KU LEUVEN

Smart meter consumption time-series forecasting

Ir. Stijn Staring

Thesis submitted for the degree of
Master of Science in Artificial
Intelligence, eg

Thesis supervisor:
Prof. dr. ir. Bart De Moor

Assessors:
Prof. dr. ir. Unknown
Prof. dr. ir. Unknown

Mentor:
Ir. Lola Botman

Academic year 2020 – 2021

Preface

I would like to thank my mentor Lola Botman for the interesting meetings and brainstorm sessions we had. Next, I thank my family that is always there to support me. At last, I want to thank everybody that reads this text. Sit back, relax and enjoy.

Ir. Stijn Staring

Contents

Preface i

Abstract iv

List of Figures and Tables v

1 Introduction 1
- 1.1 Importance of topic . . . 1
- 1.2 Problem formulation and link with previous studies . . . 1
- 1.3 Thesis objective and structure . . . 1

2 Data analysis 3
- 2.1 Introduction to dataset . . . 3
- 2.2 Preprocessing . . . 4
- 2.3 Analysis . . . 8
- 2.4 Conclusion . . . 15

3 State of the art short-term residential load forecasting techniques 17
- 3.1 Introduction to Neural Networks . . . 17
- 3.2 Short-Term residential electrical load forecasting . . . 24
- 3.3 Conclusion . . . 30

4 Forecasting the daily electricity consumption 31
- 4.1 Pre-processing . . . 31
- 4.2 Baseline models . . . 33
- 4.3 Neural network models . . . 37
- 4.4 Conclusion . . . 53

5 Model evaluation 55
- 5.1 Model selection . . . 55
- 5.2 Conclusion . . . 56

6 Conclusion 57
- 6.1 Future work . . . 57

A Introduction to the dataset 61
- A.1 Introduction to the dataset . . . 61
- A.2 Missing values . . . 62
- A.3 Daily filter . . . 62

B Forecasting the daily electricity consumption - extra 67

B.1 Baseline models 67
B.2 Parameter Search 68

C Extensions on the evaluation results **75**
C.1 Results on the testset 75
C.2 Old stuff 75
C.3 ARIMA 76

Bibliography **77**

Abstract

The `abstract` environment contains a more extensive overview of the work. But it should be limited to one page.

List of Figures and Tables

List of Figures

2.1 Resulting month of March after substitution of the missing values by the mean value of the measurements. . . . 5
2.2 One of the 9 identified meters with multiple zero daily consumptions . . 6
2.3 The maximum differences between the minimum and maximum weekly rolling averages for all the different time-series. . . . 7
2.4 The seasonality of the electrical load during the week. The blue line shows the average week over all weeks in 2017. . . . 9
2.5 Error between different pairs of weekdays. . . . 10
2.6 Figure with the comparison between holidays and business days. . . . 11
2.7 Error between a holiday and other days of the week. . . . 11
2.8 Relation between normalized daily consumption and daily temperature. 12
2.9 Figure with the comparison of the different dwelling types. . . . 14

3.1 Figure of a MLP (source [17]). . . . 18
3.2 Figure of the logical flow of a vanilla RNN with a hidden state (source: [17]). . . . 19
3.3 Exponential decrease of the gradient size of a simple RNN (red) or a LSTM (blue) (source: [16]). . . . 21
3.4 A LSTM cell that is repeated over time (source: [12]). . . . 22
3.5 A GRU cell that is repeated over time (source: [12]). . . . 23
3.6 Results obtained in paper [14] using the PDRNN method. . . . 26
3.7 Influence of the number of layers and the pooling method used in [14]. . 27
3.8 Different approaches tried in [9] and their averaged performance of 29,808 individual forecasts of half an hour individual loads. . . . 29
3.9 The importance of the different inputs as based on the average class activation score. (source [8]) . . . 30
3.10 Comparison between LSTM and CNN-LSTM. (source: [8]) . . . 30

4.1 The consumption of 2017 for the three selected series. . . . 32
4.2 Daily predictions of two baseline models. (Blue: True / Orange: Prediction) . . . 37
4.3 The generation of inputs for a stateless model. (source: [3]) . . . 39

4.4 The generation of inputs for a stateful model. (source: [3]) 40
4.5 The flow of functions that are executed in order to produce daily forecasts. 41
4.6 Model 1 - stateless model with as input a subserie of N time steps and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^l$, $\hat{y} \in \mathbb{R}^1$. 42
4.7 Model 2 - stateless model with as input a subserie of N time steps and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^l$, $\hat{y} \in \mathbb{R}^1$. 43
4.8 Model 3 - stateful model that connects single LSTM blocks and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^l$, $\hat{y} \in \mathbb{R}^1$. 43
4.9 Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_D*: regularization size of weights DENSE layer, *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM, *d_L*: dropout rate of input states LSTM, *r_d_L*: dropout rate of recurrent states LSTM, *d_D*: dropout rate of DENSE layer, *or*: best performing serie from phase one) 49
4.10 The evaluation of the error on the validation set in function of the learning rate size. 50

A.1 The amount of NaN values in all the 3248 smart meters. 61
A.2 Resulting month of March after substitution of the missing values by the mean value of the measurements. 63
A.3 Resulting month of March after substitution of the missing values by the mean value of the same moment on the next and previous day. 64
A.4 The time-serie with the original maximum difference between the minimum and maximum weekly rolling averages. 64
A.5 The time-serie with the new maximum difference between the minimum and maximum weekly rolling averages. 65
A.6 Figure that shows the seasonality of the electrical load during the day. . 65

B.1 An example histogram of the consumption in [kWh] versus count [-] used during MAPE forecast. 67
B.2 Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_D*: regularization size of weights DENSE layer, *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM, *d_L*: dropout rate of input connections LSTM, *r_d_L*: dropout rate of recurrent connections LSTM, *d_D*: dropout rate of DENSE layer input connections, *or*: best performing serie from phase one) 69
B.3 The evaluation of the error on the validation set in function of the learning rate size. 70
B.4 Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM and *or*: best performing serie from phase one) 72

B.5 The evaluation of the error on the validation set in function of the learning rate size. 73

C.1 Z-scores calculated from the yearly consumptions. 76

List of Tables

2.1 Table with information about the characteristics of the available datasets. 4

4.1 summarizing characteristics about the selected series. 32
4.2 Specifications of different CPU's and GPU tried. 33
4.3 Evaluation results for Serie 1 tested on 31 days of December. 35
4.4 Evaluation results for Serie 2 tested on 12 days of December. 36
4.5 Evaluation results for Serie 3 tested on 12 days of December. 36
4.6 Relative performance over all the 261 time series. 36
4.7 Parameters used during phase 1 for the two stateless models. 45
4.8 Different regularization added during phase 2. 46
4.9 Each value in this table shows the average error when the value of a chosen parameter was used, normalized by the biggest error of the possible values and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst possible value for a chosen parameter for each serie during phase 1 of the parameter search. For example: when 20 LSTM units are chosen, this on average gave a 12.08% lower MAE. 47
4.10 The values of the parameters that performed best for the three time series during phase 1 using model 1 based on the smallest sum of MAE errors during three runs. 48
4.11 Final values obtained after the parameter search for model 1. 50
4.12 Final values obtained after the parameter search for model 2. 51
4.13 Final values obtained after the parameter search for model 2. 53

A.1 Amount of response on the voluntary questionnaires. 62

B.1 Each value in this table shows the average error when the value of a chosen parameter was used, normalized by the biggest error of the possible values and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst possible value for a chosen parameter for each serie during phase 2 of the parameter search. 68
B.2 The values of the parameters that performed best for the three time series during phase 1 using model 2. 68

B.3 Each value in this table shows the average error when the value of a chosen parameter was used, normalized by the biggest error of the possible values and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst possible value for a chosen parameter for each serie during phase 1 of the parameter search. 71
B.4 The values of the parameters that performed best for the three time series during phase 1 using model 3. 71

Chapter 1

Introduction

The first contains a general introduction to the work. The goals are defined and the modus operandi is explained.
- Individual household forecasting is complex because of the high amount of uncertainty that it contains. To deal with this often aggregated signals are forecasted instead. If there are papers that discuss electrical household consumption forecasting, they often use a lot of information about the household which will not be scalable in practise due to privacy concerns. This thesis investigates state of the art time series forecasting techniques based on LSTM neural networks that have as goal to forecast the next day of electrical household consumption, given only limited information.

1.1 Importance of topic

Customer is better informed what the bill is going to be at the end of the month/year. Energy producer can build a better trust with its customer by sending reliable bills. (Providing good service) Producent can better estimate the energy demand of the whole customer population. This will lead to cheaper electricity production because a better planning is possible where there is less need of the more flexible but more expensive electricity installations e.g. diesel engines.

1.2 Problem formulation and link with previous studies

Now going to forecast individual houses, not aggregated signals.

1.3 Thesis objective and structure

The goal of this thesis is to do short-term load forecasting for individual households. A forecast of the electrical load of a household for 24 hours.

Chapter 2

Data analysis

In this chapter details of the dataset are introduced and an analysis is performed. Things discussed about the dataset concern assessing missing data, removing zero days, normalizing the data and removing time-series with identified fundamental changes The analysis looks at the seasonality, influence of temperature, comparing weekdays with weekends, impact of holidays and the driving households characteristics. Finally the definition of a suitable baseline model is given, which will be used during the evaluation with more elaborate models in chapter **??**.

2.1 Introduction to dataset

update pictures The data that is used in this thesis is made available for the IEEE-CIS technical challenge on energy prediction from smart data. It consists out of data from smart meters about the 1/2 hour granulated electricity consumption of 3248 households located in the United Kingdom in the year 2017. The definition of a household are all the people who occupy a single housing unit, regardless of their relationship to one another. Each smart meter collected thus a total of 17520 measurements that are performed by the the leading international energy provider, E.ON UK plc. Not all the 3248 smart meters consist of full data as can be seen in Figure A.1 in appendix A. It can be clearly seen that there are 12 steps in the amount of missing values. This is because the available data ranges from one month (only December) to a full year of data. This acknowledges that customers may have joined at different times during the year. Additionally, missing values are introduced due to errors in sending/receiving from smart meters.
Next to the electricity consumption of the different households, also information is available about the average, minimum and maximum temperature of the day on the location of the smart meter. This data is available at a daily resolution. Also, through voluntary surveys, incomplete information is collected about 2143 smart meters. This concerns e.g. dwelling type, number of occupants, number of bedrooms etc. Table A.1 displays all the attributes in appendix A.

Because of the additional information about the attributes that are summed up in

consumption.csv	
# households	3248
information	electric load
measurements	17520
granularity	$\frac{1}{2}$hour
timespan	year 2017
location	UK

weather.csv	
information	average temperature
	max temperature
	min temperature
granularity	daily
addInfo.csv	
# households	2143

Table 2.1: Table with information about the characteristics of the available datasets.

Table A.1, it can be better understood what kind of households are included in the consumption.csv. It is assumed that all the loads are measured form households of the type listed below and each household is made up of maximum four persons and has a maximum of five bedrooms. industrial loads or small businesses, a bakery for example, are not considered.

- flat
- bungalow
- detached house
- semi-detached house
- terraced house

2.2 Preprocessing

Following steps discuss the preprocessing done on the consumption time-series containing measurements for the entire year.

2.2.1 Missing data

It should be made clear that this section about missing values is only applied during the data analysis. As discussed above the consumption dataset contains additionally to the missing months also missing data due to sending/receiving errors of the smart meters. When this happens the data of the whole day is lost. It should be emphasized that a missing value should not always directly be seen as an error. It can be that the smart meter was put off because the inhabitants were on a holiday for example. The nan values then also gives information about the consumption behaviour, namely that it is possible that the inhabitants go on vacation and the electrical load will in this case normally correspond to a constant base load. However, the assumption is made that in the case of the "consumption.csv" missing data corresponds to a sending/receiving error of the smart meter. This assumption is valid because when full year data is assessed, the missing values always perfectly correspond to a day of missing values. It is therefore highly likely that the organizers of the competition manually deleted days in the consumption to increase

the difficulty of the forecasting and to model sending/receiving errors of the smart meters. That the missing values correspond to sending/receiving errors is also stated in the data description of the competition.

Two methods to impute the missing values are compared. Method one substitutes the missing values of a time-serie by the mean of all the measurements done by the meter. Method two replaces the missing values by the mean consumption value of the same moment on the next and previous day. If the next or previous day is also missing, the closest known day is used. The resulting signals can be seen in Figure A.2 and Figure A.3 in appendix A.

In order to ascertain which method of the two performs the best, a reference dataset is needed in order to compare the estimated with the true values of the missing measurements. From the original dataset which contain 3248 meters it was found that for 181 meters the month March was given without missing data. These 181 complete signals of the month March are used as reference dataset. In order to create the test data in each of the 181 meter signals 7 random days of the month March were removed and estimated by the earlier two methods. The normalized mean square errors, MSE_{AN} and MSE_{mean} given by $\sum_{i=1}^{D} e_i^2$ and normalized by MSE_{mean} are given in Figure 2.1.

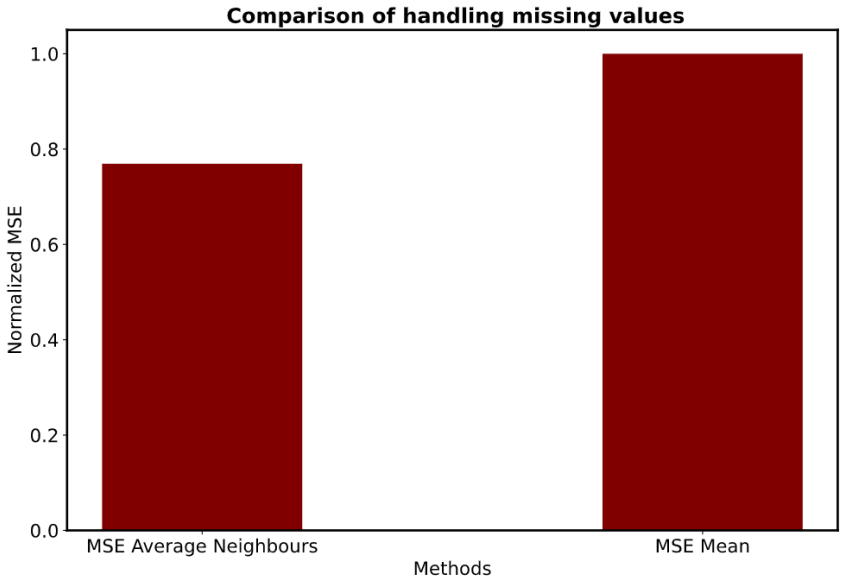


FIGURE 2.1: Resulting month of March after substitution of the missing values by the mean value of the measurements.

From Figure 2.1 it can be seen that using method 2 which estimates the missing values by the mean consumption value of the same moment on the next and previous day, outperforms method 1 which takes the mean of the signal. Therefore, all the missing values in the consumption dataset are estimated using method 2 with the

only exception the first of January and thirty-one December. If one of these two days are missing, the method 1 is used because of the absence of two neighbouring days.

2.2.2 Zero days

When processing the consumption data, some untraditional meter measurements were identified. For example there were 9 meters that had multiple days with zero day consumption measurements. Because it is unlikely that a household produces exactly zero kWh on a day all these 9 meters were removed. The consumption time-serie of one of the meters is displayed in Figure 2.2 in appendix A.

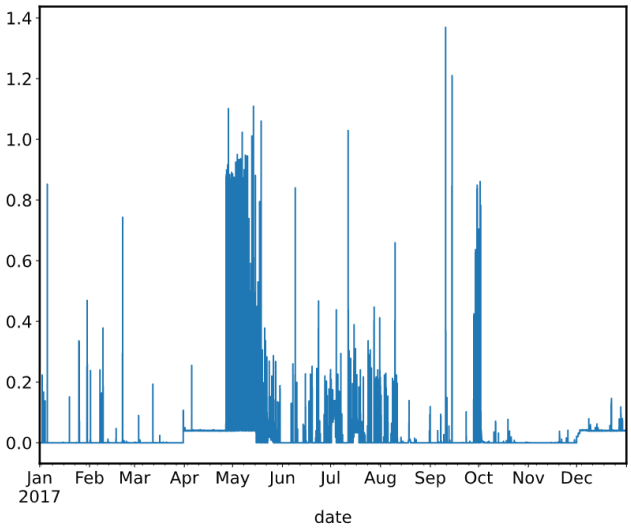


Figure 2.2: One of the 9 identified meters with multiple zero daily consumptions

Also, there has been looked if there were fundamental changes in the electricity consumption of certain meters. This is further discussed in section **??**.

2.2.3 Normalization of the data

it should be made clear that this normalization is only here used. Normalization is necessary because while absolute consumption differs, relative patterns of human behaviour are more similar [10]. The patterns in the human behaviour is what a forecasting model is trying to predict and normalization contributes by avoiding the disturbance of different magnitudes in which this human pattern may occur. Every individual household time-serie is normalized based on its yearly consumption as was done in [10]. The advantage of using the yearly consumption to normalize in comparison of the minimum and maximum values, is the robustness against measurements out shooters and every smart meter has a total consumption

of one at the end of the year

$$normalized \quad value = \frac{consumption_i}{\sum_{n=1}^{17520} consumption_i}. \tag{2.1}$$

As discussed in section 2.3 the average is taken over all the normalized time-series to obtain a single signal.

2.2.4 Removing of fundamental changes in the consumption load

After normalization of all the individual time-series it is looked for fundamental changes in the consumption load due for example when an extra person lives in the house or when systems are installed that use a lot of electricity during the year. An example of such a time-serie can be seen in Figure A.4 in appendix A. These changes are identified by looking at the maximum difference of the minimum and maximum rolling mean consumption over 7 days for each individual meter. If this difference can not anymore be explained by the dependency on the temperature and previous present appliances, it is assumed that a fundamental change in electricity consumption took place. Figure 2.3 shows all the maximum differences between the minimum and maximum weekly rolling averages. The red line on shows the cutoff and the smart meters above this line are defined as outliers and removed. The definition of an outlier that is used is the one and a half times the interquartile range. In total 256 smart meters remain.

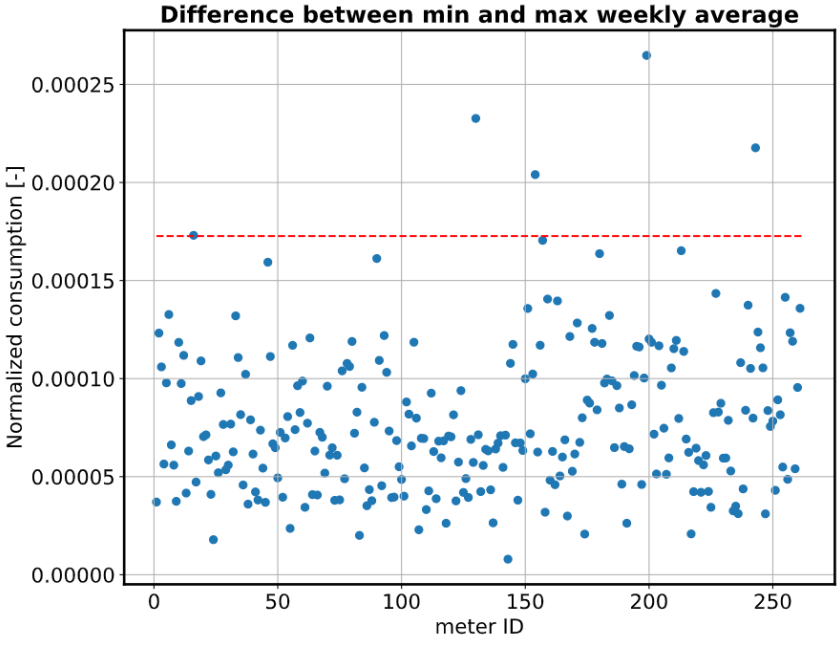


FIGURE 2.3: The maximum differences between the minimum and maximum weekly rolling averages for all the different time-series.

2.3 Analysis

Finally, the average is taken over all the remaining 256 time-series to obtain a single signal. This is done to investigate the dependency of the smart meters on seasonality, temperature, weekends and holidays. At the end of this chapter a baseline forecasting will be discussed that will be used as null-hypothesis in chapter **??** to assess if the developed models lead to an improvement.

