at home) at random times during the day. To model this scenario, the asynchronous event-driven profile steering algorithm by Hoogsteen et al. [25] is used to optimize the total (house base loads plus EV loads) aggregated load profile, because it uses an event-based approach to handle the arrival of EVs at the home charging points (see Section 3.8.2).

In the asynchronous event-driven profile steering algorithm by Hoogsteen et al. [25], predictions are committed in the first phase. For example, $\vec{r}_m$ (using notation of Section 3.8.2) is a prediction of a house baseload profile done by a local house controller in the first phase. When the first phase finishes, $\vec{r}_m$ is stored at the global controller. However, predictions can be inaccurate. As time progresses, the local house controller may predict a new house baseload profile $\vec{\hat{r}}_m$ that deviates from the original $\vec{r}_m$ (e.g. more than a certain threshold). Similar to uncertain devices, the predicted house baseload profile $\vec{r}_m$ could be updated in an event-based manner. The new house baseload profile $\vec{\hat{r}}_m$ is sent towards the global controller, which subsequently updates its *committed profile* $\vec{r}$. In this way, more accurate predictions can be incorporated when realizing the *planned profile* $\vec{q}$. Uncertain devices (such as EVs) that arrive after this, will then receive a more accurate *desired profile*, and

subsequently optimize their committed profile accordingly. This system fits well with the forecasting model that is developed in this thesis, as the forecasting model updates its predictions continuously.

The main differences of the scenario described in this section compared to the scenario described in Section 5.5.1, are that this scenario uses the asynchronous event-driven profile steering algorithm by Hoogsteen et al. [25] with 38 EVs that provide flexibility, instead of the original profile steering algorithm by Gerards et al. [33] with a single large battery that provides flexibility. However, there are also many similarities. Similar to the scenario described in Section 5.5.1, the scenario in this section also uses the predictions of the forecasting models corresponding to each household. Also, again two options are available: only using the initial prediction or updating predictions each time step. Furthermore, the SoC of all EVs are updated after each time step similar to how the SoC of the battery in Section 5.5.1 is updated after each time step.

6 Results and Discussion

Chapter Objective: This chapter discusses the results of applying the proposed forecasting model.

Chapter Contents

6.1 Introduction

This chapter discusses the results of applying the proposed forecasting model on the electric load series of the houses from the GridFlex data set [10]. Section 6.2 discusses the performance of the of the proposed forecasting model (proposed in Section 1.2) using the different architectures (explained in Section 5.3). The performance of the different architectures used for the proposed forecasting model is evaluated in several ways. First, Section 6.2.1 discusses the performance of the proposed forecasting model for different houses in the GridFlex data set [10]. Then, Section 6.2.2 discusses the performance of the proposed forecasting model over time. Section 6.2.3 discusses whether the performance is different for different components of $\vec{p}_t$. In other words, this section discusses the influence of the forecasting horizon on the performance of the proposed forecasting model. Finally, Section 6.2.4 evaluates whether the aggregation of the forecasts for all the houses (of the GridFlex data set [10]) matches the aggregation of all the actual loads of all the houses. Section 6.3 evaluates two scenarios where the performance of the profile steering algorithm [33] is evaluated, when predictions of the proposed forecasting model are used. Section 6.3.1 discusses the scenario described in Section 5.5.1 where a single large battery is scheduled by the profile steering algorithm [33]. Section 6.3.2 discusses the scenario described in Section 5.5.2 where 38 EVs are scheduled by the asynchronous event-driven profile steering algorithm [25]. In both scenarios, an analysis is

provided that compares the performance when predictions are updated versus when predictions are not updated.

6.2 Forecasting Performance

6.2.1 Comparison Of ANN Architectures For Different Households

To see if the proposed forecasting model is equally effective for different households, this section compares the performance of the proposed forecasting model for each house of the GridFlex data set [10]. Table 6.1. shows the results of using different architectures (explained in Section 5.3) of the proposed forecasting model for all the houses in the GridFlex data set [10] for the second month (September 2018). Because running simulations for the complete time frame of over two years for all houses (see Section 4.2) and all forecasting models (from Section 5.3) would take too much time, Table 6.1 only shows the results for the second month. Furthermore, the predictions of the first month would not be accurate as the forecasting models suffer from startup effects and need some time to adjust their predictions to the load of a household, as explained in Section 5.4.2. Showing the results of the second month only requires a simulation of two months for each household and each forecasting model architecture, and can be completed relatively fast. First, Table 6.1 lists all households of the GridFlex data set [10] in the first column and the standard deviation of their respective load profiles (Load St. Dev.) in the second column. Furthermore, Table 6.1 displays statistics about the absolute error between the predicted load value and actual load value, for each household and for each forecasting model architecture discussed in Section 5.3. Additionally, Table 6.1 shows the results of naive predictions. The naive predictions are equal to the load at the last time (15 minute) interval. The load at the last interval is typically close to the load at the current time interval and therefore provides a good benchmark. In the remainder of this chapter, the naive predictions are sometimes used for comparison. The statistics in Table 6.1 include the mean (μ), median (m), and standard deviation (σ) of the absolute prediction error. The mean absolute prediction error (μ) gives an indication of the performance of the forecasting model architecture on average. The median absolute prediction error (m) gives the most occurring absolute prediction error. Lastly, the standard deviation of the absolute prediction error (σ) represents the stability of the forecasting model, similar to the stability described by Javed et al. [28] (see Section 2.5).

In Table 6.1, all forecasting model architectures have the lowest prediction error (μ, m, and σ) when applied to House 58, and the highest prediction error (μ, m, and σ)

House	Load St. Dev.	Multivariate			Iterative			Multi-model			Hybrid			Naive		
		μ	m	σ	μ	m	σ	μ	m	σ	μ	m	σ	μ	m	σ
1	0.841	0.511	0.715	0.456	0.422	0.913	0.452	0.39	0.366	0.217	0.534	0.175	0.164	0.552	0.15	0.165
2	0.341	0.221	0.257	0.265	0.217	0.283	0.269	0.226	0.22	0.11	0.146	0.097	0.086	0.088	0.078	0.093
3	0.465	0.232	0.321	0.252	0.22	0.407	0.237	0.199	0.187	0.142	0.206	0.128	0.123	0.25	0.12	0.114
4	1.032	0.672	0.913	0.541	0.496	1.093	0.491	0.464	0.44	0.202	0.744	0.144	0.137	0.64	0.112	0.143
5	0.677	0.319	0.437	0.35	0.306	0.609	0.356	0.304	0.282	0.158	0.276	0.139	0.127	0.368	0.118	0.13
6	2.454	1.755	2.044	1.185	1.18	2.747	1.262	1.17	1.14	0.535	1.356	0.491	0.445	1.918	0.426	0.687
7	0.815	0.164	0.187	0.198	0.168	0.201	0.182	0.161	0.15	0.095	0.112	0.082	0.072	0.07	0.062	0.063
8	0.619	0.343	0.369	0.418	0.363	0.458	0.437	0.357	0.337	0.205	0.256	0.224	0.209	0.246	0.21	0.223
9	0.749	0.39	0.531	0.407	0.374	0.711	0.418	0.357	0.329	0.188	0.35	0.159	0.157	0.472	0.136	0.181
10	0.505	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
11	0.614	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
12	0.472	0.194	0.219	0.229	0.201	0.248	0.218	0.17	0.159	0.123	0.153	0.113	0.102	0.12	0.092	0.085
13	0.366	0.222	0.238	0.25	0.211	0.314	0.235	0.197	0.175	0.128	0.153	0.12	0.107	0.188	0.102	0.098
14	0.758	0.468	0.659	0.391	0.371	0.828	0.379	0.329	0.313	0.207	0.513	0.156	0.144	0.516	0.126	0.122
15	0.389	0.181	0.28	0.196	0.164	0.31	0.177	0.156	0.146	0.102	0.172	0.088	0.075	0.172	0.064	0.072
16	1.526	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
17	2.191	0.43	0.525	0.505	0.433	0.612	0.632	0.483	0.471	0.192	0.335	0.213	0.184	0.22	0.206	0.239
18	0.408	0.208	0.229	0.236	0.199	0.297	0.221	0.179	0.164	0.119	0.134	0.089	0.075	0.172	0.06	0.07
19	0.477	0.258	0.295	0.294	0.252	0.392	0.283	0.234	0.216	0.13	0.173	0.11	0.099	0.21	0.088	0.083
20	0.741	0.429	0.596	0.387	0.371	0.755	0.385	0.329	0.313	0.183	0.456	0.148	0.141	0.444	0.128	0.111
21	3.839	2.694	3.093	1.61	2.035	3.945	1.636	1.544	1.533	0.896	2.329	0.926	0.855	2.642	0.852	0.86
22	2.428	0.684	0.817	0.782	0.698	0.972	0.928	0.759	0.719	0.312	0.533	0.392	0.33	0.426	0.39	0.458
23	1.221	0.806	1.045	0.704	0.661	1.347	0.726	0.61	0.59	0.335	0.74	0.336	0.309	0.876	0.308	0.264
24	1.103	0.713	0.929	0.568	0.535	1.17	0.56	0.482	0.458	0.259	0.708	0.208	0.198	0.76	0.182	0.184
25	0.973	0.565	0.753	0.479	0.427	0.936	0.502	0.403	0.376	0.264	0.56	0.212	0.193	0.694	0.198	0.215
26	0.344	0.158	0.178	0.183	0.149	0.178	0.168	0.146	0.141	0.097	0.112	0.08	0.07	0.068	0.062	0.069
27	0.445	0.267	0.284	0.294	0.266	0.347	0.286	0.22	0.206	0.175	0.22	0.164	0.158	0.242	0.148	0.141
28	1.369	0.854	1.151	0.652	0.607	1.398	0.66	0.575	0.554	0.293	0.892	0.252	0.253	0.902	0.218	0.274
29	1.57	0.735	0.852	0.778	0.694	1.184	0.926	0.734	0.677	0.252	0.489	0.255	0.217	0.718	0.236	0.317
30	0.403	0.242	0.276	0.285	0.252	0.312	0.28	0.228	0.216	0.146	0.178	0.137	0.131	0.146	0.122	0.123
31	0.374	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
32	0.873	0.472	0.637	0.433	0.426	0.823	0.441	0.348	0.324	0.198	0.448	0.164	0.151	0.536	0.136	0.134
33	0.483	0.298	0.439	0.305	0.283	0.52	0.293	0.255	0.24	0.148	0.286	0.137	0.13	0.308	0.12	0.12
34	0.337	0.198	0.199	0.221	0.19	0.213	0.205	0.171	0.159	0.13	0.134	0.106	0.096	0.094	0.088	0.087
35	0.986	0.582	0.802	0.471	0.447	0.996	0.454	0.41	0.394	0.196	0.642	0.153	0.149	0.623	0.128	0.149
36	0.426	0.253	0.286	0.315	0.28	0.365	0.304	0.234	0.228	0.167	0.223	0.177	0.169	0.212	0.164	0.149
37	0.377	0.199	0.202	0.23	0.192	0.24	0.228	0.179	0.162	0.114	0.132	0.102	0.093	0.102	0.086	0.087
38	0.569	0.366	0.418	0.377	0.347	0.567	0.362	0.292	0.276	0.175	0.268	0.172	0.16	0.342	0.152	0.127
39	0.602	0.23	0.281	0.265	0.237	0.335	0.252	0.215	0.213	0.138	0.199	0.127	0.12	0.204	0.108	0.126
40	0.742	0.476	0.636	0.386	0.371	0.773	0.367	0.326	0.316	0.183	0.47	0.138	0.135	0.426	0.116	0.118
41	0.383	0.187	0.218	0.231	0.191	0.232	0.217	0.174	0.166	0.118	0.132	0.104	0.093	0.098	0.084	0.087
42	0.22	0.157	0.117	0.128	0.105	0.12	0.107	0.091	0.084	0.082	0.079	0.066	0.053	0.05	0.046	0.041
43	1.331	0.965	1.198	0.77	0.749	1.575	0.766	0.722	0.712	0.519	0.802	0.391	0.418	1.238	0.344	0.462
44	0.399	0.217	0.228	0.25	0.218	0.265	0.231	0.202	0.187	0.128	0.149	0.099	0.092	0.124	0.072	0.106
45	0.654	0.339	0.469	0.33	0.306	0.612	0.319	0.279	0.263	0.174	0.335	0.142	0.138	0.382	0.13	0.129
46	0.605	0.342	0.455	0.391	0.347	0.621	0.387	0.327	0.304	0.157	0.277	0.161	0.154	0.404	0.138	0.179
47	0.529	0.216	0.245	0.238	0.205	0.236	0.224	0.193	0.202	0.146	0.139	0.126	0.118	0.124	0.112	0.117
48	0.702	0.323	0.4	0.367	0.323	0.539	0.386	0.315	0.283	0.166	0.235	0.151	0.139	0.332	0.132	0.168
49	1.21	0.577	0.789	0.444	0.415	0.983	0.432	0.37	0.367	0.204	0.573	0.17	0.163	0.524	0.154	0.149
50	0.448	0.245	0.308	0.264	0.231	0.412	0.252	0.208	0.188	0.128	0.189	0.098	0.088	0.206	0.076	0.077
51	0.52	0.291	0.42	0.299	0.274	0.532	0.284	0.249	0.237	0.158	0.289	0.12	0.111	0.288	0.098	0.099
52	0.273	0.132	0.136	0.149	0.121	0.136	0.128	0.105	0.096	0.081	0.091	0.063	0.051	0.048	0.04	0.04
53	1.12	0.352	0.415	0.449	0.399	0.539	0.48	0.39	0.349	0.199	0.293	0.244	0.229	0.304	0.234	0.249
54	0.323	0.178	0.21	0.204	0.174	0.242	0.188	0.145	0.141	0.115	0.152	0.103	0.092	0.128	0.084	0.074
55	0.649	0.197	0.215	0.232	0.202	0.26	0.218	0.167	0.163	0.13	0.157	0.134	0.125	0.146	0.12	0.099
56	0.911	0.515	0.68	0.447	0.421	0.876	0.427	0.379	0.361	0.184	0.459	0.153	0.144	0.492	0.126	0.138
57	0.982	0.698	0.912	0.54	0.503	1.137	0.548	0.466	0.446	0.228	0.715	0.197	0.186	0.674	0.168	0.181
58	0.13	0.077	0.077	0.074	0.06	0.054	0.05	0.043	0.046	0.069	0.068	0.056	0.048	0.044	0.042	0.036
59	0.464	0.16	0.177	0.177	0.15	0.2	0.165	0.14	0.128	0.101	0.122	0.083	0.075	0.088	0.068	0.063
60	1.022	0.685	0.861	0.561	0.539	1.109	0.562	0.502	0.458	0.278	0.635	0.254	0.261	0.792	0.24	0.262
61	0.502	0.216	0.233	0.263	0.23	0.279	0.25	0.202	0.186	0.121	0.157	0.11	0.099	0.104	0.086	0.095
62	0.356	0.188	0.231	0.233	0.187	0.253	0.227	0.183	0.163	0.104	0.153	0.088	0.074	0.096	0.058	0.076
63	0.375	0.208	0.21	0.225	0.184	0.241	0.222	0.182	0.168	0.107	0.127	0.091	0.08	0.088	0.072	0.07
64	0.844	0.516	0.671	0.448	0.423	0.9	0.436	0.373	0.353	0.276	0.489	0.229	0.221	0.63	0.212	0.204
65	0.5	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
66	0.304	0.172	0.161	0.194	0.163	0.189	0.179	0.135	0.126	0.121	0.111	0.102	0.093	0.1	0.086	0.078
68	0.417	0.134	0.14	0.14	0.113	0.124	0.116	0.097	0.094	0.09	0.098	0.07	0.055	0.048	0.044	0.049
69	0.62	0.364	0.439	0.378	0.346	0.616	0.375	0.303	0.301	0.21	0.27	0.197	0.182	0.442	0.178	0.157
70	0.788	0.445	0.538	0.499	0.459	0.73	0.516	0.418	0.387	0.209	0.333	0.208	0.196	0.536	0.186	0.229
71	0.405	0.195	0.228	0.22	0.192	0.236	0.21	0.17	0.162	0.122	0.139	0.105	0.094	0.098	0.084	0.081
72	0.621	0.344	0.445	0.428	0.372	0.53	0.467	0.39	0.37	0.137	0.302	0.14	0.129	0.196	0.114	0.214
73	0.212	0.115	0.115	0.119	0.099	0.108	0.097	0.079	0.076	0.082	0.081	0.066	0.056	0.052	0.05	0.04
74	0.335	0.206	0.246	0.241	0.215	0.273	0.233	0.202	0.196	0.118	0.165	0.109	0.1	0.128	0.09	0.116
75	0.871	0.484	0.634	0.48	0.439	0.85	0.488	0.401	0.376	0.243	0.414	0.253	0.239	0.556	0.24	0.239
76	0.493	0.293	0.282	0.298	0.258	0.342	0.291	0.237	0.236	0.159	0.205	0.144	0.133	0.178	0.128	0.136
77	0.372	0.157	0.175	0.204	0.169	0.211	0.194	0.154	0.145	0.088	0.114	0.084	0.072	0.078	0.068	0.06
AVG	0.747	0.404	0.498	0.378	0.35	0.625	0.379	0.322	0.306	0.184	0.348	0.167	0.154	0.377	0.145	0.159

