KU LEUVEN

Short-term forecasting of individual household electrical consumption

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

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

List of Tables

Chapter 1

Introduction

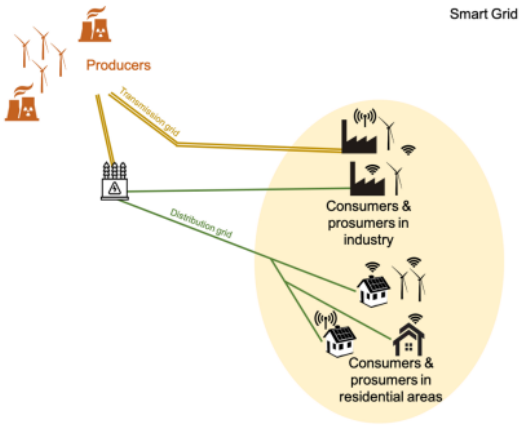


FIGURE 1.1: Smart Grid (Source: KU Leuven thesis proposal).

- different kinds of loads
- different often used metrics.
- make the transition to short term and why this is important.

1.1 Importance of topic

Individual household forecasting is a complex task because of the high amount of uncertainty and the volatility of the data. To deal with this it was found in literature that 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 practice 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.

When forecasting is improved on household scale, the customer can be 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.
- Like in pooling paper –> talk about methods that are used to reduce complexity/ already written in –> now directly forecasting the individual household consumption –> very complex task.
- see also table PL.
- working with real-life data
- wie zijn mijn assessoren?
- local prediction can be important for local changes to the net –> see oneNote goal PL.
- look at goal of thesis –> meeting and PL goal –> what do they want to do exactly
The weather and calendar data is correlated with the energy load profiles

1.2 Problem formulation and link with previous studies

Now going to forecast individual houses, not aggregated signals.
Things were I should get familiar with: scikit-learn, Tensorflow, Keras, Pandas, Anaconda, Microsoft Azure –> step in improving software skills.
Not just looking at regular data –> individual household data –> not a day will be the same and only have limited information available. These two together makes the forecasting extremely difficult.

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.
Develop a model for the future prediction of smart meter electric power consumption. The final objective is to compare several existing methods

Chapter 2

Data exploration

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 Data description

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.

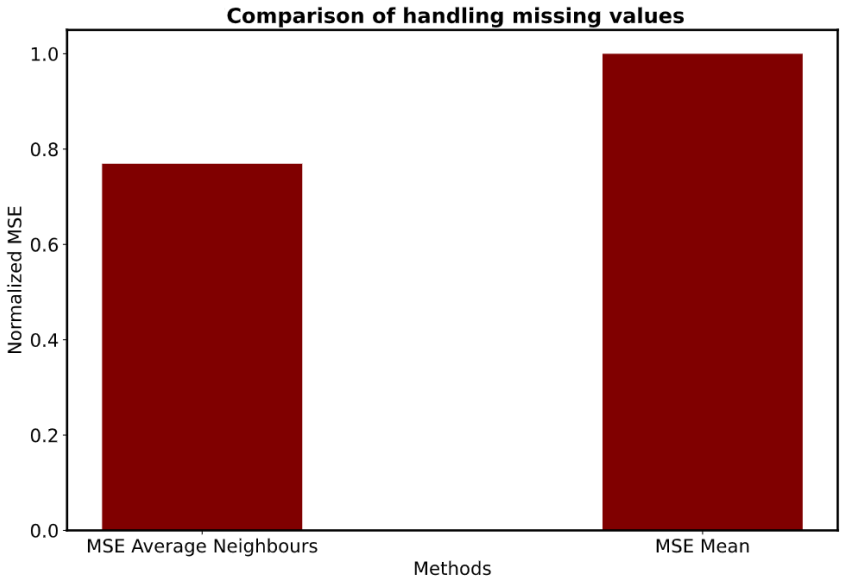


FIGURE 2.1: Resulting month of March after substitution of the missing values by the mean value of the measurements.

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.

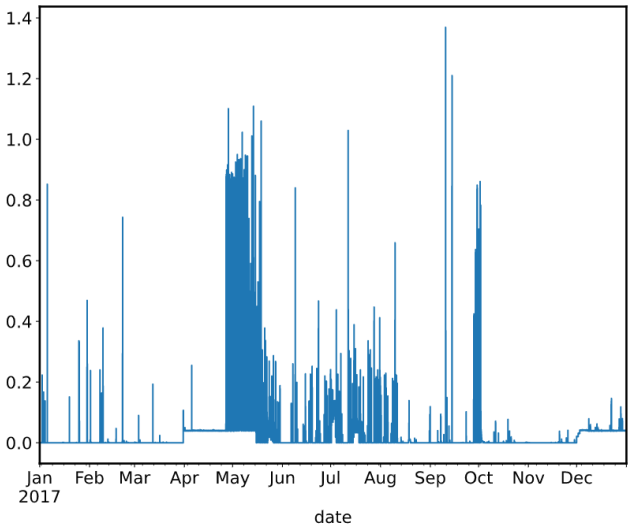


Figure 2.2: One of the 9 identified meters with multiple zero daily consumptions

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.

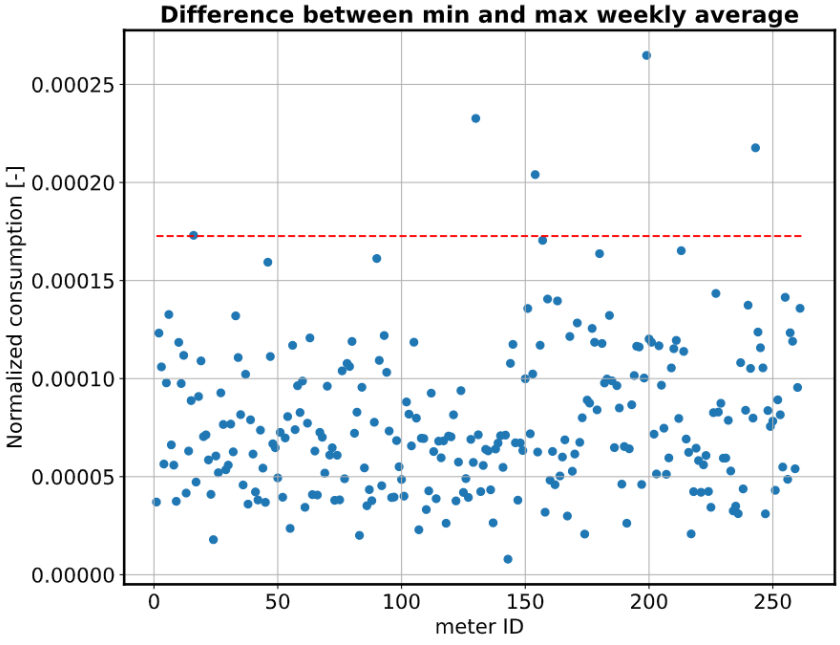


FIGURE 2.3: The maximum differences between the minimum and maximum weekly rolling averages for all the different time-series.

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

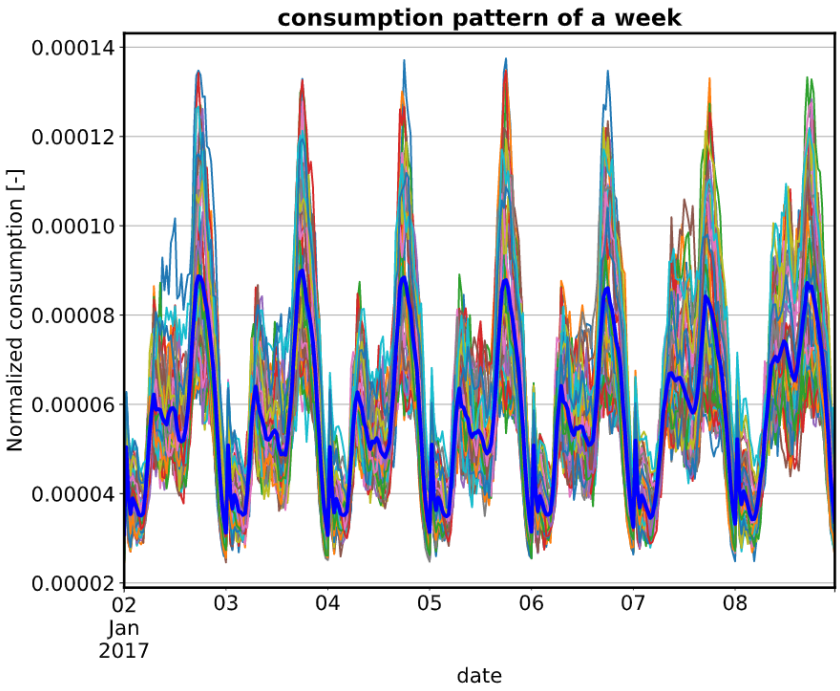


FIGURE 2.4: The seasonality of the electrical load during the week. The blue line shows the average week over all weeks in 2017.

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.
Add here more specific how the error is calculated between holiday and weekday –> MAE and normalized.

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

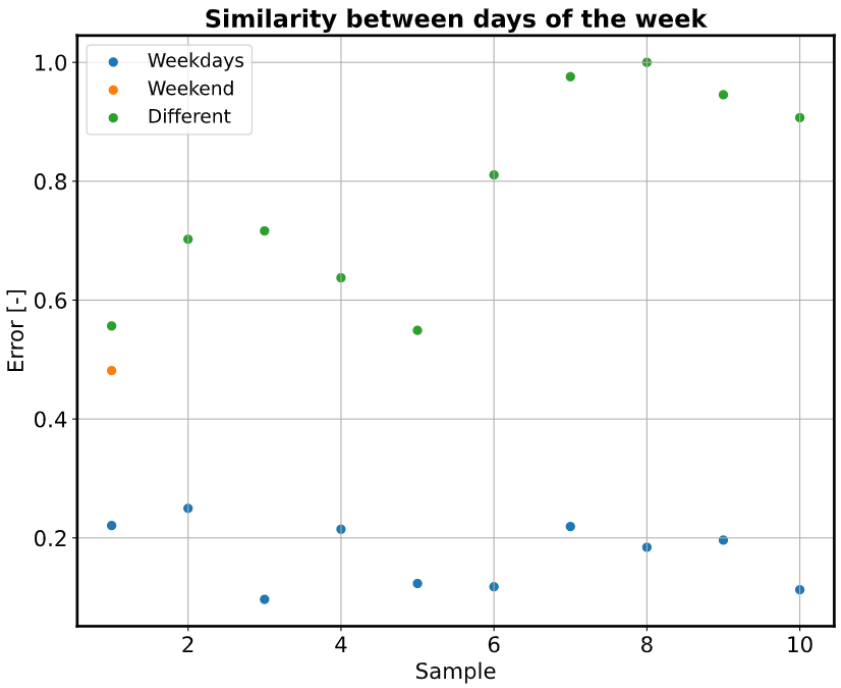


Figure 2.5: Error between different pairs of weekdays.

weekend day. The error is calculated as discussed in section 2.3.2.

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.

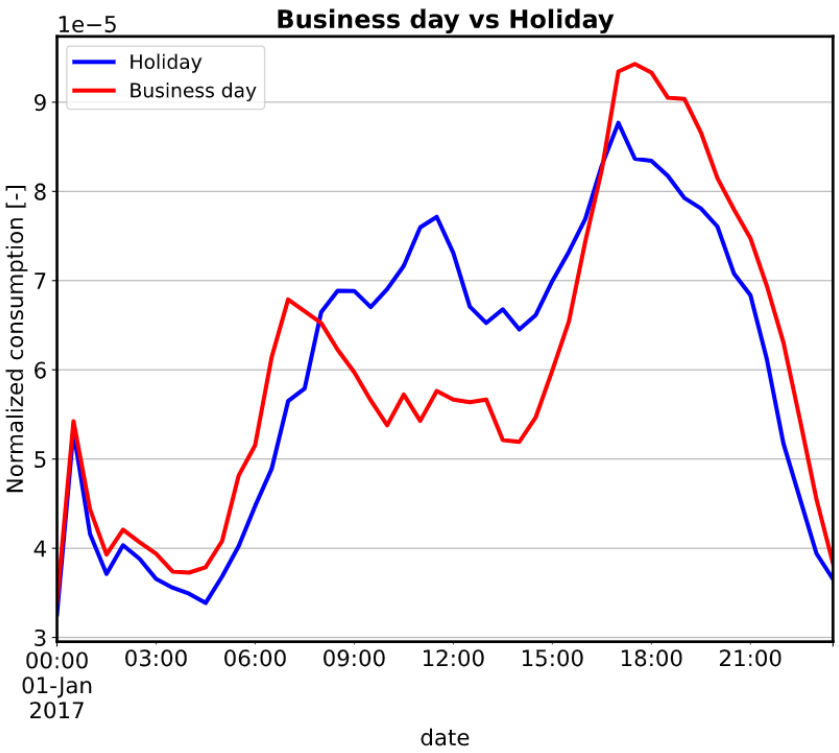


FIGURE 2.6: Figure with the comparison between holidays and business days.