2.3.1 Seasonality

In this section the seasonality of the consumption data is discussed. In [6]it was concluded that all the forecasting algorithms that were considered, produced more accurate forecasts when they were combined with a preprocessing stage that extracted the seasonality before forecasting, compared to applying the same algorithms directly on raw data. The forecasting model is left with the task of modelling the deviation from the template consumption instead of performing a forecast out of the blue. However in [6] they made forecasts of an aggregated signal which has a reasonably amount of regularity which is not the case for electrical consumption of individual households. These templates or filters are extracted from the consumption dataset by the use of equations 2.2 and 2.3. D and W gives respectively the number of days and weeks in the dataset. $\bar{y}_i$ and $\bar{y}_j$ gives the consumption of half an hour, averaged over respectively all days and weeks.

$$\bar{y}_i = \frac{1}{D}\sum_{d=1}^{D} y_{di}, \qquad i \in [1, 48], \tag{2.2}$$

$$\bar{y}_j = \frac{1}{W}\sum_{w=1}^{W} y_{wj}, \qquad j \in [1, 336]. \tag{2.3}$$

Figure A.6 shows the daily filter in appendix A. Figure 2.4 shows the weekly filter. In the daily and weekly filters there can clearly be seen a consumption peek after midnight. This is due to heat storage systems that use electricity in the hours of low tariff and that release heat during high electricity tariffs.

2.3.2 Comparing weekdays with weekends

Weekdays vs weekends can be compared with the help of Figure 2.4. The reader is reminded that in order to get this graph, all the remaining household loads after preprocessing are averaged after which all the weeks are again averaged using equation 2.3. It can be seen that the consumption of the average business day is similar to a weekend day concerning the two main peaks during the day (7 am and 6 pm) and the sharp peak at midnight. However, it can be seen that the first peak during the day is higher and goes less down again during the weekend. This effect can be seen both during a Sunday and Saterday, but is most visible during a sunday. To proof previous statements similarity is measured by calculating the hourly difference of the 21 combinations that can be made of two different days. Figure **??** shows in blue

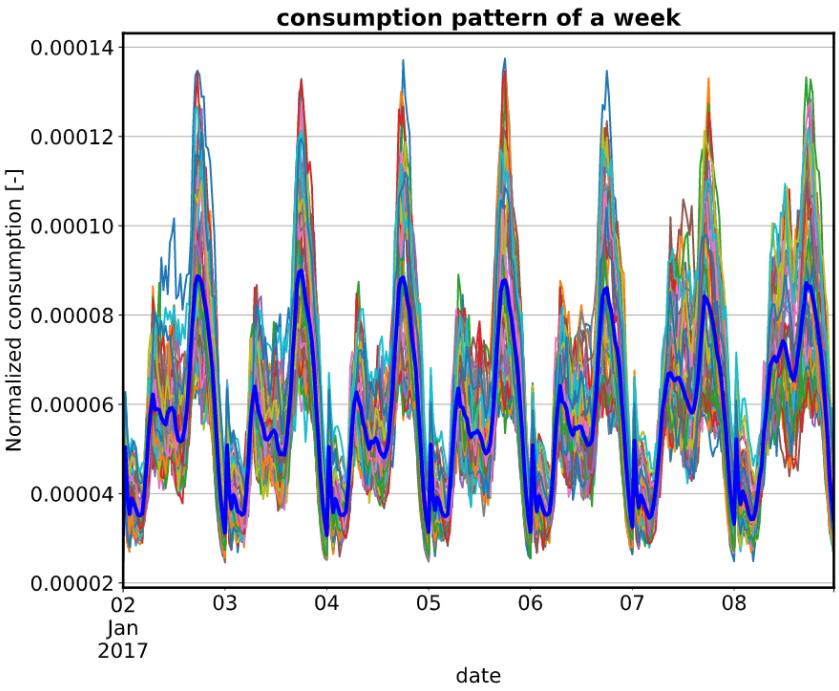


FIGURE 2.4: The seasonality of the electrical load during the week. The blue line shows the average week over all weeks in 2017.

and orange the error of combinations between business days or weekend days and in green the error of combinations between a business day and weekend day. The error value is calculated by summing the hourly errors between two days. It can be clearly seen that when a business day and weekend day are combined the error (green) is bigger and thus similarity smaller. Another thing that can be noticed is that the left cluster of green dots corresponds to a Saterday and the right to a Sunday. It can be noticed that Saterdays are more similar to a business day than a Sunday.

2.3.3 Impact of holidays

In order to look at the impact of a holiday, all the holidays of the English and welsh holiday calendar are identified for the year 2017. For each of the 8 holidays a corresponding business day is selected with an as close as possible average temperature of the day. This is done to mitigate the temperature dependency. The resulting average holiday and business day is given in Figure 2.6. A holiday behaves similar to a weekend day with the first peak load going higher and goes less down over time. Figure 2.7 shows that a holiday behaves the most similar to a Sunday .

It can be seen that the consumption of a holiday behaves similar as a weekend day. Figure shows the average error between a holiday vs business day and a holiday vs weekend day. The error is calculated as discussed in section 2.3.2.

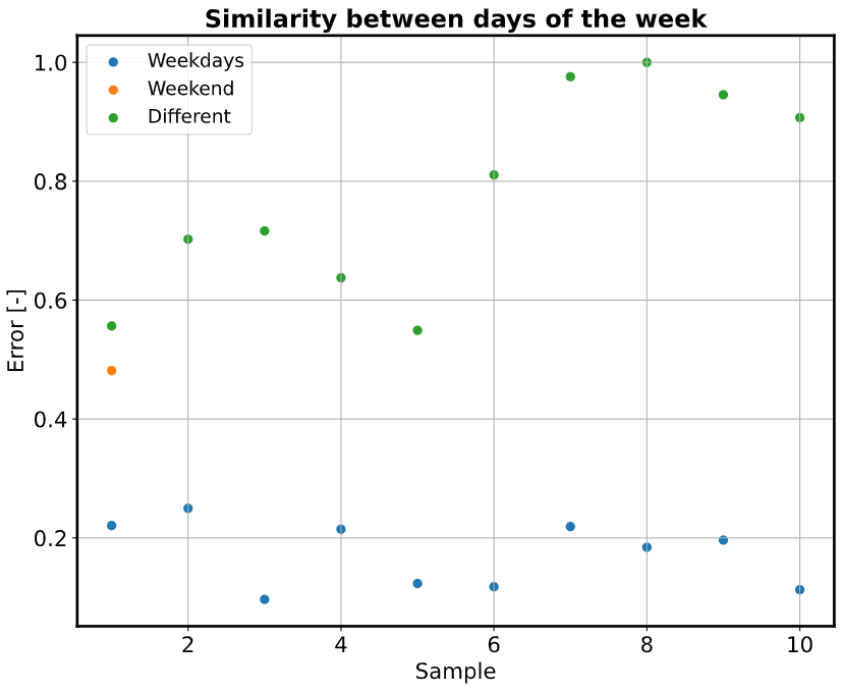


Figure 2.5: Error between different pairs of weekdays.

2.3.4 Influence of temperature

In following section the correlation between the temperature and the electricity consumption is discussed.

Pearson correlation

The Pearson correlation is a measurement of the linear dependency between two variables which is based on the covariance variable. A Pearson correlation value gives information concerning the magnitude of the association and the corresponding direction of it. A Pearson value of one and minus one give respectively a perfect positive and negative linear relation between the variables. A value of zero, corresponds to independent behaviour. Following formula is used when calculating the Pearson correlation

$$\rho_{X,Y} = \frac{\sigma_{x,y}}{\sigma_x \sigma_y}. \tag{2.4}$$

Assumptions concerning Pearson correlation are that samples used for the correlation should be independent drawn, come in pairs, follow homoscedasticity and there are no outliers. Outliers are especially undesirable when there are not a lot of samples. The variables should be normal distributed, linear related to each other and be continuous.

The samples used for the correlation are generated by calculating the daily consumptions matched with the daily average temperature. In this case the above assumptions

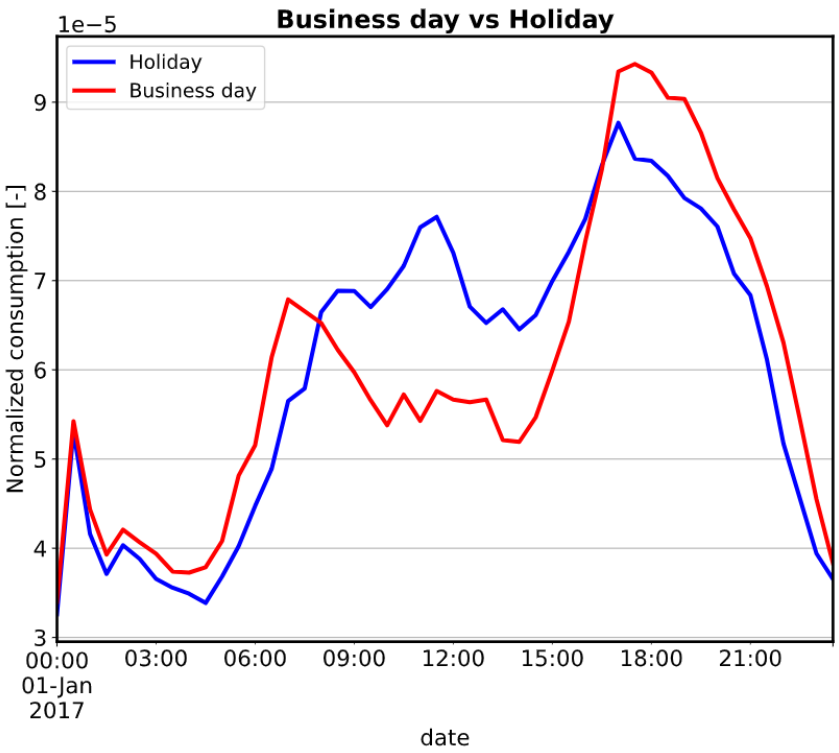


FIGURE 2.6: Figure with the comparison between holidays and business days.

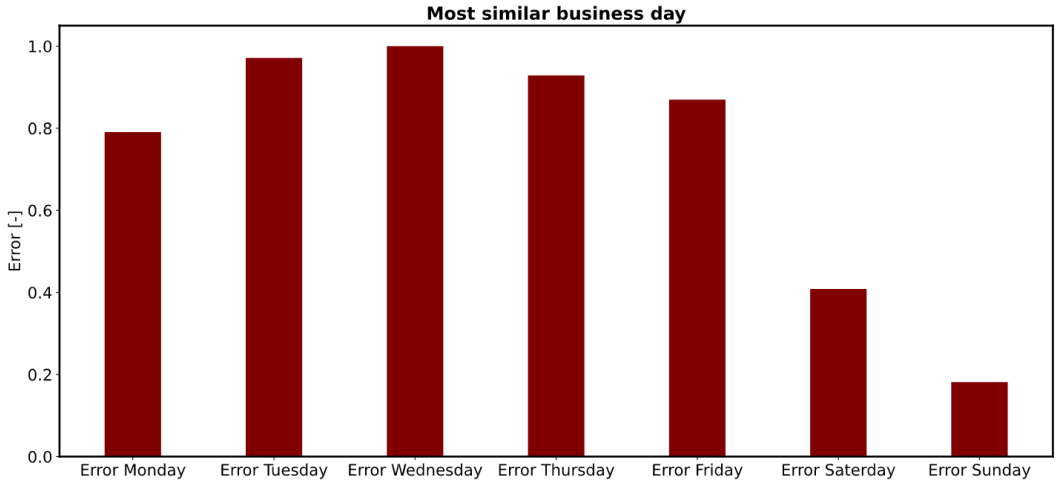


FIGURE 2.7: Error between a holiday and other days of the week.

are thus not valid. Homoscedasticity is important when performing linear regression and assumes that σ_x and σ_y are constant. This assumption is validated by making use of Figure 2.8.

This figure shows the classic cone-shaped pattern of heteroscedasticity. On days

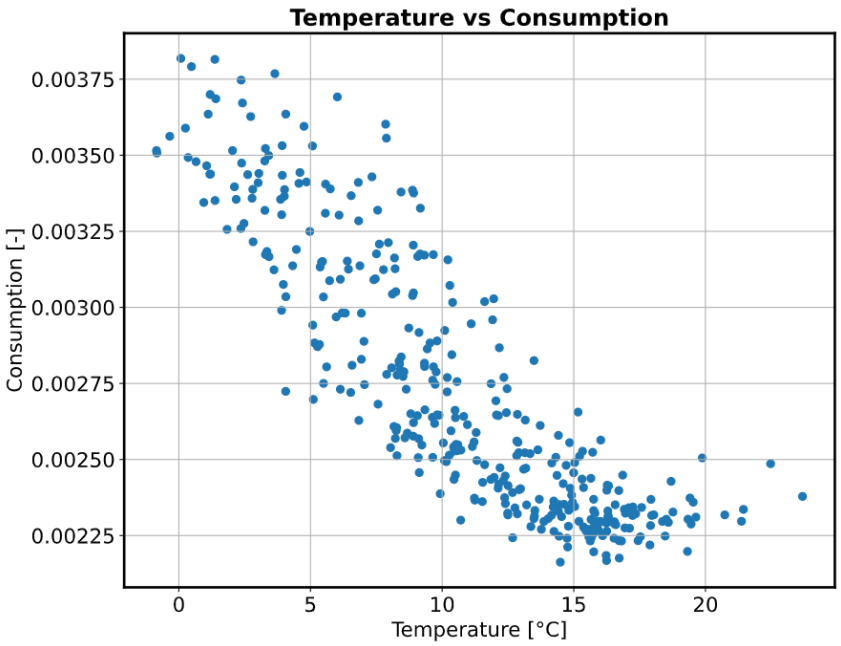


Figure 2.8: Relation between normalized daily consumption and daily temperature.

when it is warm there is overall similar human behaviour in lowering the electricity consumption. However, on colder days the variation in consumption is higher, which means that homoscedasticity is not fulfilled. Because the assumptions of the Pearson correlation are not fulfilled, care should be taken with its output.

Applying the Pearson correlation on Figure 2.8 gives a correlation value of -0.87. This means there is a reasonable linearly decreasing relation.

Spearman correlation

Spearman correlation is a "Rank correlation". This means that the ordering of the consumption and temperature in a sample are each compared in their corresponding array of measurements. When the ordering of both variables in a sample are similar, correlation is strong and positive. If the ordering is reversed, correlation is strong and negative. There is a perfect positive ordering if larger consumption always corresponds to a higher temperature. Notice that for a perfect ordering, no linear relation of the variables is necessary. The Spearman correlation coefficient is calculated using equation 2.4, but takes into account the rank of a variable in all the measurements of this variable instead of the measurement value itself.

In order to use the spearman correlation data has to be ordinal, which means that it can be ordered. The spearman correlation gives information about the monotonicity relation between the variables. $\rho = 1$ corresponds to a monotonically increasing relation.

Applying the Spearman correlation gives a correlation value of -0.89, which means there is a good negative monotone relation. This means if the temperature is higher, consumption is likely to be lower. Identically, if the temperature is lower it is likely that the consumption will be higher.

Kendal correlation The "Kendal correlation" is also a rank based correlation. Here it is looked at the pairs of observation that are concordant, discordant or neither. A correlation coefficient close to one occurs when both variables have the same ranking and similar a coefficient close to minus one occurs when rankings in one variable are the reverse of the other. Equation 2.5 gives the equation to calculate the "Kendal correlation coefficient".

$$\tau = \frac{n^+ - n^-}{\sqrt{(n^+ + n^- + n^x)(n^+ + n^- + n^y)}} \tag{2.5}$$

- n^+ is the number of concordant pairs
- n^- is the number of discordant pairs
- n^x is the number of ties only in x
- n^y is the number of ties only in y
- concordant $\rightarrow (x_i > x_j)$ and $(y_i > y_j)$ or $(x_i < x_j)$ and $(y_i < y_j)$
- discordant $\rightarrow (x_i > x_j)$ and $(y_i < y_j)$ or $(x_i < x_j)$ and $(y_i > y_j)$
- neither $\rightarrow (x_i = x_j)$ or $(y_i = y_j)$
- if both $(x_i = x_j)$ and $(y_i = y_j) \rightarrow$ not included in either n^x or n^y

Applying the Kendal correlation gives a correlation value of -0.67, which means there is a reasonable negative monotonicity relation.

2.3.5 Identification of driving attributes

In this section the influence of the extra knowledge about the kind of household where the smart meter is located, is investigated. This is not done by using a single averaged signal as was the case in the previous analysis sections. Now, every meter with additional information is considered. In Figure 2.9 the monthly consumption of the month December in function of dwelling type is shown. The month December is chosen, because this month is known for every smart meter. Missing values of the smart meters are substituted by method two, as discussed in section 2.2.1. The amount of meters used for every visualization can be seen in Table A.1.
Similar as was done in Figure 2.9 is also done for the other characteristics of the smart meters. The conclusions are listed below. As can be seen in Table A.1, some

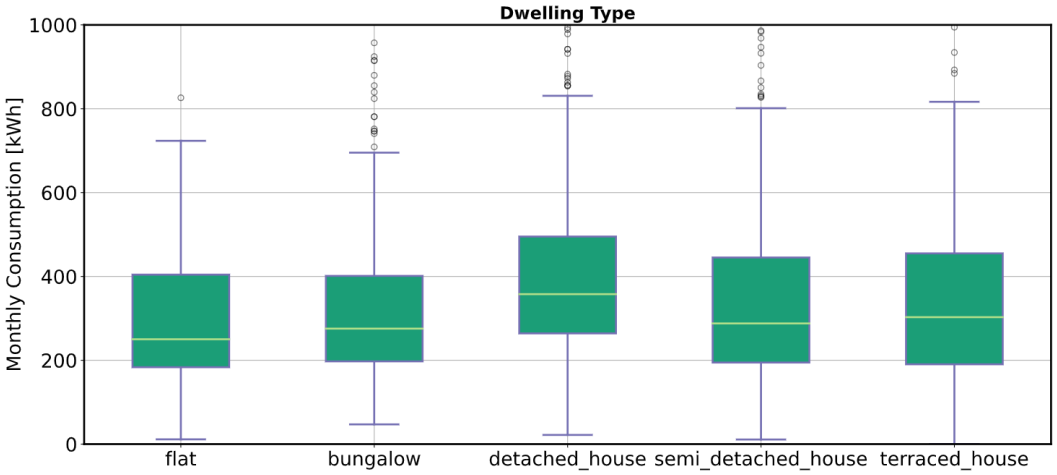


Figure 2.9: Figure with the comparison of the different dwelling types.

characteristics have not much data or the data is not much distribute over the different options of a characteristic. If this is the case, no reliable conclusions could be drawn. **Add concrete numbers!!**

- There is a lot of variance in the monthly consumption of a detached house, but it has mostly a higher consumption than other dwelling types
- A "real" house (detached, semi-detached or terraced) tends to have higher monthly consumptions than a flat or bungalow.
- The order of monthly consumption according to the mean and median values: Flat < Bungalow < Semi-detached < Terraced < Detached
- More occupants means more monthly consumption
- More rooms in the house means more monthly consumption
- Almost all houses use gas as heating fuel
- Almost all houses use gas as hot water fuel
- The age of the boiler has no clear effect on the monthly consumption
- The vast majority of the lofts are insulated
- The majority of walls are insulated
- The vast majority heats till a temperature between 18 and 20 degrees
- The majority of people has an efficient lighting percentage between 75% and 100%

2.4 Conclusion

The final section of the chapter gives an overview of the important results of this chapter. This implies that the introductory chapter and the concluding chapter don't need a conclusion.

Chapter 3

State of the art short-term residential load forecasting techniques

Forecasting the electrical load of the different individual households has a couple of challenges. There should be dealt with the missing values, as discussed in section 2.2.1. Also, the different time-series are influenced by exogenous factors as weather conditions and the day of the year. The dependency on exogenous variables can be a very non-linear relation and can have different effects on different households. For example depending on a house has solar panels, the consumption could be altered much. Only three indications of the temperature are given on a daily basis. Some additional information is know of certain households, but this data is very incomplete. Next, the individual load series have a high volatility and uncertainty with respect to a load signal on transmission level which shows more consistent seasonality and straight forward dependency on weather and calendar variables. This is because the contingency of the individual load data is mitigated due to averaging out of the uncertainty. Ofcourse, the obvious disadvantage is that only forecasts on this aggregated level can be made which is not the goal of our investigation.
To tackle the high non-linearity that is inherent to residential load forecasting in literature often "Neural Networks" are used. **See also paper TA2 –> aggregated vs individual forecasting.**

3.1 Introduction to Neural Networks

A standard multilayer feedforward neuralnetwork with locally bounded piecewise continuous activation function can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial, as stated by **Leshno et al** in **1993**. This theorem proofs that a "universal approximator" exists for continuous functions, but it lacks the recipe to construct it. In [11] it is shown that a feedforward network with a single layer is enough to approximate any function by a specified accuracy if the hidden layer has the possibility to add an

unlimited amount of hidden neurons in its layer. It is discussed that when a function is discontinuous, which means that it makes sudden, sharps jumps, it is not possible to approximate the function by any prescribed accuracy. However, in practise a continuous approximation is often good enough.