Table 6.1: Absolute error statistics for different houses and forecasting model architectures

when applied to House 21. House 58, house 21, and their corresponding prediction errors for all the forecasting model architectures are highlighted in Table 6.1. Note that, unsurprisingly, house 58 has the lowest load standard deviation and house 21 has the highest load standard deviation. There exist normalized error metrics that divide the prediction error by the load standard deviation (see Section 2.5), but this thesis simply uses the absolute prediction error as its value is easier to explain. In general, many houses that have a more volatile load profile (higher load profile standard deviation) also have higher prediction error values. This is in line with the discussions in Section 2.2.2. Figure 6.1 displays scatter plots that indicate the relation between a statistic (μ, m, σ) of the absolute prediction error and the standard deviation of the load of a household. Each dot in the scatter plot represents one household trained with one forecasting model architecture of Table 6.1 (e.g. one dot may represent the standard deviation of the load of house 8, and the error of the multivariate forecasting model). Figure 6.1 also displays trend lines through each

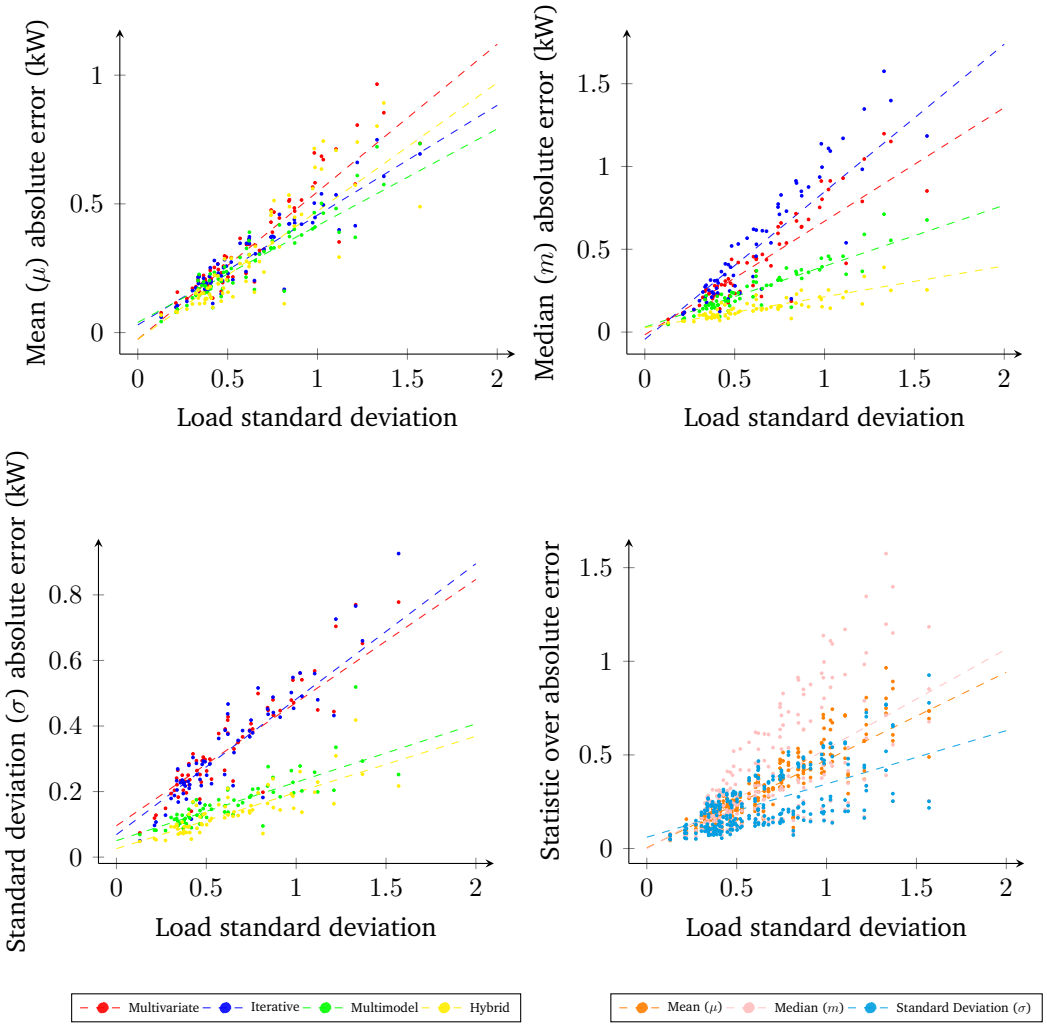


Figure 6.1: Error versus standard deviation of the load profile

of the scatter plots. Although the trend lines are based on all the houses from the GridFlex data set [10], the graphs of Figure 6.1 zoom in on the interval $[0, 2]$ of the domain, as most houses have a load standard deviation within in this interval. The graphs in the upper left corner, upper right corner, and lower left corner show the relationship between μ and the load standard deviation, m and the load standard deviation, and σ and the load standard deviation, respectively, for the different forecasting model architectures. The graphs in Figure 6.1 suggest that there is a linear relationship between the standard deviation of the load of a house and the absolute prediction error of the forecasting model, regardless of the forecasting model architecture that was used. A larger house load standard deviation correlates with a larger prediction error. The graph in the upper left corner of Figure 6.1 shows this as well, but there appears to be no large difference between the forecasting model architectures.

The graph in the upper right corner of Figure 6.1 shows that the strength of the correlation of the median (m) of the absolute prediction error with the house load standard deviation is different for different forecasting model architectures. The median error of the iterative model has the strongest correlation, then the correlation of the median error of the multivariate model, then the correlation of the median error of the multimodel model, and finally the median error of the hybrid model has the weakest correlation. The graph in the lower left corner shows that the standard deviation σ of the prediction error is less affected by a larger load standard deviation, when a mulitmodel or hybrid architecture is used compared to when a multivariate or iterative architecture is used. This indicates that the forecasting model produces more stable predictions with the multimodel and the hybrid architecture compared to with the multivariate and the iterative model. The graph in the lower right corner of Figure 6.1 shows the relationship between the value of a statistic and the standard deviation of a house load. This graphs shows that the stability (standard deviation, σ in Table 6.1) of the prediction error is the least affected by the standard deviation of a house load. The median of the prediction error is the most affected by the standard deviation of a house load.

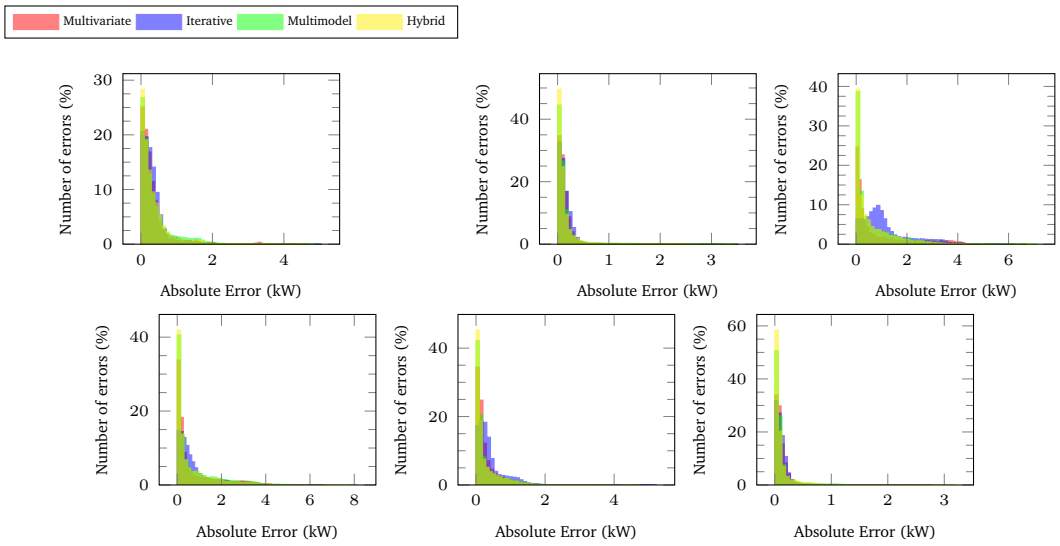


Figure 6.2: Distribution of errors for house 8, 26, 28, 29, 51, 52.

Both the mean (μ) and the median (m) of the absolute error are evaluated in Table 6.1 and Figure 6.1, because they are significantly different for many houses. The reason for this is the large skewness of the distribution of errors. Figure 6.2 shows histograms of the error distribution for different forecasting model architectures. The histograms are based on the errors of house 8, 26, 28, 29, 51, and 52, respectively. All histograms show that the error distribution is skewed to the right. When comparing the different forecasting model architectures, the hybrid and multimodel architectures have a relatively high peak on the left, indicating a larger skewness. The iterative architecture has a relatively low peak on the left, indicating a lower

skewness. Especially for the hybrid and multimodel architecutre, the skewness illustrated by Figure 6.2 indicates that, although most predictions have a very small error, the prediction error is very large for a very small number of predictions. In practice, this means that the median (m) prediction error may represent the real performance of the forecasting model better in most cases compared the mean (μ) prediction error, especially for the hybrid and multimodel architecture.

6.2.2 Performance Convergence

This section discusses how the performance of the forecasting model changes over time. As explained in Section 5.4.2, depending on the value of hyperparameters such as the learning rate (η), the forecasting model needs some time to adjust its predictions to the house load, before reaching a reasonable accuracy. Figure 6.3 shows the mean absolute prediction error of the load of house 28, for different forecasting model architectures. This error is not only shown per 15-minute interval, but is also averaged over larger time periods (per day and per week) to better show the overall trend. For all forecasting model architectures, the error trend decreases during the first 20 days, indicating that all forecasting model architectures suffer from a "start-up" time of approximately 20 days.

Figure 6.4 and Figure 6.5 display the mean absolute error for house 8, 26, 28, 29, 51, and 52, over the complete time period (of two years). The error is shown for different forecasting model architectures and is (again) averaged over two larger time periods (per week and per month) to capture the trend. The plots in these figures give an indication of the performance of the forecasting model over time. In all plots in Figure 6.4 and Figure 6.5, the startup period is not very visible anymore, because the plots display the average per week and per month. Still, these figures show that the error is not stable over time, but differs over the course of the two years. Especially for the houses with solar panels (see house 28, 29, 51 in Table 1.1), the performance differs during different time periods. During late spring and early summer (March, April, May, June), the average error per week and per month is significantly higher than during other seasons. This difference is especially large for the iterative architecture and the multivariate architecture. The error of the multimodel architecture and the hybrid architecture are less affected by this season. For the models without solar panels (see house 8, 26, 52 in Table 1.1), there is much less difference in error between seasons. Also, the architectures differ only slightly in performance over time. Although the difference is minimal, the multivariate architecture performs best and the iterative architecture performs worst for these houses.

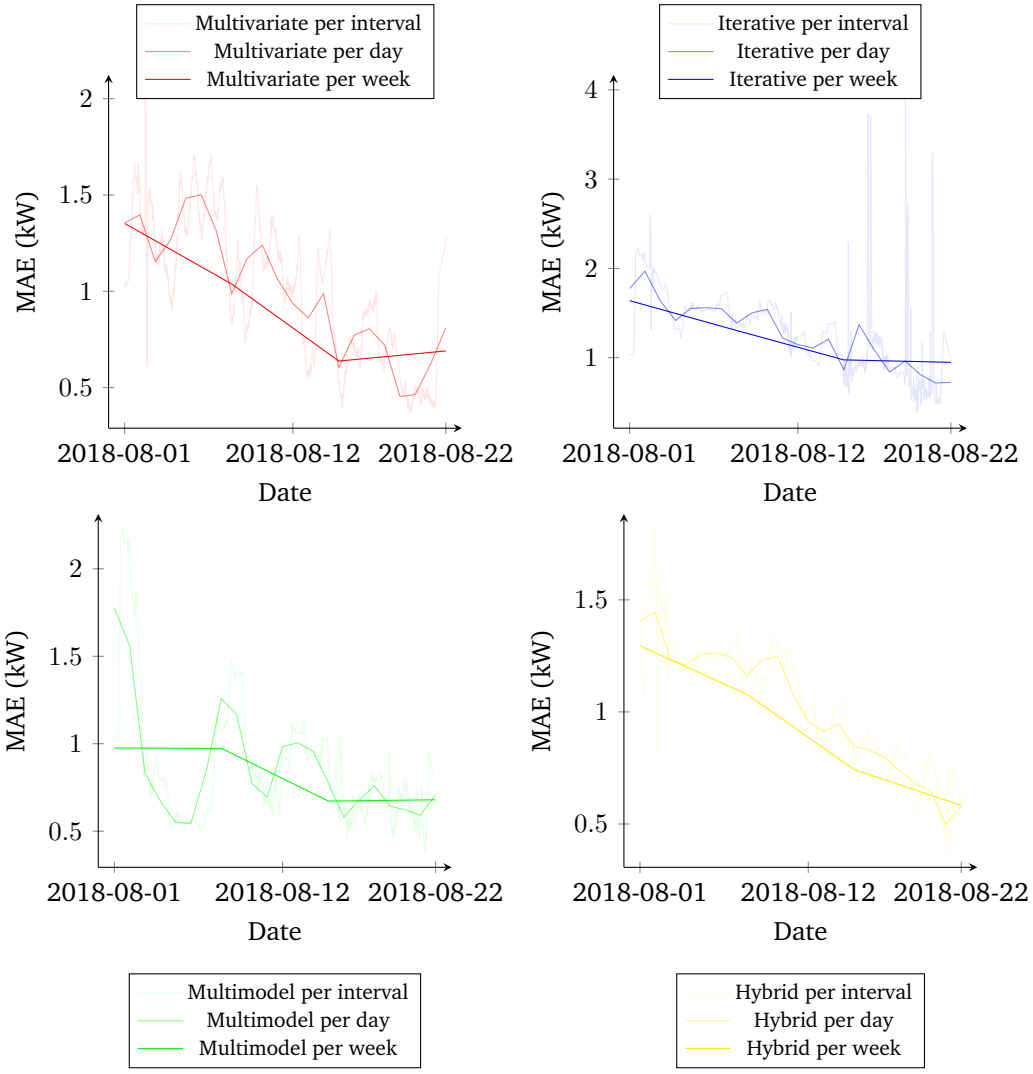


Figure 6.3: Startup effects when applying different architectures of the forecasting model to the load of House 28