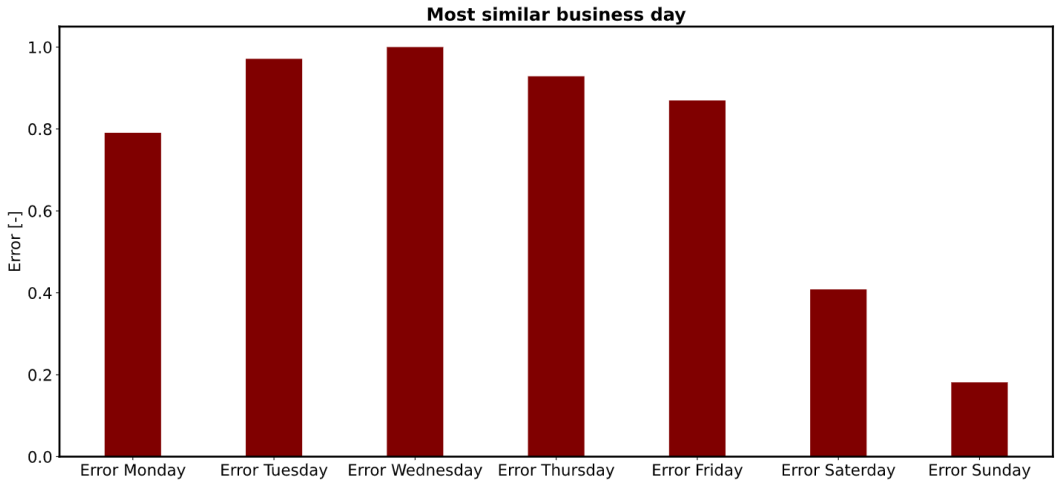


FIGURE 2.7: Error between a holiday and other days of the week.

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 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.

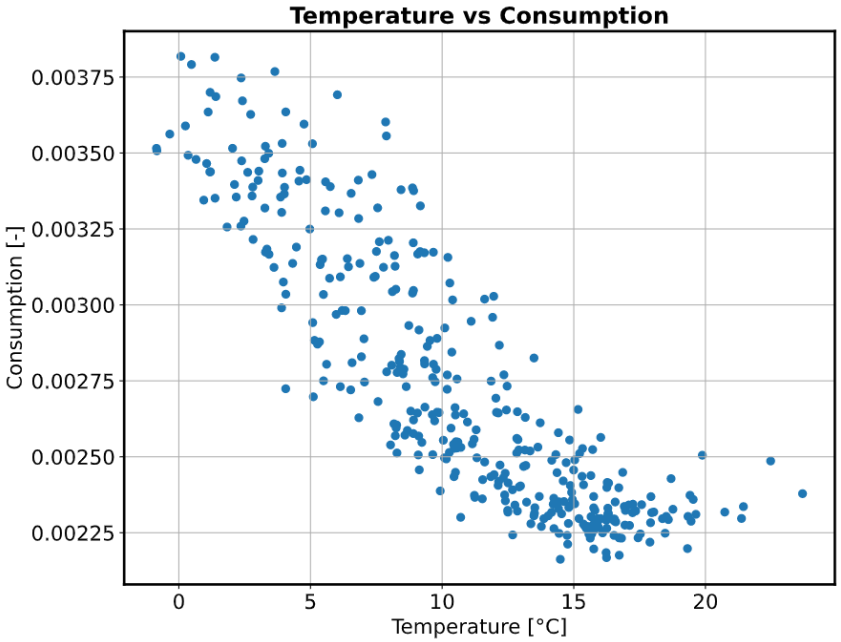


Figure 2.8: Relation between normalized daily consumption and daily temperature.

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

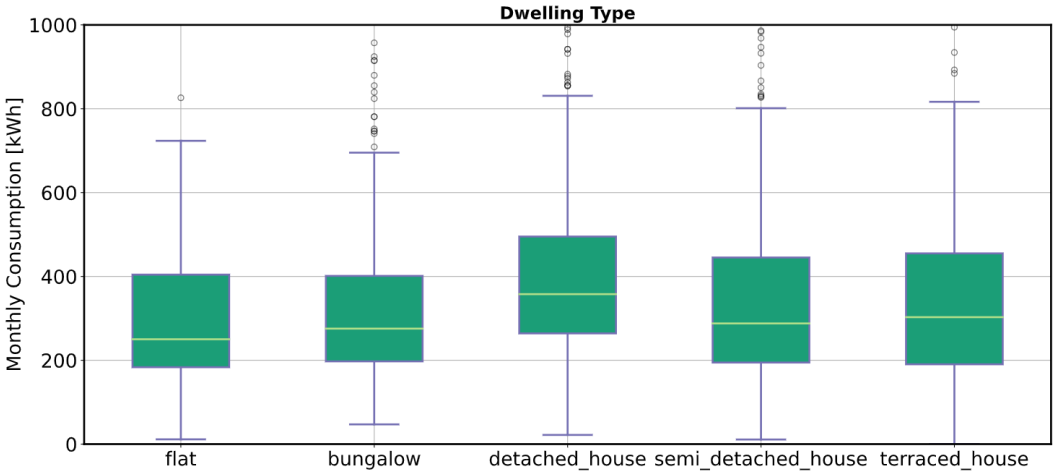


Figure 2.9: Figure with the comparison of the different dwelling types.

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 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.

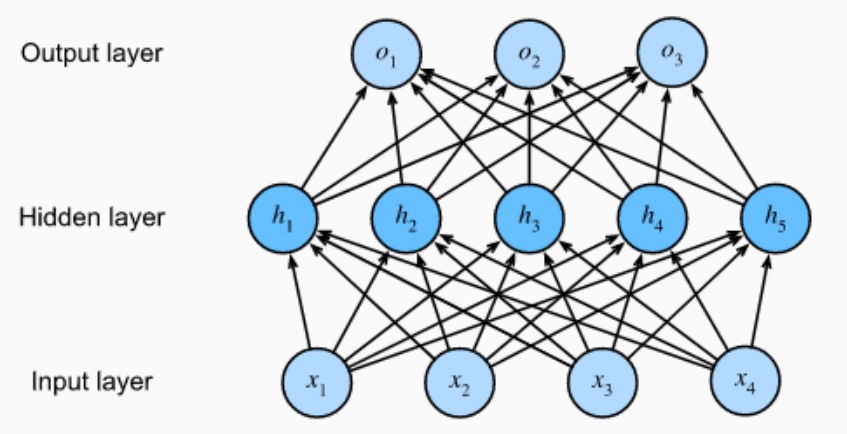


Figure 3.1: Figure of a MLP (source [18]).

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.

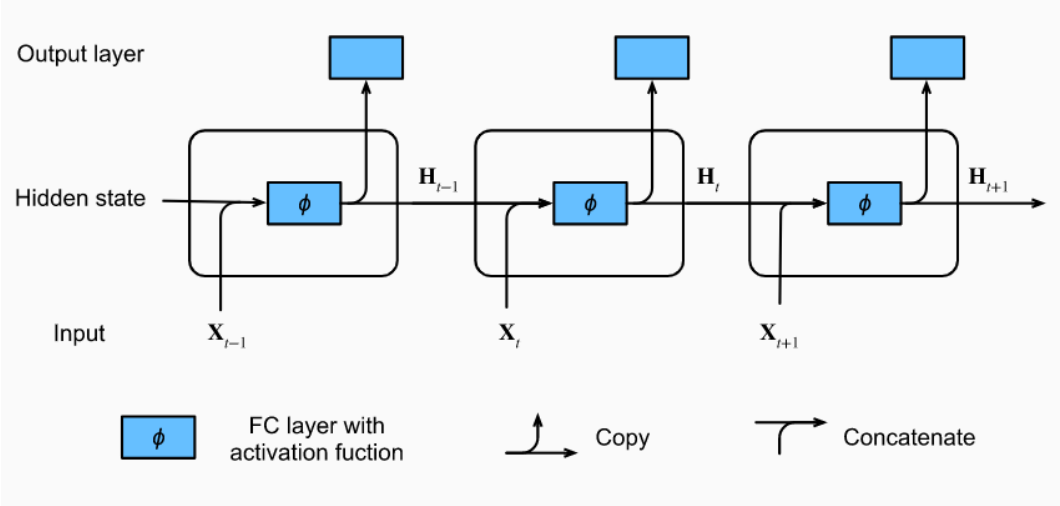


FIGURE 3.2: Figure of the logical flow of a vanilla RNN with a hidden state (source: [18]).

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 [17]

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 [17]. 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 [17], 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.

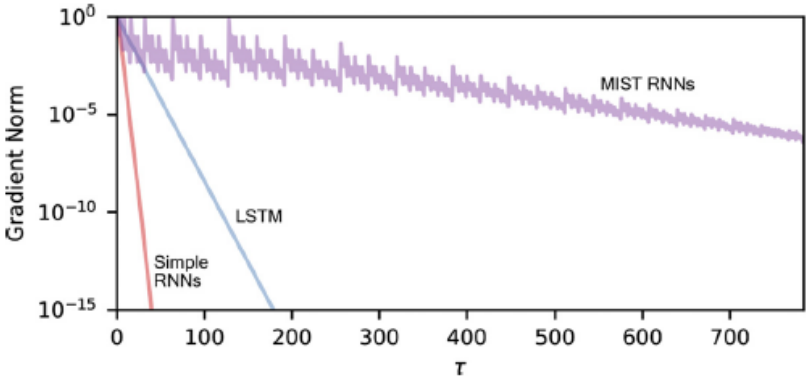


FIGURE 3.3: Exponential decrease of the gradient size of a simple RNN (red) or a LSTM (blue) (source: [17]).

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

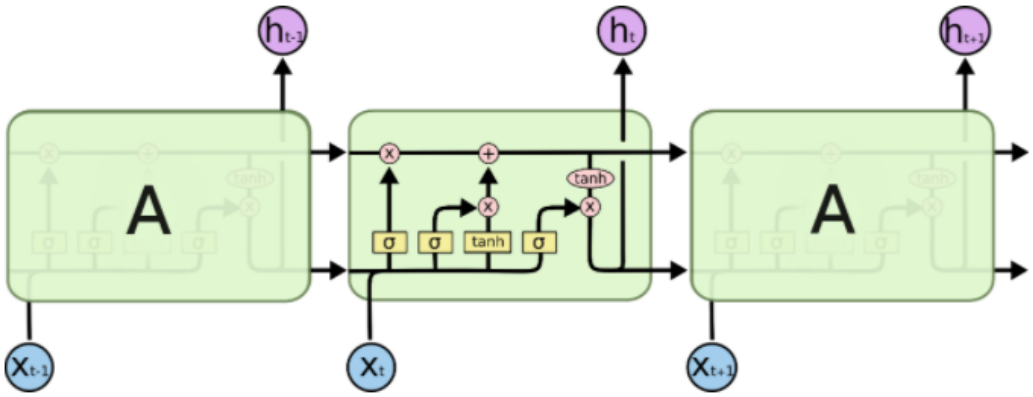


Figure 3.4: A LSTM cell that is repeated over time (source: [12]).

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 [17]:

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

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

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

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

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

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

3.1.6 GRU

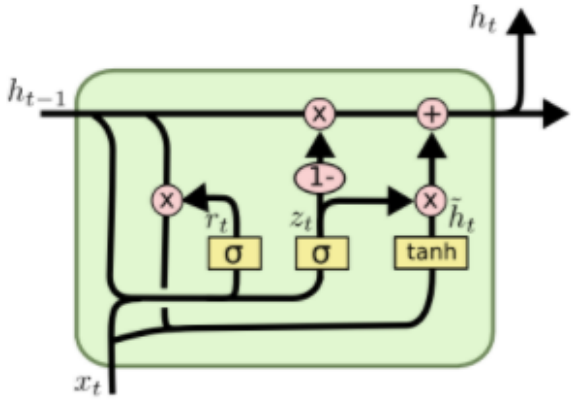


FIGURE 3.5: A GRU cell that is repeated over time (source: [12]).