Neural networks are suitable of learning very non-linear mappings between inputs and outputs. The difference between "Deep Neural Networks" and "Shallow Neural Networks" is the amount of layers of neurons are used inside the network. These layers of neurons, that are not inputs or outputs are called "hidden neurons". Because a "Deep Neural Network" has a hierarchical layout of the different hidden layers, it not only learning features from the non-linear combinations of inputs, but uses other layers to learn features of combinations of features learned in lower hidden layers. This is possible because higher hidden layers get the outputs of lower hidden layers as input. As discussed in [14] due to this characteristic, deep learning is suitable to learn multiple uncertainties with differing sharing levels over different households e.g. the amount of sunshine. However, because of the higher expressiveness (and often the amount of the to learn parameters), a "Deep Neural Network" with respect to a "Shallow Neural Network", suffers more of overfitting as is discussed in section 3.1.4.

3.1.1 MLP

The simples configuration of deep networks are multilayer perceptrons and they are made up out of multiple fully connected layers of neurons. Figure 3.1 shows a MLP with one hidden layer.

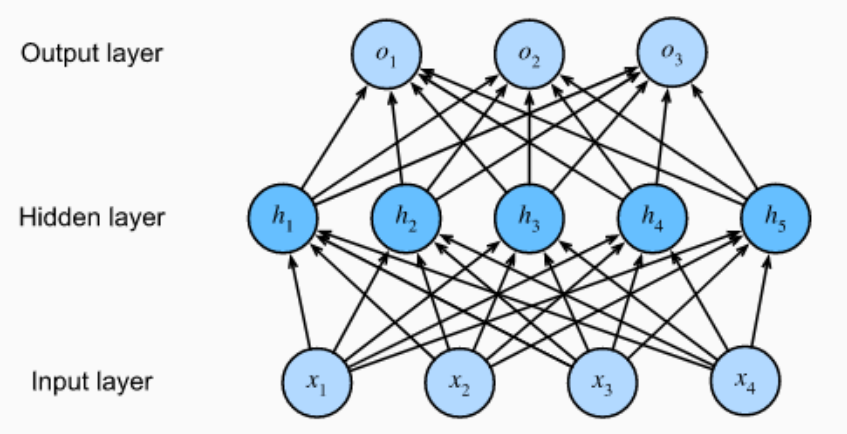


Figure 3.1: Figure of a MLP (source [17]).

All layers are connected to the next layer by the means of an affine function together with a non-linear activation function represented by sigma as shown by equation 3.1 with $\mathbf{L}^{(N)}$ the vector with outputs of the Nthe layer, $\mathbf{W}^(N)$ the Nth weight matrix and $\mathbf{b}^{(N)}$ the Nth bias

$$\mathbf{L}^{N+1} = \sigma(\mathbf{W}^{(N)}\mathbf{L}^{N} + \mathbf{b}^{(N)}). \tag{3.1}$$

3.1.2 CNN

See oneNote

3.1.3 RNN

A recurrent Neural Network is a specialized neural network to deal with sequential information. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. In order to take past information from previous inputs into account, a hidden variable h_t is used. By making use of this variable which makes a summary of the previous seen information, an exponential increase in the number of model parameters is avoided. Cited from [**?**]: "Hidden states are technically speaking inputs to whatever we do at a given step, and they can only be computed by looking at data at previous time steps". Equation 3.2 shows how the previous hidden state and the current information are merged in the next hidden state with $\mathbf{X}^t \in \mathbb{R}^{d\times 1}$, $\mathbf{H}^t \in \mathbb{R}^{h\times 1}$, $\mathbf{W}_1 \in \mathbb{R}^{h\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{h\times h}$ and $\mathbf{b} \in \mathbb{R}^{h\times 1}$

$$\mathbf{H}^{t+1} = \tanh\Big(\mathbf{W}_1\mathbf{X}^t + \mathbf{W}_2\mathbf{H}^t + \mathbf{b}\Big). \tag{3.2}$$

The equation $\mathbf{X}^t$ corresponds to one example at time step t with dimensionality d. Also a deep RNN is possible, where multiple hidden state per time step are used.

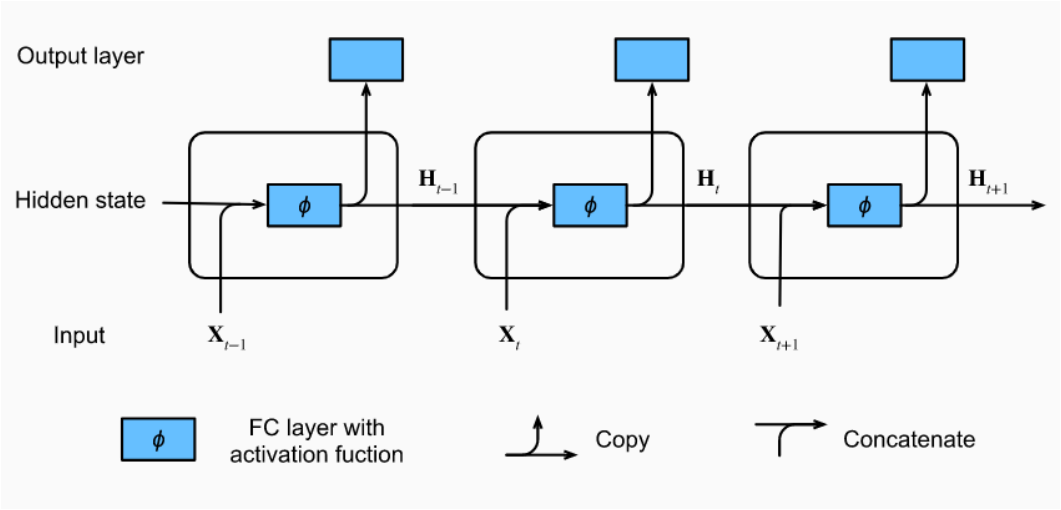


FIGURE 3.2: Figure of the logical flow of a vanilla RNN with a hidden state (source: [17]).

As was discussed in Section 3.1 a standard neural network can act as a "universal approximator" when given enough hidden states. A similar result exist for a recurrent neural network which states that it is capable to approximate a sequence-to-sequence mapping to an arbitrary accuracy as discussed in [5]. However, as discussed in [16]

even if expressiveness of the simple model is very powerful in theory, this doesn't indicate that such a representation can be learn in a reasonably amount of time from a dataset. As will be discussed in Section 3.1.4, the main drawback of the vanilla recurrent neural network is that it forgets fast, important information in function of the amount of time steps. When using "backpropagation through time" for updating the weights, the gradients that corresponds to inputs seen a lot of time steps ago will become very small due to the multiplication of small gradients over the time steps. Therefore, there contribution of updating the weights of the recurrent neural network will be very small and thus this information will be "forgotten".

3.1.4 Difficulties & Solutions of neural networks

Neural Networks have a high expressiveness but comes at the cost of overfitting and a vanishing gradient. when the NN is learning from training data, every epoch the error between the input and output of the training exemples is reduced. In the beginning the generalization error reduces simultaneously with the generalization error. The generalization error is the error that the model makes on data that is not in the training set. However, on a certain point during the training the generalization error increases while the training error still decreases. This means that the model is no longer learning "intelligent" general rules and patterns in the data, but is just remembering the training data and will therefore not apply in general. This is often the case in a model with high expressiveness because the model is less pushed to make generalizations and has the ability to just to remember the training data. Solutions to overfitting can be regularization which includes the parameter norms as a cost in the objective function. Typical choices for resembling the size of a parameter are the L_1 and the L_2 norms. Other methods that can be used are: early stopping, dropout and pruning.
It should also be noted that the gradient can increase very much over the different time steps, which in literature is called gradient explosion. The solution strategy for this is applying gradient clipping by norm or by value. Gradient clipping by norm means that when the two norm of the gradient $\boldsymbol{\xi}$ exceeds a threshold value θ, the two norm of the gradient is scaled to equal the threshold value. The mathematical formulation is given by equation 3.3:

$$\boldsymbol{\xi} = min(1, \frac{\theta}{\|\boldsymbol{\xi}\|}) \times \boldsymbol{\xi}. \tag{3.3}$$

An alternative method to avoid gradient explosion is using gradient value clipping. The second problem is the vanishing gradient problem which originates because while using the backpropagation algorithm to calculate the gradient which is used in different update methods of the weights, the gradient is calculated at the end of the NN and propagated back using every time the previous calculated gradient values which exponentially decreases in function of the time steps. Therefore, at the first layers of the network, the gradient has become so small that the weights are almost not updated anymore. In a RNN setting this corresponds to having a short term memory which means that initial inputs that were presented to the NN

are being forgotten. Mitigation strategies often proposed in literature are LSTM and GRU. Both techniques have in common that they can learn which data in the sequence is important and should be retained and which information can be thrown away. It is important to state that LSTM and GRU are not solving the vanishing gradient problem as explained in [16]. The gradient is still exponentially decreasing, but the effect is less pronounced as can be seen for LSTM in Figure 3.3. When the forget gate, that sits inside a LSTM cell outputs a value that is close to one, the exponential decay will have also a base close to one. τ gives the number of epochs. Also, the complexity of the recurrent models grows linearly with the amount of time steps that are processed in the sequence. As discussed in [16], the amount of memory and calculation effort needed to do a gradient update also increases linearly with the amount of time steps. Memory and calculation load can be mitigated by making use of "truncated backpropagation through time".
Can put further explanation in attachment –> see assignament ANN.

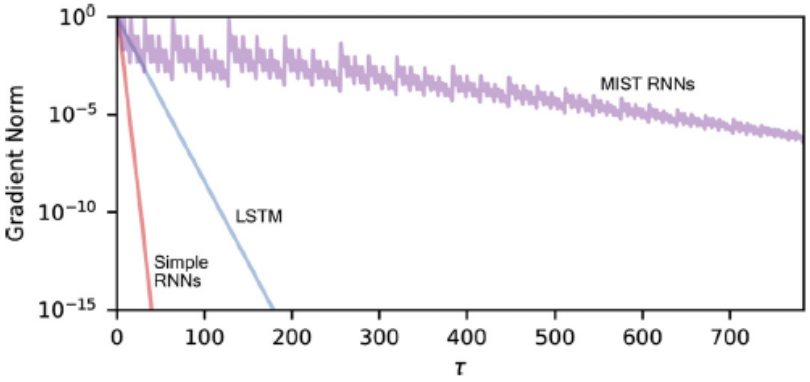


FIGURE 3.3: Exponential decrease of the gradient size of a simple RNN (red) or a LSTM (blue) (source: [16]).

3.1.5 LSTM

As discussed in Section 3.1.4 the LSTM is an updated version of the conventional RNN first proposed by **Hochreiter & Schmidhuber** in 1997 to deal with the short term memory it suffers from. A LSTM can longer take important aspects of the presented time series into account while outputting a current prediction. To do this a LSTM makes use of three gates: forget gate $\mathbf{f}_t$, input gate $\mathbf{i}_t$ and an output gate $\mathbf{o}_t$. When comparing the three gates with equation 3.2, it is clear that every gate is by itself a recurrent neural network, with the only difference that a sigmoid function is used instead of a hyperbolic tangent. The core concept of the LSTM is that it makes use of a memory cell that is passed on through the different time steps. The memory cell contains important information that is seen before in the data and should be taken into account at the current new output. The three gates can delete, write and read information from this memory cell. It can also be noted that equation 3.7

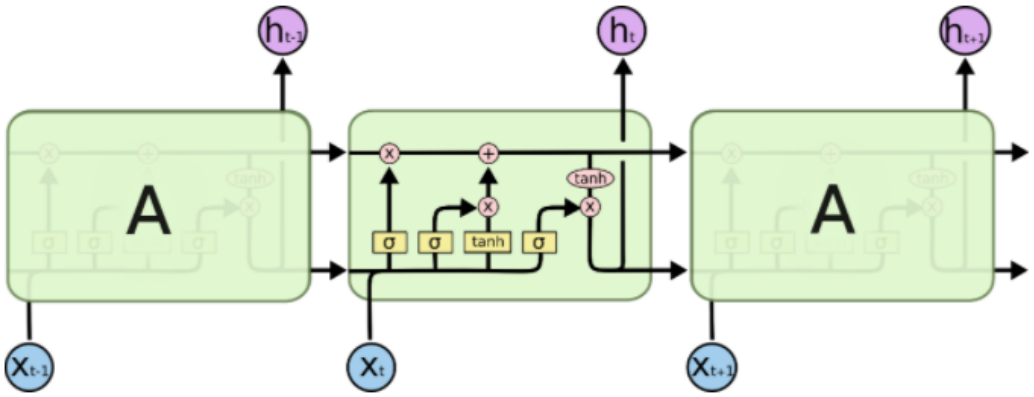


Figure 3.4: A LSTM cell that is repeated over time (source: [12]).

is exactly equal to the conventional RNN described by equation 3.2. Equation 3.7 processes the hidden states H_t and the new input X_t to propose an update $\tilde{\mathbf{c}}_t$ to the previous memory cell. The input gate (Eq. 3.5) decides what will be preserved of the proposal and actually updated. The forget gate (Eq. 3.4) decides what will be preserved from the original memory cell $\mathbf{c}_t$. When both the old memory cell and the proposal are pruned, they are combined to one new memory cell. This new memory cell is together with the output gate (Eq. 3.6) used to output new hidden states.

In order to train a LSTM neural network there are considerably more parameters that have to be learned. There are now four different weight matrices for both the hidden states and the inputs. Because by this increase of weights also the expressiveness of the model has increased with respect to the vanilla recurrent neural network of Section 3.1.3. Therefore, overfitting of the data should be extra monitored. Further, it can also be noted when looking to the LSTM equations that when the parameter that determines the amount of hidden states this will have a higher effect on the calculation load of a LSTM than the vanilla RNN. This is similar when more inputs are added.
The LSTM equations are given as follows as they were found in [16]:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fH}\mathbf{H}_{t-1} + \mathbf{W}_{fX}\mathbf{X}_{t-1} + \mathbf{b}_f), \tag{3.4}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{iH}\mathbf{H}_{t-1} + \mathbf{W}_{iX}\mathbf{X}_{t-1} + \mathbf{b}_i), \tag{3.5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{oH}\mathbf{H}_{t-1} + \mathbf{W}_{oX}\mathbf{X}_{t-1} + \mathbf{b}_o), \tag{3.6}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{cH}\mathbf{H}_{t-1} + \mathbf{W}_{cX}\mathbf{X}_{t-1} + \mathbf{b}_c), \tag{3.7}$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{t-1} + \mathbf{i}_t \times \tilde{\mathbf{c}}_t, \tag{3.8}$$

$$\mathbf{H}_t = \mathbf{o}_t \times \tanh(\mathbf{c}_t). \tag{3.9}$$

3.1.6 GRU

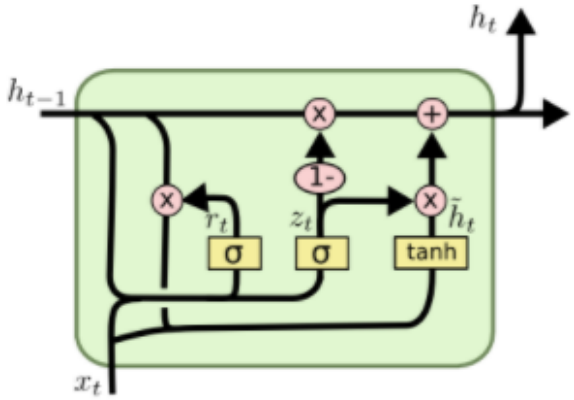


FIGURE 3.5: A GRU cell that is repeated over time (source: [12]).

A gated recurrent unit neural network is a newer, simplified version of the LSTM that deals with the short term memory problem of a vanilla recurrent neural network. It was introduced by **Cho et al.** in 2014. The LSTM is changed by merging the forget and input gate into an update gate. Also, the seperate memory cell and hidden states are combined. The different performance between the variations of the LSTM neural network are discussed in Section 3.1.7. The GRU equations are given as follows as was found in [16]:

$$\mathbf{z}_t = \sigma(\mathbf{W}_{zH}\mathbf{H}_{t-1} + \mathbf{W}_{zX}\mathbf{X}_{t-1} + \mathbf{b}_z), \tag{3.10}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{rH}\mathbf{H}_{t-1} + \mathbf{W}_{rX}\mathbf{X}_{t-1} + \mathbf{b}_r), \tag{3.11}$$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{W}_{HH}(\mathbf{r}_t \times \mathbf{H}_{t-1}) + \mathbf{W}_{HX}\mathbf{X}_t + \mathbf{b}_H), \tag{3.12}$$

$$\mathbf{H}_t = \mathbf{z}_t \times \mathbf{H}_{t-1} + (1 - \mathbf{z}_t) \times \tilde{\mathbf{H}}_t. \tag{3.13}$$

3.1.7 Performance of different parameter settings and variations of LSTM neural network models from literature

Paper [1] conducts an empirical evaluation of the GRU and compares it with the older LSTM. It was found that it outperformed the vanilla RNN and attained similar performance as the LSTM on the task of polyphonic music modeling and speech signal modeling. According to [12], the next step in sequence modelling can be the use of attention models or grid LSTM's.

There exists a lot of variations of the LSTM neural networks. Paper [4] discusses a large-scale analysis of eight LSTM variants on the tasks of: speech recognition, handwritting recognition and polyphonic music modelling. The hyperparameters of the models were optimized using a random search method. The influence of each of the hyperparameters was assessed using the fANOVA toolbox. This toolbox is explained in [7] It was found that none of the assessed variants of the conventional LSTM architecture could significantly outperform the latter. However, it was stated that the LSTM variants in some occasions were able to simpify the LSTM and its calculation load and number of parameters, without the lost of performance.
Next, it was found that the forget and output gate were the most crucial gates of the LSTM network. When one of the two was removed, a significant lost of performance occurred. There was also an hyper parameter search conducted with following hyperparameters:

- amount of LSTM hidden states
- learning rate
- momentum term
- standard deviation of Gaussian input noise

It was concluded that it could be assumed that there was no interaction between the different parameters.The largest interaction could be found between the learning rate and the size of the network which was still small. Therefore, parameters can be varied individually which can drastically reduce the amount of runs that had to be performed to see the effect on the model. Next, it was concluded that the learning rate was the most important parameter and could be tuned by setting it high e.g. equal to one, after which the size was decreased using a early stopping approach. This is means that decreasing was stopped when performance was also decreases. The use of a momentum term, which takes previous values of the weights into account when updating, was found to be unimportant in their setting of online gradient descent. The use of gaussian input noise to avoid overfitting was found to be not helpful.

3.2 Short-Term residential electrical load forecasting

Pooling paper Classical ways to deal with uncertainty.
Residential electrical load series have a high amount of volatility and uncertainty due to the contingency of the electrical consumption. Classical ways to deal with this are discussed in [14] and listed as follows:

1. Clustering to group similar houses based on historic load or exogenous consumption driving variables. Because the load or driving variables are similar in a cluster, the variance of uncertainty is also decreased. However, performance is very dependent of the dataset. **But the uncertainty on the whole is reduced −> on single household stays the same!!**

2. Aggregating the residential loads to cancel out the uncertainties. The aggregated signal will show more regular patterns which means that is easier to predict.The downside is that the aggregated forecast will do a poor job of serving as forecast for a household

3. A spectral analysis e.g. wavelet analysis, Fourier transforms and empirical mode decomposition aim at seperating a load serie into a regular pattern, an uncertain signal and noise. Because the amount of regularity is low in a residential load serie, this method is infeasible.

In this paper [14] a novel pooling-based deep recurrent neural network is proposed which collects load profiles of neighbouring houses into a pool of training inputs. Pooling of neighbouring households historical loads to serve as input of the "Deep Recurrent Neural Network", is proposed to increase the data volume and diversity of load forecasting, which mitigates the effect of overfitting present in a DRNN. The idea is as quoted by [14] to use the interconnected spacial information to compensate insufficient temporal information.Thereby, the pool of data allows to learn the correlations between neighbouring households and the shared uncertainties coming from external factors e.g. temperature. Also, due to the pooling of different households during training the DRNN is able to learn common uncertainties. In paper [14] pools consisting of 10 households are used. From the pool of inputs every epoch a randomly chosen batch of load signals are fed to the network. LSTM is applied to mittigate the short term memory of the RNN. Additionally, there is been made use of early stopping to further avoid overfitting. To implement early stopping there has been looked at the "MSE" for k iterations, obtained by cross-validation. When the variance of this sequence gets smaller than a specified variable, training stops. When the training ends, performance is tested on each household by using the learned network to perform a feed-forward prediction of the electrical load.