6.2.3 Forecasting Horizon

The forecast window $\vec{p_t}$ made at time t contains 96 components $p_{t,i}$ for $i \in \{1, ..., 96\}$. Each of the 96 predicted values corresponds to a 15 minute interval in a day. The absolute error between one component $p_{t,i}$ and the corresponding actual value $d_{t,i}$ can be calculated using $E_{t,i} = |p_{t,i} - d_{t,i}|$ for $i \in \{1, ..., 96\}$ and $t \in \{1, ..., T\}$. Here, T is the last time step of the full two years time frame of the GridFlex data set [10]. This way, each component of the error $\vec{E_t} = [E_{t,1}, ..., E_{t,96}]$ corresponding to the forecast window $\vec{p_t} = [p_{t,1}, ..., p_{t,96}]$ can be calculated for each time t. To see how well the forecasting model can predict the electricity consumption of a household for different forecasting horizons in general, the average of each error component $E_{t,i}$ over all time steps $t \in \{1, ..., T\}$ can be calculated: $\mu_t(E_i) = \frac{1}{T} \sum_{t=1}^{T} |p_{t,i} - d_{t,i}|$. The average forecast window error $\mu_t(\vec{E}) = [\mu_t(E_1), ..., \mu_t(E_{96})]$

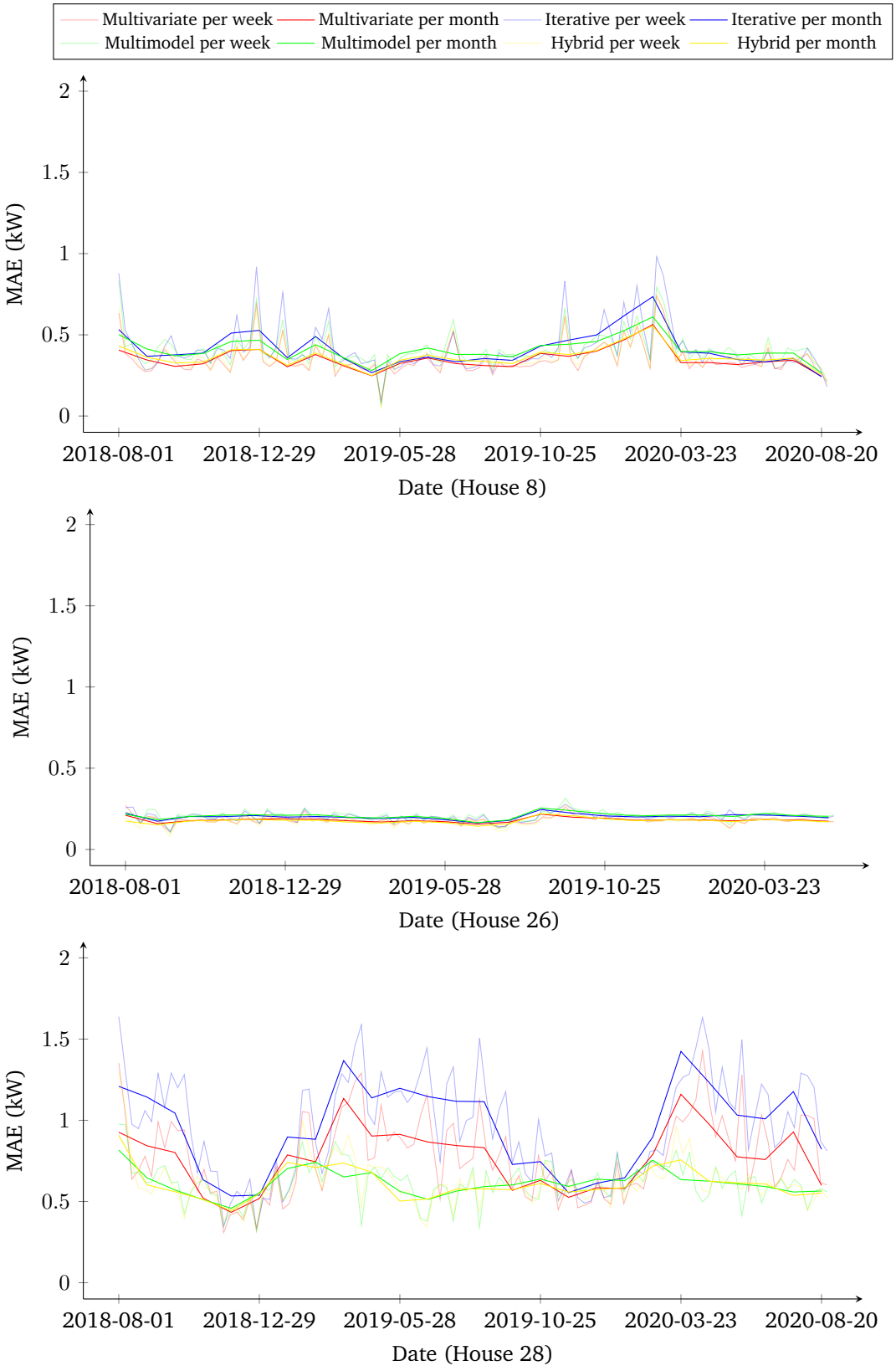


Figure 6.4: Error over time for different forecasting model architectures for house 8, 26, 28

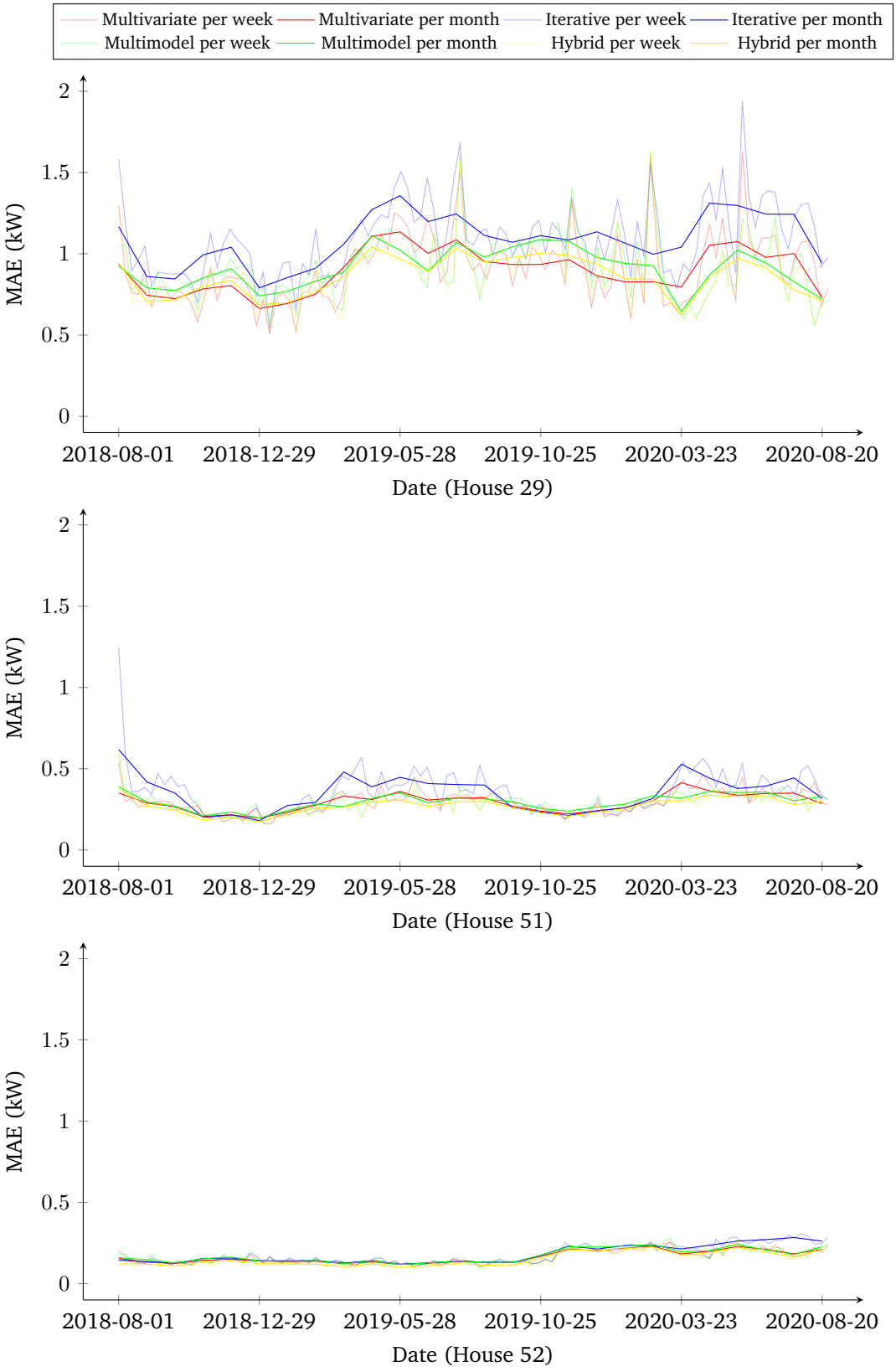
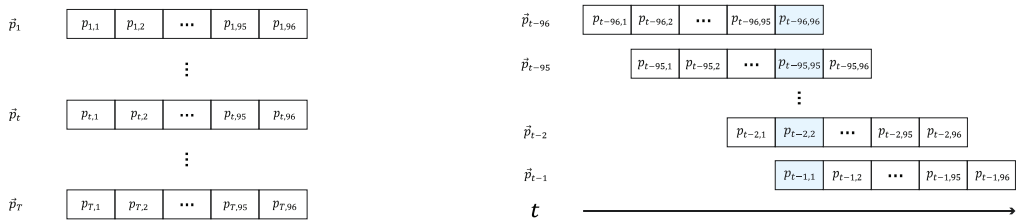


Figure 6.5: Error over time for different forecasting model architectures for house 29, 51, 52

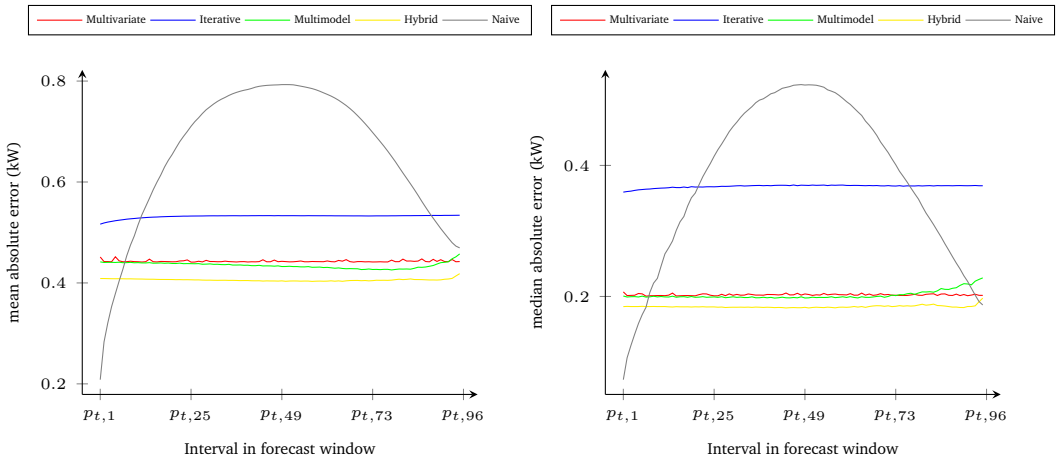
then consists of each of these average error components. Figure 6.6a presents the



(a) Method 1 **(b)** Method 2

Figure 6.6: Aligning forecast windows $\vec{p}_t$ for all $t \in \{1, ..., T\}$

forecasting windows $\vec{p}_t$ for all $t \in \{1, ..., T\}$ on top of each other. By aligning the ith components $p_{t,i}$ of $\vec{p}_t$ for all $t \in \{1, ..., T\}$ vertically, it illustrates that the average error component $\mu_t(E_i)$ is calculated using information from each column. Figure 6.7a



(a) mean **(b)** median

Figure 6.7: Mean (left) and median (right) absolute error for each component of $\vec{p}_t$ (96 in total), averaged over house 8, 26, 28, 29, 51, 52, based on the full two years, for different forecasting model architectures

plots the average forecast window error $\mu_t(\vec{E}) = [\mu_t(E_1), ..., \mu_t(E_{96})]$ averaged over house 8, 26, 28, 29, 51, and 52, for different forecasting model architectures. The x-axis shows the components $p_{t,i}$ within the forecast window corresponding to the mean absolute error (MAE) components $\mu_t(E_i)$, for $i \in \{1, ..., 96\}$. The median forecast window error $m_t(\vec{E}) = [m_t(E_1), ..., m_t(E_{96})]$ is calculated in a similar fashion and plotted in Figure 6.7b. For both Figure 6.7a and Figure 6.7b, the error does not differ significantly for different forecasting horizons (number of intervals in the future) within the forecast window. This applies to all four architectures (multivariate, iterative, multimodel, hybrid). When comparing the four forecasting architectures, the iterative architecture performs worst with an average MAE of approximately 0.5kW (corresponding median of 0.36kW), and shows a slight error increase during the first few intervals of the forecasting window. The average MAE of the multivariate model is relatively stable over time, around

approximately 0.45kW (corresponding median of 0.2kW), similar to the average MAE of the multimodel architecture. The hybrid architecture performs best overall with an average MAE of approximately 0.4kW (corresponding median of 0.18kW). Both the hybrid and multimodel architecture show a slight increase in error during the last few intervals. Next to the four forecasting model architectures, also the results for the naive predictions are shown (introduced in Section 6.2.1). A naive prediction is a constant vector $\vec{p}_t$ Of this constant vector, all components are equal to the last value of the load that was measured by the smart meter of the house. For example, if the household load is 3.2kW at time t, then $\vec{p}_t = [3.2, 3.2, ..., 3.2]$ for the naive predictions. The error curve of the naive model in Figure 6.7 can be explained by the auto-correlation curve (Figure 4.10) of Section 4.4.2) of the household load. The current load has a relatively high correlation with the load on another day at exactly the same time. For example, if $t = 12{:}00$ in the afternoon on the 1st of January, then the future load of the forecasting window that is likely to correlate most, is the load at approximately $t = 12{:}00$ on the 2nd of January. This load corresponds to the last few intervals of the forecasting window. Thus, Figure 6.7 shows that the error in the first and last few intervals is relatively low. Similarly, in the center of the forecast window the error is highest, as the load during that time is most likely to correlate least with the current load. The lowest average MAE for the naive predictions is approximately 0.2kW at $p_{t,1}$. This is lower than any of the forecasting model architectures at $p_{t,1}$. However, at most other intervals in the forecast window, the average MAE is much higher than the average MAE for the other forecasting model architectures. Also, the median error at $p_{t,1}$ for the naive predictions does not differ much from the median error at $p_{t,1}$ for the multivariate and multimodel architecture, and the hybrid architecture has even a lower median error.