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 [17]:

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

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

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

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

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 [17]. Stochastic gradient descent means that the approximated gradient is calculated from a random subset of

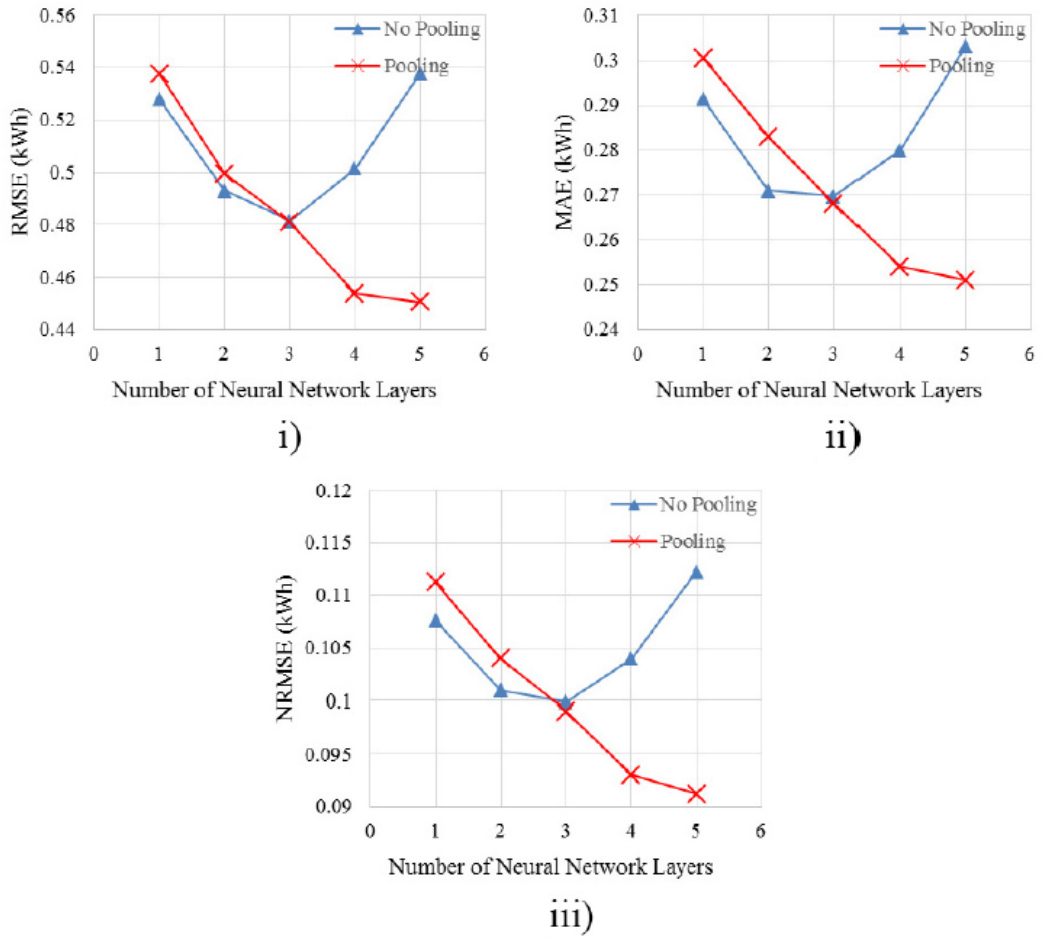


FIGURE 3.7: Influence of the number of layers and the pooling method used in [14].

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

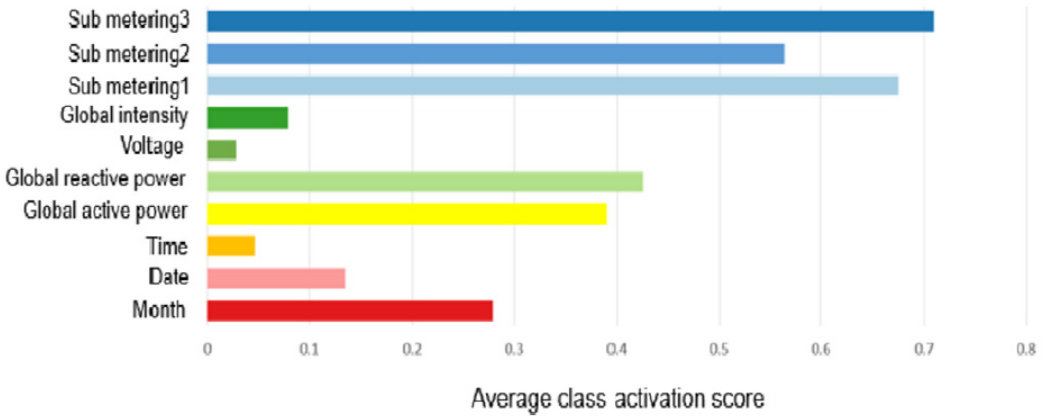


Figure 3.9: The importance of the different inputs as based on the average class activation score. (source [8])

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.

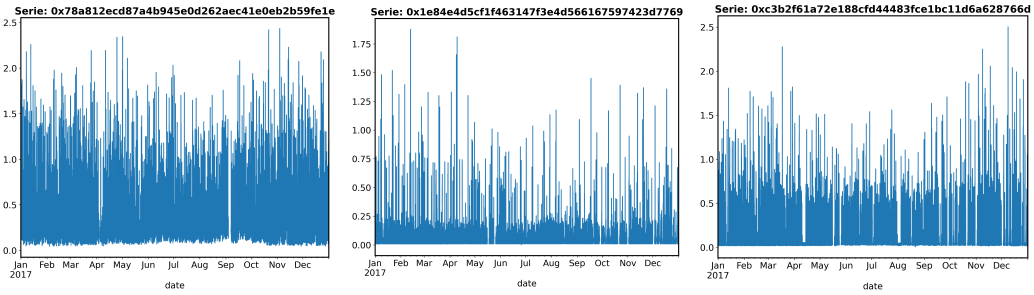
Chapter 4

Forecasting the daily electricity consumption

In this chapter different forecasting techniques to perform 24 hours ahead predictions for individual household electrical consumption are discussed. The time series have a half hourly frequency, which means that 48 data points have to be estimated during each prediction of a day. The day we want to forecast is further indicated as the "desired day". First the raw data is introduced and preprocessing steps done are explained in Section 4.1. Section 4.4 presents the baseline models. These models are implemented using a straightforward approach. They serve as a benchmark for more complex models in Section 4.5. "Long Short-Term Memory" neural networks are most suitable to process time series and have therefore been chosen as the core model which is analysed with different design choices. Finally, a parameter search is conducted which consists of analysing the best parameters for optimal results.

4.1 Preprocessing

The available raw data is summarized by Table 2.1. Three series are selected from the *consumption.csv* for which the important characteristics are summarized by Table 4.1. The chosen time series have the least amount of missing data and don't contain large shifts of the mean consumption during the year. Figure 4.1 shows the electrical consumption of the three selected series. The validation set used during the parameter search consists out of the 10 last days in November. Only 10 days are selected to reduce the amount of predictions the models have to calculate. Also, these days don't contain any missing values for the three series. The months January till November compose the training set and December is taken as test set.
The corresponding average temperature series are also used. They don't contain any missing values.
During this chapter the series containing the electrical consumption are not aggregated as was the case in Section 2.3. Therefore, the min-max normalization is a suitable normalization method for each serie. It is chosen to estimate the missing values in the training set and not leave them out. In other words, a larger trainings



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

Figure 4.1: The electrical consumption in 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 (21-30 November)	10	10	10
Days in Test set (December)	31	23	23
Days in training set (Rest)	320	307	306
Mean average temperature [$°C$]	10.37	10.61	10.22
Standard deviation average temperature [$°C$]	5.09	5.20	5.01
Median average temperature [$°C$]	10.46	10.61	10.35
Maximum average temperature [$°C$]	23.99	24.51	22.95
Minimum average temperature [$°C$]	-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.

set with an additional estimation error is preferred. By imputing the missing values more training data is available but also time jumps are avoided. As discussed earlier, data that is missing are always complete days. These are estimated using baseline models that will be discussed in Section 4.4 in following order: "previous week forecast", "previous day forecast" and "mean forecast". When the first baseline model can't make a prediction for the missing day i.e. the day during previous week is not available, the next baseline model is used and so on.

4.2 Error metrics

The metrics that can be used to evaluate the performance of predictions are RMSE (Eq. 3.14), NRMSE (Eq. 3.15), MAE (Eq. 3.16), MSE (Eq. 4.1) and MAPE (Eq. 4.2).

$$MSE = \frac{\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}{N} \tag{4.1}$$

$$MAPE = \frac{\sum_{t=1}^{N}|\hat{y}_t - y_t|/y_t}{N} \tag{4.2}$$

It is expected that the *MAE* penalizes outliers less severe than *MSE*, which takes the error squared. Therefore, using the *MSE* metric, more emphasize is put on predicting the peaks of the reference signal correctly with respect to *MAE*. The advantage of using RMSE and MAE is that they both have an error in kWh which is intuitive. NRMSE and MAPE take the reference signal into account. NRMSE gives a bigger error, when the reference signal has a small difference between the y_{max} and y_{min}. MAPE penalizes a small MAE more severe on a small reference value than on a larger one due to the division by y.

4.3 Microsoft Azure cloud

The calculations executed in this chapter are performed on a virtual machine using the Microsoft Azure services. This was possible because Microsoft provides students with $100 of free credit. Table 4.2 shows the different features of the CPU/GPU's used during this thesis.

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 used.

To help transferring the calculations to the cloud, free tutorials hosted by Microsoft Azure are available. Before python scripts can be run a workspace and computation cluster have to be set up. It is also important that a correct python environment is set on the virtual machine. This can be accomplished using a YAML file where commands are given which packages should be installed and which channels should be used by Conda to find these packages. Conda is part of Anaconda and is an open source package management system. It differentiates with Pip in following points according to the Anaconda documentation:

- Conda can install packages of any language and not only Python.
- Conda makes it easy to create isolated environments containing different Python versions and packages.
- Pip installs dependencies in a serial loop while Conda fulfils all package dependencies simultaneously using a SAT-solver.

However, it is sometimes still necessary to use Pip because this can be the only option when installing a package. Using an isolated environment on a local machine makes it easy to transfer this environment automatically to a YAML file that is send to the virtual machine.

Finally, it is noted that while performing the calculations a warning occurred that the TensorFlow binary was not compiled to support CPU instructions as SSE4.1 SSE4.2 AVX AVX2 AVX512F FMA. This means that the binaries in the tensorflow package are from a different machine and that the CPU used to run the calculations on this moment, is not able to use all its features. This warning was not fixed, but is given as a possible suggestion to the reader to improve calculation speeds when reusing the developed Python scripts.
- give also an indiction of the model time –> MAPE needs long to train - because of the simplicity of the baseline models –> don't have a validation set–> data of training and validation are taken together. - give examples

4.4 Baseline models

As was discussed in the introduction of the chapter, the baseline models use a straightforward approach to make a forecast for a desired day. Therefore, they serve as a benchmark for more complex models that are explained in Section 4.5. Theses are the different baseline models that are considered:

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

The models consider all the days preceding the desired day to be training data. As will be seen in Section 4.5, the prediction signal for a day is calculated one value at a time and the total 48 values are calculated recursively. This consecutive predicting will not apply for the baseline models. Here, all the 48 values are calculated at once, meaning one full day ahead. In the case of Model 5: "MAPE forecast", the predictions are done consecutively but the values that are calculated are not used

during next predictions.

Model 1: "1 day ago forecast"
This model considers the consumption values of the day preceding the desired day as the forecast for the desired day. For example, the 24 hours ahead prediction starting from Tuesday 19/12/2021 00.00 am is taken equal to the 48 consumption values that are recorded between Monday 18/12/2021 at 00:00 am and Tuesday 19/12/2021 at 00:00 am. The philosophy of the model is that the most recent consumption data serves as a a good prediction. However, one can think logically and know that the most recent data is not guaranteed to behave the same. For example, as was found in Section 2.3.2 the consumption of a weekend day behaves much different than the consumption of a weekday. Therefore, violating the assumption that the most recent data is also the most adequate to use as prediction. It is expected that people have a reasonable routine and it is likely that this routine will also be present in the electricity consumption. Therefore, from a human behaviour perspective it makes more sense to use the same day the previous week as prediction for the desired day.

Model 2: "7 days ago forecast"
This model tries to capture the human weekly routine and therefore it takes the same weekday of the previous week as the prediction for the desired day.

Model 3: "Closest day forecast"
This model looks for a past day that would be representative to serve as prediction for the desired day based on following features:

- Day of the week.
- Holiday
- Average temperature

If 24 hours ahead of 19/12/2017 at 00:00 am has to be predicted, the following calendar information can be extracted from this time stamp : it is a weekday, more specifically a Tuesday and it is not a holiday. From the exogenous variable, it is known that the average temperature was 12.6°C. From the training set, which means, all the days before 19/12/2017, select all the previous Tuesdays which are not holidays, and find the day with the closest average temperature based on euclidean distance. It could be the Tuesday 10/01/2017 with an average temperature of 11.4°C for example. The consumption values of Tuesday 10/01/2017 day are chosen to serve as prediction for the desired day, namely Tuesday 19/12/2017.
It should be noted that this method expects a good prediction of the average temperature of 19/12/2017. As can be seen in Table 4.1, no temperature values are missing for the three selected series. This table also shows that there are only 8 holidays. To increase the amount of days when the desired day is a holiday, also all Sundays are considered. This is according what was found in Section 2.3.3 where

Figure 2.7 shows the normalized MAE between a holiday and a Sunday is the smallest.

Model 4: "Mean forecast"
This model again uses calendar information to group the same kind of days as was done for Model 3. For example, if the desired day is 19/12/2017, all the previous Tuesdays which are not holidays, are selected. Instead of looking to the most similar day based on the closest average temperature in euclidean distance, the mean is calculated and used as prediction for the desired day. In this model no extra Sundays are used when the desired day is a holiday.
The choice of this model is inspired by [9]. It is expected that it can identify frequently recurring behaviour. The model can be able to identify trends as for example an increased consumption around 8 am and 6 pm as shown in Figure A.6, but the performance on predicting correct peak values is expected to be lower due to the averaging of a large variation of peak sizes.