An overview of the different steps that were done during the proposed method are: data cleaning and preprocessing $\rightarrow$ data pooling $\rightarrow$ data sampling $\rightarrow$ data training $\rightarrow$ benchmarking.

Performance of the proposed method was finally evaluated based on a test set of the last 30 days and consisting out of :

1. performance of the proposed method with respect to Vanilla RNN, SVR and DRNN (without pooling)

2. the effect of the neural network depth and pooling

The proposed DRNN with pooling outperforms all other four methods based on following three metrics:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}{N}} \tag{3.14}$$

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \tag{3.15}$$

$$MAE = \frac{\sum_{t=1}^{N} |\hat{y}_t - y_t|}{N} \tag{3.16}$$

Actually LSTM network The amount of which the PDRNN outperformed the other methods can be seen in Table **??**. The effect of the depth of the DRNN and the pooling method is depicted in Figure 3.7. It can be seen that without the pooling method the DRNN only benefits from extra layers till three are used. This is because from that point, overfitting will reduce the generalization capacity of the DRNN. With the pooling technique, extra layers stays beneficial. It can thus be concluded that introducing extra hidden layers is a good choice to model the non-linear relations, but this can only be done efficiently when overfitting is mittigated by the use of a pooling strategy.The RNN with pooling used for benchmarking consisted out of five layers and thirty hidden units in each layer.

Network Architecture	*RMSE (kWh)*	*NRMSE (kWh)*	*MAE (kWh)*
ARIMA	0.5593	0.1132	0.2998
RNN	0.5280	0.1076	0.2913
SVR	0.5180	0.1048	0.2855
DRNN	0.4815	0.0974	0.2698
PDRNN	0.4505	0.0912	0.2510
Improvement from DRNN to PDRNN	6.45%		6.96%
Improvement from ARIMA to PDRNN	19.46%		16.28%

Figure 3.6: Results obtained in paper [14] using the PDRNN method.

GRU (Gated Reset Update) or LSTM (Long Short Term Memory) can be implemented. They are both enhancements of the vanilla RNN which suffers from a vanishing gradient which causes it to to behave without a long term memory. In practise to know which one works often both are tried [16]. Stochastic gradient descent means that the approximated gradient is calculated from a random subset of

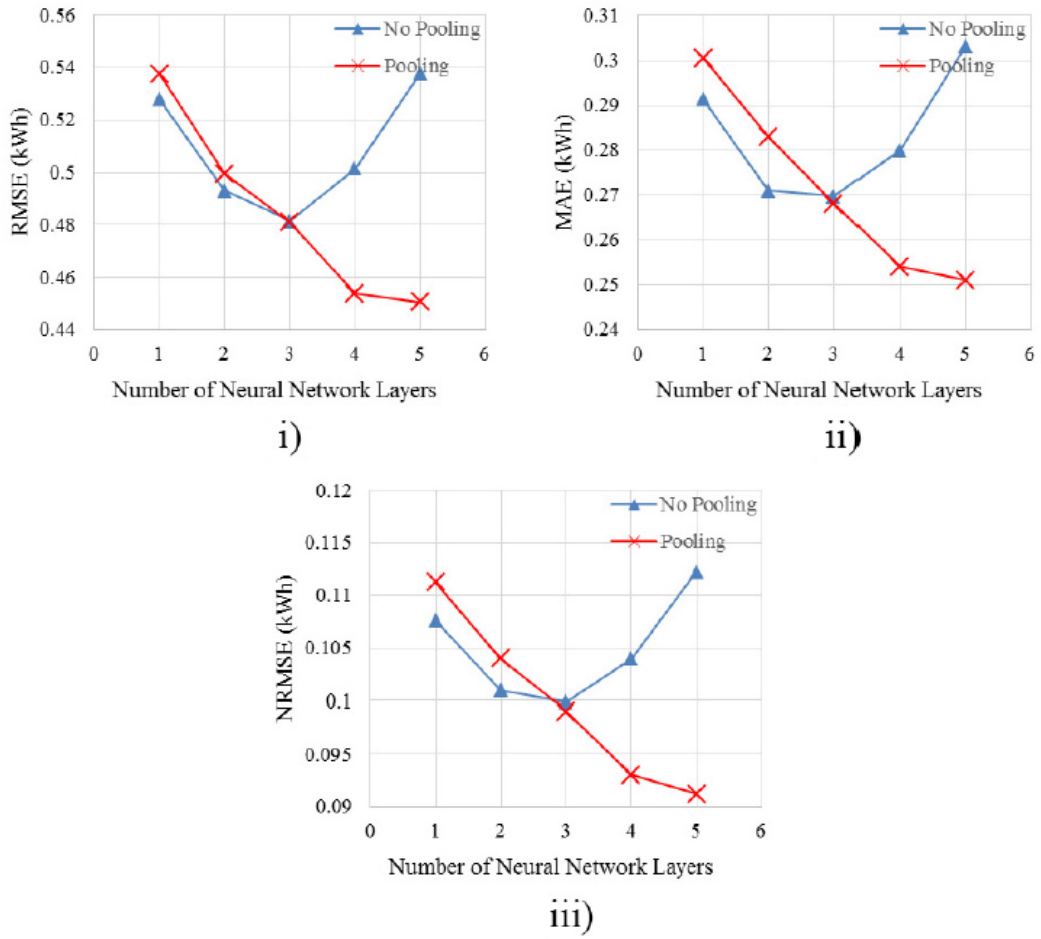


FIGURE 3.7: Influence of the number of layers and the pooling method used in [14].

the available data instead from the entire dataset.

Short-term Residential load forecasting based on LSTM RNN paper
In [9] it is chosen for a LSTM approach to forecast the complex temporal consumption pattern which characterises a single household electricity load. It is discussed that the diversity in the aggregated level of the individual electrical loads, smooths the daily load profile. This has as effect that the aggregated electrical load time-serie becomes more predictable, while a single household electrical load is more dependent on the human behaviour of its residents. This is substantiated by making use of a density based clustering technique where it was shown that the different daily consumptions of the aggregated signal could be described by one cluster and no outliers. An outlier means that a daily consumption could not be assigned to a cluster. On the other hand for individual time series the amount of outliers could range to over 80. To compare the consistency of different individual load signals the amount of outliers could therefore be used.

Because the residents daily routine is characterizing the household load so much, this is tried to be learned directly inside the LSTM RNN.
Inputs that are given to the LSTM are k past half hour load measurements, the time of when this measurements were taken, the day of the week of the measurements and if this day is a holiday or not. In table 3.8 the results are shown of the LSTM RNN method in comparison with other forecasting techniques. It can be noted that the proposed technique outperforms the rest based on the average performance of $29,808$ individual forecasts of half an hour individual loads. Forecasting was performed on 69 different electrical loads coming from households in Australia. However, for individual load series forecasting the MAPE minimization is also remarkable when considering its simplicity in comparison with LSTM. Next, it was concluded that learning methods that had good performance on aggregated time-series e.g. IS-HF and KNN, perform much worse when predicting individual loads.
Further, by making use of a regresion technique in function of the amount of outliers it is shown that LSTM and BPNN (Back-Propagation Neural Network) perform similar for, as previously discussed, consistent individual loads. The LSTM only starts to differentiate in performance when inconsistency grows. To conclude things that lack in [9] are practical useful forecasts of a timespan of 24h instead of only half an hour and making use of a rule of thumb when parameter tuning. Hyperparameters that can be tuned in LSTM are: learning rate, lag variable, amount of hidden layers and the amount of hidden nodes.

CNN-LSTM paper
In [8] a novel technique is proposed which makes use of a convolutional neural network from which the outputs are given to a LSTM recurrent network after which a fully connected neural network is used to produce the outputs. The purpose of the CNN is to extract the features that are the main drivers of energy consumption and to remove the noise that comes initially together with the raw inputs. The CNN is made up out of convolution layers and pooling layers and makes use of the "ReLU" activation function. The main purpose of a convolution layer is to extract features while the pooling layer reduces the number of parameters by making use of the "max pooling principle". Using the "max pooling principle" means taking the max value of each neuron cluster of the previous layer. As discussed in paper [9] LSTM is suitable to alleviate the problem of a vanishing or exploding gradient which characterized a simple RNN. LSTM is able to preserve long-term memory by making use of memory states that is used in the calculation of hidden states. It is therefore suitable to remembering the irregular trend of the electrical load time-serie. Finally, a fully connected time-serie predicts the load forecast.
Paper [8] further showed superiority with respect to only making use of the LSTM layers as can be seen in Table **??**. The Inputs that were used to forecast the household load which is located in France are: three submeters with historical loads, global intensity, voltage, global reactive power, global active power, time, data and month. At last, also an analysis is performed to investigate the influence of the different inputs by calculating the average class activation score over the inputs. The results are shown in Figure **??**. It can be seen that especially "Sub metering 3" has a big

Method/Scenario	*Avg. MAPE individual forecasts*	*Avg. MAPE Aggregating forecasts*	*Avg. MAPE forecasting the aggregate*
LSTM/2 time steps	**44.39 %**	**8.18%**	**9.14%**
LSTM/6 time steps	**44.31%**	**8.39%**	**8.95%**
LSTM/12 time steps	**44.06%**	**8.64%**	**8.58%**
Empirical mean	136.46%	32.54%	32.54%
MAPE minimisation	**46.00%**	34.91%	27.28%
BPNN-D/1 day	80.02%	11.69%	14.50%
BPNN-D/2 days	75.28%	11.67%	14.48%
BPNN-D/3 days	74.10%	11.66%	14.42%
BPNN-T/2 time steps	49.62%	**8.37%**	9.54%
BPNN-T/6 time steps	49.04%	**8.29%**	9.55%
BPNN-T/12 time steps	49.49%	**8.36%**	9.17%
KNN/2 time steps	74.83%	15.37%	11.23%
KNN/6 time steps	71.19%	14.61%	12.10%
KNN/12 time steps	81.13%	15.23%	15.30%
ELM/2 time steps	122.90%	33.68%	Not tested
ELM/6 time steps	136.49%	35.35%	Not tested
ELM/12 time steps	123.45%	30.05%	Not tested
IS-HF	96.76%	20.43%	32.09%

FIGURE 3.8: Different approaches tried in [9] and their averaged performance of 29,808 individual forecasts of half an hour individual loads.

influence on the final forecasts. This sub meter corresponds to the the electric water heater and air conditioner of the house. As was shown in Section A.1 the dataset used in this thesis gives only information about the presence of a hot water heater. Discussed limitations in the paper are the definition of the hyper parameters that were set by trail and error instead of using an automated method e.g. a genetic algorithm. A further limitation is the lack of household characteristics e.g. the amount of residents living in the house. It has previously been shown by **C. Beckel et al.** that household occupancy is one of the primarily drivers of electrical consumption in a household.

CNN-GRU paper
[13]

See oneNote for the summary of the paper and say that it is showed that CNN-GRU performs even better than CNN-LSTM

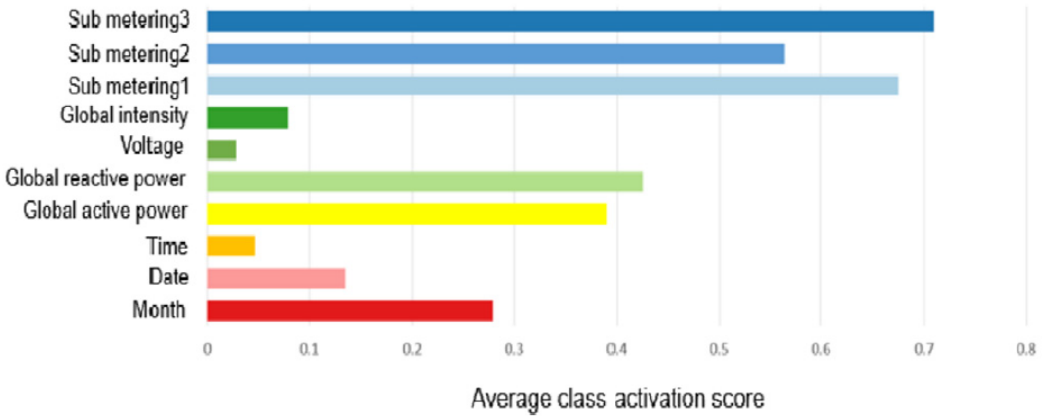


Figure 3.9: The importance of the different inputs as based on the average class activation score. (source [8])

Prediction performance with time resolution change.

Method	Resolution	MSE	RMSE	MAE	MAPE
Linear Regression	Minutely	0.4046	0.6361	0.4176	74.52
	Hourly	0.4247	0.6517	0.5022	83.74
	Daily	0.2526	0.5026	0.3915	52.69
	Weekly	0.1480	0.3847	0.3199	41.33
LSTM	Minutely	0.7480	0.8649	0.6278	51.45
	Hourly	0.5145	0.7173	0.5260	44.37
	Daily	0.2406	0.4905	0.4125	38.72
	Weekly	0.1049	0.3239	0.2438	35.78
CNN-LSTM	Minutely	**0.3738**	**0.6114**	**0.3493**	**34.84**
	Hourly	**0.3549**	**0.5957**	**0.3317**	**32.83**
	Daily	**0.1037**	**0.3221**	**0.2569**	**31.83**
	Weekly	**0.0952**	**0.3085**	**0.2382**	**31.84**

Figure 3.10: Comparison between LSTM and CNN-LSTM. (source: [8])

3.3 Conclusion

The final section of the chapter gives an overview of the important results of this chapter. This implies that the introductory chapter and the concluding chapter don't need a conclusion.

Chapter 4

Forecasting the daily electricity consumption

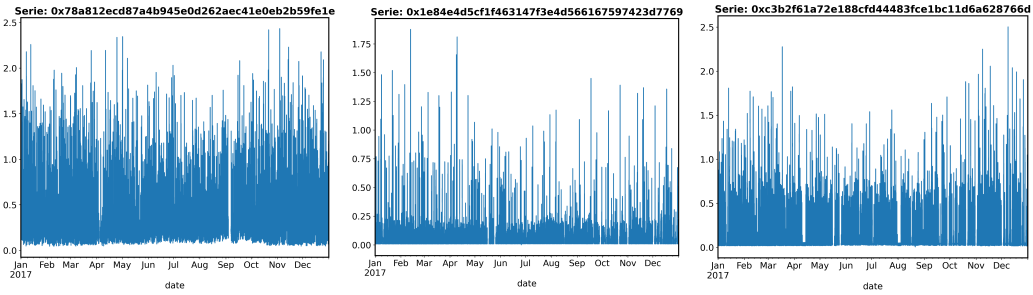
In this chapter the different forecasting techniques to perform a 24 hour prediction for an individual household are discussed. One daily prediction has a data granularity of an half hour, which means that 48 data points have to be estimated for each prediction.
The day we want to forecast is further indicated as the "desired day". First pre-processing is done in Section 4.1 and the time series used are discussed. Afterwards the baseline models are looked into in Section 4.2. These models are characterised by a low calculation load during training and they therefore serve as an easy to obtain result where a more complex model can be compared with. Next, more complex models based on a neural network philosophy are discussed in Section 4.3. "Long Short-Term Memory" and "Gated Recurrent Unit" are most suitable to process time series and therefore serve as the core model which is analysed with different design choices. Finally, a parameter search is conducted. Here, an analysis is made of the sensitivity of the choice of different parameters is made.

4.1 Pre-processing

The data that is available is summarized by Table 2.1.

4.1.1 Data

In order to reduce the calculation load to do the parameter search in Section **??**, three series are selected from the *consumption.csv* that is listed in Table **??**. The series are chosen based on the least missing values of the historic electrical consumption serie and the absence of a big shift. Figure **??** shows the three figures and Table **??** summarizes their characteristics.



(A) Serie 1 (B) Serie 2 (C) Serie 3

Figure 4.1: The consumption of 2017 for the three selected series.

Characteristic	**Serie** 1	**Serie** 2	**Serie** 3
Mean daily consumption [kWh]	14.55	3.17	6.58
Standard deviation daily consumption [kWh]	3.21	0.99	2.57
Median daily consumption [kWh]	14.09	2.96	5.88
Maximum daily consumption [kWh]	30.08	6.60	17.15
Minimum daily consumption [kWh]	7.51	1.83	1.50
Total missing days (consumption)	4	25	26
Days in validation set (November)	30	29	29
Days in Test set (December)	31	23	23
Days in training set (Rest)	300	288	287
Mean average temperature [$^{\circ}C$]	10.37	10.61	10.22
Standard deviation average temperature [$^{\circ}C$]	5.09	5.20	5.01
Median average temperature [$^{\circ}C$]	10.46	10.61	10.35
Maximum average temperature [$^{\circ}C$]	23.99	24.51	22.95
Minimum average temperature [$^{\circ}C$]	-1.07	-1.28	-1.42
Missing average temperature days	0	0	0
Amount of holidays in 2017	8	8	8

Table 4.1: summarizing characteristics about the selected series.

- have chosen to substitute the missing values and use them as training data. An option could have been to learn from a more accurate, smaller dataset. **should change validation set that is used is november –> baseline no validation, NN –> 10 last days of November.**

The three time series are divided in a training, validation and test set. As validation and test set are respectively the months November and December chosen. Because in this chapter the time series are not aggregated to a single signal as was the case in Section 2.3 the min-max normalization can be used. The missing values in the consumption are substituted by making use of following baseline models in order: "previous week" model, "previous day" model and the mean model. When a baseline can't make a forecast because the data is not available, a next baseline model is used.

Similar, the missing values of the temperature are substituted by first looking at the temperature yesterday, then tomorrow and as last the mean temperature.

The simulations that are done in the chapter are performed on a virtual machine through the Microsoft Azure service. Table 4.2 shows the different features of the hired machine.

Name	**Logical cores**	**RAM (GB)**	**Storage (GB)**	**Cost**
F4s v2 (CPU - Azure)	4	8	32	\$0.194/*hour*
NVIDIA Tesla K80 (GPU - Azure)	6	56	380	\$1.166/*hour*
i7-5500U@ 2.40 GHz (CPU - local)	4	12	32	–

TABLE 4.2: Specifications of different CPU's and GPU tried.

- Make a table that summarizes the environment to do simulations.(baseline and othe models) (can be put in the appendix)

4.2 Baseline models

4.2.1 Models

As earlier discussed, the baseline models are characterised by a low calculation load during training and therefore serve as a baseline to compare more complex models with. The different baseline models tried are listed as follows:

- Model 1: "closest day forecast"
- Model 2: "1 day ago forecast"
- Model 3: "7 days ago forecast"
- Model 4: "Mean forecast"
- Model 5: "MAPE forecast"

For all the models listed here, the training set for the forecast of the next day are all the days before this day of the year 2017. These models can therefore be categorized as "lazy learning models" because they only do work when they are asked a query. In contrast, the models discussed in Section 4.3 generalize the training data without knowing the actual query. They belong to the "eager learning methods" class. The 24 hour predictions made by the 5 models are done all at once. This is in contrast to the models described in Section 4.3, where the prediction is made sample per sample and where the predictions done for an earlier hour of the day are taken into account.

Model 1: "closest day forecast"
This model looks for the most similar day in the training set based on following metrics to make a prediction:

- Holiday
- Day of the week

All the days in the training are categorized according to these metrics. Then it is looked in which category the desired day belongs i.e. which day of the week and if it is a holiday. Inside the selected category, an assessment of the difference in average temperature is made for all days with respect to the desired day. It is assumed that the average temperature of the desired day is already available, which is a very plausible assumption. Finally, the day with the closest euclidean distance in temperature is selected and the electrical consumption signal is copied to serve as the prediction of the desired day. It should be noted that there are only a maximum of 8 holidays as can be seen in Table 4.1. Therefore, when a holiday should be predicted all Sundays are also included in the training set because a Sunday behaves most similar to a holiday as can be seen in Figure 2.7.

Model 2: "1 day ago forecast"
This model simply looks at the consumption of the day before the desired day. The philosophy of the model is that the most recent consumption data serves as a a good predictor.
Model 3: "7 days ago forecast"
This model looks at the most recent household consumption of the previous corresponding day of the week. It is expected that people have a reasonably fixed routine during the week and therefore it is likely that this routine will also be found back in the electrical consumption.

Model 4: "Mean forecast"
In the mean forecast the different days are again categorized as was done in Model 1, but instead of selecting a single day out of the group of days, a mean day is calculated and used as prediction of the desired day. No extra Sundays are included to forecast a holiday.