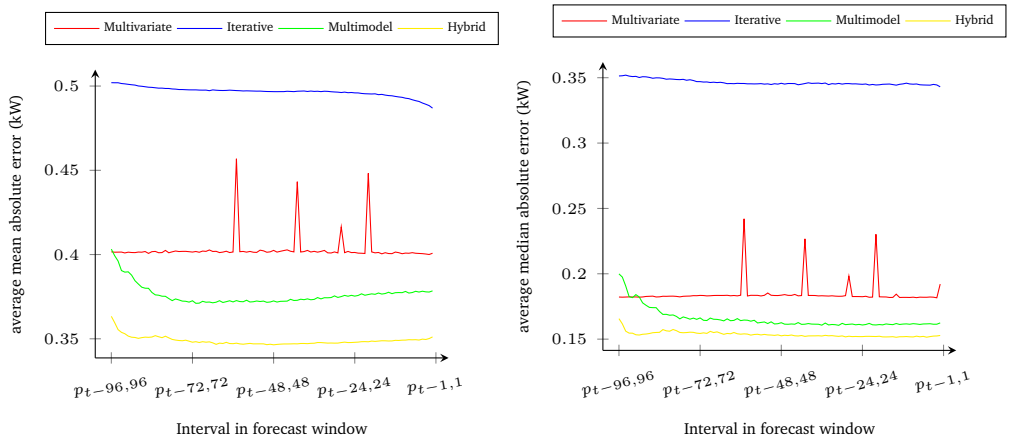


Figure 6.8: The average mean (left) and median (right) error for a prediction of the load at time t across different forecast windows $p_{t-i,i}$ $(i \in \{1, ..., 96\})$

The performance of different forecasting horizons within the forecast window can also be plotted in an alternative way. Since the forecasting model has a sliding

window and a forecasting horizon of 96 time steps, different predictions are made of the load at time t. Figure 6.6b presents the forecasting windows $\vec{p}_k$ for $k \in \{t-96, ..., t-1\}$ on top of each other and aligns them by time, similar to the rolling horizon depicted in Figure 5.2. The components of different forecast windows $\vec{p}_k$ for $k \in \{t-96, ..., t-1\}$ are aligned vertically if they predict the same (actual/real) load. For example, the highlighted components in Figure 6.6b are all predictions of the actual load at time t. Again, the mean and median error corresponding to the i^{th} component of forecast window $\vec{p}_{t-i}$ can be plotted. This is done in Figure 6.8. Here, the results are averaged over all 76 houses of the GridFlex data set [10] for a time period of the first two months. The results in Figure 6.8 are (approximately) a mirrored version of the results in Figure 6.7. The slight differences can be explained by the fact that the results of Figure 6.8 are based on the first two months of data of all 76 houses whereas the result of Figure 6.7 are based on the full two years of data of house 8, 26, 28, 29, 51, and 52.

6.2.4 Aggregation

Figure 6.9 plots the aggregated load (black) of all households of Table 6.1 for the first two months (August and September) of 2018 and the aggregated predicted load (coloured) for this time period. First of all, Figure 6.9 shows that the aggregated predicted load is far from the aggregated actual load in the first few days. This corresponds to the startup effects or adjustment period, which was discussed in Section 5.4.2 and Section 6.2.2. On this aggregated level, the multivariate, iterative, multimodel, and hybrid architecture need approximately 7, 17, 10, and 20 days, respectively, to converge to a reasonable performance.

In Section 2.2.2, the bottom-up approach by Wijaya et al. [31] was discussed. Gerards et al. [32] identified that a potential risk of the bottom-up approach is that household prediction errors could accumulate when the forecasts are aggregated. This happens if the household prediction errors correlate in some way. As each proposed forecasting model is applied to a single household and all forecasts are aggregated, a bottom-up approach is indeed used in this research. Figure 6.9 can be used to see if the errors accumulate. It shows that the forecasting models applied to the households structurally under-predict the absolute peak value (in both positive and negative direction) when using the multivariate architecture. The aggregated house load peaks are larger in both directions than their corresponding aggregated predictions, especially the negative (solar production) peaks. The aggregated predictions of the forecast models are an even worse approximation of the aggregated house loads when the iterative architecture is used. The aggregated predictions of the forecasting models do not follow the seasonal pattern that the actual aggregated load follows, when using this architecture. In contrast, the aggregated predictions

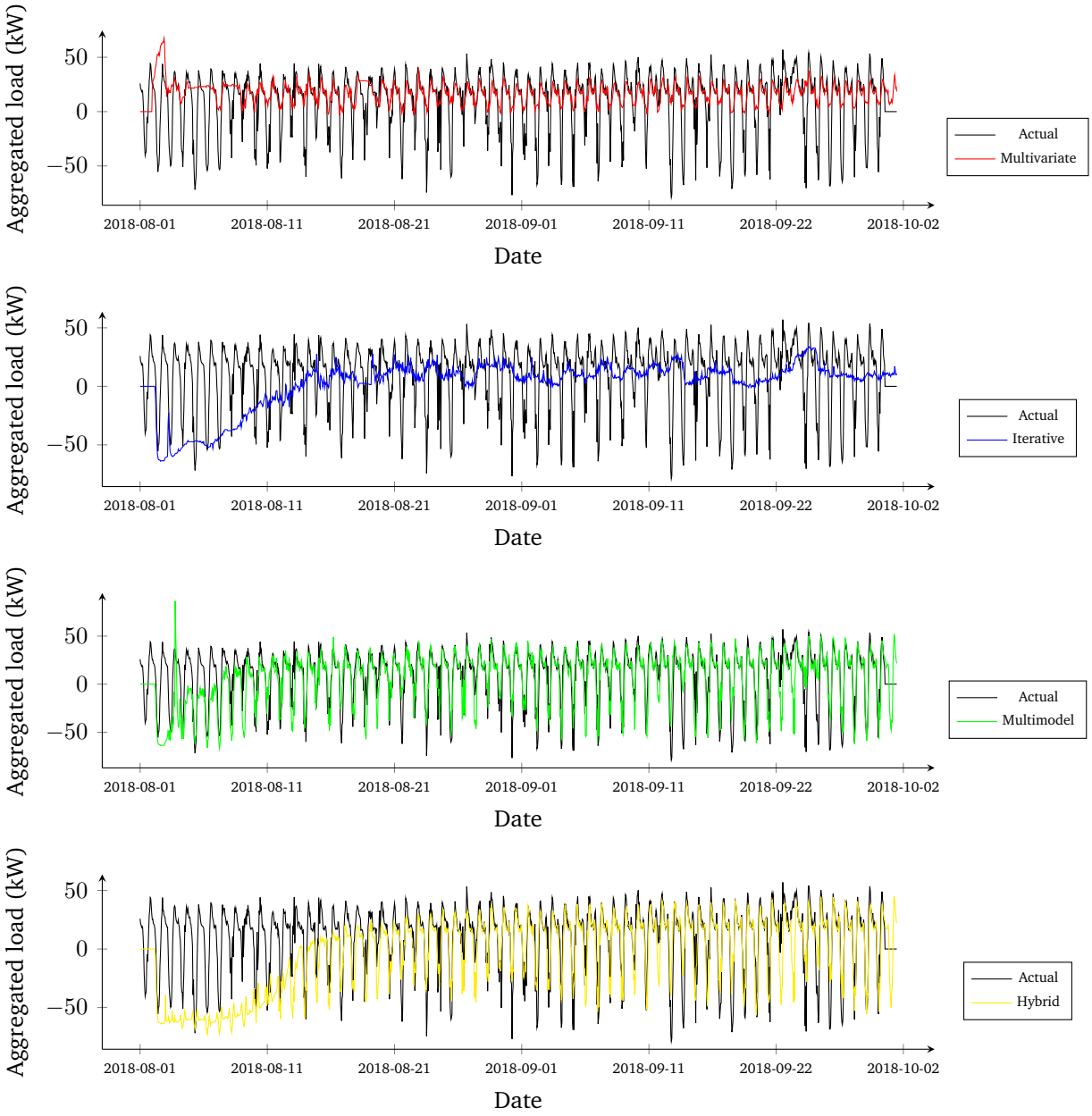


Figure 6.9: Aggregation of all 76 households from the GridFlex data set [10], for the first two months (August and September)

of the forecasting models appear to reasonably approximate the aggregated house load when the multimodel or hybrid architecture is used. Only on some days, the aggregated predictions of the forecast models over- or under-predict the aggregated actual house load. On this aggregated level, the multimodel architecture and the hybrid architecture best approximate the actual aggregated house load.

6.3 Profile Steering Simulation

In this section, the results of the scenarios explained in Section 5.5 are discussed. Section 6.3.1 discusses the results of running a simulation of the scenario explained in Section 5.5.1. Section 6.3.2 discusses the results of running a simulation of the scenario explained in Section 5.5.2. To evaluate the impact of predictions on the performance of the profile steering algorithm, three different types of predictions are used:

- **Perfect predictions**
  In this case, a scenario is simulated where predictions are exactly correct. This means that the sum of the predictions of the house loads (made by the forecasting models of all households) is exactly equal to the sum of all actual house loads. These perfect predictions are used to determine what the profile steering algorithm can achieve in a best case scenario.
- **Predictions from the proposed forecasting model**
  In this case, a scenario is simulated where predictions of all the house loads, made by the proposed forecasting models applied to each house load, are aggregated. This aggregated prediction is an estimate of the total aggregated house load.
- **Synthetic predictions**
  In this case, a scenario is simulated where the predictions are fabricated to obtain a "prediction" with certain properties that is used in the analysis later. Such a synthetic prediction is used to compare the two options discussed in Section 5.5: only using the initial predictions or updating predictions each time step. The synthetic predictions are created as follows. First, a perfect prediction $\vec{p^*}_t$ is created. Then for each component (interval) $p^*_{t,i}$ in this perfect prediction $\vec{p^*}_t$, some semi-random noise is added according to: $p'_{t,i} = p^*_{t,i} \pm \epsilon \frac{4i}{96}$ for $i \in \{1, ..., 96\}$. Here, ϵ is a random number sampled uniformly from the range $0 \leq \epsilon < 1$. Figure 6.10 shows the mean error of each component $p'_{t,i}$ of $\vec{p'}_t$ over all t in the simulation, similar to Figure 6.7 from Section 6.2.3. Note that within the synthetic prediction $\vec{p'}_t$, the first components $p'_{t,i}$ are close to the perfect prediction $p^*_{t,i}$ (small error), but the last components are far away from the perfect prediction (large error). This is different from the predictions of the proposed forecasting model (see Figure 6.7 from Section 6.2.3), where the components have approximately the same error across all intervals $i \in \{1, ..., 96\}$ within the forecast window (i.e. Figure 6.7 shows a near-horizontal line).

For each of these three types of predictions, two options are discussed (see Section 5.5.1): (i) only the initial predictions are used and (ii) the predictions are

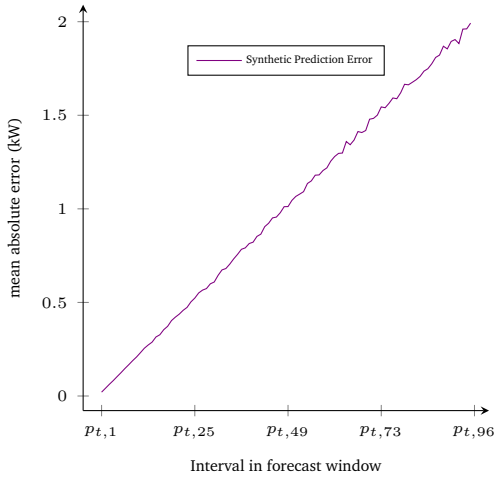


Figure 6.10: Mean absolute error for each component of the synthetic prediction $\vec{p'}_t$ (96 in total)

updated each time step. This means that in total, Section 6.3.1 discusses six different versions of the battery scenario and Section 6.3.2 discusses six different versions of the realistic neighbourhood scenario. For each version of both scenarios, five days are simulated, from 2018-09-01 to 2018-09-06. However, the flexible devices (battery and EVs) start showing extreme behaviour in the last intervals of the simulation, because there exist no time after the "end time" of the simulation. In practice, there exists no such "end time". Therefore, the last day is cut off after each simulation is executed. For each figure that is explained in this section, the same legend is used. The gray continuous line represents the battery load $\vec{x}_B$ ("Battery") in Section 6.3.1 and the aggregated load of all EVs $\vec{x}_{EV}$ ("EVs") in Section 6.3.2. The cyan line ("Houses") represents the (actual) aggregated house load $\vec{x}_H$. The gray dotted line ("Prediction Houses") represents the aggregated predictions of house loads $\vec{x}_{PH}$ (made by the forecasting models corresponding to each household). The red line ("Total Realized") represents the total load (all houses plus battery) $\vec{x}_{TR}$ that was realized. In Section 6.3.1, $\vec{x}_{TR} = \vec{x}_H + \vec{x}_B$ and in Section 6.3.2, $\vec{x}_{TR} = \vec{x}_H + \vec{x}_{EV}$. Finally, the black dashed line ("Prediction Realized") represents the total load (all houses plus battery) that would have been realized $\vec{x}_{PR}$ if the actual house load exactly matched the predicted house load. In Section 6.3.1, $\vec{x}_{PR} = \vec{x}_{PH} + \vec{x}_B$ and in Section 6.3.2, $\vec{x}_{PR} = \vec{x}_{PH} + \vec{x}_{EV}$.

Based on this, error metrics $E1$, $E2$, $E3$, $E4$, and $E5$ are defined:

$$E1 = \frac{1}{t}\sum_{t=1}^{T} |x_{H,t} - x_{PH,t}|$$

$$E2 = \frac{1}{t}\sum_{t=1}^{T} |x_{TR,t} - x_{PR,t}|$$

$$E3 = \frac{1}{t}\sum_{t=1}^{T} |x_{TR,_t} - 0_t|$$

$$E4 = \frac{1}{t}\sum_{t=1}^{T} |x_{PR,t} - 0_t|$$

$$E5 = E3 - E4$$

All errors metrics ($E1$, $E2$, $E3$, $E4$, and $E5$) give the mean absolute error (MAE) of a 15-minute interval of the simulation, as the mean is taken over all time steps in the simulation $t \in \{1, ..., T\}$. Note that $E1$ is simply the prediction error. The metric $E2$ represents how close the total (all houses, plus all EVs/battery) realized load ("Total Realized") is to the predicted realized load ("Prediction Realized"). Note that error $E2$ is a direct consequence of the error $E1$. The error metric $E3$ represents how close the total realized load ("Total Realized") is to a flat zero-profile ($[0, 0, ..., 0]$). The smaller the value of $E3$, the flatter the total realized load profile. The error metric $E4$ represents how close the predicted realized load ("Prediction Realized") is to a flat zero-profile. The metric $E4$ represents the best case scenario, i.e. when the actual aggregated house load perfectly matches the aggregated predicted house loads. Finally, $E5$ is the most important error. The metric $E5$ represents difference between how flat the total realized profile is ($E3$) and how flat the total realized profile could have been in a best case scenario ($E4$).

6.3.1 Battery Scenario

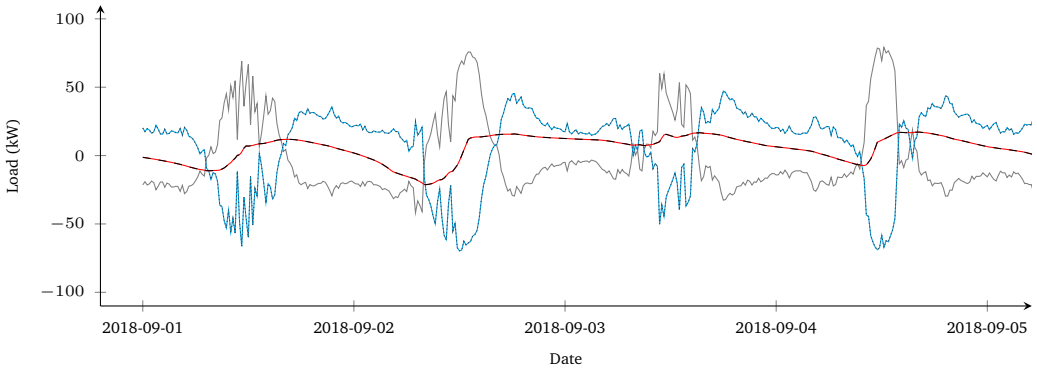
This section discusses the results of simulating the scenario described in Section 5.5.1. Recall that this scenario describes how a single (unrealistically) large battery may flatten the profile of the aggregated load of all houses in a neighbourhood using the profile steering algorithm [33]. Table 6.2 presents the results of using five error

Predictions	No Update					Update				
	$E1$	$E2$	$E3$	$E4$	$E5$	$E1$	$E2$	$E3$	$E4$	$E5$
Perfect	0.0	0.0	10.071	10.071	0.0	0.0	0.0	10.071	10.071	0.0
Synthetic	14.211	14.211	16.487	10.202	6.285	0.169	0.169	10.256	10.259	-0.004
Multivariate	15.415	15.415	22.181	15.775	6.405	15.82	15.82	23.018	15.338	7.679
Iterative	23.29	23.29	27.083	8.842	18.241	22.496	22.496	26.632	9.467	17.165
Multimodel	11.208	11.208	13.015	8.445	4.57	11.335	11.335	13.3	8.571	4.73
Hybrid	10.76	10.76	12.247	6.453	5.794	10.691	10.691	12.236	6.357	5.88
Naive	14.357	14.357	17.236	11.104	6.132	5.34	5.34	26.635	26.63	0.005

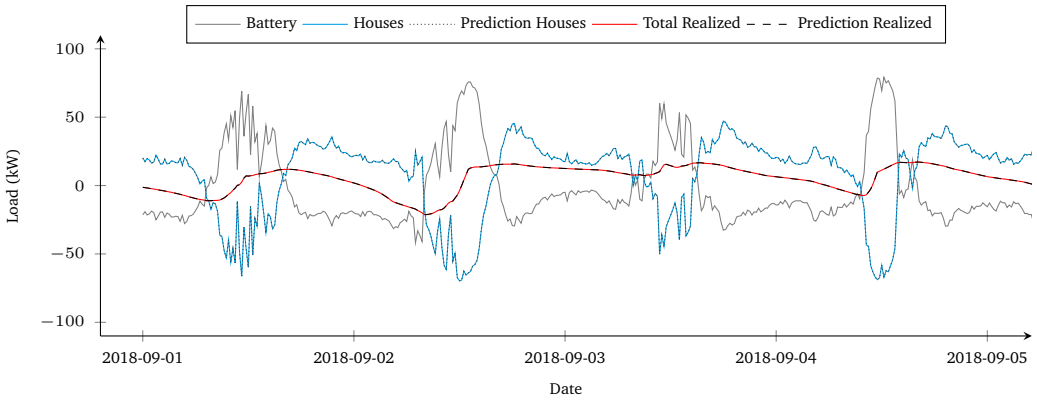
Table 6.2: Battery Scenario

metrics ($E1$, $E2$, $E3$, $E4$, and $E5$) described in Section 6.3. Because $E2$ is a direct

consequence of $E1$, their values are the same. This section discusses the results in Table 6.2 using a few examples.