Model 5: "MAPE forecast"
This model solves a non-linear optimization problem for each step to be predicted. The objective function 4.3 should be minimized to obtain an estimated load $\hat{y}$. To obtain a 24 hour ahead prediction , 48 predictions are consecutively calculated. This model served as baseline model in [9]. The same type of days as the desired day is selected as was done in Model 3. When calculating a prediction for time t of the desired day, an empirical probability mass function is derived using all the available historic predictions of time t originating from the selected days.

$$\epsilon_t(p) = \sum_{i=1}^{K} \zeta_t(p_i) \left| \frac{(p - p_i)}{p_i} \right| \tag{4.3}$$

Eq. 4.3 shows the non-linear optimization with ϵ_t the MAPE expectation for time t, $\zeta_t(p_i)$ an empirical probability mass function in function of the p_ith possible discretized consumption value and p a load value within the empirical range. $\zeta_t(p_i)$ is calculated by making a histogram with the "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 p_i is taken as the midpoint of two bin edges. From the count in the histogram, the value of the probability mass function for each discretized consumption value is found. The prediction value p, at time t of the desired day, is found by solving Eq. 4.4. To solve the optimization an IPOPT solver is used in a casADi software environment. A practical consideration of using this model is that the calculation load due to solving each time an optimization is higher than in the previous models.

$$\hat{y} = \underset{p}{\operatorname{argmin}}(\epsilon_t(p)) \tag{4.4}$$

4.4.1 Results of baseline models

The performance of the baseline models on the different test sets with respect to different error metrics is given by Tables 4.3, 4.4 and 4.5. Only the days of December where all models could produce a prediction for are included during the calculation of the error. The models with the worst and best performance are respectively indicated with red and green.

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: Baseline results for Serie 1 tested on 31 days of December.

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: Baseline results for Serie 2 tested on 12 days of December.

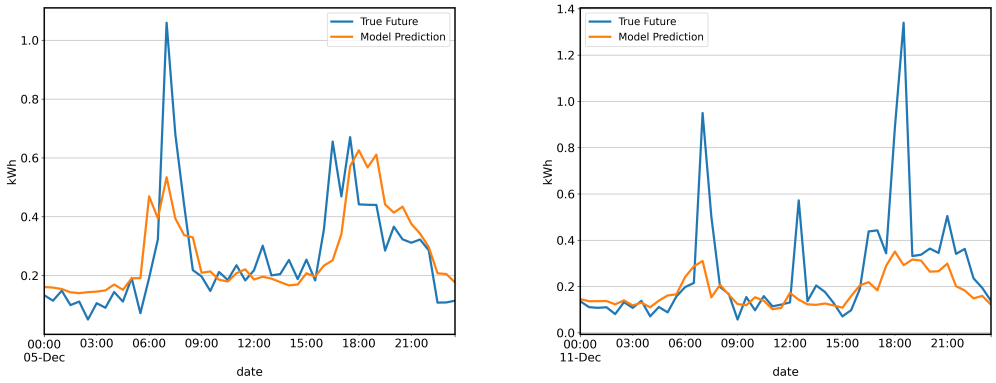
Error metric	**Closest day**	**1 day**	**7 days**	**mean**	**MAPE**
Mean absolute error	0.1267	0.1370	0.1323	0.1038	0.1130
Mean squared error	0.0824	0.0846	0.0895	0.0521	0.0743
Normalized root mean squared error	0.1453	0.1472	0.1514	0.1155	0.1380
Root mean square error	0.2871	0.2909	0.2991	0.2282	0.2726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

TABLE 4.5: Baseline results for Serie 3 tested on 12 days of December.

In the tables it can be seen that the "mean forecast" outperforms all other models on all the different error metrics except for MAPE. For all three series the "MAPE forecast", performs best on the MAPE metric. This is a logically result because this model is optimized to reduce the MAPE. It is observed that the "closest day forecast" together with the "mean forecast" or "MAPE forecast" perform second and third best for Serie 2 and 3. "1 day ago forecast" and "7 days ago forecast" perform worst for these series. This ranking of models is not valid for Serie 1. Here, the "closest

day forecast" performs overall as one of the worst models. It can thus be concluded that the performance of the models is certainly serie dependent. Because "mean forecast" performs best on all the different error metrics except MAPE and "MAPE forecast" gives the lowest MAPE, both models are included in the comparison with the more complex models of Section 4.5 in Chapter 5.

By inspection it is observed that model 1, 2 and 3, who copy the consumption of another day, show more peaks in their prediction than model 4 and 5. Predictions during the test set of models 4 and 5 can be seen in Figure 4.2. A practical downside of models 1 and 2 during calculation of the predictions for the test set was that sometimes no prediction is possible due to the missing of the day one day or week ago in the reference signal.



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

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

In Table 4.6 the overall performance of the baseline models on all 261 time series that contain a full year of measurements in the "consumption.csv", is given. The MAPE metric is not used because 99 series contain reference values of zero, which leads to a division by zero according to Eq. 4.2. Relative performance of a model on one serie is assessed by normalizing the error by division by the biggest error produced by one of the models. Therefore, the error of each serie is a value between one and zero and the model that performed worst will have an error of one. Finally, an average value is calculated over the different time series for each model. The closer the calculated value is to one, the more often this model performed worst with comparison to the other models. By applying this normalization, the NRMSE leads to the same result as RMSE. It is observed that the order of model performance based on the three metrics in Table 4.6 over all the time series is: "mean forecast", "MAPE forecast", "7 days ago forecast", "1 day ago forecast" and finally "closest day forecast". From Table 4.6 it is concluded that "mean forecast" is expected to give

the best results on a randomly chosen time serie.

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 with a full year of measurements.

4.5 Deep LSTM models

In this section deep LSTM models are presented in different variations. LSTM models are specialized in time series learning as discussed in Section 3.1.5. The term "deep" refers to multiple LSTM layers that are stacked on top of each other. The goal of the developed models is to make a 24 hours ahead forecast of the electrical consumption of an individual household. The advantage of using a "deep" architecture is that it can learn load features hierarchically as discussed in [14]. This means that load features learned in a higher layer will be the combinations of features learned in lower layers.
First, practical considerations during the implementation of the models using Keras are discussed. Keras is a neural network library in Python which can be backend by Tensorflow which is an library for a number of various tasks in machine learning. This section covers the used inputs, batch size, choice between a stateless and stateful model, initialization, measures taken to avoid overfitting and the chosen error metrics. Next, the specific LSTM models developed are explained in Section 4.5.2. Finally, the conducted parameter search is described in Section 4.5.5.

4.5.1 Different practical considerations of the models in Keras

Inputs

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

- Historic electrical consumption (min-max normalization)
- Average daily temperature (min-max normalization)
- Day of the week (one hot encoding)
- Time of the day (one hot encoding)
- Holiday (one hot encoding)

How far the model will look back into the history of electrical consumptions during a prediction, is determined by the lag value. This value corresponds to the amount of time steps that will be taken into account. The LSTM model in Keras expects a matrix $\boldsymbol{X}$ as input with three dimensions: sample number, amount of time steps and amount of features. One sample of the $\boldsymbol{X}$ matrix e.g. $(1, lag\ value, features)$, equals a 2D matix of dimensions *time steps* $\times$ *features*. Every column contains a different time serie that serves as input to the model. 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 temperature is only known on daily basis and the frequency of the values on the reference signal is one every thirty minutes, the temperature value is often the same in the second column of the 2D matrix. The values originating from the reference signal and daily average temperature serie are normalized by using a min-max normalization. Every value will now lay between zero and one.
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. Finally, this is also done to indicate if it is a holiday. In total the input matrix $\boldsymbol{X}$ has 59 columns which corresponds to the amount of features. Every sample of $\boldsymbol{X}$ is used for a prediction of one time step of half an hour of the desired day. To make a 24 hours ahead prediction, the total 48 values are calculated recursively. Previous predictions are used in next ones by adding them to the history of electrical consumptions. The assumption is taken that the reference signal is available till the desired day. For example, if a prediction the electrical consumption of 19/12/2017 is desired, it is assumed that the reference signal is available till 18/12/2017 24:00.

Batch size

A batch size has to be specified in the Keras fit function. The batch size determines how many samples of $\boldsymbol{X}$ are feeded to the model at once and their outputs compared with their reference to define an objective function. This objective function is used to calculate the gradients for weight updates. Because every sample of $\boldsymbol{X}$ outputs one prediction value, the amount of samples of $\boldsymbol{X}$ correspond to the amount of predictions made.

Stateless versus Stateful

Keras introduces a concept of stateless and stateful. As explained in Section 4.5.1 the input given to a LSTM model is a matrix with three dimensions: sample number, time steps, features. As explained in [3] a stateless model interprets every sample of

$\boldsymbol{X}$ as if they have no relations with each other. A stateless model is not able to see that the different samples of $\boldsymbol{X}$ originate from the same time series. Therefore, the hidden and memory states are reset when a new sample of $\boldsymbol{X}$ is considered during model training and prediction. When the hidden states and memory states are reset, they are again initialized to zero vectors. If there is no relation between the different samples of $\boldsymbol{X}$, it is logical to collect as much samples as possible and the input data is generated by moving a window every time one step further on the original time series as can be seen in Figure 4.3.

A stateful model doesn't reset its hidden and memory states when considering a new sample of $\boldsymbol{X}$. Because of this an additional condition is imposed to the input data that now, when all the samples of $\boldsymbol{X}$ are glued behind each other, has to return the original times series sampled from. If this condition is not fulfilled, the hidden states and memory states would traverse a different serie than the original one. After one epoch during training the states are reset because the end of the original time serie has no connection with its start. During predictions the hidden and memory states are also preserved. Therefore, seeding of the model before prediction is possible as explained in Section 4.5.4.

The advantage of using a stateful model in comparison to a stateless one, is that during training and prediction more data can be presented to the hidden and memory states before they are reset. Therefore, the capability of the LSTM of long term remembering important features of the data, is more exploited. However, a practical disadvantage of using a stateful model is that Keras expects that the batch size chosen during training, equals the one during prediction. Each prediction uses one sample of input matrix $\boldsymbol{X}$ to output one prediction value. This means that the batch size during training is forced to be equal to one. It was tried to circumvent this by first training on a model with chosen batch size and then transfer the learned model weights to a new, identical model with batch size one which would be used during the prediction. However, after testing it was found that the *get_weights()* and *set_weights()* commands in Keras didn't reproduce the same prediction results. 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. In literature it was found that setting the weights on different models is possible when using the underlying Tensorflow library, but not in Keras. In addition, a stateful model expects that the training and validation sets are divisible by the batch size. To satisfy this condition samples of the input matrix can be thrown away. As will be seen in Section 4.5.4, it was chosen to set the batch size of the developed stateful model 3 equal to one which is called "online learning" in literature. Both the previous conditions are then fulfilled.

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.

It was chosen to set the batch size issues for a stateful model by setting the batch size

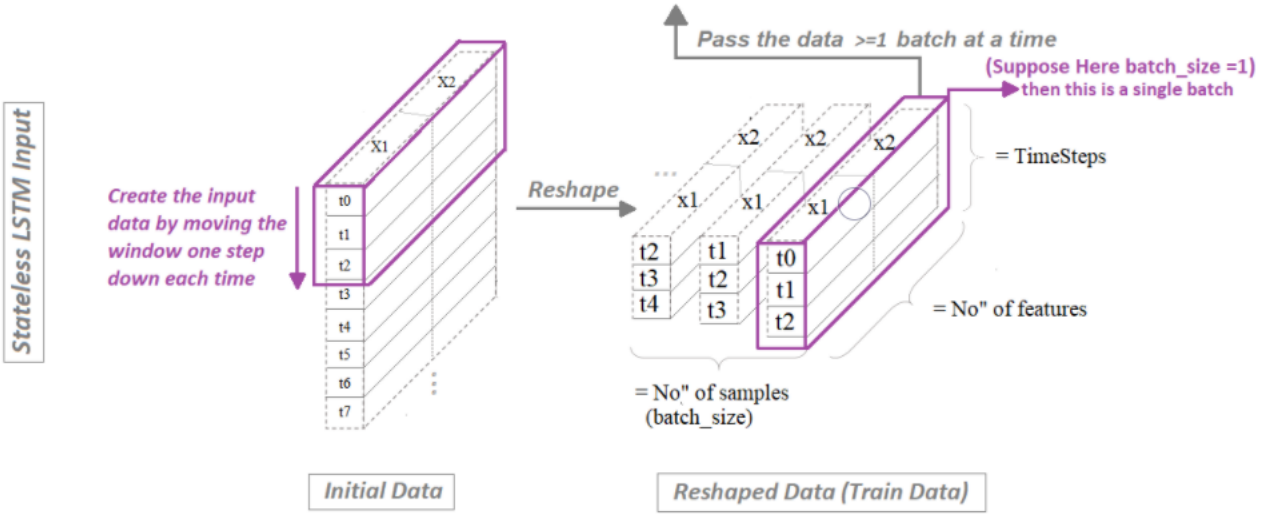


Figure 4.3: The generation of inputs for a stateless model. (source: [3])

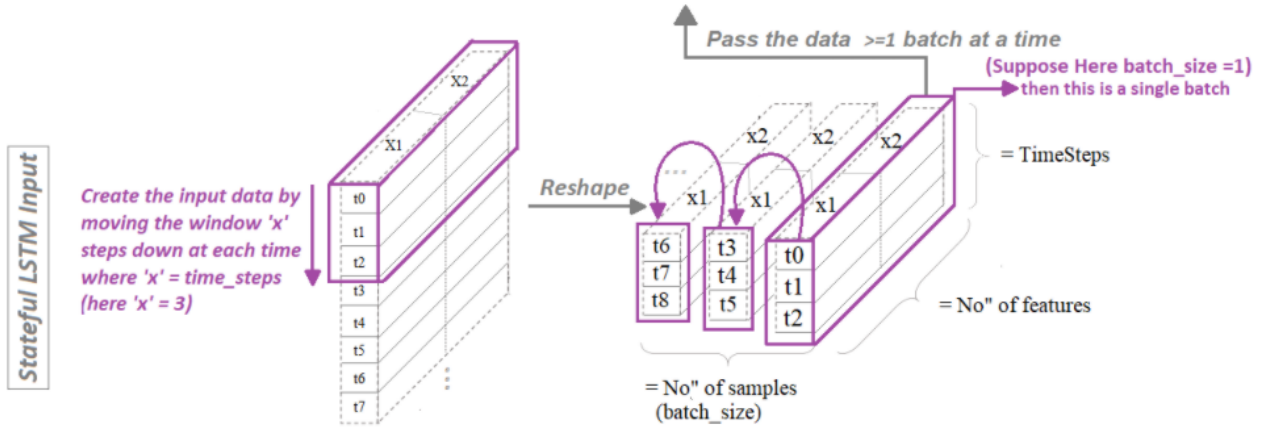


Figure 4.4: The generation of inputs for a stateful model. (source: [3])

Initialization

The LSTM equations in Section 3.1.5 show four weight matrices $\boldsymbol{W}$ with an index H and four weight matrices with an index X. These are respectively the recurrent and kernel weights and they are both initialized in a different way. A recurrent weight matrix is initialized using an orthogonal initialization and the kernel weight matrix uses a 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 equations are all initialized by arrays filled with zeros.