Model 5: "MAPE forecast"
This model solves for each half hour of the desired day a small non-linear optimization problem displayed by (Eq. 4.1). This model served as baseline model in paper [9].

$$objective = \sum_{i=1}^{K} \zeta_{pi} \left| \frac{(\hat{y} - p_i)}{pi} \right| \tag{4.1}$$

Again a group of days of size M, corresponding the desired day is selected based on the metrics of weekday and if the desired day is a holiday. Also Sundays are added to the group of holidays for this model. Next, the consumption at time t is extracted

out of this group of days, which gives a list of length M with historic consumption values. From these values an empirical probability mass function ζ_{pi} is derived by making use of a histogram using "Freedman-Diaconis rule" to decide the bin size. Figure B.1 shows an example of such an histogram. The amount of discretized values p_i is equal to the K bins and taken as the midpoint of two bin edges. From the count in the histogram the probability mass function for each discretized value is found. $\hat{y}$ is found by minimizing equation 4.1.

The metrics used to evaluate the predictions performance of the baseline models are $RMSE$ (Eq. 3.14), $NRMSE$ (Eq. 3.15), MAE (Eq. 3.16), MSE (Eq. 4.2) and $MAPE$ (Eq. 4.3).

$$MSE = \frac{\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}{N} \tag{4.2}$$

$$MAPE = \frac{\sum_{t=1}^{N}|\hat{y}_t - y_t|/y_t}{N} \tag{4.3}$$

It is expected that the MAE punishes prediction errors more proportional than the MSE, which takes the error squared. When an outlier occurs, MSE will see this as a bigger error than MAE does. The downside of using the absolute function is that it is not a smooth function. The advantage of using $RMSE$ and MAE is that both have an error in kWh which is intuitive. $NRMSE$ and $MAPE$ take also the signal to predict into account. $NRMSE$ gives a bigger error, when the true signal has not a lot of variation. $MAPE$ takes into account that a small error value of on a signal with small amplitude has more impact than a small error on a signal with a big amplitude. Therefore, the latter is considered as a smaller error.

4.2.2 Results of baseline models

The results of the different forecasts of the month December are summarized by Table 4.3, 4.4 and 4.5. In order to make a fair comparison only the days of December where all models could produce a forecast are included in the error metrics.

Error metric	**Closest day**	**1 day**	**7 days**	**mean**	**MAPE**
Mean absolute error	0.2049	0.1954	0.1896	0.1542	0.1920
Mean squared error	0.1148	0.1090	0.1011	0.0701	0.1079
Normalized root mean squared error	0.1591	0.1550	0.1493	0.1243	0.1542
Root mean square error	0.3389	0.3302	0.3180	0.2648	0.3285
Mean absolute percentage error	0.5709	0.6163	0.5840	0.4594	0.4138

TABLE 4.3: Evaluation results for Serie 1 tested on 31 days of December.

The rule that the mean forecast outperforms the other methods can be concluded out of all the above tables when not looking to the MAPE metric. The order of the other forecasting techniques are in this case more serie dependent. The MAPE

Error metric	Closest day	1 day	7 days	mean	MAPE
Mean absolute error	0.0559	0.0750	0.0693	0.0473	0.0507
Mean squared error	0.0123	0.0264	0.0188	0.0085	0.0125
Normalized root mean squared error	0.0823	0.1205	0.1017	0.0681	0.0828
Root mean square error	0.1111	0.1625	0.1373	0.0919	0.1117
Mean absolute percentage error	1.601	2.1993	2.3123	1.6657	0.7841

Table 4.4: Evaluation results for Serie 2 tested on 12 days of December.

Error metric	Closest day	1 day	7 days	mean	MAPE
Mean absolute error	0.1267	0.1370	0.1323	0.1038	0.1130
Mean squared error	0.0824	0.0846	0.0895	0.0521	0.0743
Normalized root mean squared error	0.1453	0.1472	0.1514	0.1155	0.1380
Root mean square error	0.2871	0.2909	0.2991	0.2282	0.2726
Mean absolute percentage error	0.8609	1.3153	0.9271	0.8094	0.4792

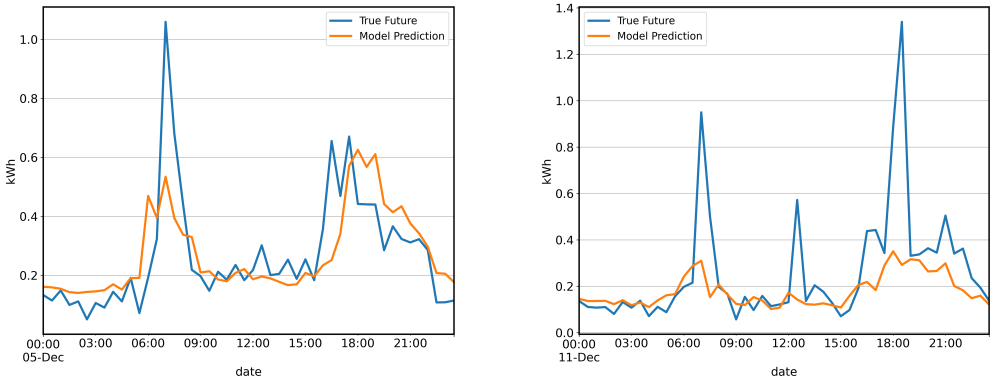
Table 4.5: Evaluation results for Serie 3 tested on 12 days of December.

minimization technique logically performs for all time series best on the MAPE error metric. The mean forecast and the MAPE forecast will consequently be the models that are chosen as baseline models. It can be seen by inspecting the predictions that model 1,2 and 3, which copy the consumption of another day, show more peaks in their prediction than model 4 and 5 that make use of the mean and MAPE techniques. Predictions of of the last mentioned models can be seen in Figure 4.2 Also, a practical downside of model 1,2 and 3 is that it is possible that they give no output e.g. when no temperature or consumption yesterday is available.

Table 4.6 shows the average performance of the baseline models on all 261 time series in the "consumption.csv" of Table 2.1 that contain a full year of measurements. The MAPE metric is not used because 99 series contain a true half hour consumption of zero, which would lead to a division by zero. The resulting error metric of every time serie is divided by the error of the model that performs worst. Then the average is taken over all the time series. The closer the averaged value is to one, the more often this model was behaving worst. By applying this normalization, the NRMSE leads to the same result as for RMSE. Therefore, only RMSE is mentioned.

Error metric	Closest day	1 day	7 days	mean	MAPE
Mean absolute error	0.950	0.8298	0.8224	0.7274	0.7918
Mean squared error	0.8377	0.7771	0.7363	0.4996	0.6907
Root mean square error	0.9084	0.8621	0.8426	0.6993	0.8155

Table 4.6: Relative performance over all the 261 time series.



(A) Model 4: "Mean forecast" (B) Model 5: "MAPE forecast"

FIGURE 4.2: Daily predictions of two baseline models. (Blue: True / Orange: Prediction)

From Table 4.6 it can again be concluded that the model performing a mean forecast attains the best results. The second best general applicable model over all the three metrics is the MAPE forecast model.
-add initialization in intro

4.3 Neural network models

In this section deep LSTM models, which are models that are specialized in time series learning, are presented in different variations. The term deep is used because multiple LSTM layers are stacked on top of each other. The goal of the developed models is to make a 24 hour electrical consumption forecast for a single household. First, the general context in which the deep LSTM models are trained is explained. This covers the inputs that are feeded to the models, measures taken to avoid overfitting, the explanation of a stateless model versus a stateful model, initialization and the metrics used for evaluation. Next, the specific LSTM models developed are discussed in Section 4.3.2. Further, follows the discussion of the conducted parameter search in Section 4.3.5.

4.3.1 Different practical considerations of the models in Keras

Inputs

The inputs to theses models are similar as was suggested in papers [2] and [9]:

- historic training values (min-max normalization)
- average daily temperature (min-max normalization)
- which day of the week? (one hot encoding)

- which time of the day? (one hot encoding)
- is it a holiday? (one hot encoding)

How far will be looked back into the history of the electrical consumption is decided by the lag value. This value corresponds to the number of historic time steps that will be taken into account. The LSTM models are developed using the Keras toolbox backed by Tensorflow in Python. The Keras model expects a matrix $\boldsymbol{X}$ as input with three dimensions: sample number, amount of time steps and amount of features. The amount of time steps equals thus the lag value.
One sample of the $\boldsymbol{X}$ matrix e.g. $(1, lag\ value, features)$, equals a 2D matix. Time steps correspond to the rows of this matrix and features to the columns. Every column contains a different time serie that serves as input. On the first column an amount of $lag\ value$ of previous consumption values is stored. The further down the rows of the first column, the closer time gets to the actual value that needs to be predicted. This is also the case for the temperature, but because the temperature is only known on daily basis and the history of the electricity consumption is known for every 30 minutes, the temperature value is often the same in the second column of the 2D matrix. The historic consumption and daily average temperature input are normalized by scaling between 0 and 1, using a min-max normalization.
In the next 7 columns of the 2D matrix, information is given about which day of the week it is. This is encoded making use of an one hot encoding.For every row one of the 7 columns which corresponds with the day of the week gets an one and the rest will be set to zero. Very similar the information about the time of the day is indicated. There are 48 columns and for every row only one column gets an one to indicate the time of the day. This makes that the 2D matrix has 59 columns which corresponds to the amount of features. Every 2D matrix in X serves as input to forecast a single next half hour electrical consumption.

When the 24 hour prediction is made, the assumption is taken that information about the real electrical consumption is known until the moment of prediction. In case of this thesis, that means until midnight the previous day.

Batch size

A batch size has to be specified in the Keras fit function. The batch size determines how many inputs are feeded to the model and their outputs compared with their reference to define an objective function that is used when calculating the gradients for weight update. Because here only one sample is predicted per 2D input matrix, the amount of samples of $\boldsymbol{X}$ correspond to the amount of outputs.

Stateless versus Stateful

Keras introduces a concept of stateless and stateful. As explained above the input given to a LSTM model is an array of data of three dimensions: sample number,

amount of timesteps, amount of features. As explained in [3] a stateless model interprets every sample of $\boldsymbol{X}$, which is a 2D matrix of inputs, to contain in every column a serie that has nothing to do with the serie on the same column in a other sample of $\boldsymbol{X}$. When processing the next sample of $\boldsymbol{X}$, the hidden states and memory states are therefore again initialized to zero vectors. Because there is no relation between the input series over the different samples, it is logical to collect as much as possible samples for training and the input data is generated by moving a window every time one step further as can be seen in Figure 4.3. It was found that when predictions were done also the hidden states and memory states are reset for every prediction.

In contrast with a stateless model, a stateful model doesn't reset its hidden states and memory states when going to the next sample in $\boldsymbol{X}$. This means that when all the series that are contained in the separate 2D matrices are glued behind each other, the initial input signals should be recovered. How this is done can be seen in Figure 4.4 for a *lag value* of three. It can however be desired to reset hidden states and memory states during training e.g. after one epoch. To achieve this, it has to be set manually. During predictions, also the hidden states and memory states are preserved. Therefore the concept of seeding before predictions becomes possible as explained in Section 4.3.4.

Using a stateful model introduces the additional complexity that the amount of training samples in $\boldsymbol{X}_{training}$ and $\boldsymbol{X}_{validation}$ should be dividable by the batch size that is chosen. This batch size then also has to be used during prediction. The circumvent these issues, a batch size of 1 is chosen for model 3 which is discussed in Section 4.3.4.

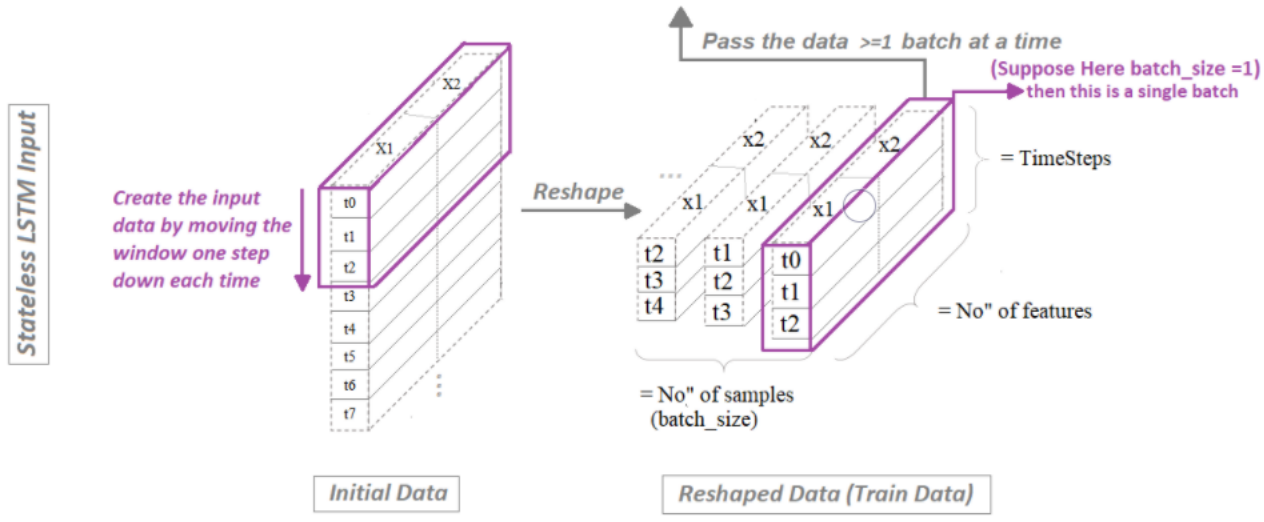


FIGURE 4.3: The generation of inputs for a stateless model. (source: [3])

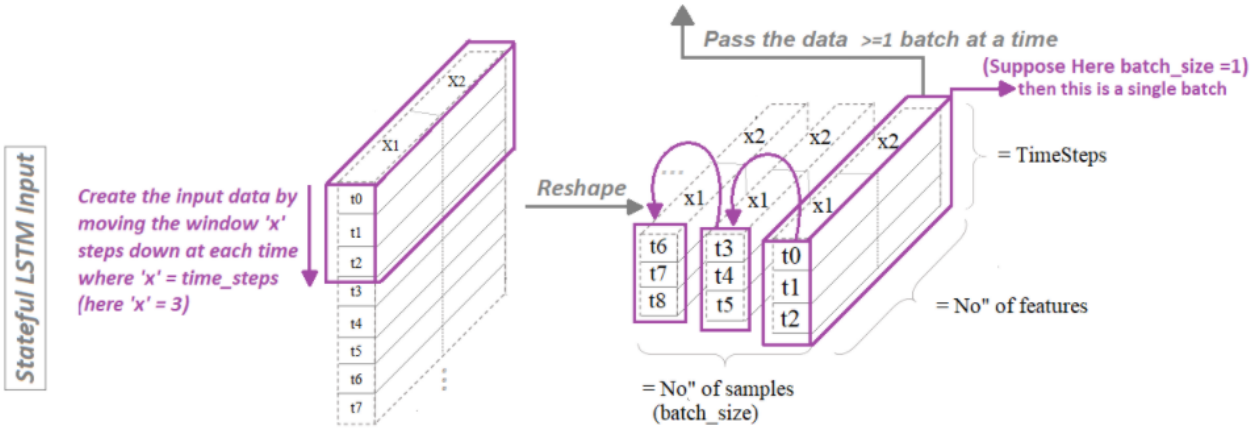


Figure 4.4: The generation of inputs for a stateful model. (source: [3])

Initialization

When looking at the LSTM equations in Section 3.1.5, four weight matrices $\boldsymbol{W}$ with an index H and four weight matrices with an index X can be identified. These are respectively the recurrent and kernel weights and they are both initialized in a different way. A recurrent weight matrix is initialized using a "orthogonal" initialization and the kernel weight matrix uses "glorot uniform" initialization. Both methods make use of random sampling of a distribution during the generation of the initial values. The biases $\boldsymbol{b}$, memory states $\boldsymbol{c}_t$ and hidden states $\boldsymbol{H}_t$ of the LSTM equation are all initialized by arrays filled with zeros.

Overfitting avoidance in Keras

Different ways to avoid overfitting can be applied in Keras and is listed as follows:

- Early Stopping
- Dropout and recurrent dropout in the LSTM layer
- Dropout in the Dense layers
- $l2$ regularization on the kernel weights, recurrent weight, bias and activity.

Metrics to evaluate the models

The metrics that are used to evaluate the prediction performance of the models are the same as the ones used by the baseline models in Section 4.2:

- Mean absolute error

- Mean squared error
- Normalized root mean squared error
- Root mean square error
- Mean absolute percentage error

During training also an error measure is used to accordingly update the weights of the neural network. It was found in literature that much of the time MSE is used for this.

4.3.2 Deep LSTM

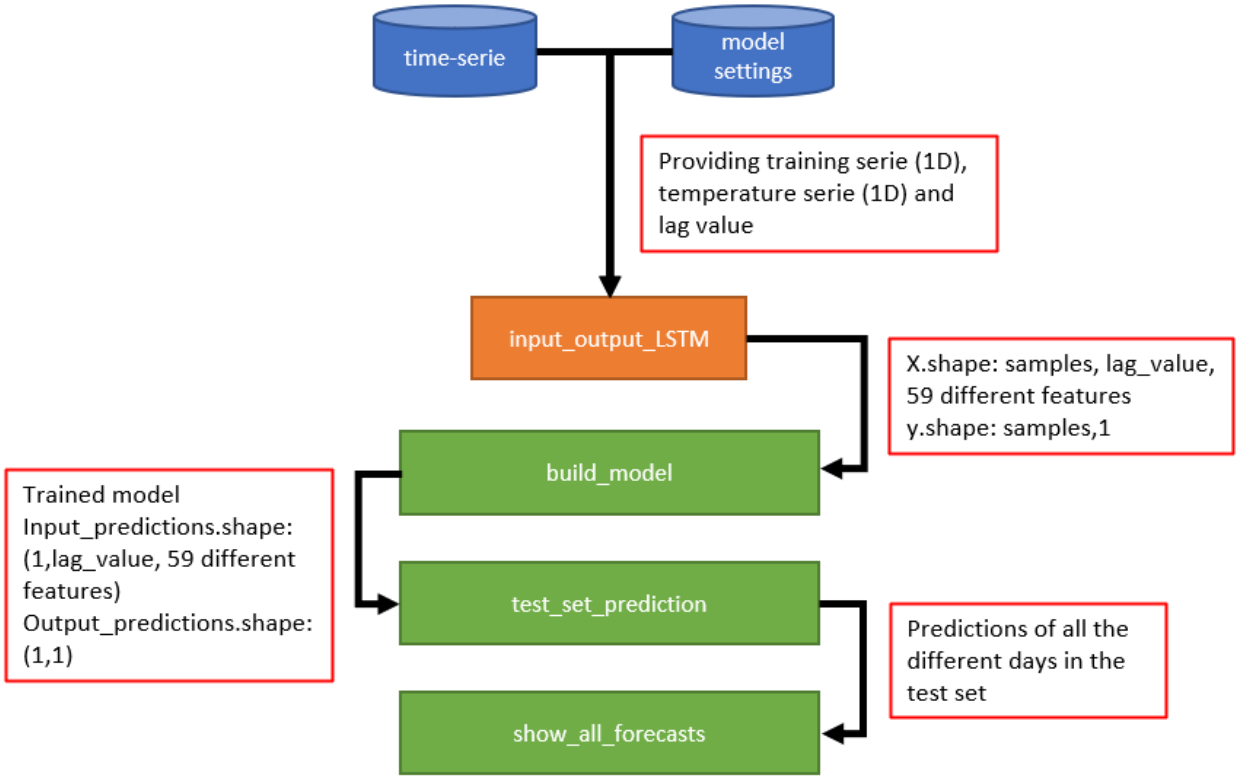


FIGURE 4.5: The flow of functions that are executed in order to produce daily forecasts.

First, the input data of the LSTM is calculated in *input_output_LSTM* which should be done in different ways depending on the model as is showed in Figures 4.3 and 4.4. This is done before training by making use the GPU that is listed in Table 4.2. The inputs that are used are discussed in Section 4.3.1.
Next, the models are build. Three different models will be shown further from which

the first two are stateless models and the third is a stateful model. For the stateless models it is possible to shuffle the input data. It was found that when it is desired that before splitting the inputs of the LSTM in a training and validation set the data is shuffled and not just the last part is used as validation set, shuffling should be done manually before inputted to the fit function in Keras according to Deeplizard's blog page.
During prediction, every model will predict only one half hour of electrical consumption at a time as was proposed as a good methodology in [15]. This concrete means that one input is feeded to the model and one output is coming out. To predict all 48 consumption values in a day, the previous predictions are used to predict consumptions later on the day. Also, each model makes use of early stopping on a validation set to avoid overfitting. The models can have multiple LSTM layers on top of each other. The hidden state vector $\boldsymbol{H}_j$ serves then as input instead of $\boldsymbol{X}_k$ for the layer above. The specific parameters and layout that is chosen for each of the three models is discussed in Section 4.3.5.