(a) Not updating predictions



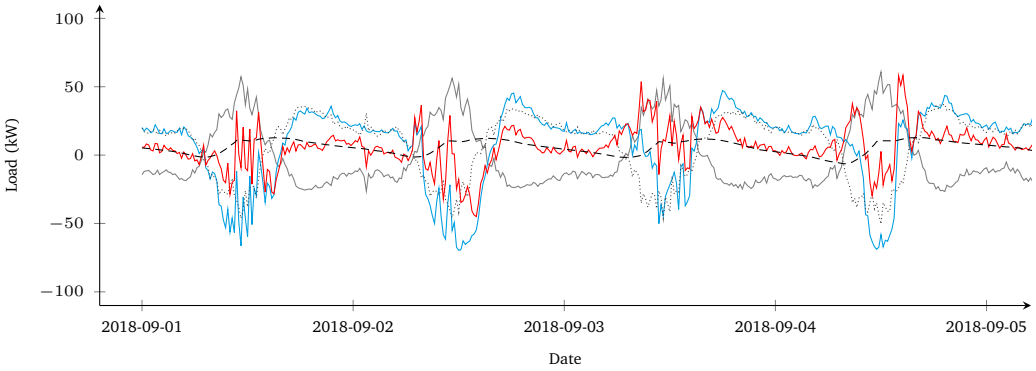
(b) Updating predictions

Figure 6.11: Perfect predictions

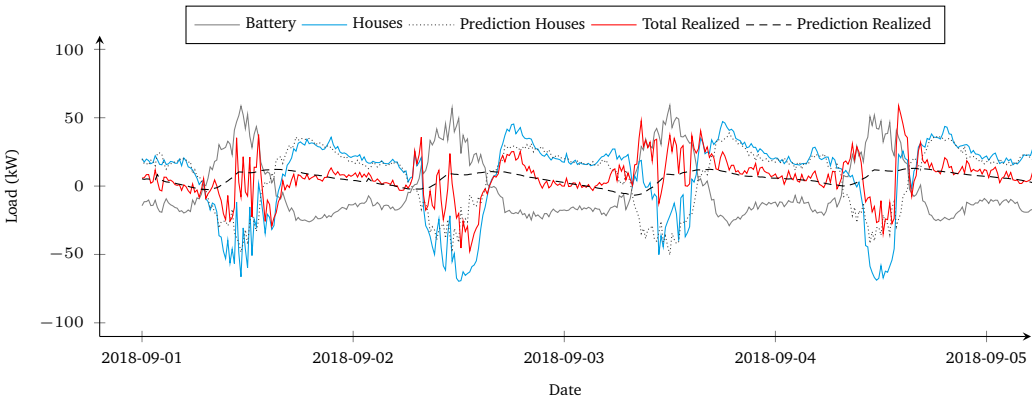
Figure 6.11 shows two different graphs displaying the results of the scenario described in Section 5.5.1, when perfect predictions are used. Figure 6.11a shows the results when the first option (see Section 5.5.1) is chosen: only using initial predictions and not updating the predictions each time step. Figure 6.11b shows the results when the second option is chosen: updating the predictions each time step. Since Figure 6.11 shows the case where perfect predictions are used, the actual aggregated house load ("Houses") and the aggregated predictions of the house loads ("Prediction Houses") overlap. Table 6.2 also shows that $E1 = 0.0$. The profile steering algorithm scheduled the battery load ("Battery") in such a way that its profile counters (approximately reflects over the x-axis) the aggregated predictions of the house load. Especially, when there is a lot of PV production (valleys of the aggregated house load in Figure 6.11), the profile steering algorithm tries to counter this by scheduling the battery to charge during these times. Additionally, the total (houses plus battery) realized load ("Total Realized") overlaps with the total predicted realized load ("Prediction Realized"). This is a direct consequence of having perfect predictions. Therefore, Table 6.2 shows that $E2 = E1 = 0.0$. Furthermore, note that Figure 6.11a and Figure 6.11b are exactly the same. This means that

it does not matter if predictions are updated each time step when predictions are perfect.

More importantly, Figure 6.11 indicates that even with perfect predictions, the (predicted) realized load profile is not entirely flat. In other words, the realized load profile will not be flat in a best case scenario. Table 6.2 shows that $E4 = 10.071 > 0$. Figure 6.11 shows that the (predicted) realized load profile has a repeating pattern each day. At the beginning of the day, the load increases. And at the end of a day, there is a small valley. At the beginning of the next day, the load increases again. This profile shape can most likely be explained by forecast horizon of a day that is used in this thesis (forecast window of 96 15-minute intervals). As a result, the profile steering creates a planning of a day after each time step. When increasing this horizon (to e.g. two days ahead), this shape may be flattened.



(a) Not updating predictions



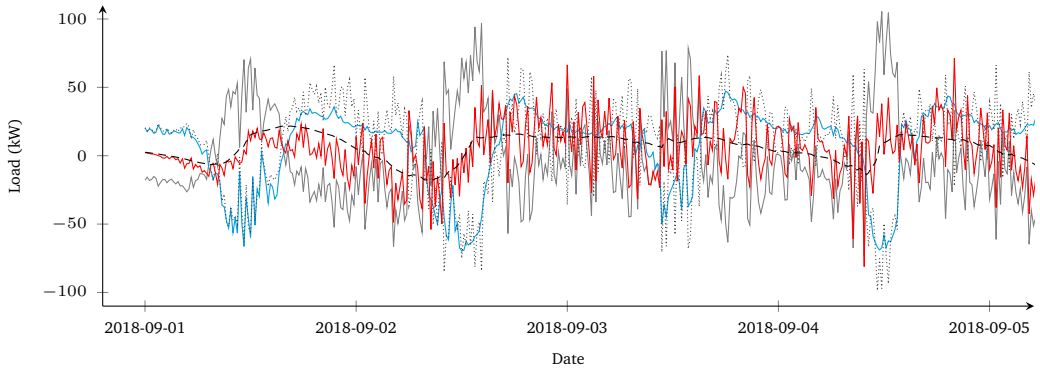
(b) Updating predictions

Figure 6.12: Predictions from the proposed forecasting model using the hybrid architecture

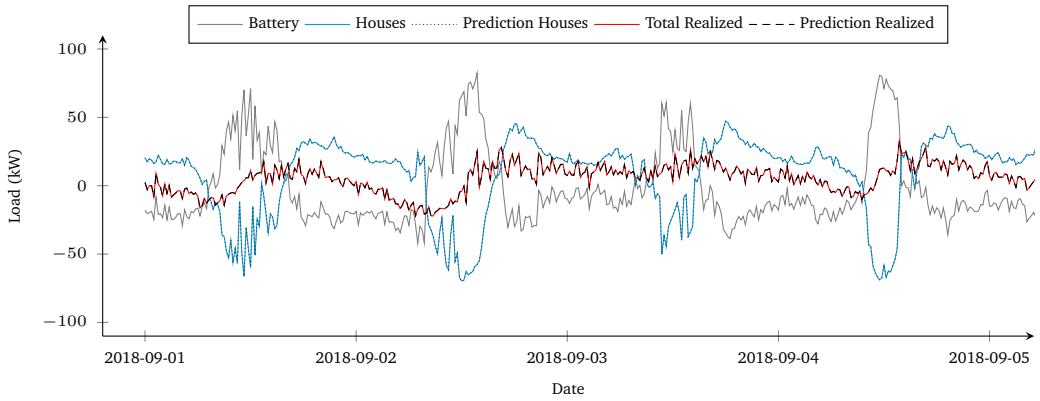
Figure 6.12 shows the results of the scenario described in Section 5.5.1, when predictions of the proposed forecasting model with a hybrid architecture are used. Table 6.2 also shows the results for other architectures (multivariate, iterative, and multimodel). Different from Figure 6.11, the lines in the graphs do not overlap anymore, as the predictions are not perfect. Because the aggregated predictions of the house loads ("Prediction Houses") differs from the actual aggregated house

load (“Houses”), the total (houses plus battery) realized load (“Total Realized”) is different from the total predicted realized load (“Prediction Realized”) which would have happened when the actual aggregated house load would perfectly match the aggregated predictions of the house loads. Table 6.2 shows that $E1 = E2 = 10.76$. Figure 6.12 shows that during the nights, the total realized load (“Total Realized”) matches the predicted realized load (“Prediction Realized”) quite well. However, when there is PV production (valleys of the aggregated house loads) during the day, Figure 6.12 shows that the total realized load (“Total Realized”) often deviates from the predicted realized load (“Prediction Realized”). For example, the PV production on 2018-09-02 and 2018-09-02 of the aggregated house loads (“Houses”) is more than was anticipated. In other words, the valley of the aggregated predictions of the house loads (“Prediction Houses”) is not as deep as the actual aggregated house loads (“Houses”). Because of this unexpected PV production, the battery does not charge enough to compensate this. Thus, the PV valley is not entirely filled. Even though Figure 6.12 shows the aggregated house load (“Houses”) of 76 houses of the GridFlex data set [10], the aggregated house load profile is very volatile during times of PV production. For example, within those valleys there exist load differences between local 15-minute intervals of over 50kW. This makes it very hard to predict the shape of the valley and thus hard to fill the valley correctly.

Again, Figure 6.12a shows the results when the first option (see Section 5.5.1) is chosen: only using initial predictions and not updating the predictions each time step. Figure 6.11b shows the results when the second option is chosen: updating the predictions each time step. However, there is no clear difference in performance between the two options (updating or not updating predictions each time step). The results in Table 6.2 confirm this. This may be explained with forecast window error shown in Figure 6.7. The error of each component $p_{t,i}$ of the prediction $\vec{p}_t$ does not differ between intervals $i \in \{1, ..., 96\}$ within the forecast window. For each forecasting model architecture shown in Figure 6.7, the line is near horizontal. This means that the error of a component $p_{t,i}$ is on average equal to the error of any other component $p_{t,j}$, where $i, j \in \{1, ..., 96\}$. For example, the error of the first component $p_{t,1}$ is on average equally large as the error of the last component $p_{t,96}$. Updating predictions will change the prediction information available to the profile steering algorithm, but does not necessarily improve the quality of this prediction information. Using this reasoning, it is not useful to update predictions. This indicates that updating predictions for the profile steering algorithm does not improve the resulting realized load profile (“Total Realized”), when the predictions from the proposed forecasting model are used. However, if one would use predictions where the error of a component $p_{t,i}$ increases for larger $i \in \{1, ..., 96\}$, updating predictions may be useful. Therefore, Figure 6.13 shows the results of the scenario described in Section 5.5.1, when synthetic predictions are used.



(a) Not updating predictions



(b) Updating predictions

Figure 6.13: Synthetic predictions

Again, Figure 6.13a shows the results when the first option (see Section 5.5.1) is chosen: only using initial predictions and not updating the predictions each time step. It shows that the aggregated predictions of the house loads ("Prediction Houses") are very capricious. This is caused by the random noise that was added to fabricate these synthetic predictions. The load of the battery ("Battery") is scheduled by the profile steering algorithm in a way to counter this predicted volatile profile and achieve a smooth and, most importantly, flat total (houses plus battery) load profile ("Prediction Realized"). However, the real aggregated house load ("Houses") is not so capricious as was predicted by the synthetic predictions. Hence, the battery is scheduled in such a way to counter predicted volatility that does not exist in reality. As a result, the realized load ("Total Realized") becomes very capricious.

Figure 6.13b shows the results when the second option is chosen: updating the predictions each time step. This time, there is a clear difference between Figure 6.13a and Figure 6.13b. Figure 6.13b shows that the aggregated predictions of the house loads ("Prediction Houses") is very close to the actual aggregated house load ("Houses"). This is caused by the frequent updates of the house load predictions used by the profile steering algorithm. Figure 6.10 shows that the components $p'_{t,i}$ that are located in the beginning (i.e. i is close to 1) of the forecasting window $\vec{p'}_t$

have a smaller error on average than the components $p'_{t,i}$ that are located near the end (i.e. i is close to 96). This is why updating causes the error between the prediction of the load and the actual load to decrease, when using synthetic predictions. In general, updating is useful for any prediction with this property. The forecast window of the naive prediction (see Figure 6.7 from Section 6.2.3) also has this property, which is why the error metrics of the naive predictions are also so low in Table 6.2 when predictions are updated at each time step. Because the aggregated (naive and synthetic) predictions of the house loads ("Prediction Houses") are so close to the actual aggregated house load ("Houses"), the total (houses plus battery) realized load ("Total Realized") is also very close to the predicted realized load ("Prediction Realized"). Table 6.2 shows that in this case, the error metrics are very small: $E1 = E2 = 0.169$.