Overfitting avoidance in Keras

Different ways to avoid overfitting can be applied in Keras::

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

Error metrics

The MAE metric is used to evaluate the prediction performance on the validation set during the parameter search in Section 4.5.5. During training the MSE is used in the objective function which is used to derive weight updates.

4.5.2 Deep LSTM

First, the input data of the LSTM is calculated in *input_output_LSTM*. This is done before training using the GPU that is listed in Table 4.2. The inputs that are used are discussed in Section 4.5.1. 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. 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 taken as validation set, shuffling should be done manually before the inputs are assigned to the Keras fit function according to Deeplizard's blog page. During prediction, every model will predict one value at at time as was proposed as a good methodology in [16]. This means that 48 predictions are needed for a 24 hours ahead prediction. 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 4.5.5.

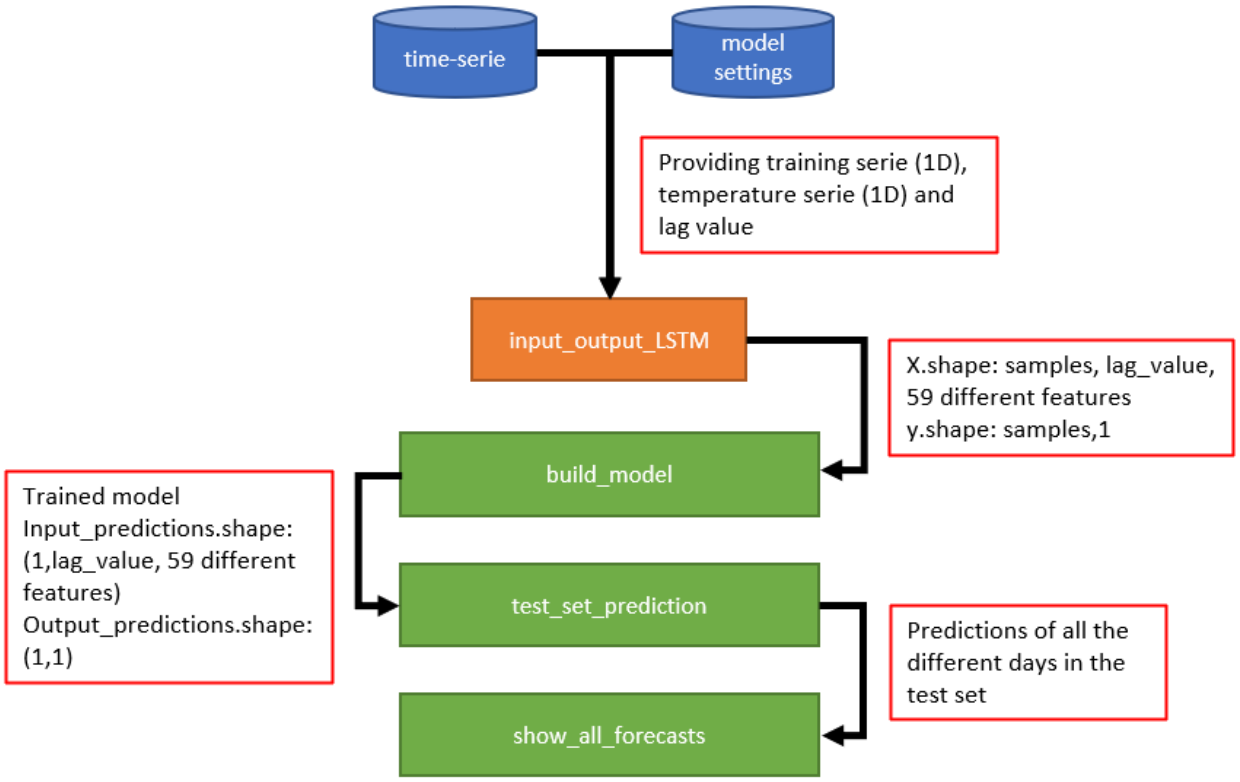


Figure 4.5: The flow of functions that are executed in order during the prediction process with LSTM models.

Model 1: Stateless with no flatten layer

One sample of the input matrix $\boldsymbol{X}$ consists of N timesteps which are feeded to the LSTM layer as is shown in Figure 4.6. N equals to the chosen lag value. 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 as is explained in Section 4.5.1. Because Model 1 is stateless $\boldsymbol{C}_0$ and $\boldsymbol{H}_0$ will be initialized as zero vectors. 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 amount of layers and hidden recurrent states are chosen, all the three models will have the same amount of trainable parameters.

4.5.3 Model 2: Stateless with flatten layer

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

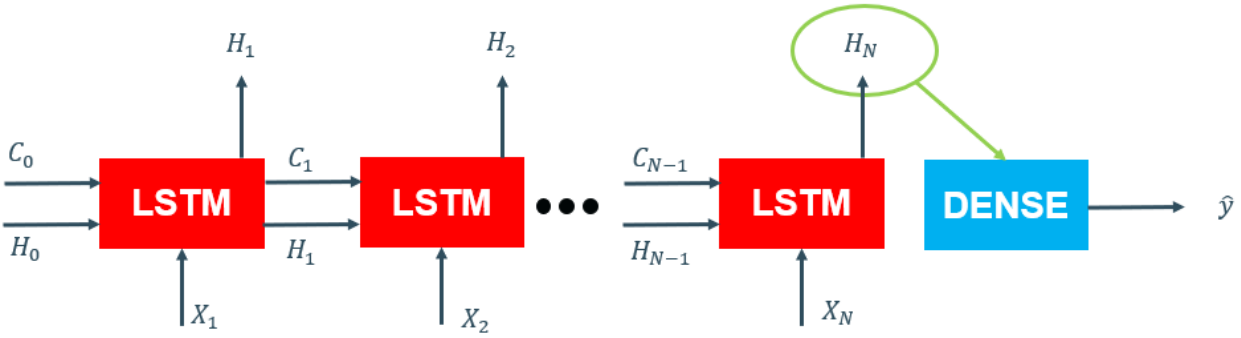


FIGURE 4.6: Model 1 - stateless model with as input a subserie of N time steps and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^{59}$, $\hat{y} \in \mathbb{R}^1$.

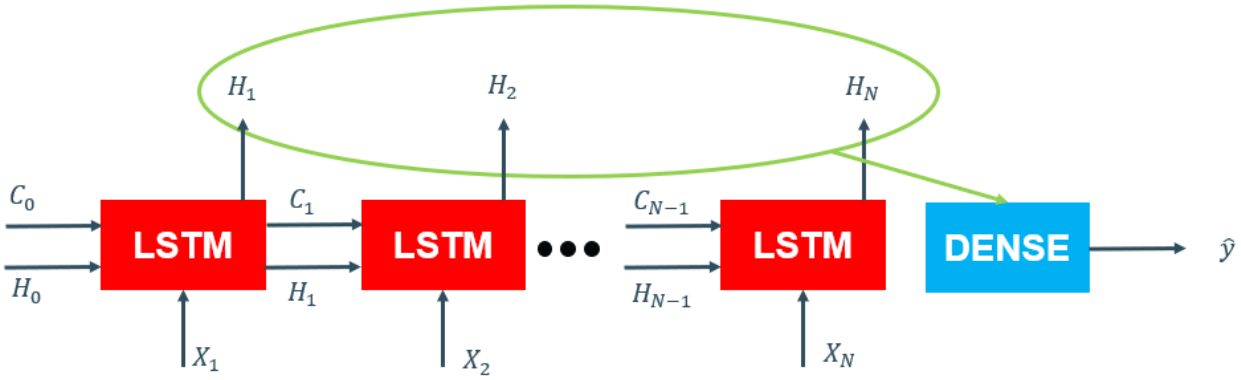


FIGURE 4.7: Model 2 - stateless model with as input a serie of N time steps and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^{59}$, $\hat{y} \in \mathbb{R}^1$.

depicted in [9].

4.5.4 Model 3: Stateful

Where the previous two models only go through a subserie before making a prediction, this model sees the whole original time serie when making predictions. This is possible because the model looks one by one at each sample of the original time serie. For example $\boldsymbol{X}_1$ is feeded to the model and C_1 and H_1 come out. C_1 and H_1 are now again set as C_0 and H_0 when the next sample $\boldsymbol{X}_2$ 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 $(15500, 1, 59)$, so 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. 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

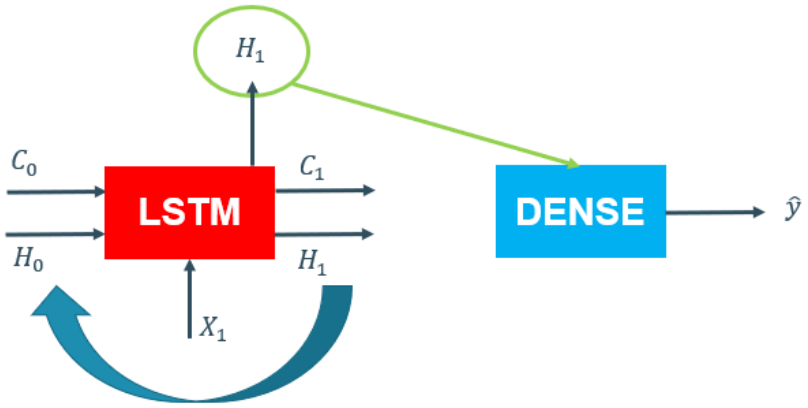


Figure 4.8: Model 3 - stateful model that connects single LSTM blocks and $\boldsymbol{C}_i \in \mathbb{R}^m$, $\boldsymbol{H}_j \in \mathbb{R}^n$, $\boldsymbol{X}_k \in \mathbb{R}^{59}$, $\hat{y} \in \mathbb{R}^1$.

and the hidden states that can be used during the prediction. During the consecutive predicting of a 24 hours ahead forecast, the model only uses previous predictions as input in comparison to Model 1 and 2, who use a serie as input of each prediction with at the end the previous predicted values.

4.5.5 Parameter search

This section will show that the different LSTM models are dependent on the choice of good parameters to perform well. During a parameter search different values for the parameters 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 assumed to depend on each other. If all the different parameters are varied simultaneously the calculation load will rapidly blow up. To deal with this, the parameter search is conducted in three stages. The cost of this reduction in calculation load will be that dependencies between parameters are neglected. During the parameter search every combination 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 values behaves better or worse, is based on the MAE on the 10 last days of November, averaged over three runs. This validation set is used for all three models on all three series. Only 10 days are chosen to reduce the calculation load. An advantage of using this validation set in comparison with cross-validation, is that it can be used for all models. Cross-validation for Model 3, is not possible because it would cause interruptions in time. Inspiration for the values tried during the parameter search is found in [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 stages, the best set of values is chosen and the learning rate is more precisely tuned (calculation time $\approx$ 7 hours)

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.

The values tried for the different parameters are displayed in Table 4.7. A total of 24 combinations is tried during phase one and each combination is run 3 times. The batch size chosen is only one number to make a fair comparison between the different settings 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 see a clear effect when a different amount of LSTM layers is used. Only two epochs are considered to reduce the calculation load and therefore early stopping is not used during the parameter search.

Changes to the parameters in Table 4.7 are made for model 3. In model 3 a batch size of one, a lag value of one and shuffling "False" are used. A batch size of one is chosen because then the prerequisites for a stateful model as discussed in Section 4.5.4, are fulfilled.

In phase two, the assumption is made that a more complex model is able to better capture the non-linear relations in the data and regulation is added to avoid overfitting. The values for the learning rate and lag value that were part of the lowest average MAE during phase one, are selected during phase two. Also, the amount of hidden states is taken equal to 50 and the amount of LSTM layers to 3. The regulation 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 l2-norm of the weights is added in the objective function. This means that big weight values 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 reducing the error on

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.

the training set and the l2-norm 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 set respectively a rate of its inputs $\boldsymbol{X}_k$ and $\boldsymbol{H}_j$ to zero. The dropout rate values that are chosen between 0.2 and 0.5 as is advised in the original paper [15].