Model 1: Stateless with no flatten layer

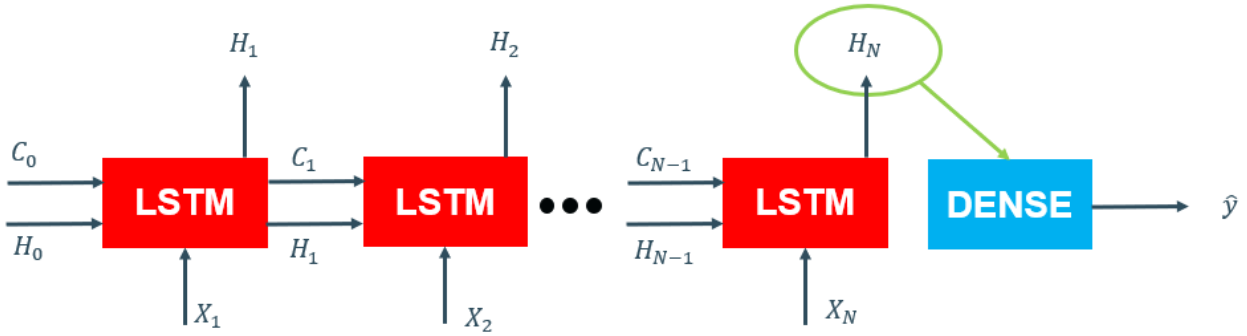


Figure 4.6: Model 1 - stateless model with as input a subserie of N time steps and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^l$, $\hat{y} \in \mathbb{R}^1$.

Every subserie of the original time serie has N time steps and is feeded to the LSTM layer. When the final LSTM block is reached the output of this block will be translated to a single prediction value by a conventional layer of hidden neurons (Dense layer). The subseries are created by shifting the window one time step further in time on the original time serie. Because Model 1 is stateless $\boldsymbol{C}_0$ and $\boldsymbol{H}_0$ will be initialized by zero vectors. The amount of N LSTM blocks that are depicted in Figure 4.6 equals the lag value that determines the amount of previous time steps that will be taken into account for the next prediction. Although that Figure 4.6 gives the impression that there are multiple LSTM blocks, this only serves as a visual interpretation. There is actually only one LSTM block that is reused for every $\boldsymbol{X}_k$. This means that when the same size of layers and hidden recurrent states are chosen, all the three models will have the same amount of trainable parameters.

4.3.3 Model 2: Stateless with flatten layer

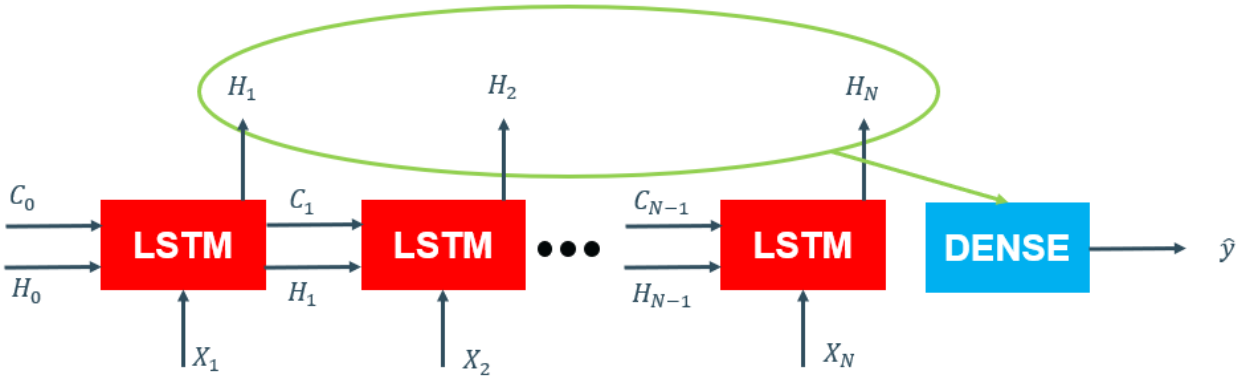


FIGURE 4.7: Model 2 - stateless model with as input a subserie of N time steps and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^l$, $\hat{y} \in \mathbb{R}^1$.

In this model additionally to the output of the last LSTM block also the previous outputs of the LSTM blocks are inputs to the dense layer. This model corresponds to the model depicted and explained in [9].

4.3.4 Model 3: Stateful

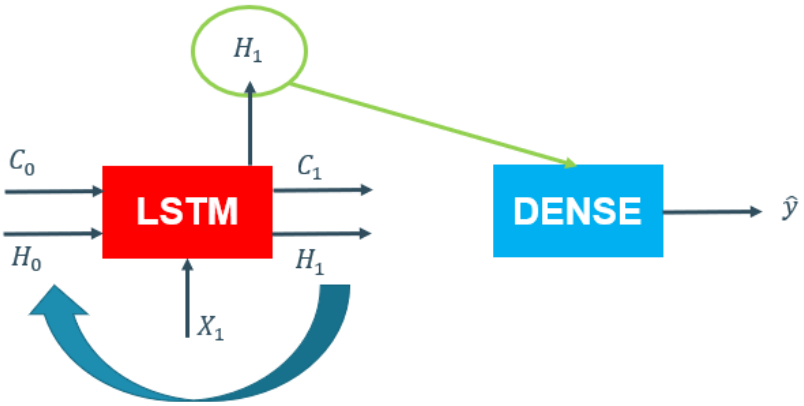


FIGURE 4.8: Model 3 - stateful model that connects single LSTM blocks and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^l$, $\hat{y} \in \mathbb{R}^1$.

Where the previous two models only go through a small subserie of 48 or 96 time steps before making a prediction, this model sees the whole original time serie. This is possible because the model looks one by one at each sample of the original time serie e.g. $\boldsymbol{X}_1$ and sets the C_1 and H_1 that come out as C_0 and H_0 when the next sample e.g. $\boldsymbol{X}_2$ of the original time serie is used. This means that the model glues every subserie of one value together, which recreates the original serie. The shape

of $\boldsymbol{X}$ is now $(15500, 1, 59)$, which means that each sample now is not a 2D matrix but an array. After training and before prediction of the test set or validation set, it is important that the memory states $\boldsymbol{C}_i$ and hidden states $\boldsymbol{H}_j$ are set "right" for prediction of the daily consumption. Therefore, the model is first seeded which means that first all the inputs before the desired day are shown to the model. When the model then reaches the desired day, it had the chance to collect important information in the memory states and the hidden states that it can use during the prediction. A downside of using this model is that during prediction it looks only at the previous predicted electrical consumption and uses this as input to predict the next.

4.3.5 Parameter search

As will be shown in this section, the different LSTM models are dependent on the choice of good parameters to be able to perform well. During a parameter search different combinations of values for the models are tried and the values that give the best results are retained. Because it is not straightforward what the influence of one parameter exactly is on the model, all the different parameters are expected to depend on each other. If all the different parameters are varied the calculation load to try all the different combinations extremely quickly increases. In order to deal with this the parameter search that is conducted is done in three stages. The cost that comes with reducing the calculation load, is that the domain of possible values is not totally searched and dependencies are neglected. During the parameter search every combination of settings is repeated three times to reduce the influence of the random initialization of the weights in the weight matrices. The error value that is used to decide if a combination of parameters behaves better or worse, is based on the average MAE on the 10 last days of November of the three runs. The 10 last days of November are thus used as validation set during the parameter search and can be used for all three models as validation set. Inspiration for the values chosen for the different parameters is based on [14].

- Stage1: different model layouts are tried (calculation time $\approx$ 10 hours)
- Stage 2: the most complex model is chosen and regulation is added (calculation time $\approx 6 \times 4$ hours)
- Stage 3: after comparing the previous model, the best is chosen and the learning rate is more precise tuned (calculation time $\approx$ 7 hours)

The different values of the parameters that are used can be seen in Table 4.7. A total of 24 combinations are tried and each combination was run 3 times. The batch size chosen is only one number to make a fair comparison between the different setting with respect to the amount of weight updates that will be performed. With a batch size of 48, there are around $\frac{15500}{48} \approx 320$ weight updates per epoch and 640 in total. One LSTM layer is compared with three LSTM layers on top of each other in order to catch a clear difference in the influence of the amount of layers used. Because

Parameter	**Values**
Hidden states	$[20, 50]$
LSTM layers	$[1, 3]$
Neurons dense layer	$[50]$
Dense layers	$[1]$
Lag value	$[48, 96]$
Number epochs	$[2]$
Batch size	$[48]$
Learning rates	$[10^{-4}, 10^{-3}, 10^{-2}]$
Shuffle	$[True]$
Repeat	$[3]$

TABLE 4.7: Parameters used during phase 1 for the two stateless models.

only two epochs are considered to reduce the calculation load during the parameter search, no early stopping is used during the parameter search.

Minor changes to the parameters in Table 4.7 are made for model 3. In model 3 a batch size, a lag value of one and shuffling "False" are used. A batch size of one is chosen because then the prerequisite for a stateful model is fulfilled that the training set and validation set must be divisible by the batch size and more importantly, that the batch size during training equals the one during prediction. To avoid these issues, it was first experimented with using a batch size during training that was bigger than one and then rebuilding the model with a batch size of one. The latter model would then be used during prediction. The trained weights of the first model where transferred to the second model. However, after testing it was found that the *get_weights()* and *set_weights()* commands in Keras didn't reproduce the same prediction results when used on different models. It could only be used on the same model to set the weights to the ones a chosen amount of epochs back during training. This is useful e.g. during the manual implementation of early stopping to restore the weights that performed best. Setting the weights on different models is however possible in the underlying tensorflow package, but not in Keras. In this thesis, it was chosen to solve the batch size issues for a stateful model by setting the batch size equal to one.

In phase two, the assumption is made that a more complex model will be able to better capture the non-linear relations in the data and regulation is added to avoid overfitting behaviour. The learning rate that was part of combination of parameter values that performed best during phase one, is selected together with the lag value. During phase two the amount of hidden states is taken equal to 50 and the amount of LSTM layers to 3. The regulation that is shown in Table 4.8, is always added one at a time. The synergy between different regularization methods is thus neglected.

When l2-norm regularization is added to a weight matrix, the size of the weight using the l2-norm is added in the objective function. This means that big weight values

Kind of regularization	**Kind of dropout**
regularizer on input weight matrix LSTM	dropout inputs LSTM
regularizer on recurrent weight matrix LSTM	dropout recurrent states
regularizer on DENSE layer	dropout dense layer
[10^{-2},10^{-3},10^{-4}, 10^{-5}]	[0.2,0.3,0.4, 0.5]

Table 4.8: Different regularization added during phase 2.

are seen as an artificial error and the model tries to keep the weights small. The regularization parameter defines the ratio between the model focussing on the error with the reference signal and the size of the weights. Setting a big regularization parameter will give a model that becomes less "expressive". Therefore, the regularization parameter will contribute in the avoidance of overfitting.
The dropout layer that is used after the LSTM layer and after the Dense layer sets a rate of its inputs to zero. Similar, the dropout and recurrent dropout parameters in the LSTM layer drop respectively a rate of its inputs $\boldsymbol{X}_k$ and $\boldsymbol{H}_j$ to zero. The dropout rate values that are chosen are chosen between 0.2 and 0.5 as is advised in [**?**].

As will be further shown and as was found in [4], the learning rate is a very important parameter that has to be tuned correctly to enhance the model performance. Therefore a bigger range of learning rates is tried for the model that performed best after stage one and two of the parameter search. Afterwards, the learning rate which leads to the the smallest error is selected. The different learning rates that are tried are 10^{-2}, 5×10^{-3}, 2×10^{-3}, 10^{-3}, 10^{-4} and 10^{-5}.

In order to speed up the calculations, multithreading is implemented to calculate separate runs in different threads. It was found that one thread takes considerably longer than when the full CPU is used to calculate the same run, but still a time gain is obtained in a total of 4 runs.

The importance of setting good parameters is stressed by the fact that for certain parameters the loss during training becomes "not a number". This is because the model becomes unstable and gets a very big loss which eventually becomes a nan. When this occurs the model is tried again with different initialization of the weights, which often solves the problem or a nan value is included as error.

Model 1: Stateless with no flatten layer

Phase1:
Table 4.9 shows the parameter search using the parameters listed in Table 4.7. The search was ran on a local machine (see Table 4.2) for 9 hours and 48 minutes. It is clear from Table 4.9 that when the learning rate is varied this has the most impact on the average performance of the model. This result corresponds with paper [4] where it was concluded that this parameter is the most important when tuning

a LSTM. Next, it can be seen that the lag value of 96 time steps didn't lead to much more improvement with respect to looking at 48 historic time steps during every prediction. A reason for this can be that the day, two days ago, is not very representative for the desired day. It can be argued that the desired day a week ago would add more value, because this better captures the human weekly routine. In comparison Model 3 traverses the whole historic signal before making predictions. It is therefore expected that that model better captures the weekly routine.
An unexpected result that was found is that less LSTM units and only one LSTM layer in the case of Serie 1 gives raise to a lower MAE. The reason for this could be the presence of overfitting when using a more complex model. Table 4.10 gives the combination of settings that performed best during phase one. In the next phase of the parameter search, regulation is added and compared with the results obtained during phase one of the parameter search.

Model 1: Stateless (no flatten layer)				
Chosen parameter	**Value**	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	20	12.08	1.24	1.48
	50			
layers LSTM	1	9.82		
	3		4.26	5.80
Lag value	48	4.25		0.390
	96		2.06	
Learning rate	10^{-2}			
	10^{-3}	0.0594	17.2	10.6
	10^{-4}	8.74	25.0	12.2

TABLE 4.9: Each value in this table shows the average error when the value of a chosen parameter was used, normalized by the biggest error of the possible values and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst possible value for a chosen parameter for each serie during phase 1 of the parameter search. For example: when 20 LSTM units are chosen, this on average gave a 12.08% lower MAE.

Phase 2
In this section the learning rate and the lag value of the best performing model are combined with the most complex model with regulation. With most complex model, it is meant that 50 LSTM units and 3 LSTM layers are used. A sensitivity analysis is done to look which regulation has the most effect. Finally, the results of phase two are compared with the best results of phase one. Note, that using this approach assumes that adding regulation can nullify the use of a too complex model. When LSTM regularization is added this done the same on each LSTM layer.
From Figure 4.9 it follows that:

Model 1: Stateless (no flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	20	50	20
layers LSTM	1	3	3
Lag value	96	96	48
Learning rate	0.01	0.0001	0.0001
MAE error 1	0.133	0.0426	0.100
MAE error 2	0.135	0.0433	0.100
MAE error 3	0.138	0.0428	0.101

Table 4.10: The values of the parameters that performed best for the three time series during phase 1 using model 1 based on the smallest sum of MAE errors during three runs.

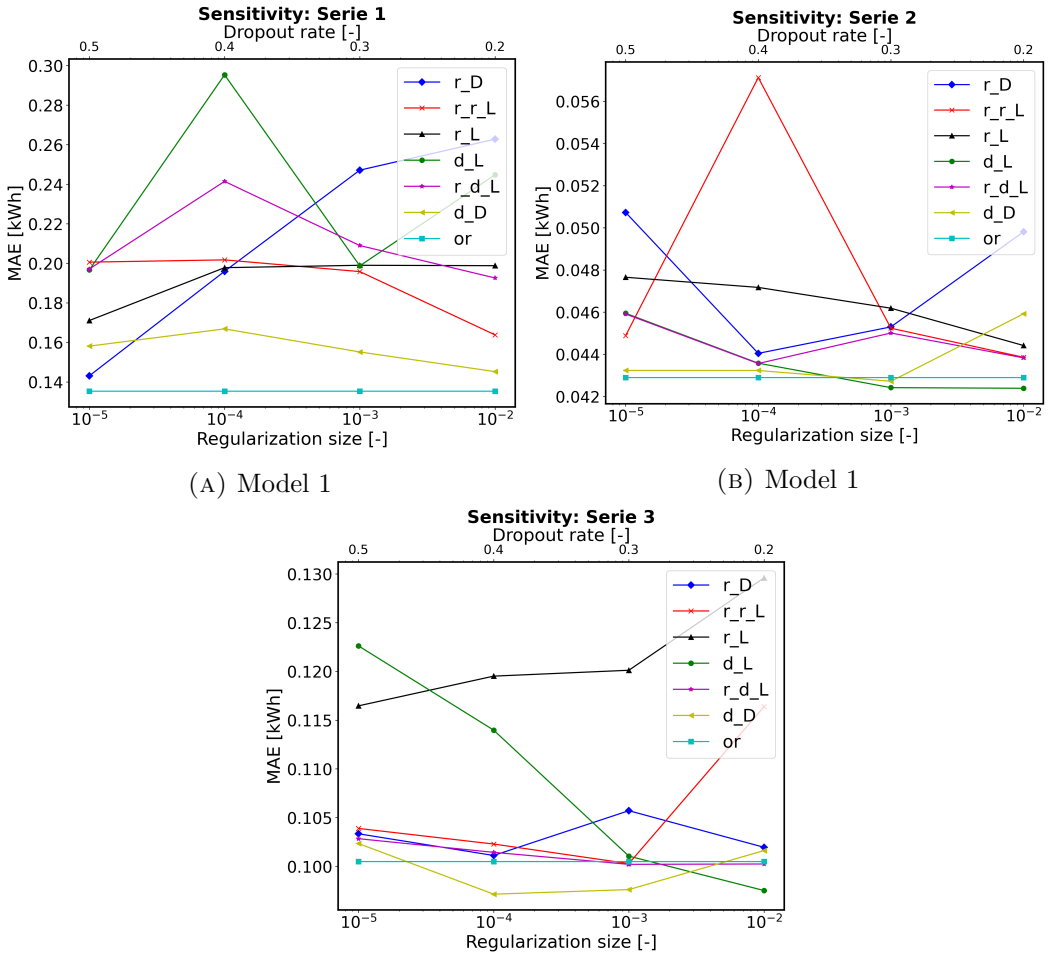
- Serie 1: The best setting during phase one as shown in Table 4.10 outperformed all settings during phase two.
- Serie 2: The best setting after phase one and two is when a dropout rate of 0.2 is added on the input states of the LSTM together with 3 LSTM layers and 50 hidden states.
- Serie 3: The best setting after phase one and two is when a dropout rate of 0.4 is added on the dense layer together with 3 LSTM layers and 50 hidden states.

Phase 3

Finally, during this phase special attention is devoted to the learning rate. As was displayed in Table 4.9 changing the learning rate could lead to significant differences in model performance. Therefore, a sensitivity analysis is performed and the best learning rate is chosen for the best model after phase one and two of the parameter search. Figure 4.10 shows as expected a U-shape error in function of the learning rate. A learning rate that is chosen too large is vulnerable to oscillations and will not convergence and a learning rate that is too small will take a very long time to attain good results and therefore have an increased error when compared on a fixed amount of epochs. The resulting U-shape corresponds to what is found in [4] .

Final model

The final parameters for model one are displayed in Table 4.11.

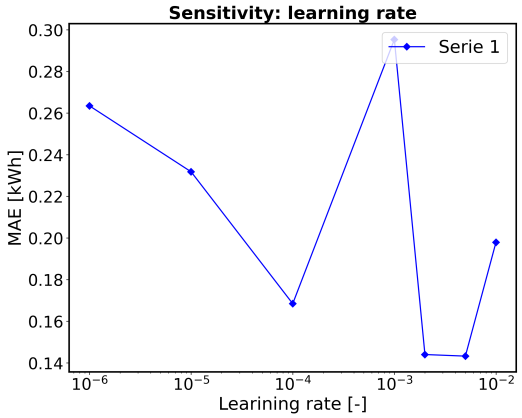


(A) Model 1

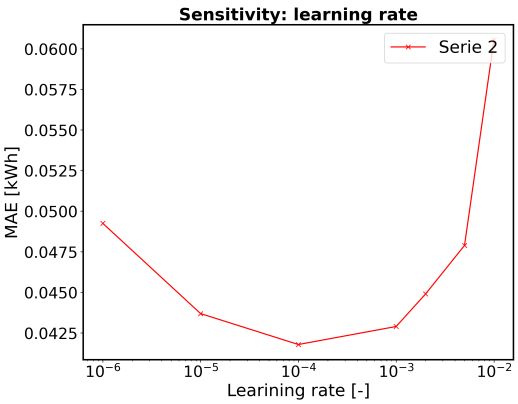
(B) Model 1

(C) Model 1

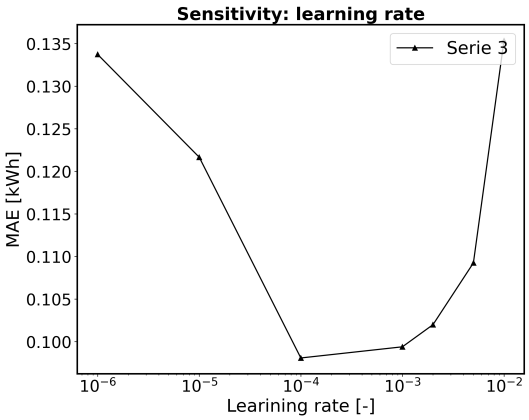
FIGURE 4.9: Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_D*: regularization size of weights DENSE layer, *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM, *d_L*: dropout rate of input states LSTM, *r_d_L*: dropout rate of recurrent states LSTM, *d_D*: dropout rate of DENSE layer, *or*: best performing serie from phase one)



(a) Model 1



(b) Model 1



(c) Model 1

Figure 4.10: The evaluation of the error on the validation set in function of the learning rate size.