6.3.2 Realistic Neighbourhood Scenario

This section discusses the results of simulating the scenario described in Section 5.5.2. Recall that this scenario describes how 38 EVs together may fill valleys of the profile of the aggregated load of all houses in a neighbourhood using the asynchronous event-driven profile steering algorithm [25]. Table 6.3 presents the results of using

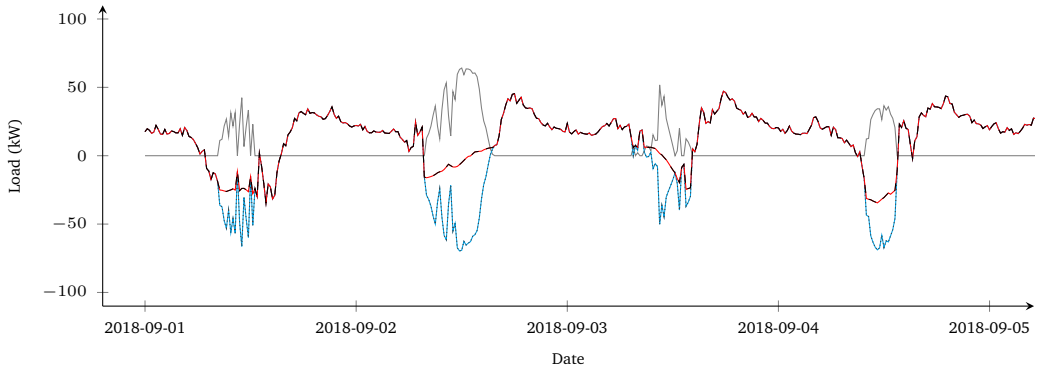
Predictions	No Update					Update				
	$E1$	$E2$	$E3$	$E4$	$E5$	$E1$	$E2$	$E3$	$E4$	$E5$
Perfect	0.0	0.0	20.414	20.414	0.0	0.0	0.0	20.414	20.414	0.0
Synthetic	14.255	14.255	22.011	22.773	-0.761	0.168	0.168	20.702	20.708	-0.006
Multivariate	15.418	15.418	25.24	19.612	5.628	15.826	15.826	25.472	18.694	6.778
Iterative	23.265	23.265	29.102	12.548	16.554	22.495	22.495	28.607	13.236	15.37
Multimodel	11.218	11.218	22.639	19.558	3.081	11.33	11.33	22.917	19.113	3.805
Hybrid	10.773	10.773	22.985	16.186	6.799	10.701	10.701	22.909	16.22	6.689
Naive	14.354	14.354	24.553	18.452	6.101	5.336	5.336	28.61	28.601	0.009

Table 6.3: Realistic Neighbourhood scenario

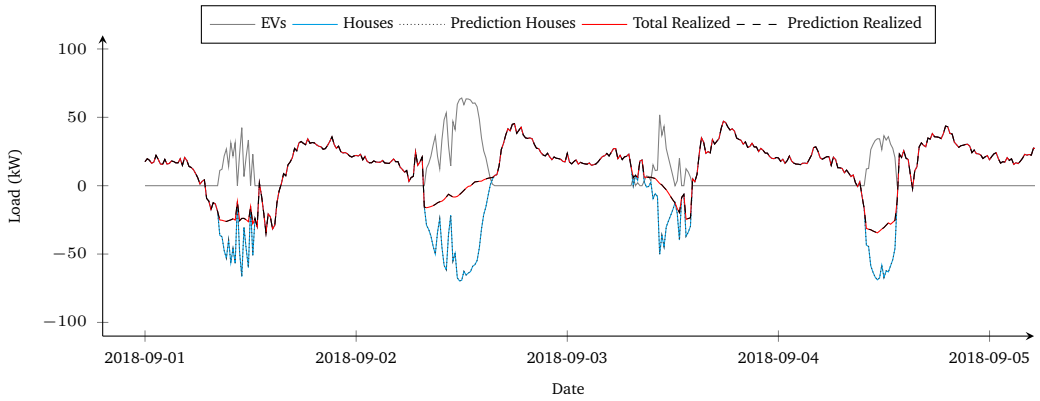
five error metrics ($E1$, $E2$, $E3$, $E4$, and $E5$) described in Section 6.3. Similar to Section 6.3.1, this section discusses the results in Table 6.3 using a few examples.

Figure 6.14 shows two different graphs displaying the results of the scenario described in Section 5.5.2, when perfect predictions are used. Figure 6.14a shows the results when the first option (see Section 5.5.2) is chosen: only using initial predictions and not updating the predictions each time step. Figure 6.14b shows the results when the second option is chosen: updating the predictions each time step. Similar to Section 6.3.1, the lines corresponding to "Houses" and "Prediction Houses" overlap, because predictions are perfect. Therefore, also the lines corresponding to "Total Realized" and "Prediction Realized" overlap. Additionally (also similar to Section 6.3.1) there is no difference between Figure 6.14a and Figure 6.14b.

Because EVs start charging (arrive at the home charging point) at different (randomly chosen) times, there is not always enough flexibility available to completely fill the



(a) Not updating predictions

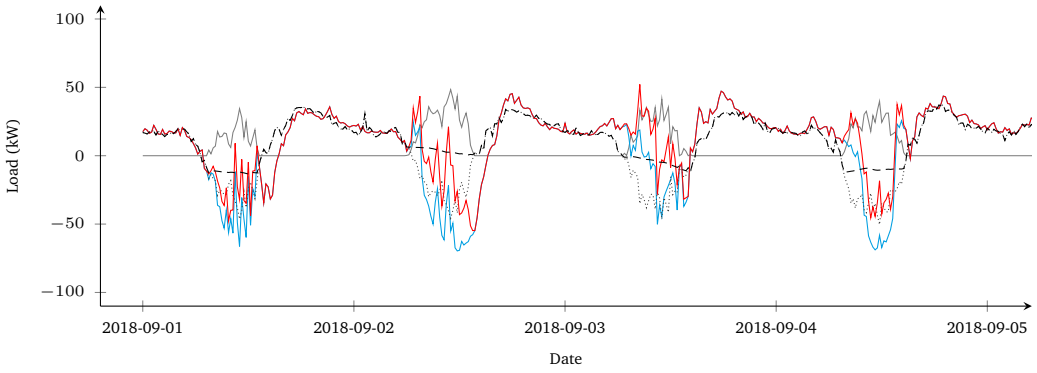


(b) Updating predictions

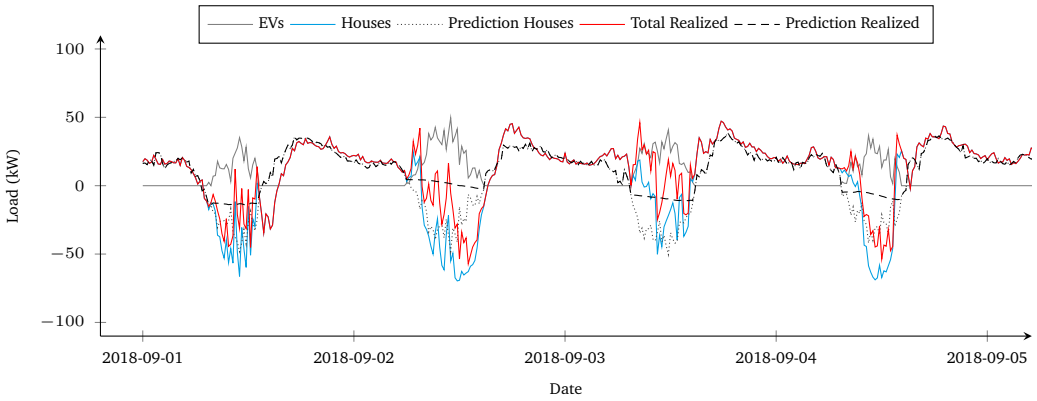
Figure 6.14: Perfect predictions

valleys. Even with perfect predictions, the valleys of the aggregated house load profile ("Houses") are not completely filled. Therefore, the total (all houses plus all EVs) realized profile ("Total Realized") still shows valleys, albeit these valleys are less deep.

Figure 6.15 shows the results of the scenario described in Section 5.5.2, when predictions of the proposed forecasting model with a hybrid architecture are used. Again, Figure 6.15a shows the results when the first option (see Section 5.5.2) is chosen: only using initial predictions and not updating the predictions each time step. Figure 6.15b shows the results when the second option is chosen: updating the predictions each time step. Similar to the results described in Section 6.3.1, there is no clear difference in performance between the two options (updating or not updating predictions each time step). In both graphs, the asynchronous event-driven profile steering algorithm is unable to match the total (all houses plus all EVs) realized load ("Total Realized") with the predicted realized load profile ("Prediction Realized"). This is partly because it is uncertain when EVs become available (arrive at home), partly because the EVs together only provide a limited amount of flexibility (more capacity is required to completely fill the valleys of the aggregated house load profile), and partly because of the prediction errors of the proposed forecasting



(a) Not updating predictions

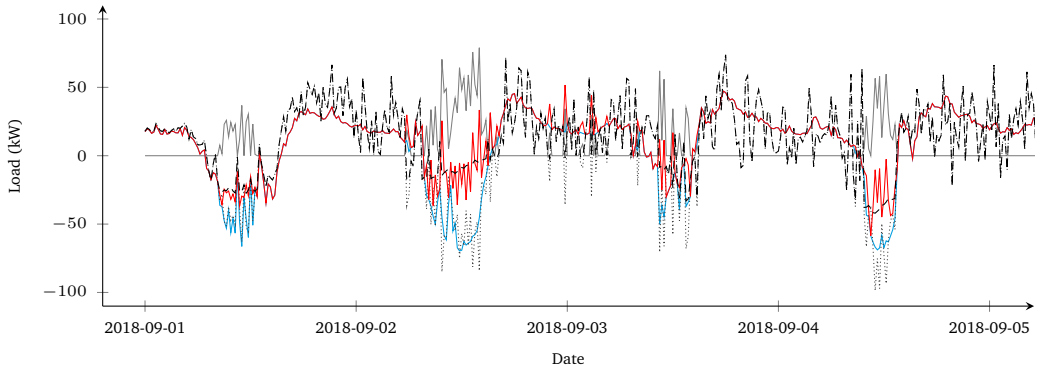


(b) Updating predictions

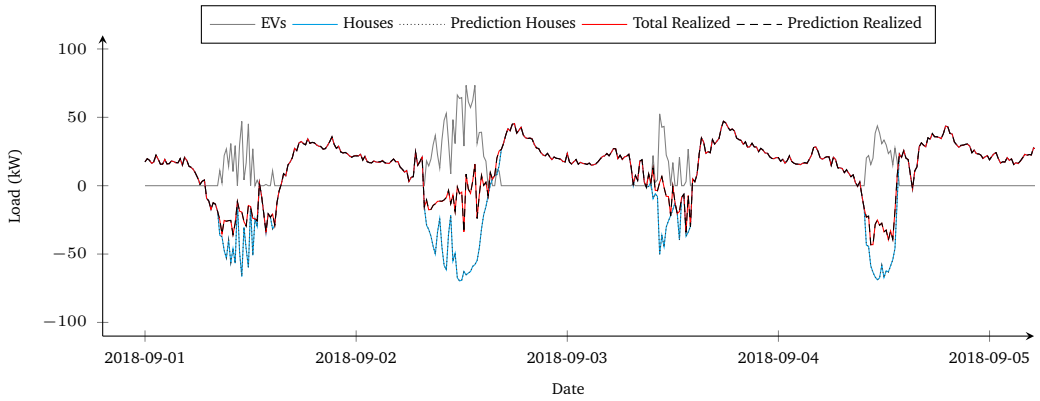
Figure 6.15: Predictions from the proposed forecasting model using the hybrid architecture

model. Similar to Section 6.3.1, the large differences (sometimes of 50kW) between the load at intervals within a valley (during PV production) makes it difficult to schedule EVs to accommodate these local peaks. However in most intervals, the total (all houses plus all EVs) load profile ("Total Realized") is slightly better than the total aggregated house load profile ("Houses") as valleys are filled to some extent.

Finally, Figure 6.16 shows the results of the scenario described in Section 5.5.2, when synthetic predictions are used. Again, Figure 6.16a shows the results when the first option (see Section 5.5.2) is chosen: only using initial predictions and not updating the predictions each time step. And Figure 6.16b shows the results when the second option is chosen: updating the predictions each time step. Similar to the results in Section 6.3.1, there is a clear difference between Figure 6.16a and Figure 6.16b. The total (all houses plus all EVs) realized load profile ("Total Realized") shown in Figure 6.16a is more capricious than the total realized load profile ("Total Realized") shown in Figure 6.16b. Also, Figure 6.16b shows that the asynchronous event-driven profile steering algorithm is better able to match the total realized load profile ("Total Realized") with the predicted realized load profile ("Prediction Realized") when predictions are updated.



(a) Not updating predictions



(b) Updating predictions

Figure 6.16: Synthetic predictions

However, note that Table 6.3 shows that $E5$ is a negative value and also the lowest value out of all other prediction types, even when predictions are not updated. It seems strange that the synthetic predictions have the lowest $E5$ value when predictions are not updated, but this can be explained. In general, $E4$ is always less than or equal to $E3$ for any type of prediction in the battery scenario if the predictions are not updated. This is because the predicted realized load ("Prediction Realized") is typically flatter than the actually realized load ("Total Realized"). However, in the realistic neighbourhood scenario, EVs can only fill valleys and not shave peaks, so their batteries are not scheduled to counter the capricious synthetic predictions of the house loads when there is no PV production (no valley to fill). Therefore, the predicted realized load ("Prediction Realized") matches the aggregated predictions of the house loads ("Prediction Houses") perfectly at these times. However, the aggregated predictions of the house loads ("Prediction Houses") is very capricious due to the nature of the synthetic predictions. Furthermore, the curve of the actually realized load ("Total Realized") matches the curve of the actual aggregated house load ("Houses") perfectly at these times. Since the aggregated house load ("Houses") is much flatter than its synthetic prediction ("Prediction Houses"), the total realized load ("Total Realized") is also much flatter than the

predicted realized load (“Prediction Realized”). Thus, the value of $E5$ is somewhat misleading in this specific case.

7 Conclusions And Future Work

Chapter Objective: *Based on the findings in this thesis, this chapter draws conclusions to answer the research questions defined in Chapter 1 and suggests directions for future work.*

Chapter Contents

7.1 Introduction

As a result of the energy transition, many residents have installed rooftop solar panels and started using more electrical appliances, such as electric vehicles (EVs) and heat pumps. This decentralized production and consumption puts pressure on the current electricity grid. To flatten load peaks and prevent overloading of grid components, demand side management (DSM) techniques can be employed to match electricity supply and demand. Some DSM techniques require house load predictions. However, there are challenges regarding the application of load predictions, the variation between household loads, and the volatility of a load within a single household. To overcome these challenges, this thesis proposes an online "rolling horizon" forecasting model that is continuously being trained on the smart meter data of a single household. It can therefore adapt to the load profile behaviour of that household and to structural changes in electricity consumption over time.

Chapter 1 introduces the house load forecasting challenges in more detail and states the research questions. Chapter 2 discusses some characteristics specific to electricity consumption forecasting and reviews several time series forecasting models that are used for predicting electricity consumption. In particular, a motivation is provided for the use of artificial neural networks (ANNs). Chapter 3 introduces demand side

management (DSM) and reviews several DSM techniques. Chapter 4 provides an analysis of the input data for the proposed forecasting model. Chapter 5 explains the structure of the proposed forecasting model. The simulation results of applying the proposed forecasting model on a set of households are discussed in Chapter 6. This chapter also evaluates a specific DSM technique that uses the predictions of the proposed forecasting model in several neighbourhood scenarios. The remainder of this chapter answers the research questions in Section 7.2. Finally, Section 7.3 presents recommendations for future work.

7.2 Conclusions

This section answers the research questions defined in Section 1.3. This section lists all research questions and presents their corresponding answers below the question.

RQ 1 What input data correlates the most with the residential electricity consumption that needs to be predicted?

Firstly, residential historic electricity consumption correlates with the electricity consumption that needs to be predicted, and could therefore be used to make predictions. Section 2.2 reviews literature related to electricity consumption forecasting and states that an electricity load series exhibits seasonality, because it is strongly auto-correlated. Section 4.4.2 confirms this by presenting an auto-correlation analysis for the houses of the GridFlex data set [10]. This analysis helps to identify what part of the historic consumption data is most useful when making house load predictions. The results of this analysis show no strong weekly seasonality, but do show a strong yearly and daily seasonality. On average, the house load at the exact same time interval on a previous day correlates the most with the current house load. In other words, there is a specific lag that is most relevant to predict a load value at a specific interval in the future. Since the proposed forecasting model predicts multiple (96) values within a forecast window, multiple lags are relevant. To employ a larger subset of the historic house load data when making a prediction and to constraint the number of input variables for the proposed forecasting model, window variables are used. Since the correlation analysis identifies a declining correlation trend after a few months, the lag and window input variables only cover a period of the last a month at most. The trend is captured using trend variables. Yearly seasonality has not been taken into account, because the GridFlex data set [10] only covers two years of data.