It was found in [4], that the learning rate is a very important parameter that has to be tuned correctly to enhance the model performance. Therefore, during stage three a range of learning rates is tried for the best model after inspection of stage one and two. The learning rate which leads to the the smallest MAE on the validation set 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}. The selection of multiple learning rates of size of magnitude 10^{-3}, is conform the parameter search in [14].

In order to speed up the calculations, multithreading is implemented to calculate the parameter search in parallel. For the implementation the Python Multiprocessing library is used. 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 when a total of 4 runs is considered.

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. Because this increased the calculation time, for model three, nan values are also included as training errors.

Model 1: Stateless with no flatten layer

Phase1:
Table 4.9 shows phase one of the parameter search for Model 1, using the parameters listed in Table 4.7. Each value shows a percentage of improvement with respect to the worst value for one parameter for each serie during phase 1 of the parameter search. For example, when 20 hidden LSTM states $\boldsymbol{H}_t$ according to the equations in

Section 3.1.5 are chosen, an average improvement in MAE on the validation set of 12.08% is obtained for Serie 1 with respect to when 50 hidden LSTM states are used during all the runs of the parameter search. Because the choice of 50 hidden states gives an improvement of 0%, these values are left out.
The calculations for Model 1 were done 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 for Serie 2 and 3. 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 using a lag value of 48. A reason for this can be that the day, two days ago, is not very representative for the desired day and the hidden states and memory states of the LSTM don't collect much more valuable information to use during the prediction in comparison to a lag value of 48. It can be argued that the day a week ago would add more value, because of the human weekly routine. In comparison Model 3 traverses the whole historic signal before making predictions. It is therefore expected that the model better captures the weekly routine in the hidden states and memory states that can be used during the prediction.
An unexpected result 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 too complex model. Table 4.10 gives the combination of values for the parameters that performed best during phase one. In the next phase of the parameter search, regulation is added and compared with the results in Table 4.10.

Model 1: Stateless (no flatten layer)				
Chosen parameter	**Value**	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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 corresponding parameter value is used, normalized by the biggest error of the possible values of one parameter and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst value for one parameter for each serie during phase 1 of the parameter search.

Model 1: Stateless (no flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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

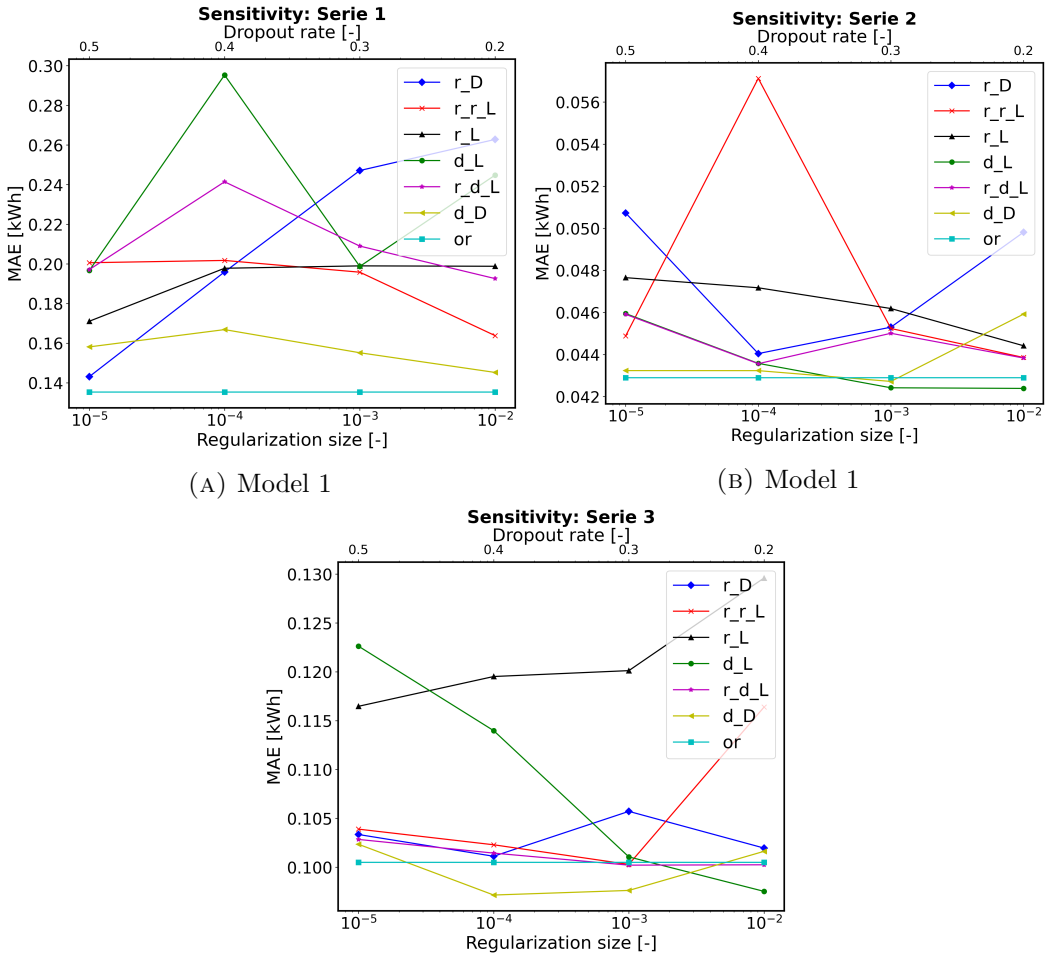
Table 4.10: The values of the parameters with the lowest average MAE on the validation set over three runs.

Phase 2
In this section the learning rate and the lag value out Table 4.10 are combined with the most complex model with regulation. With most complex model, it is meant that 50 hidden LSTM states and 3 LSTM layers are used. A sensitivity analysis is done to look which kind of regulation has the most effect with respect to the MAE. Finally, the results of phase two are compared with Table 4.10. This approach assumes that adding regulation can nullify the use of a possible too complex model. The amount of regularization added is always the same for each of the LSTM layers. The performance is calculated as an average MAE on the 10 last days of November over three runs.

From Figure 4.9 it can be derived that:

- Serie 1: The best setting during phase one according to 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
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 for the learning rate is performed on the best models after phase one and two. Figure 4.10 shows as expected an 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 too a good result and a learning rate that is too small will take a very long time to attain good results and therefore



(A) Model 1

(B) Model 1

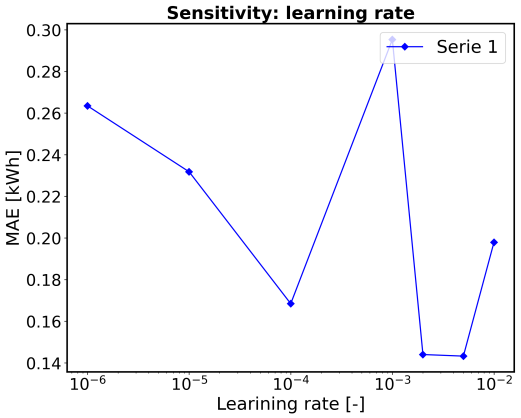
(C) Model 1

FIGURE 4.9: Results of the sensitivity analysis on the size of the regularization parameter and the dropout rate according to MAE. (Legend: *r_D*: regularization size of weights of 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 inputs LSTM, *r_d_L*: dropout rate of hidden states LSTM, *d_D*: dropout rate of DENSE layer, *or*: best performing serie from phase one)

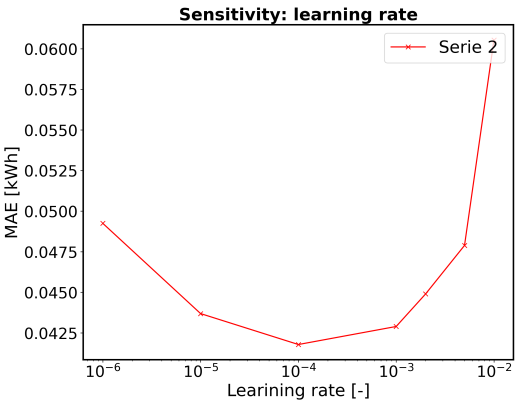
has an increased error when compared on a fixed amount of epochs. The resulting U-shape corresponds to what is found in [4]. The same workflow to conduct the parameter search will be followed for Model 2 and 3.

Final model

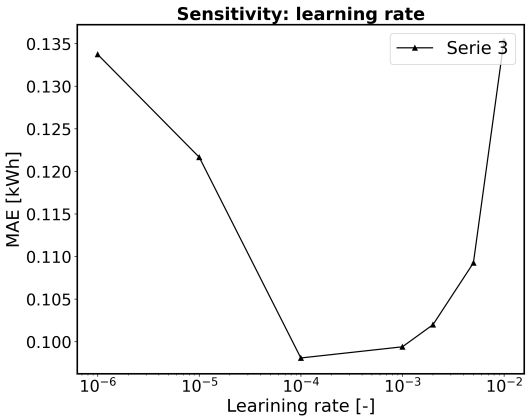
The final parameters for model one are displayed in Table 4.11.



(a) Model 1



(b) Model 1



(c) Model 1

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

Model 1: Stateless (no flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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 found after the parameter search for model 1.

Model 2: Stateless with flatten layer

Phase 1
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 found when a regularization parameter on the input weight 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 found when a regularization parameter on the input weight 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. Again an U-shape is observed.

Final model
The final parameters for Model 2 is displayed in Table 4.12.

Model 2: Stateless (no flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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 weight matrices LSTM	10^{-5}	10^{-3}	0
Dropout DENSE	0	0	0.4

TABLE 4.12: Final values found 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 are updated by comparing each output immediately with its reference. Because this model is stateful, it is first seeded before predictions are done.

Phase: 1
As discussed in the beginning of Section 4.5.5, the values used during phase one of Model 3 differ form the ones in Table 4.7. As can be seen in Figure B.3 the learning rate is still an important parameter to tune. Table B.4 shows the values that give the best results after phase one.

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 found 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 seen that the model when three layers are used together with 50 hidden recurrent states, often produced a very big loss during training. If this is the case, the output of the training loss becomes not a number. Only when a regularization parameter on the hidden state weigh matrices or on the input weight matrices is added the obtained performance was better. The results of the two regularizers are shown in Figure B.4.

For Serie 1 and 3 no improvement with respect to the best results of phase one was found. It is not necessarily the case that the addition of regularization parameters 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 phase two. Only for serie 2 the best result 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. An U-shape is observed for Serie 1 and 3. To find a U-shape for Serie 2, even smaller learning rates have to be considered.

Final model
The final parameters for Model 3 are displayed in Table 4.13.

4.6 Conclusion

This chapter discussed the data to conduct the electricity consumption forecast on and which models were used. First, the baseline models were discussed. It was found that the mean forecast performed best for the error metrics: MAE, MSE, NRMSE

Model 3: Stateful			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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 found after the parameter search for model 3.

and RMSE. "MAPE forecast" performed best when the MAPE error metric was used. 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 LSTM 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 often the most to the model performance.

Chapter 5

Model evaluation

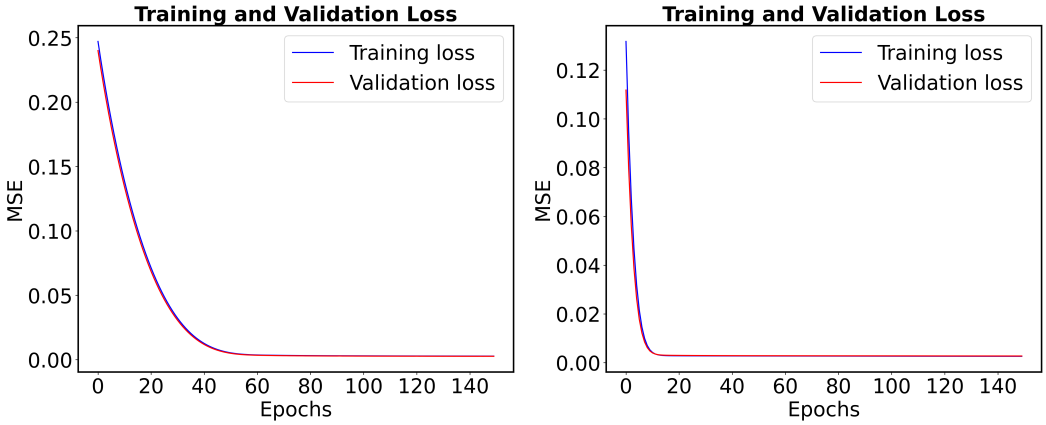
This chapter discusses the performance of the models that were introduced in Chapter 4 on the test set. As was shown in Table 4.1, the test set consists out of the days of the month December. The goal of the test set is to assess the model performance on new data and it is therefore important that the models are not trained and their parameters are not tuned using this data. Missing days in the test set are removed to avoid the influence of the estimation error of the reference signal on the model performance. 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 using the MAE metric is selected. As validation set the 10 last days of November are used. Also, during training early stopping is applied wherefore an additional 10% of the training data is taken to serve as a second validation set. For a stateless model this 10% is randomly taken from the remaining training data. For a stateful model this 10% originates form the end of the training data. 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 set to 150. The parameter values obtained after tuning for the three time series can be found in Chapter 4. Table 5.1 summarizes the amount of epochs that the selected model ran. There can be a big difference in the amount of training epoch and Table 5.1 shows that Serie 2 needs often a lot of epochs. Although, it was found that for both Model 2 and 3 there is not much improvement anymore on the training and validation sets after respectively epoch 60 and epoch 20 as shown in Figure 5.1. Model 3 takes longer to predict than the other two models due to the additional seeding that is required.