Model 1: Stateless (no flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	20	50	50
layers LSTM	1	3	3
Lag value	96	96	48
Learning rate	0.005	0.0001	0.0001
Dropout inputs LSTM	0	0.2	0
Dropout DENSE	0	0	0.4

Table 4.11: Final values obtained after the parameter search for model 1.

Model 2: Stateless with flatten layer

Phase 1
The previous explained procedure is repeated for the other models. From Table B.1 it can again be seen that the choice of learning rate has the potential to give a big improvement and the use of an increased lag value of 96 didn't give a major improvement.
Phase 2
From Figure B.2 it follows that:

- Serie 1: The best setting after phase one and two is when a regularization parameter on the input weigh matrices is added of 10^{-5}together with 3 LSTM layers and 50 hidden states.
- Serie 2: The best setting after phase one and two is when a regularization parameter on the input weigh matrices is added of 10^{-3}together with 3 LSTM layers and 50 hidden states.
- Serie 3: The best setting after phase one and two is when a dropout rate of 0.4 is added on the dense layer together with 3 LSTM layers and 50 hidden states.

Phase 3
Figure B.3 shows the sensitivity of the size of the learning rate in function of the MAE.

Final model
The final parameters for model two is displayed in Table 4.12.

Model 2: Stateless (no flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	50	50	50
layers LSTM	3	3	3
Lag value	96	48	96
Learning rate	0.002	10^{-5}	0.0001
Regularization on input weigh matrices LSTM	10^{-5}	10^{-3}	0
Dropout DENSE	0	0	0.4

TABLE 4.12: Final values obtained after the parameter search for model 2.

Model 3: Stateful model

The model is trained by making use of a batch size of one, which means that the weights of the LSTM block is updated by comparing each output immediately with its reference. The downside of the use of this model is that before the prediction of the consumption of a day, the model first needs to be seeded.

Phase: 1
During phase one the values chosen for the parameters as mentioned before, slightly different. The batch size and the lag value are both put to one. As can be seen in Figure B.3 the learning rate is still an important parameter to tune, but is also seen in Table B.4 that one LSTM layer performed better than using three for the different series.
Phase: 2
From Figure B.4 it follows that:

- Serie 1: The best setting during phase one as shown in Table B.4 outperformed all settings during phase two.
- Serie 2: The best setting after phase one and two is when a regularization parameter on the hidden state weigh matrices is added of 10^{-3}together with 3 LSTM layers and 50 hidden states.
- Serie 3: The best setting during phase one as shown in Table B.4 outperformed all settings during phase two.

It was found that the model when the three layers are used together with 50 hidden recurrent states, often produces a very big loss during training. If this is the case, the output of the training loss becomes not a number. Only for the regularizer on the recurrent hidden states of the LSTM and the regularizer on the inputs of the LSTM the results were better. The results of the two regularizers are shown in Figure B.4.

As can be seen the parameter search of phase two for serie one and two performs worse than the parameter search of phase one. It is not necessarily the case that the addition of regularizers or dropout layers is the only cause of this bad behaviour. As could be seen in Tables B.3 and B.4 one layer performed often better than three layers, which could also contribute to the worst results obtained during parameter search two. Only for serie 2 the best setting of phase one is outperformed.

Phase: 3
Figure B.5 shows the sensitivity of the size of the learning rate in function of the MAE.

Final model
The final parameters for model two is displayed in Table 4.13.

Model 3: Stateful			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	50	50	20
layers LSTM	1	3	1
Lag value	1	1	1
Learning rate	10^{-4}	10^{-6}	10^{-4}
Regularization on hidden states weigh matrices LSTM	0	10^{-3}	0

TABLE 4.13: Final values obtained after the parameter search for model 2.

4.4 Conclusion

This chapter discussed the data to perform the daily electricity consumption forecast on and which models are used. First, the baseline models were discussed. It was found that the mean forecast performed best for the error metrics: MAE, MSE, NRMSE and RMSE. "MAPE forecast" performed best when the MAPE error metric was considered. Both models predict the trend line and don't predict peaks in contrary to the closest day, 1 day and 7 days models. Next, the LSTM models that were developed were discussed. This included the practical considerations of the implementation of the models in Keras and a parameter search. The parameter search was done in three phases in order to reduce calculation load. It was found that the learning rate was the parameter that contributed the most to model performance.

Chapter 5

Model evaluation

This chapter discusses the models that were introduced in Chapter 4 on the test set. As was shown in Table 4.1, the test set consists out of the days of the month December. Missing days in December are removed to avoid the influence of the estimation error of the reference signal. In this chapter first the model selection is explained after which a discussion of the performance on the test set follows.

5.1 Model selection

From Chapter 4 the model parameters are tuned, but there is still a factor of random model performance due to the random initialization of the weight matrices. To reduce this influence, the model is trained 10 times and the model that performed best on a validation set is selected. As validation set the 10 last days of November are used. Also, during the training early stopping is performed wherefore an additional 10% of the training data taken to serve as a second validation set. The patience parameter is taken as 5, which means that the validation error can increase 5 times before the model is stopped. The maximum amount of epochs that is allowed is 150. The values of the parameters for each of the three time series can be found in Chapter 4. Table **??** summarizes the results of the model selection.
Add here a table with the model selection results. - epochs - loss on validation set

Performance on the test set

In this section the results on the test set are discussed of the model that was selected in Section 5.1.
- The model, last ten days are used for getting best model run and 10% of the training is used for early stopping. Then model that get after training is run on the test set. The neural models have the downside in comparison with the baseline models that they have to remove training data that is very close to the test set during training. For example, the base line models could use all the previous data to perform training till the desired day

For example model three, only uses training data till end October. Model one and model two use training data till 20 November and miss a random 10% of the data during the year. This loss of data was necessary for model tuning and the fact that the LSTM model make generalizations before they know the query i.e. the day to forecast (eager models) and the base models learn from the previous data when they know the query (lazy models).
Say that here use the models that obtained in the previous chapter. Make MSE plot for the different NN and the baseline models for each serie and make a comparison of the MAPE with the other baseline model. (by making use of barplots –> normalized with the worst performer) Performance on the test set.
Make a plot of the day forecast for each NN model –> four graphs bundled. - for comparing performance in the same time serie –> mae is used. To compare performance between different time series –> MAPE is used. - for stateless random 10 percent is used as validation set. Remember that should shuffle the training set beforehand otherwise j - for stateful –> the last 30 days of November will be used –> no shuffling is allowed.
- make a graph which shows all the different days that have to be forecasted on the x-axis and the MAE error on the y-axis for the different models. (line graph)
- give an indication how long the models trained –> amount of epochs.
- when there are a lot of missing days in the test data: serie 1 (0 days) and the other two (8days) –>it is naturally that the model performs worse, the previous day it bases its forecast on is just an estimate. For the day that is forecasted are always days where the true reference signal is available.
- bar plot for general performance (MAE and MAPE - basemodels and NN)
- line plot for the error on each day (MAE and MAPE - basemodels and NN)
- the signal for the forecast of a day for each model
- in some plots will see that there are gaps –> reason is that it is not useful to predict on the 8 missing days in the test set. There will only be an estimated signal available to calculate an error on.
- the line plot shows the distribution of the error per day on the predictions per day.
- also the mape metric is considered because it allows to compare between different series.

5.2 Conclusion

Chapter 6

Conclusion

The final chapter contains the overall conclusion. It also contains suggestions for future work and industrial applications.

6.1 Future work

- genetic algorithm to tune the parameters

Appendices

Appendix A

Introduction to the dataset

In this appendix extra information and Figures are added that are not necessary to understand the work discussed in Chapter 2.

A.1 Introduction to the dataset

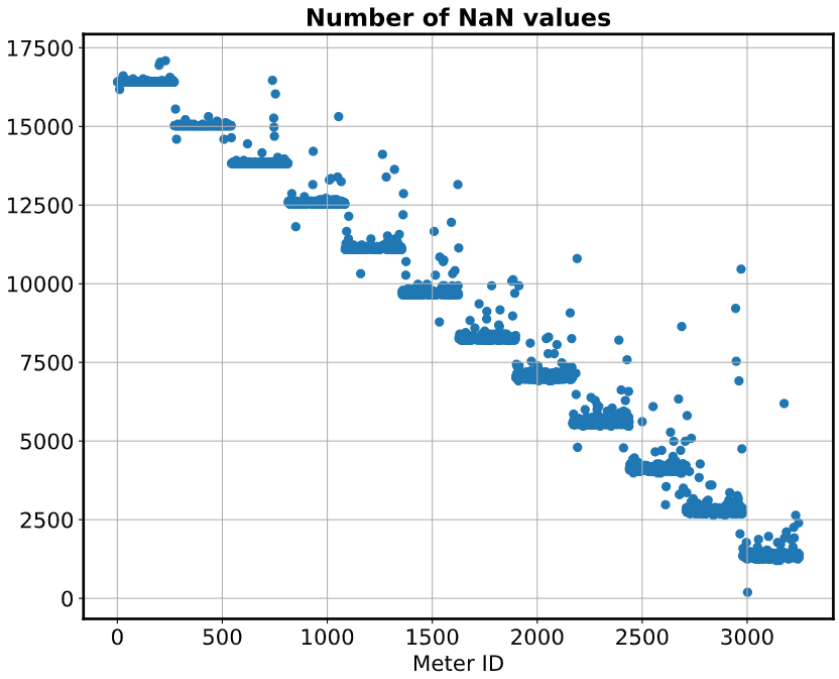


FIGURE A.1: The amount of NaN values in all the 3248 smart meters.

Attribute	Filled places
Dwelling type (5 cat.)	1702
# Occupants (max 4)	74
# Bedrooms (max 5)	1859
Heating fuel (4 cat.)	78
Hot water fuel (3 cat.)	76
Boiler age (2 cat.)	74
Loft insulation (2 cat.)	75
Wall insulation (5 cat.)	75
Heating temperature (4 cat.)	74
Efficient lighting percentage (4 cat.)	73
Dishwasher (0,1,2)	76
Freezer (0,1,2)	70
Fridge freezer (0,1,2)	70
Refrigerator (0,1,2)	73
Tumble Dryer (0,1,2)	76
Washing machine (0,1,2)	76
Game console (0,1,2,3)	72
Laptop (0,1,2,3,4)	70
Pc (0,1,2,3)	70
Router (0,1,2)	69
Set top box (0,1,2,3)	70
Tablet (0,1,2,3,4)	70
Tv (0,1,2,3,4)	75

Table A.1: Amount of response on the voluntary questionnaires.

A.2 Missing values

A.2.1 Fundamental change

A.3 Daily filter

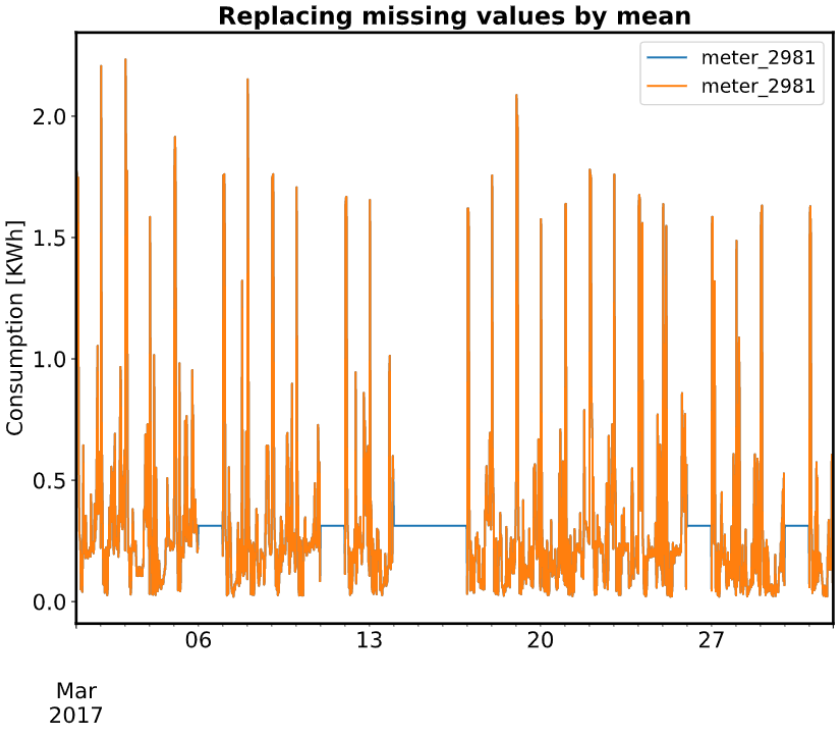


FIGURE A.2: Resulting month of March after substitution of the missing values by the mean value of the measurements.

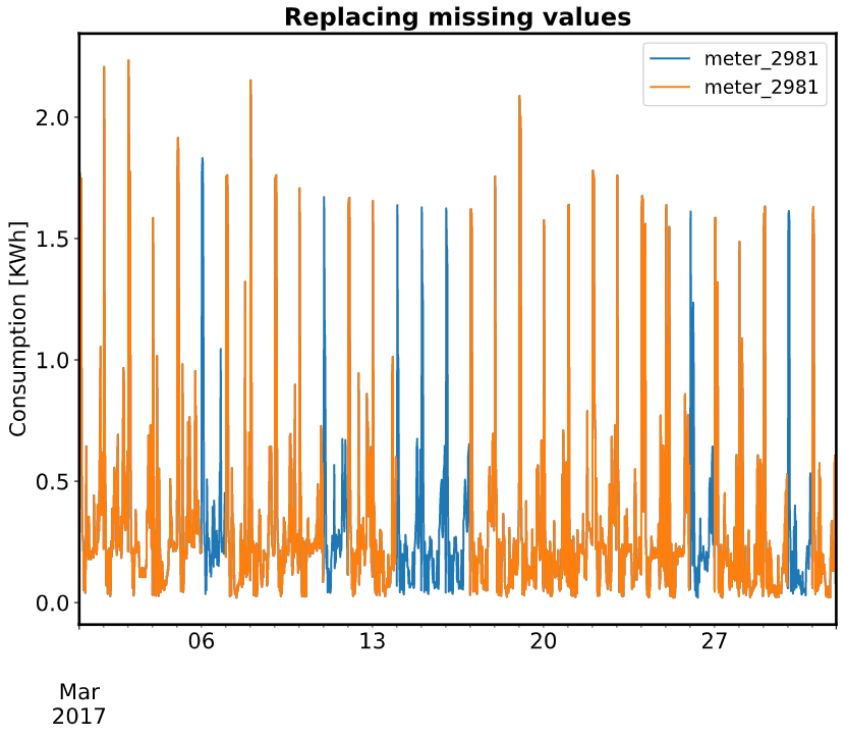


Figure A.3: Resulting month of March after substitution of the missing values by the mean value of the same moment on the next and previous day.

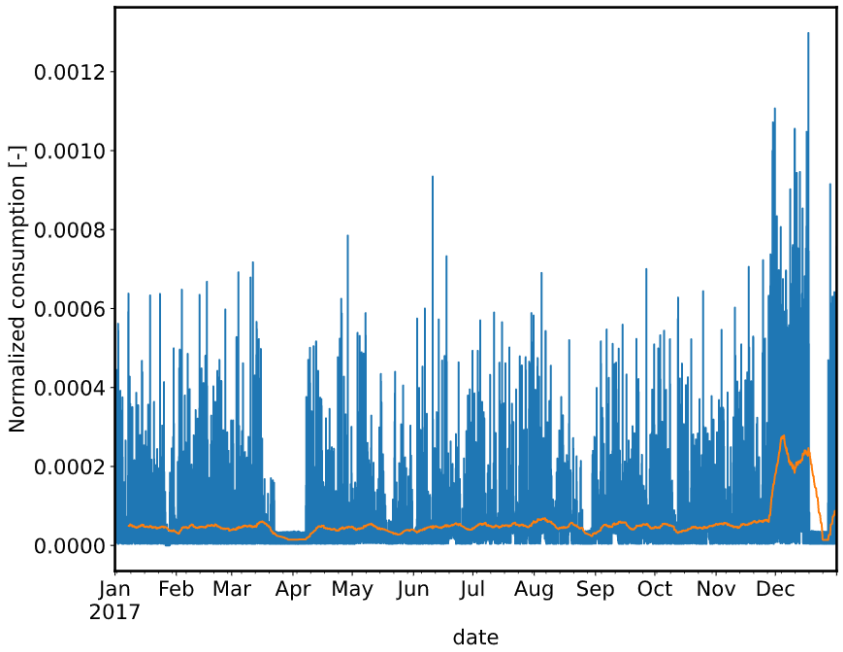


Figure A.4: The time-serie with the original maximum difference between the minimum and maximum weekly rolling averages.

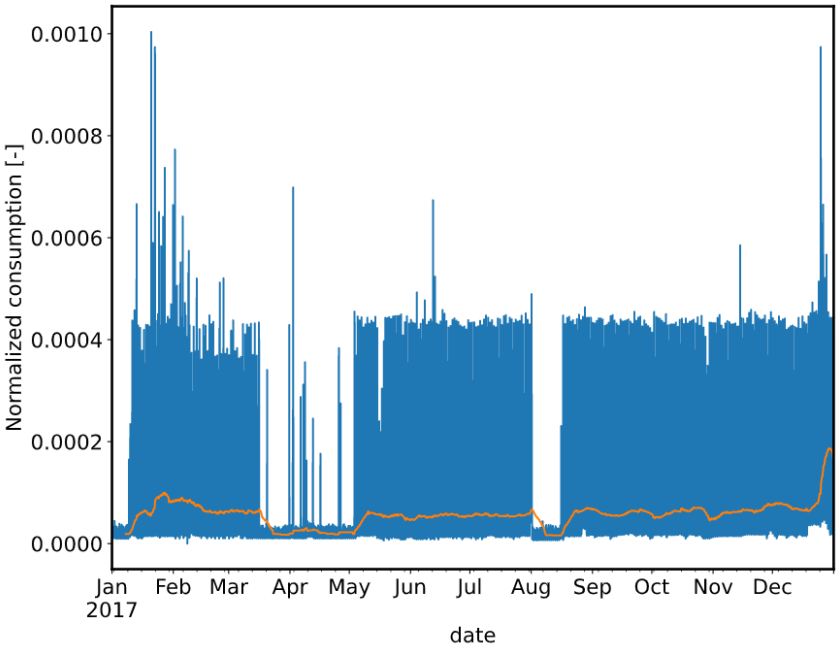


FIGURE A.5: The time-serie with the new maximum difference between the minimum and maximum weekly rolling averages.

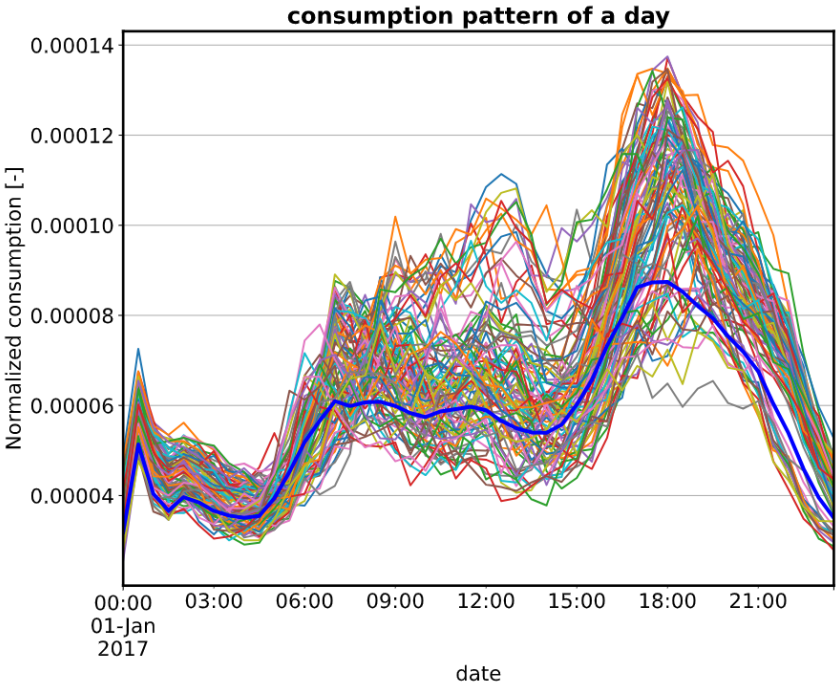


FIGURE A.6: Figure that shows the seasonality of the electrical load during the day.