Apart from historic electricity consumption data, there exist exogenous variables that correlate with the residential electricity consumption that needs to be predicted. Section 2.2.3 reviews different exogenous variables that affect the load profile of a house. For example, building characteristics have an impact on the house load profile. However, since these variables are static or fixed, we expect that the proposed forecasting model will learn this information from the load profile of the house. Additionally, demographic information about the household occupants has influence on the house load profile. However, the proposed forecasting model learns autonomously and can adapt to structural changes within the household (e.g. a new family moves in). Furthermore, this requires issuing a survey that asks residents for personal data and this should be avoided for an autonomous model. In contrast, publicly available data such as time data and weather data is used by the proposed forecasting model. Time data may indicate whether residents are at home or not (e.g. public holidays) or whether they are using specific electric appliances. Weather data may impact the use of heating devices and has a significant effect on the yield of solar panels (PV). Since many houses of the GridFlex data set [10] have rooftop solar panels, weather data from the KNMI [74] was used. To find out what weather variables correlate with the residential electricity consumption, a weather correlation analysis was carried out. Based on the correlation with the house load of a subset of houses, a set of weather variables was selected (see highlighted entries of Table 4.2). This set includes mostly information about sunshine duration and radiation, but also soil temperature, air temperature, humidity, wind speed and direction, clouds, and visibility. It is important to note that these variables only had a high correlation with the load of houses with rooftop solar panels, but the correlation was minimal for houses without rooftop solar panels. Since many houses of the GridFlex data set [10] have solar panels, these variables are used as input to the proposed forecasting model.

RQ 2 How can an online short-term forecasting model be determined to predict household electricity consumption?

Chapter 5 explains the proposed forecasting model. Using a black-box paradigm, Section 5.2 explains that an online forecasting model can be created using a rolling-horizon with discrete time steps. Each time step, the proposed forecasting model needs to do three things: update/calculate the input data, make a prediction of the house load, and plan a training update.

- The input consists of a set of weather, time, and historic electricity consumption variables that are summarized in Section 4.4.4. The values of the weather, time, and historic electricity consumption variables need to be updated each time step of the rolling horizon and are explained in Section 4.4.1, Section 4.4.3,

and Section 4.4.2, respectively. Section 4.4.2 also explains how some of these variables (window and trend variables) need to be re-calculated each time step.

- Each time step, a house load prediction is made with a forecasting horizon of a day. This prediction is based on input data of a month ago and the output of the prediction is a forecast window consisting of 96 15-minute intervals of the next day.
- Each time step, a training update should be planned so that the forecasting model can learn from its prediction mistakes and improve its performance for a future prediction. This process allows the proposed forecasting model to adapt to the specific characteristics of the house load profile and to structural changes in the house load data. As training the forecasting model cannot be done based on data from the future, a training update of the forecasting model needs to be scheduled ahead, such that it can be carried out a day later. These training updates use information of the actual house load of the last day and its corresponding prediction (the forecast window) that was made exactly a day ago.

Section 2.3 motivates that Artficial Neural Networks (ANNs) are most useful for predicting household electricity consumption in an online setting, because of their ability to approximate nonlinear functions well (better than e.g. ARIMA). Since a house load profile is very volatile, this property very useful for short-term house load forecasting. Also, ANNs are computationally faster than deep complex networks (e.g. LSTMs), which makes them more suitable when predicting the house load in an online setting at a (perhaps computationally slow) local house controller. Section 2.4 explains a more detailed background on ANNs. Section 5.3 explains four different forecasting model architectures: multivariate, iterative, multimodel, and hybrid. Each architecture uses ANNs in a different way to make a prediction. Chapter 6 shows that on average (of all houses of the GridFlex data set [10]) out of the forecasting model architectures, the mean error (μ) is lowest for the multimodel architecture, but the median error (m) is lowest for the hybrid architecture. However, Section 6.2.1 shows that the results differ significantly per household and indicates a relationship between the house load volatility and the performance of the forecasting model. Chapter 6 also shows that all architectures suffer from start up effects (large errors in the beginning of the simulation) and need some time to adapt their predictions to a house load. Section 5.4 shows that start up effects and feedback sensitivity are mainly caused by the choice of hyperparameters. When the optimal hyperparameters are selected based on a grid search, the multivariate model appears to have the lowest startup time (approximately one week) compared to other architectures. Next to the start up phase, Section 6.2.2 discusses how each forecasting model architecture performs over time. It shows that the error of the architectures during late spring and early summer is significantly higher than other

periods, especially for houses with rooftop solar panels. This suggests that prediction errors are higher when there is solar production. The iterative (and to a lesser extent the multivariate) architecture is more erroneous than the multimodel and hybrid architecture, especially during these periods. This is also reflected in the stability (σ) of their predictions, which is significantly better for the multimodel and hybrid architecture compared to the iterative and multivariate architecture. Section 6.2.3 shows that (for all architectures), there is no difference between the average error of the 96 components within the forecast window. When comparing the architectures, the hybrid architecture has the lowest forecast window error on average. Finally, Section 6.2.4 shows that the aggregated predictions of the house load best approximate the actual aggregated house load, when the multimodel or hybrid architecture is used by the forecasting models. Considering the above, the hybrid architecture appears to be most suitable architecture for the proposed forecasting model out of the four architectures, to make house load predictions.

RQ 3 How to make meaningful predictions of household electricity consumption that can be used as input to a DSM scheduling algorithm?

Section 5.5 explains two neighbourhood scenarios that use the profile steering algorithm [33]. Section 6.3 discusses the results of simulating these scenarios. When only initial predictions are used (predictions are not updated), the hybrid architecture has the lowest prediction error and thus the lowest error metrics $E1$ and $E2$. However, the predictions made by the forecasting model with the multimodel architecture has the lowest $E5$ value in the both scenarios (ignoring the misleading $E5$ value of the synthetic predictions without updates in the EV scenario, explained in Section 6.3.2). This indicates that a better forecast does not always result in the best outcome, when used by the profile steering algorithm.

If predictions are updated, then the synthetic predictions have the best results (lowest errors $E1$ and $E5$). This is explained by the shape (small component error in the beginning and large component error in the end) of the forecast window error of the synthetic predictions, as explained in Section 6.3. However, synthetic predictions cannot be used in practice, because they serve simply as a theoretical example to show that the shape of the forecast window determines how much the performance of the profile steering algorithm improves when updating predictions. The naive predictions also have relatively good results (low errors $E1$ and $E5$) when predictions are updated, because the forecast window error of the naive predictions has the same property. This indicates that, if predictions are updated each time step, the error size of the first interval/component of the forecast window has the most influence on the performance of the profile steering algorithm. Thus for the profile steering algorithm, the accuracy of the predictions of the upcoming interval(s) is

more important than later intervals, if those predictions are updated each time step. If predictions are not updated each time step, then the accuracy of the first interval is not necessarily more important than the accuracy of later intervals in the forecast window.

The results of Section 5.5 also show that achieving a flat profile is most difficult during periods of PV production. Even with an aggregated house load (of all 76 houses of the GridFlex data set [10]), there exist load differences between 15-minute intervals of more than 50kW. This makes those periods hard to predict (there is a larger prediction error during these periods) and indicates that simply "scheduling" the load for the upcoming interval(s) is not sufficient to realize a flat load profile.

7.3 Recommendations For Future Work

7.3.1 Historic Electric Consumption Data Selection

Section 4.4.2 explains how parts of the historic electricity consumption data can be chosen based on a load correlation analysis. However, an alternative way to find the most relevant lags is to use Dynamic Time Warping (DTW) instead of correlation analysis. DTW is an algorithm that measures the similarities between two time series. The electric house load of a house from the GridFlex data set [10] is also a time series. When comparing the house load series with a lagged version of that same house load series, DTW may measure high similarities at different lags. Therefore, DTW may provide new insights into what historic electricity consumption may be useful for predictions.

7.3.2 Different Forecasting Model Architectures

In this thesis four different forecasting model architectures are discussed: multivariate, iterative, multimodel, and hybrid. However, these are not the only four options that exist. The multivariate architecture has an output layer consisting of 96 output neurons, corresponding to the 96 components of the forecast window. For future work, it is interesting to look at the error of each component when a larger forecast window is used, e.g. for two days (consisting of 192 components). In that case, the output layer of the ANN would have 192 output neurons. The multimodel and hybrid architecture are both split based on a classification: 15-minute intervals in a day and hours in a day, respectively. For future work, it would be interesting to evaluate the performance of architectures based on other classifications. For example, one ANN for each day of the week (seven ANNs in total). Another architecture may be one

ANN for work days and one ANN for weekend days. In general, one can replace any of the time variables in Table 4.7 (in Section 4.4.3), and create a new forecasting model architecture instead.

7.3.3 Other Machine Learning Algorithms

In this research, ANNs are used in the proposed forecasting model. However, there exist more machine learning algorithms that may fit in the black-box paradigm explained in Section 5.2. For example, a random forest approach or deeper neural networks may be used in a future work. For example, a deep learning technique such as a transformer or LSTM can be used if it is computationally fast enough to be applied (train and predict) in the online setting that is described in this thesis. Deep learning techniques generally have a higher accuracy and may therefore provide better predictions.

7.3.4 Dynamic Hyperparameters

As explained in Section 5.4.2, the value of the hyperparameters influences the length of the startup phase and the sensitivity towards feedback. Instead of choosing hyperparameter values as a fixed number based on a grid search, it would be interesting to look at hyperparameters that can change dynamically over the course of the simulation. For example, Chapter 6 shows that the startup effects are between approximately one week and one month for different forecasting model architectures. When increasing the learning rate in the beginning of the simulation, the length of the startup effects may be reduced. Also when a structural change to the household took place, the forecasting model should adapt quickly. In such a case, the learning rate could also be increased for a small period of time to quickly adapt the predictions to the changed load shape. Furthermore, Section 6.2.2 shows that the prediction error differs over time. This suggests that different hyperparameter values (than the ones chosen based on the grid search) may be optimal in these periods. Based on the prediction error, all hyperparameters may be changed dynamically to have an optimal performance at all times.

7.3.5 The Fill-Level

Another direction for future work concerns the fill-level Z that is discussed in Section 3.7. Schoot Uiterkamp et al. [72] present a method to predict this optimal value of Z using an empirically created distribution of the optimal fill-level Z. However, a forecasting model like the one that is proposed in this thesis may also be

used to predict this optimal fill-level Z instead of predicting the load of a household on the grid, which is hard to predict.

7.3.6 Load Volume And Upcoming Interval

Section 3.7 also argues that a prediction of the base load profile $\vec{p}$ at multiple intervals in the future is not required when provided with an estimate of the optimal fill-level $\hat{Z}$. Only of the base load of the upcoming time interval p_t and the sum of complete total energy consumption (over all later intervals) is needed to make a schedule for an EV that is close to optimal [34]. This is why it may be interesting for future work to develop a forecasting model (ANN) that predicts the total volume of the load (over all later intervals). In line with Gerards et al. [34] Section 7.2 concludes that an accurate prediction of the upcoming interval is most important when the DSM algorithm (profile steering algorithm in this case) has access to updated predictions (if the property holds that the predictions are more accurate in the upcoming intervals compared to later intervals). Since weather predictions are typically more accurate for the upcoming intervals compared to later intervals, a DSM algorithm (like profile steering) may perform better when the weather predictions are updated each time step. Therefore, it would be interesting for future work to develop a forecasting model (ANN) that focuses on predicting purely the load at the upcoming interval.

7.3.7 Separate PV Predictor And Operational Control

Section 7.2 concludes that achieving a flat profile is most difficult during periods of PV production. Firstly, this suggests that a separate PV predictor may improve the prediction performance. One predictor could focus on only predicting electricity consumption and one predictor could focus on only predicting PV production. Secondly, as the prediction error during periods of PV production is larger, simply "scheduling" the load for the upcoming interval(s) may not be sufficient to realize a flat load profile. We therefore suggest that future work investigates how operational control may help to flatten the profile in these periods. For example, if there is an error between the predicted load and actual load, a flexible device (e.g. a battery) could accommodate this error by adapting its load real-time. In such an approach, the flexible devices (e.g. batteries and EVs) should never be scheduled in a way that they can reach their capacity limit (either completely empty or completely full), but always leave some room to accommodate prediction errors.

Bibliography

[4] WindEurope, „Wind energy in Europe: 2021 Statistics and the outlook for 2022-2026,“ EWEA, Tech. Rep., Feb. 2022 (cit. on p. 1).

[5] The International Renewable Energy Agency, „Renewable Capacity Statistics 2022,“ IRENA, Abu Dhabi, Tech. Rep., 2022 (cit. on p. 1).

[10] V. M. J. J. Reijnders, M. E. T. Gerards, and J. L. Hurink, *Energy consumption data of the GridFlex Heeten project*, Accessed: 07-07-2022, Enschede, 2021 (cit. on pp. 2, 4, 5, 9, 13, 16, 37–43, 45, 46, 49–51, 56, 69, 71, 73, 74, 76, 79, 84, 85, 91, 100–102, 104).

[12] N. Wei, C. Li, X. Peng, F. Zeng, and X. Lu, „Conventional models and artificial intelligence-based models for energy consumption forecasting: A review,“ *Journal of Petroleum Science and Engineering*, vol. 181, p. 106 187, Oct. 2019 (cit. on pp. 4, 17–19, 24).

[13] K. Amasyali and N. M. El-Gohary, „A review of data-driven building energy consumption prediction studies,“ *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 1192–1205, Jan. 2018 (cit. on pp. 4, 6, 7, 16, 17, 19, 24).

[14] D. Mariano-Hernández, L. Hernández-Callejo, F. S. García, O. Duque-Perez, and A. L. Zorita-Lamadrid, „A Review of Energy Consumption Forecasting in Smart Buildings: Methods, Input Variables, Forecasting Horizon and Metrics,“ *Applied Sciences 2020, Vol. 10, Page 8323*, vol. 10, no. 23, p. 8323, Nov. 2020 (cit. on pp. 4, 7, 16, 17).

[15] L. Hernandez, C. Baladrón, J. M. Aguiar, *et al.*, „A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings,“ *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1460–1495, 2014 (cit. on pp. 4, 17).

[16] H. Ziekow, C. Goebel, J. Strüker, and H. A. Jacobsen, „The potential of smart home sensors in forecasting household electricity demand,“ *2013 IEEE International Conference on Smart Grid Communications, SmartGridComm 2013*, pp. 229–234, 2013 (cit. on p. 5).

[17] A. Molderink, V. Bakker, M. G. C. Bosman, J. L. Hurink, and G. J. M. Smit, „Management and Control of Domestic Smart Grid Technology,“ *IEEE transactions on smart grid*, vol. 1, no. 2, pp. 109–119, Sep. 2010 (cit. on pp. 7, 15).

[18] S. Singh and A. Yassine, „Big Data Mining of Energy Time Series for Behavioral Analytics and Energy Consumption Forecasting,“ *Energies 2018, Vol. 11, Page 452*, vol. 11, no. 2, p. 452, Feb. 2018 (cit. on pp. 7, 17).

[19] H. S. Hippert, C. E. Pedreira, and R. C. Souza, „Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, Feb. 2001 (cit. on pp. 12, 16, 18, 19, 22–25, 38, 49, 59).

[20] G. Zhang, B. Eddy Patuwo, and M. Y. Hu, „Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, Mar. 1998 (cit. on pp. 12, 18, 19, 22–24).

[21] J. L. Yuan and T. L. Fine, „Neural-network design for small training sets of high dimension," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 266–280, 1998 (cit. on p. 12).

[22] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, New Jersey: John Wiley & Sons, 2016, p. 177 (cit. on p. 12).

[24] S. Humeau, T. K. Wijaya, M. Vasirani, and K. Aberer, „Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households," *2013 Sustainable Internet and ICT for Sustainability, SustainIT 2013*, 2013 (cit. on pp. 12, 14, 16, 25).

[25] G. Hoogsteen, A. Molderink, J. L. Hurink, and G. J. M. Smit, „Asynchronous event driven distributed energy management using profile steering," *2017 IEEE Manchester PowerTech, Powertech 2017*, p. 7 980 986, Jun. 2017 (cit. on pp. 13, 36, 56, 69, 71–73, 93).