	Model 1	Model 2	Model 3
Serie 1	5	16	5
Serie 2	7	150	150
Serie 3	19	11	6

Table 5.1: The amount of training epochs for each selected model.



(a) Model 2 - Serie 2

(b) Model 3 - Serie 2

Figure 5.1: The evolution of the MSE on the training and validation sets.

5.2 Performance on the test set

In this section the results on the test set are discussed for the models that were selected in section 5.1. Also, the two best baseline models "mean forecast" and "MAPE forecast", that were explained in Section 4.4, are included in the comparison. An advantage that the baseline models have in comparison to the LSTM neural networks is that they use the previous data till the day to forecast to make predictions while the LSTM neural networks only trains on data till November. (Seeding not taken into account) This is because it needs data for the validation set to tune parameters and implement early stopping.

The baseline models and the LSTM neural networks both belong to a different group of models respectively to the lazy models and eager models. A lazy model only looks at the data when the query is known i.e. what day to forecast. For example a "mean forecast" looks after it knows which day to forecast to the same weekdays and takes the average. In comparison an eager model already makes generalizations on a training set before it knows which day it has to predict. This applies to the LSTM models.

Model 1 and Model 2 make use of a lag value of 48 or 96 and no seeding. When the day after a missing day(s) is predicted the inputs used during prediction could

originate entirely from an estimated reference signal for the missing day(s). If this is the case, it is expected that the error on the desired day will be larger. The estimation of the reference signal is done by substituting the missing values as described in Section **??**. For both Serie 1 and Serie 2 there are 8 missing days in the test set.

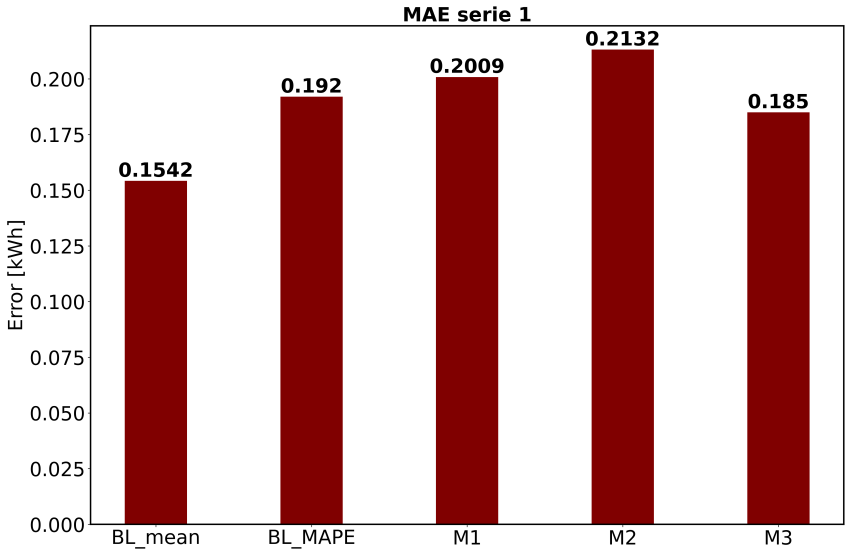


FIGURE 5.2: The MAE performance on all the days of the test set for Serie 1.

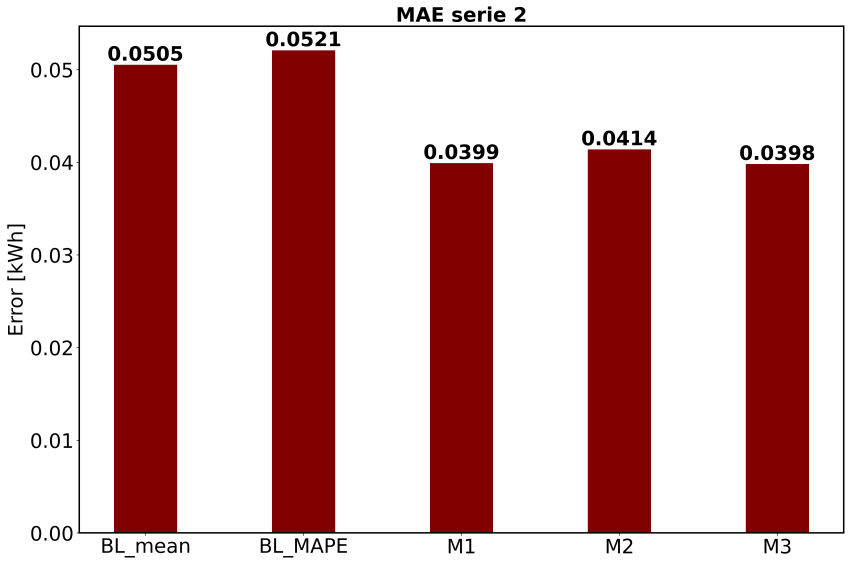


FIGURE 5.3: The MAE performance on all the days of the test set for Serie 2.

Figures 5.2, 5.3 and 5.4 give a comparison of the three LSTM models and the two baseline models. It can be seen that there is a reduction of the MAE for Series 2 and 3 for all the LSTM models and for Serie 1, Model 3 attains a smaller error than

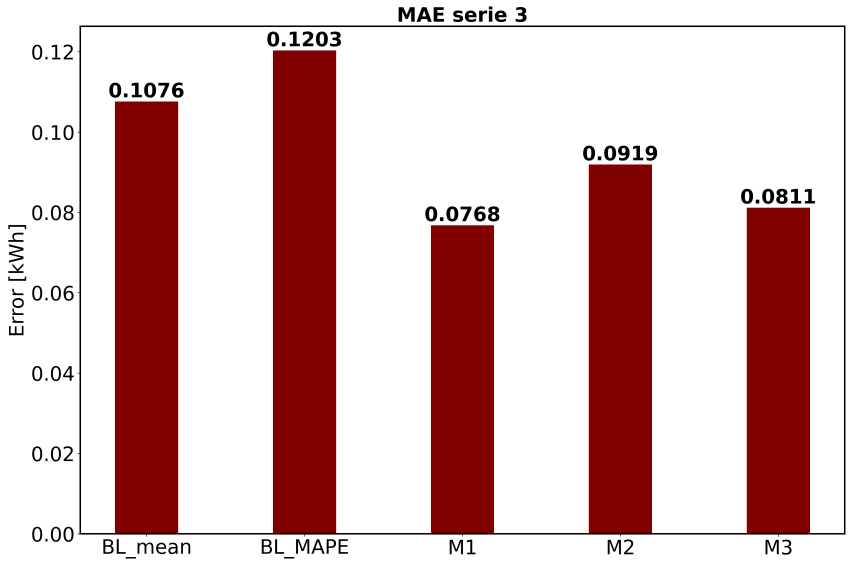


Figure 5.4: The MAE performance on all the days of the test set for Serie 3.

the "MAPE forecast" model. Also, it can be noticed that Model 2 for all the three series behaves slightly worse than the other two LSTM models. It is clear that the performance of the models is serie dependent.

To get more insight in how the MAE error is distributed over the test set, it is calculated for each individual day and displayed in Figures C.1, C.2 and C.3 in Appendix C. Figures C.2 and C.3 are discontinuous due to the missing days that are present in the reference signal in the month December. When the reference values of a day are not known, the day is removed from the test set to remove an additional estimation error of the reference signal during the calculation of the MAE. This is also the case for the calculation of the MAE in Figures 5.2, 5.3 and 5.4.

Next, it is noted that the MAPE for the LSTM models is much higher than for the baseline models as is shown in Table 5.2. As displayed in Figure 5.5 this is due to an overestimation of the reference signal when the values are small.

	Mean forecast	**MAPE forecast**	**Model 1**	**Model 2**	**Model 3**
Serie 1	0.46	0.41	3.07	3.37	3.27
Serie 2	1.82	0.83	3.67	3.64	3.65
Serie 3	0.80	0.48	3.30	3.44	3.21

Table 5.2: The MAPE for each Model and serie.

To get better insight in the actual output of the different models, Figure 5.5 shows the prediction of the different models on a chosen day in the test set.

First, it is again stressed that the "mean squared error" is chosen as metric during training of the LSTM models. Because of this, the LSTM models are more pushed towards learning the peaks of the reference signal due to the squared error. It can be seen that on the 7^{th} of December, the peaks of Serie 1 and Serie 3 are present in the predicted signal. It was found that especially for Serie 1, the LSTM models are predicting a higher consumption than the reference signal. The shape of the reference and prediction signals are similar, but there is an offset between the two signals. A reason for the shift could be that the LSTM models are trained using the MSE metric. It is found in literature that when a least square fitting is performed on data points the curve that is fitted can be much disturbed by the presence of outliers. The disturbance resulted also in a shift of the fitted curve. Because during training the MSE is used, the peaks during the day also act as outliers that could be responsible for the model shift. A solution to solve this is to get rid of the squared error and punish more proportional by using the MAE metric instead. On the other hand it can be argued that it is better to predict a higher consumption that one that is too low, because it is better to anticipate to a worse scenario then expected.

The reader is reminded that there is no regulation added for Model 1: Serie 1 and Model 3: Serie 1 and 3. When regularization is added this is clearly visible in Figure 5.5 because without regulation the predicted signal is much more choppy. A clear example can be seen for Model 2 and 3 in serie 3. These two models both show a small bump in the predicted signal before the larger peak. Model 2 where a form of regularization is added shows a very smooth bump while in Model 3 without regularization, the bump is much more choppy.

It can be seen that the "mean forecast" also is able to identify the two peaks in Serie 1 and the big peak in Serie 3, but as is expected from a mean, this peak is smaller due to averaging. The "mean forecast" method however suffers less from the offset than the LSTM models in Serie 1.

As expected the "MAPE forecast" will focus on correctly predicting the low values in the reference signal because these will get in the denominator of the MAPE metric according to Eq. 4.2. It can be seen that the peaks will be ignored when this method is applied.

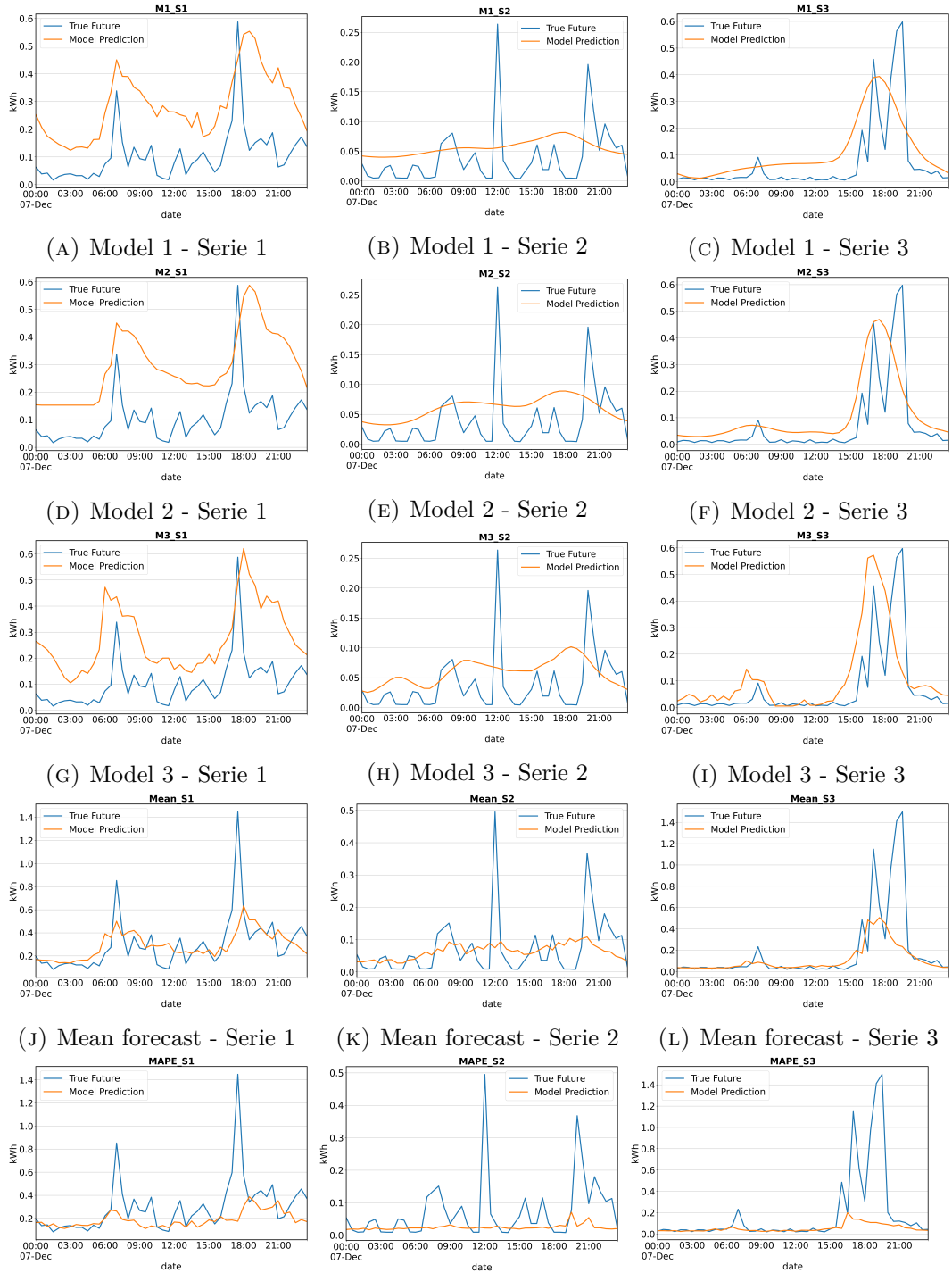
5.3 Conclusion

In this chapter the performance of the developed models is assessed. First, the model selection discussed and it was explained that the models obtained after the parameter search are ran ten times with different initialized weight matrices. The one that performed best on the validation set is selected. The number of epochs that the models trained were low except for model 2 and 3 on Serie 2.