Appendix B

Forecasting the daily electricity consumption - extra

In this appendix extra information and Figures are added that are not necessary to understand the work discussed in Chapter 4.

B.1 Baseline models

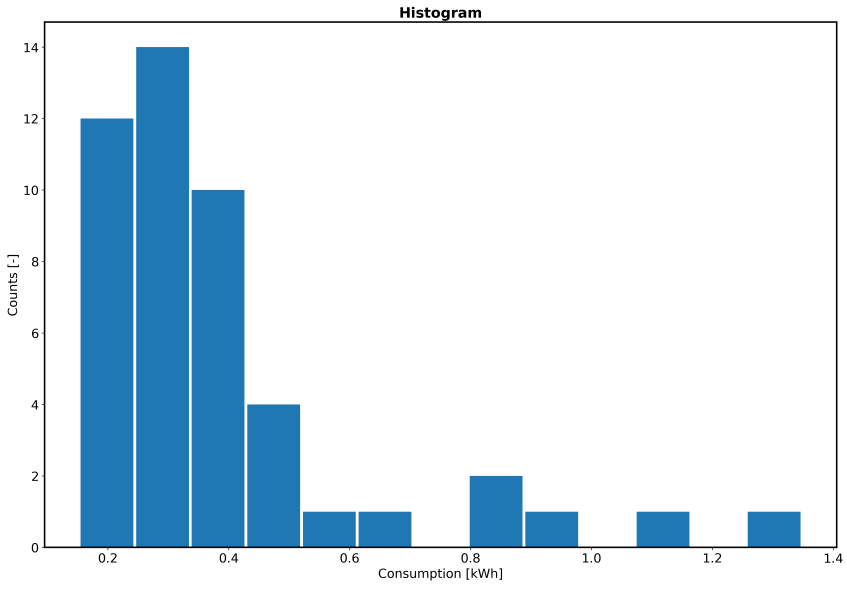


FIGURE B.1: An example histogram of the consumption in [kWh] versus count [-] used during MAPE forecast.

B.2 Parameter Search

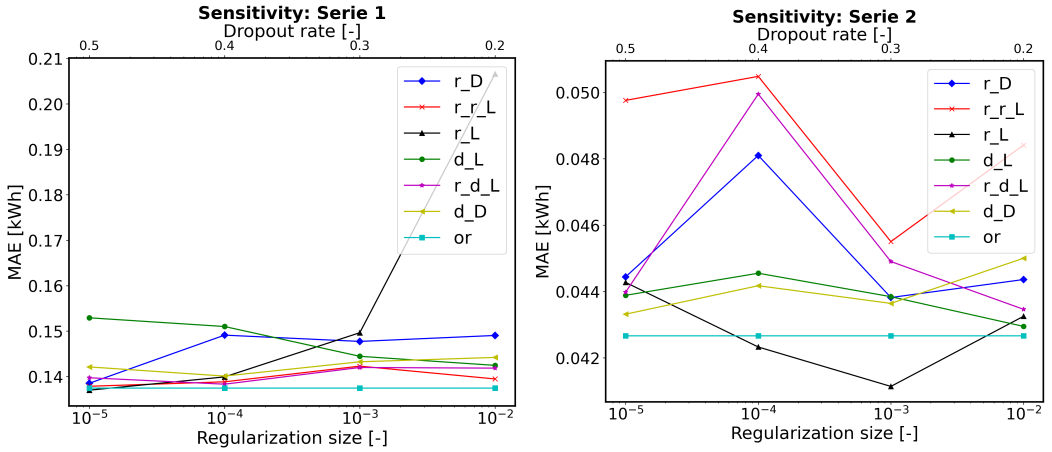
B.2.1 Model 2

Model 2: Stateless (flatten layer)				
Chosen parameter	**Value**	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	20	0.446	0.0622	
	50			0.319
layers LSTM	1			
	3	21.3	10.60	9.98
Lag value	48	7.51	2.84	
	96			1.76
Learning rate	10^{-2}			
	10^{-3}	23.4	2.51	13.2
	10^{-4}	43.0	12.8	17.9

Table B.1: Each value in this table shows the average error when the value of a chosen parameter was used, normalized by the biggest error of the possible values and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst possible value for a chosen parameter for each serie during phase 2 of the parameter search.

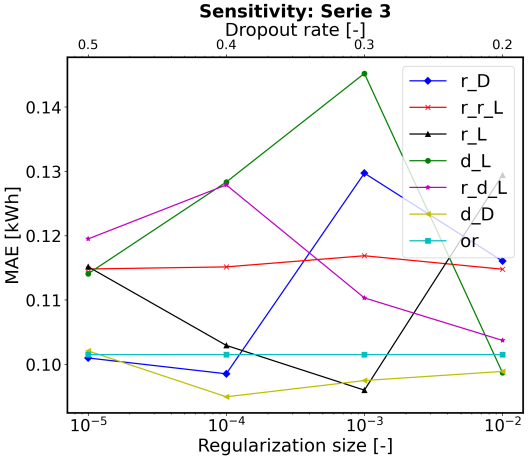
Model 2: Stateless (flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	20	50	50
layers LSTM	3	3	3
Lag value	96	48	96
Learning rate	0.001	0.0001	0.001
MAE error 1	0.137	0.0426	0.0973
MAE error 2	0.139	0.0430	0.107
MAE error 3	0.136	0.0424	0.100

Table B.2: The values of the parameters that performed best for the three time series during phase 1 using model 2.



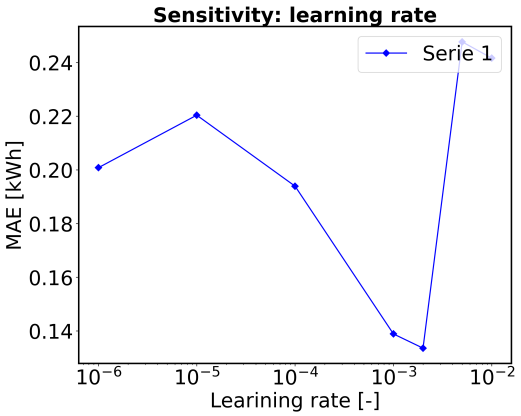
(A) Model 2

(B) Model 2

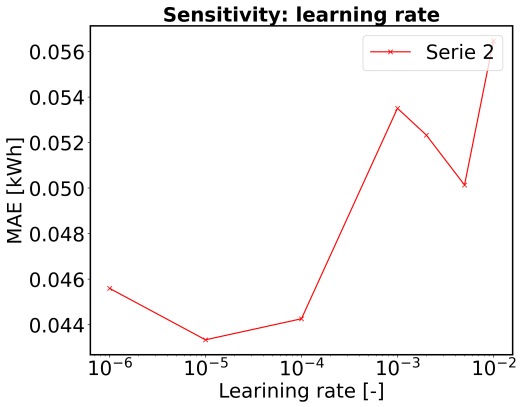


(C) Model 2

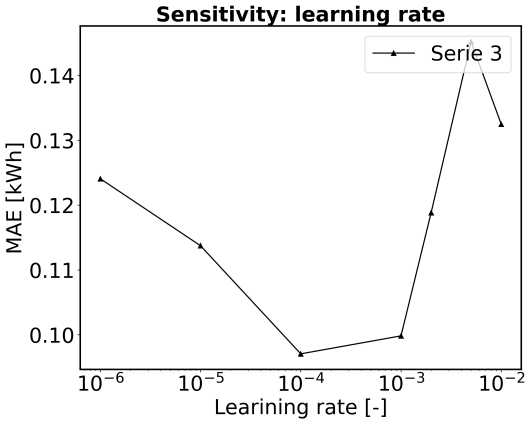
FIGURE B.2: Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_D*: regularization size of weights DENSE layer, *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM, *d_L*: dropout rate of input connections LSTM, *r_d_L*: dropout rate of recurrent connections LSTM, *d_D*: dropout rate of DENSE layer input connections, *or*: best performing serie from phase one)



(A) Model 2



(B) Model 2



(C) Model 2

Figure B.3: The evaluation of the error on the validation set in function of the learning rate size.

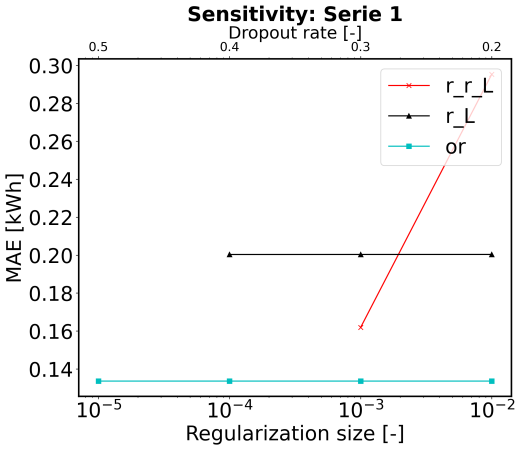
B.2.2 Model 3

Model 3: Stateful (1 time step)				
Chosen parameter	**Value**	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states LSTM	20	6.66		0.55
	50		0.51	
layers LSTM	1	30.00	1.52	7.42
	3			
Learning rate	10^{-2}			
	10^{-3}	17.72	0.0	0.0
	10^{-4}	27.28	2.28	11.13

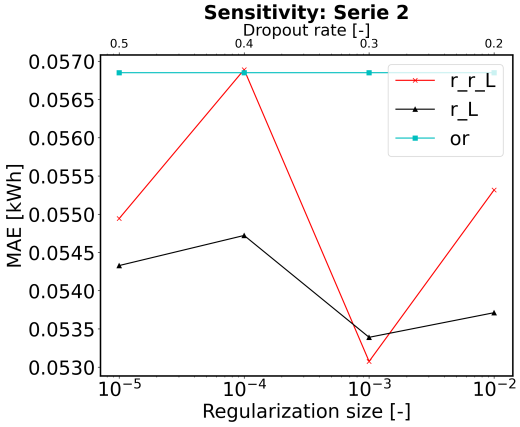
TABLE B.3: Each value in this table shows the average error when the value of a chosen parameter was used, normalized by the biggest error of the possible values and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst possible value for a chosen parameter for each serie during phase 1 of the parameter search.

Model 3: Stateful (1 time step)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Units LSTM	50	50	20
layers LSTM	1	1	1
Learning rate	0.0001	0.0001	0.0001
MAE error 1	0.132	0.0522	0.109
MAE error 2	0.130	0.0613	0.111
MAE error 3	0.138	0.0570	0.112

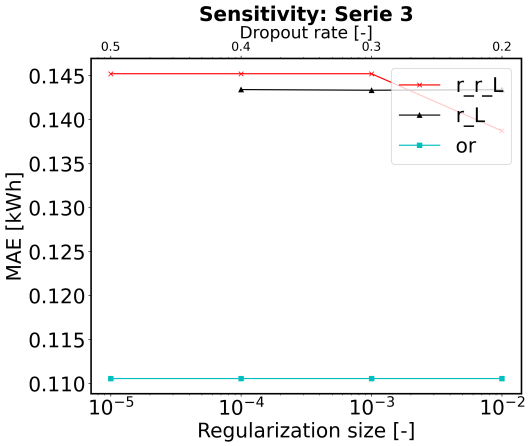
TABLE B.4: The values of the parameters that performed best for the three time series during phase 1 using model 3.



(A) Model 3

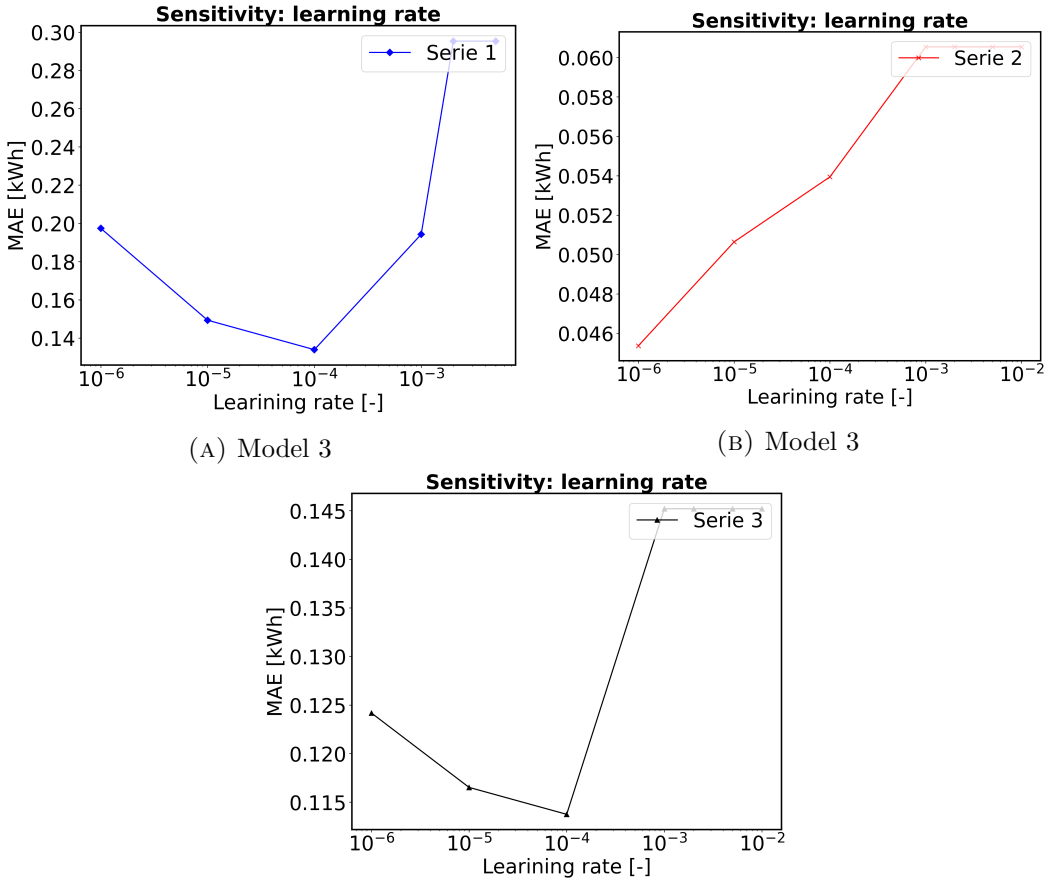


(B) Model 3



(C) Model 3

Figure B.4: Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM and *or*: best performing serie from phase one)



(A) Model 3

(B) Model 3

(C) Model 3

FIGURE B.5: The evaluation of the error on the validation set in function of the learning rate size.

Appendix C

Extensions on the evaluation results

C.1 Results on the testset

C.2 Old stuff

C.2.1 Removing outliers

After the missing values are replaced by estimations, the outliers of the electricity consumption signals are identified. This is done by looking at the z-scores of the yearly consumptions. A z-score is calculated using equation **??** and assumes that the yearly consumptions are normally distributed around the average consumption. Consumptions that have a very low probability to occur are removed by imposing that $|z - score| < 3$.

$$z - score = \frac{x - \mu}{\sigma} \tag{C.1}$$

Figure C.1 gives the obtained z-values. It can be seen that 6 meters with an unlikely high or low consumption are removed.

C.2.2 Normalization of the data

Normalization is necessary because while absolute consumption differs, relative patterns of human behaviour are more similar [10]. The patterns in the human behaviour is what a forecasting model is trying to predict and normalization contributes by avoiding the disturbance of different magnitudes in which this human pattern may occur. Every individual household time-serie is normalized based on its maximum and minimum value according to equation C.2.

$$normalizedvalue = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{C.2}$$

As discussed in section **??** the average is taken over all the normalized time-series to obtain a single signal.**Ask if this is good??** Because the maximum is taken into

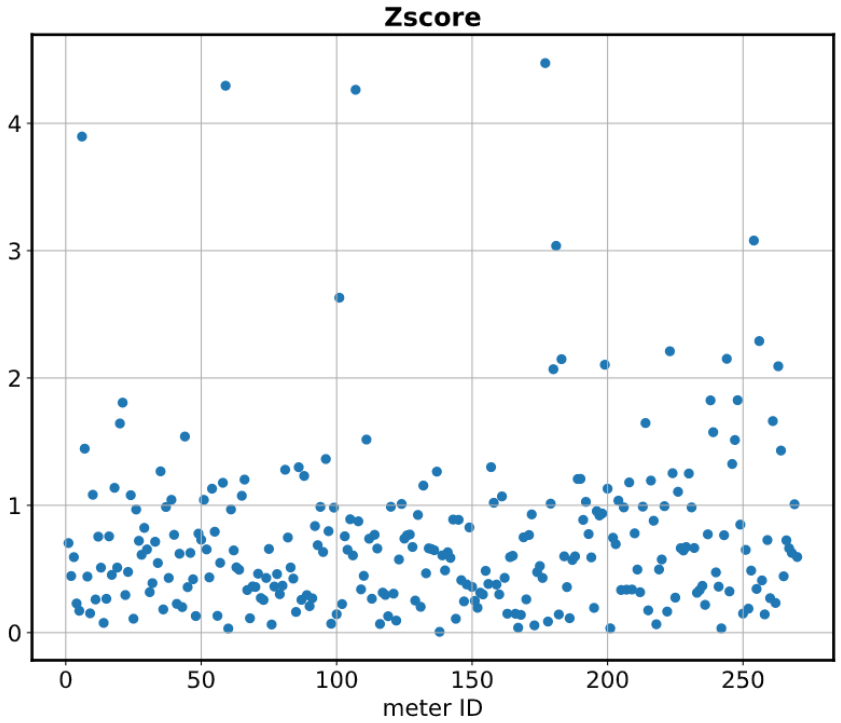


Figure C.1: Z-scores calculated from the yearly consumptions.

account during the normalization, measurement out shooters have an influence on the normalization.

C.3 ARIMA

What is ARIMA. Assumptions of ARIMA...

Stationarity

https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/ When data is modelled it is assumed that the statistics of the data are consistent or stationary. This means the mean and standard deviation is not changing in time. However, because time series are often subdued to a trend or seasonality this assumption of stationarity is violated. In order to model not stationary observations by a stationary model as ARIMA, trends and seasonal effects should be removed. A way to check the stationarity of your observations, the "Dicky-Fuller test" can be used. A way to remove non-stationarity is by using "Difference Transform". Here the trend and seasonality is subtracted from the observations leaving behind a stationary dataset.

Bibliography

[1] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. pages 1–9, 2014.

[2] M. Espinoza, J. Suykens, R. Belmans, and B. De Moor. Electric Load Forecasting. *IEEE control systems magazine*, (October 2007):43–57, 2007.

[3] M. Fneish. Keras_LSTM_Diagram: Understanding Keras Recurrent Nets' structure and data flow in a single diagram.

[4] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[5] B. Hammer. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1-4):107–123, mar 2000.

[6] B. A. Hoverstad, A. Tidemann, H. Langseth, and P. Ozturk. Short-Term Load Forecasting With Seasonal Decomposition Using Evolution for Parameter Tuning. *IEEE Transactions on Smart Grid*, 6(4):1904–1913, 2015.

[7] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. *31st International Conference on Machine Learning, ICML 2014*, 2:1130–1144, 2014.

[8] T. Y. Kim and S. B. Cho. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy*, 182:72–81, 2019.

[9] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid*, 10(1):841–851, 2019.

[10] J. Lago. A ratios and clustering based approach to forecast electricity consumption, 2020.

[11] M. A. Nielsen. Neural Networks and Deep Learning, 2015.

[12] C. Olah. Understanding LSTM Networks.

[13] M. Sajjad, Z. A. Khan, A. Ullah, T. Hussain, W. Ullah, M. Y. Lee, and S. W. Baik. A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting. *IEEE Access*, 8:143759–143768, 2020.

[14] H. Shi, M. Xu, and R. Li. Deep Learning for Household Load Forecasting-A Novel Pooling Deep RNN. *IEEE Transactions on Smart Grid*, 9(5):5271–5280, 2018.

[15] J. Suykens. Recurrent neural networks - ANN lecture. 0, 2021.

[16] J. Teuwen and N. Moriakov. *Convolutional neural networks*. Elsevier Inc., 2019.

[17] M. Zhang, Aston Lipton, Zachary Li and A. Smola. *Dive Into Deep Learning*, volume 17. 2020.