[26] B. Yildiz, J. I. Bilbao, J. Dore, and A. B. Sproul, „Short-term forecasting of individual household electricity loads with investigating impact of data resolution and forecast horizon," *Renewable Energy and Environmental Sustainability*, vol. 3, p. 3, 2018 (cit. on pp. 13, 14, 16, 25).

[27] M. Rossi and D. Brunelli, „Electricity demand forecasting of single residential units," *2013 IEEE Workshop on Environmental, Energy and Structural Monitoring Systems, EESMS 2013 - Proceedings*, 2013 (cit. on pp. 14, 25).

[28] F. Javed, N. Arshad, F. Wallin, I. Vassileva, and E. Dahlquist, „Forecasting for demand response in smart grids: An analysis on use of anthropologic and structural data and short term multiple loads forecasting," *Applied Energy*, vol. 96, pp. 150–160, Aug. 2012 (cit. on pp. 14–16, 22, 24, 25, 44, 74).

[29] R. Sevlian and R. Rajagopal, „A scaling law for short term load forecasting on varying levels of aggregation," *International Journal of Electrical Power and Energy Systems*, vol. 98, pp. 350–361, Jun. 2018 (cit. on p. 14).

[30] A. Tidemann, B. A. Høverstad, H. Langseth, and P. Öztürk, „Effects of scale on load prediction algorithms," *IET Conference Publications*, vol. 2013, no. 615 CP, 2013 (cit. on pp. 14, 16).

[31] T. K. Wijaya, M. Vasirani, S. Humeau, and K. Aberer, „Cluster-based aggregate forecasting for residential electricity demand using smart meter data," *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pp. 879–887, Dec. 2015 (cit. on pp. 14, 16, 84).

[32] M. E. T. Gerards and J. L. Hurink, „Robust peak-shaving for a neighborhood with electric vehicles," *Energies*, vol. 9, no. 8, 2016 (cit. on pp. 15, 29, 31, 33–35, 84).

[33] M. E. T. Gerards, H. A. Toersche, G. Hoogsteen, *et al.*, „Demand side management using profile steering," *2015 IEEE Eindhoven PowerTech, PowerTech 2015*, pp. 1–457 759, Jun. 2015 (cit. on pp. 15, 31, 35, 36, 56, 69, 72, 73, 88, 103).

[34] M. E. T. Gerards, J. L. Hurink, and R. Hübner, „Demand side management in a field test: lessons learned," *CIRED, Open Access Proc. J., IET Journals*, vol. 2017, no. 1, pp. 1678–1681, Jun. 2017 (cit. on pp. 15, 34, 41, 106).

[35] „NEN-EN 50160:2010/A3:2019 Spanningskarakteristieken in openbare elektriciteitsnetten," NEN, Tech. Rep., Oct. 2019 (cit. on p. 15).

[37] K. Gajowniczek and T. Zabkowski, „Short term electricity forecasting using individual smart meter data," *Procedia Computer Science*, vol. 35, no. C, pp. 589–597, 2014 (cit. on pp. 16, 18).

[38] B. Yildiz, J. I. Bilbao, J. Dore, and A. B. Sproul, „Recent advances in the analysis of residential electricity consumption and applications of smart meter data," *Applied Energy*, vol. 208, pp. 402–427, Dec. 2017 (cit. on pp. 16, 19, 24, 25, 31).

[39] A. Marinescu, C. Harris, I. Dusparic, S. Clarke, and V. Cahill, „Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods," *2013 2nd International Workshop on Software Engineering Challenges for the Smart Grid, SE4SG 2013 - Proceedings*, pp. 25–32, 2013 (cit. on pp. 16, 18).

[40] V. Bakker, A. Molderink, J. L. Hurink, and G. J. M. Smit, „Domestic heat demand prediction using neural networks," *Proceedings of 19th International Conference on Systems Engineering, ICSEng 2008*, pp. 189–194, 2008 (cit. on pp. 16, 23, 45).

[41] V. Bakker, M. G. C. Bosman, A. Molderink, J. L. Hurink, and G. J. M. Smit, „Improved Heat Demand Prediction of Individual Households," *IFAC Proceedings Volumes*, vol. 43, no. 1, pp. 110–115, Jan. 2010 (cit. on pp. 16, 23, 45).

[42] A. M. Barua and P. K. Goswami, „A survey on electric power consumption prediction techniques," *International Journal of Engineering Research and Technology*, vol. 13, no. 10, pp. 2568–2575, 2020 (cit. on pp. 17–19, 24).

[43] N. Fumo and M. A. Rafe Biswas, „Regression analysis for prediction of residential energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 47, pp. 332–343, 2015 (cit. on p. 17).

[44] H. X. Zhao and F. Magoulès, „A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586–3592, Aug. 2012 (cit. on p. 18).

[45] A. Kapoor and A. Sharma, „A Comparison of Short-Term Load Forecasting Techniques," *International Conference on Innovative Smart Grid Technologies, ISGT Asia 2018*, pp. 1189–1194, Sep. 2018 (cit. on p. 18).

[46] K. Yan, W. Li, Z. Ji, M. Qi, and Y. Du, „A Hybrid LSTM Neural Network for Energy Consumption Forecasting of Individual Households," *IEEE Access*, vol. 7, pp. 157 633–157 642, 2019 (cit. on p. 18).

[47] W. He, „Load Forecasting via Deep Neural Networks," *Procedia Computer Science*, vol. 122, pp. 308–314, 2017 (cit. on pp. 18, 19).

[48] K. Yan, X. Wang, Y. Du, *et al.*, „Multi-step short-term power consumption forecasting with a hybrid deep learning strategy," *Energies*, vol. 11, no. 11, Nov. 2018 (cit. on p. 18).

[49] H. K. Alfares and M. Nazeeruddin, „Electric load forecasting: Literature survey and classification of methods," *Internationa l Journal of Systems Science*, vol. 33, no. 1, pp. 23–34, 2002 (cit. on p. 19).

[50] K. Liu, S. Subbarayan, R. R. Shoults, *et al.*, „Comparison of very short-term load forecasting techniques," *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 877–882, 1996 (cit. on p. 19).

[51] M. R. Kok, „Biologically Realistic Artificial Neural Networks," Bachelor's Thesis, University of Twente, Enschede, Jun. 2020 (cit. on pp. 20, 21).

[52] M. A. Nielsen, *Neural Networks and Deep Learning*, Dec. 2015 (cit. on pp. 21, 22, 67, 68).

[53] G. Sanderson, *Backpropagation calculus | Deep learning, chapter 4*, Nov. 2017 (cit. on p. 21).

[54] R. J. Hyndman and A. B. Koehler, „Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006 (cit. on pp. 24, 25).

[55] K. Abdulla, K. Steer, A. Wirth, and S. Halgamuge, „Improving the on-line control of energy storage via forecast error metric customization," *Journal of Energy Storage*, vol. 8, pp. 51–59, Nov. 2016 (cit. on p. 25).

[56] S. Haben, J. Ward, D. Vukadinovic Greetham, C. Singleton, and P. Grindrod, „A new error measure for forecasts of household-level, high resolution electrical energy consumption," *International Journal of Forecasting*, vol. 30, no. 2, pp. 246–256, Apr. 2014 (cit. on p. 25).

[64] A. Molderink, V. Bakker, J. L. Hurink, and G. J. M. Smit, „Comparing Demand Side Management approaches," *IEEE PES Innovative Smart Grid Technologies Conference Europe*, pp. 1–8, Oct. 2012 (cit. on pp. 30–33).

[65] I. A. M. Varenhorst, „GridShield: Robust Control Algorithms to Prevent Power Outages," M.S. thesis, University of Twente, Enschede, Jun. 2022 (cit. on p. 30).

[66] T. Van Der Klauw, M. E. T. Gerards, G. J. M. Smit, and J. L. Hurink, „Optimal scheduling of electrical vehicle charging under two types of steering signals," *IEEE PES Innovative Smart Grid Technologies Conference Europe*, vol. 2015-January, no. January, Jan. 2015 (cit. on pp. 31, 33, 34).

[67] M. Beaudin and H. Zareipour, „Home energy management systems: A review of modelling and complexity," *Renewable and Sustainable Energy Reviews*, vol. 45, pp. 318–335, 2015 (cit. on p. 31).

[68] H. T. Yang, J. T. Liao, and C. I. Lin, „A load forecasting method for HEMS applications," *2013 IEEE Grenoble Conference PowerTech, POWERTECH 2013*, 2013 (cit. on p. 31).

[69] A. H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, „Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320–331, Dec. 2010 (cit. on p. 32).

[70] R. Tang, S. Wang, and H. Li, „Game theory based interactive demand side management responding to dynamic pricing in price-based demand response of smart grids,“ *Applied Energy*, vol. 250, pp. 118–130, Sep. 2019 (cit. on p. 32).

[71] T. van der Klauw, M. E. T. Gerards, and J. L. Hurink, „Resource allocation problems in decentralized energy management,“ *OR Spectrum*, vol. 39, no. 3, pp. 749–773, Jul. 2017 (cit. on p. 32).

[72] M. H. H. Schoot Uiterkamp, M. E. T. Gerards, and J. L. Hurink, „Fill-level prediction in online valley-filling algorithms for electric vehicle charging,“ *Proceedings - 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2018*, Dec. 2018 (cit. on pp. 33, 34, 105).

[76] K. Manani, *Feature Engineering for Time Series Forecasting*, Jul. 2022 (cit. on pp. 52, 63).

Webpages

[1] Directorate-General for Energy. „Renewable energy targets.“ (), [Online]. Available: https://energy.ec.europa.eu/topics/renewable-energy/renewable-energy-directive-targets-and-rules/renewable-energy-targets_en (visited on Dec. 9, 2022) (cit. on p. 1).

[2] Directorate-General for Climate Action. „2030 climate & energy framework.“ (), [Online]. Available: https://climate.ec.europa.eu/eu-action/climate-strategies-targets/2030-climate-energy-framework_en (visited on Dec. 9, 2022) (cit. on p. 1).

[3] Climate Action Network Europe. „Building a Paris Agreement Compatible (PAC) energy scenario - CAN Europe.“ (Jun. 2020), [Online]. Available: https://caneurope.org/building-a-paris-agreement-compatible-pac-energy-scenario/ (visited on Dec. 9, 2022) (cit. on p. 1).

[6] CBS. „Zonnestroom; vermogen zonnepanelen woningen, wijken en buurten, 2019.“ (Jun. 2021), [Online]. Available: https://opendata.cbs.nl/#/CBS/nl/dataset/85010NED/table?ts=1673357781523 (visited on Jan. 10, 2023) (cit. on pp. 1, 47).

[7] CBS. „Zonnestroom; vermogen zonnepanelen woningen, wijken en buurten, 2020.“ (Dec. 2022), [Online]. Available: https://opendata.cbs.nl/#/CBS/nl/dataset/85447NED/table?ts=1673357634160 (visited on Jan. 10, 2023) (cit. on pp. 1, 47).

[8] European Environment Agency. „New registrations of electric vehicles in Europe.“ (Oct. 2022), [Online]. Available: https://www.eea.europa.eu/ims/new-registrations-of-electric-vehicles (visited on Jan. 7, 2023) (cit. on p. 2).

[9] European Heat Pump Association. „Market data.“ (2022), [Online]. Available: https://www.ehpa.org/market-data/ (visited on Jan. 7, 2023) (cit. on p. 2).

[11] TenneT. „Congestie management.“ (Sep. 2021), [Online]. Available: https://www.tennet.eu/nl/de-elektriciteitsmarkt/congestie-management (visited on Jan. 7, 2023) (cit. on p. 3).

[23] J. Brownlee. „A Gentle Introduction to the Box-Jenkins Method for Time Series Forecasting.“ (Jan. 2017), [Online]. Available: https://machinelearningmastery.com/gentle-introduction-box-jenkins-method-time-series-forecasting/ (visited on Jan. 8, 2023) (cit. on p. 12).

[36] Office for National Statistics. „Energy prices and their effect on households.“ (Feb. 2022), [Online]. Available: https://www.ons.gov.uk/economy/inflationandpriceindices/articles/energypricesandtheireffectonhouseholds/2022-02-01 (visited on Jan. 8, 2023) (cit. on p. 16).

[57] J. Marsh. „How many watts does an electric car charger use? | EnergySage Blog.“ (Mar. 2022), [Online]. Available: https://news.energysage.com/how-many-watts-does-an-electric-car-charger-use/ (visited on Dec. 9, 2022) (cit. on p. 28).

[58] CBS. „De energierekening in januari 2022: hogere leveringstarieven en lagere belastingen.“ (Feb. 2022), [Online]. Available: https://www.cbs.nl/nl-nl/longread/rapportages/2022/de-energierekening-in-januari-2022-hogere-leveringstarieven-en-lagere-belastingen/4-verschillen-tussen-huishoudens (visited on Jan. 11, 2023) (cit. on pp. 28, 29).

[59] Milieu Centraal. „Gemiddeld energieverbruik in Nederland.“ (), [Online]. Available: https://www.milieucentraal.nl/energie-besparen/inzicht-in-je-energierekening/gemiddeld-energieverbruik/#gemiddeld-energieverbruik-verschilt-per-huis-en-aantal-bewoners (visited on Jan. 11, 2023) (cit. on p. 28).

[60] StroomVanDeZon. „1 of 3 fasen groepenkast.“ (), [Online]. Available: https://stroomvandezon.nl/1-fase-of-3-fasen-groepenkast/ (visited on Dec. 9, 2022) (cit. on p. 28).

[61] „Mogelijkheden voor koken op inductie | Koken op inductie.“ (), [Online]. Available: https://www.kookinductie.nl/mogelijkheden/ (visited on Dec. 9, 2022) (cit. on p. 28).

[62] Engie. „1-fase of 3-fase laadpaal | ENGIE.“ (), [Online]. Available: https://www.engie.nl/product-advies/laadpalen/orientatie/1-fase-3-fase-laadpaal#/ (visited on Dec. 9, 2022) (cit. on p. 28).

[63] ElaadNL. „V2G: the Power Recycling Car.“ (2023), [Online]. Available: https://elaad.nl/projecten/v2g-the-power-recycling-car/ (visited on Jan. 7, 2023) (cit. on p. 29).

[73] KNMI. „Open Data API.“ (), [Online]. Available: https://api.dataplatform.knmi.nl/open-data (visited on Oct. 18, 2022) (cit. on p. 45).

[74] „Welcome - KNMI Data Platform - KNMI Data Platform.“ (), [Online]. Available: https://dataplatform.knmi.nl/nl/ (visited on Aug. 23, 2022) (cit. on pp. 45, 46, 101).

[75] „KNMI - Automatische weerstations.“ (), [Online]. Available: https://www.knmi.nl/kennis-en-datacentrum/uitleg/automatische-weerstations (visited on Aug. 23, 2022) (cit. on p. 45).

[77] HSMA. „Bonus Lecture. Time Series Cross Validation.“ (Mar. 2020), [Online]. Available: https://www.youtube.com/watch?v=g9iO2AwTXyI (visited on Jan. 16, 2023) (cit. on p. 64).

[78] L. Jiang. „A Visual Explanation of Gradient Descent Methods (Momentum, AdaGrad, RMSProp, Adam).“ (Jun. 2020), [Online]. Available: https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c (visited on Jan. 16, 2023) (cit. on p. 67).

[79] A. L. Chandra. „Learning Parameters, Part 5: AdaGrad, RMSProp, and Adam.“ (Sep. 2019), [Online]. Available: https://towardsdatascience.com/learning-parameters-part-5-65a2f3583f7d (visited on Jan. 16, 2023) (cit. on p. 67).

[80] V. Bushaev. „Understanding RMSprop — faster neural network learning.“ (Sep. 2018), [Online]. Available: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a (visited on Jan. 16, 2023) (cit. on p. 67).

[81] R. Bhat. „Gradient Descent With Momentum.“ (Oct. 2020), [Online]. Available: https://towardsdatascience.com/gradient-descent-with-momentum-59420f626c8f (visited on Jan. 16, 2023) (cit. on p. 67).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.