Secondly, the performance on the test set is discussed for the LSTM models and two baseline models using bar plots that displayed the MAE on the test set for each serie. It was concluded that there is a reduction of all the three LSTM models in comparison to both the baseline models for Serie 2 and 3. Also, Model 2 always performed worse then the other LSTM models.
7 December is chosen for which all the predicted signals are grouped for the different models. From this figure is was clear that the LSTM models are able to identify peaks of the reference signal and have the same overall shape. Predicting the peaks correctly is an important, practical feature e.g. to anticipate electrical peak consumption demands. It was also noticed that the predictions are often an overestimation of the reference signal. Especially for Serie 1, the shape of the reference and prediction signals are similar, but there is an offset between the two signals. The offset could possibly be reduced with another choice of error metric during training. Because of the overestimation on small values, it was found that the MAPE of the LSTM models was worse than the baseline models. Also, the influence of adding regularization on the predicted signal could be seen which lead to more smooth signals.

The "mean forecast" baseline model was not able to predict the peaks in the reference signal as good as the LSTM models, but it has a lower offset error in Serie 1.

The "MAPE forecast" is focussed on predicting all the small values correct to minimize the MAPE, but ignores all the peaks of the reference signal.



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

(D) Model 2 - Serie 1 (E) Model 2 - Serie 2 (F) Model 2 - Serie 3

(G) Model 3 - Serie 1 (H) Model 3 - Serie 2 (I) Model 3 - Serie 3

(J) Mean forecast - Serie 1 (K) Mean forecast - Serie 2 (L) Mean forecast - Serie 3

(M) MAPE forecast - Serie 1 (N) MAPE forecast - Serie 2 (O) MAPE forecast - Serie 3

FIGURE 5.5: The prediction result of the different models on 7 December. (Reference: blue/ Prediction: orange)

Chapter 6

Conclusion

The final chapter contains the overall conclusion. It also contains suggestions for future work and industrial applications.
- say that because only use a small amount of simplistic inputs it found that after learning for a non-linear combination of previous consumptions the model was able to find a pattern in in the electricity consumption and make improvements with respect to the baseline models.
- stress the difficulty of the task –> not much informative data –> really a lot of uncertainty –> say that could identify a pattern in this data –> clear prediction of peaks.
- also look at comments of Swevers and build up of my previous thesis.

6.1 Future work

- because it was seen that much forecasts have the correct form with regard to the amount of peaks a first thing to try to solve this could be to adjust the error metric that is used during training from MSE to MAE. The peaks will be more proportional punished and this could lead that the model shifts down. - genetic algorithm to tune the parameters
- Track down the influence of each of the inputs. - Do a more extensive parameter search –> not neglecting synergy between different regularizations. Better to simultaneously try all the different 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

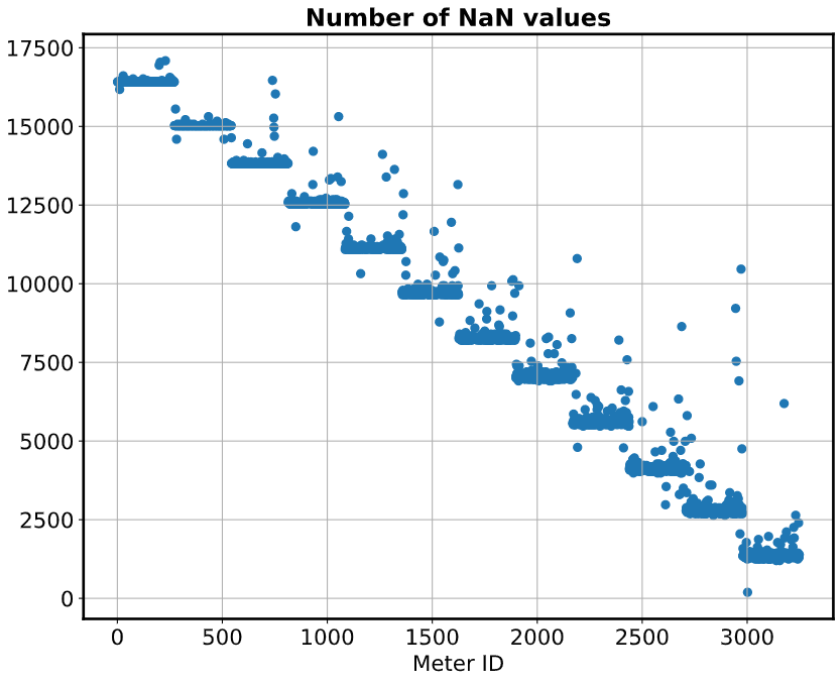


FIGURE A.1: The amount of NaN values in all the 3248 smart meters.

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

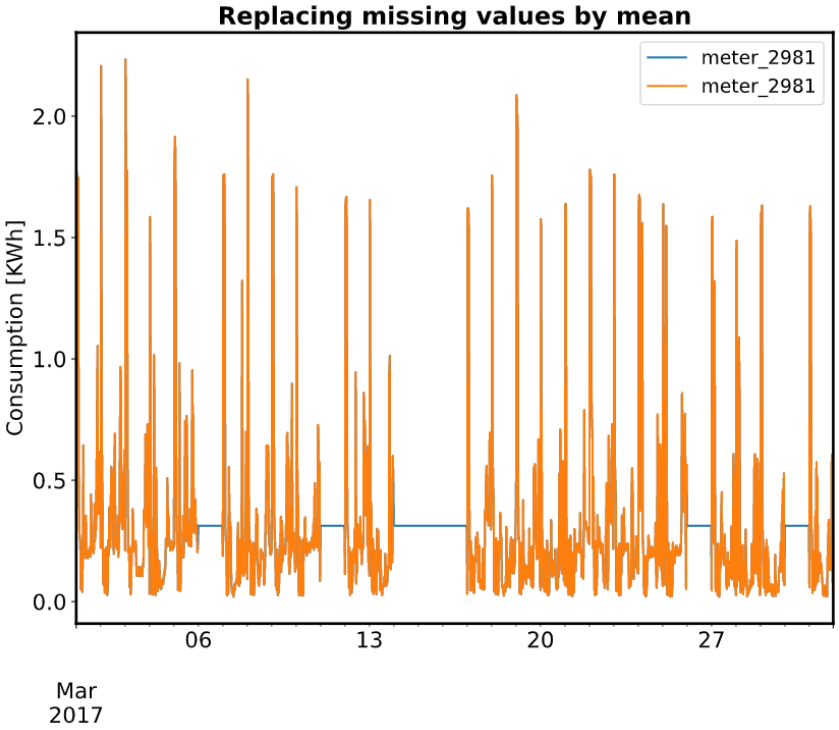


FIGURE A.2: Resulting month of March after substitution of the missing values by the mean value of the measurements.

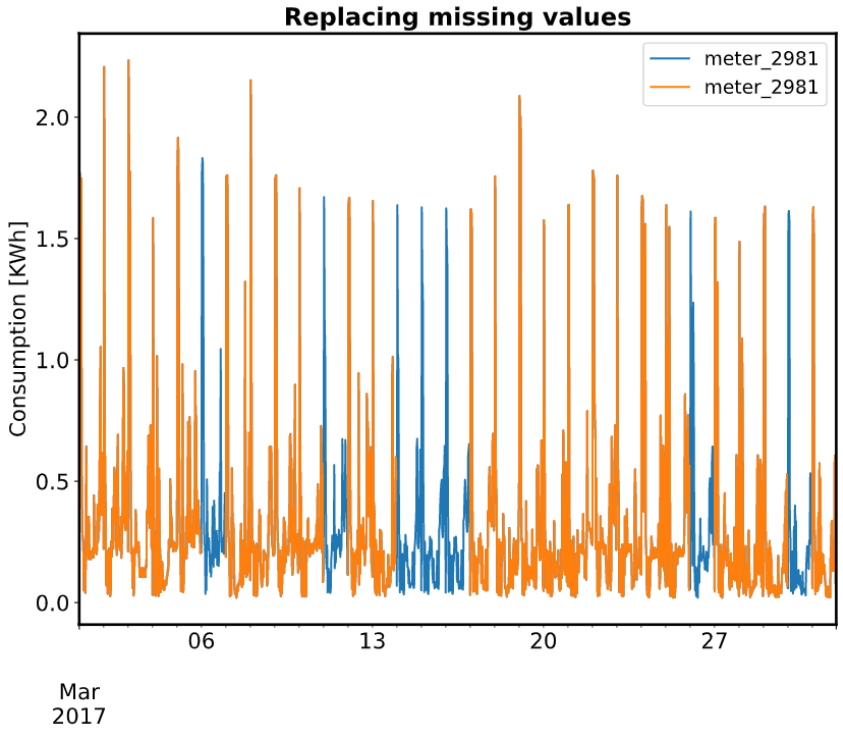


Figure A.3: Resulting month of March after substitution of the missing values by the mean value of the same moment on the next and previous day.

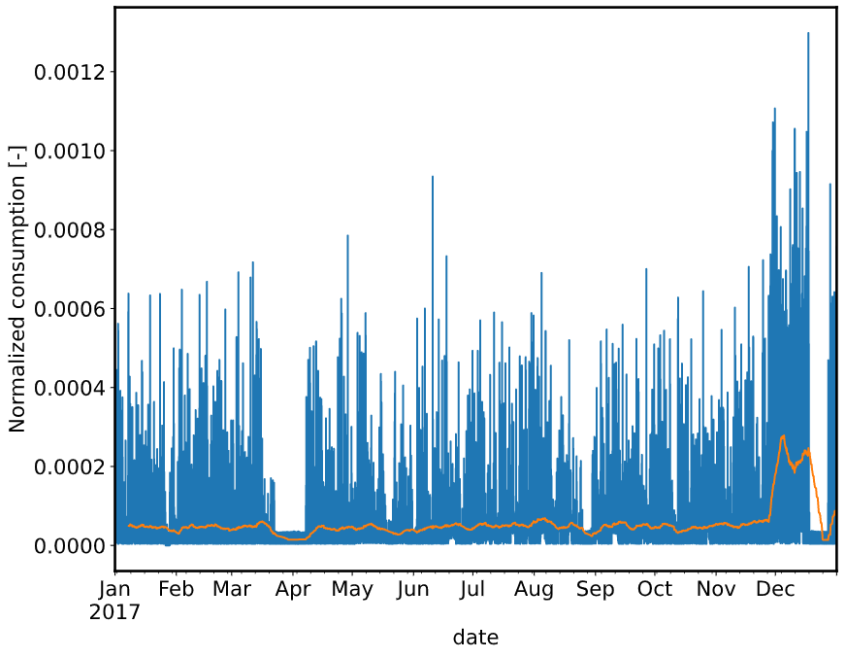


Figure A.4: The time-serie with the original maximum difference between the minimum and maximum weekly rolling averages.

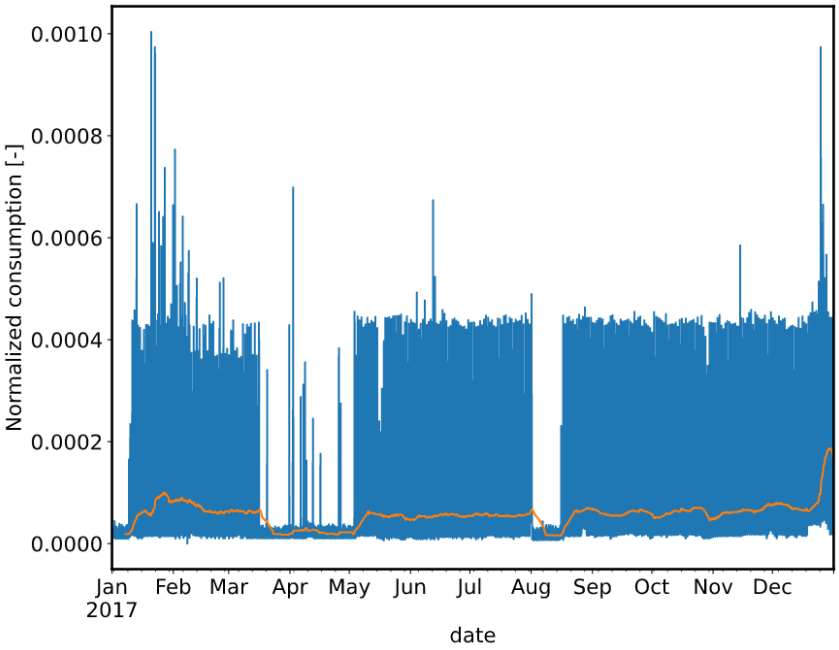


FIGURE A.5: The time-serie with the new maximum difference between the minimum and maximum weekly rolling averages.

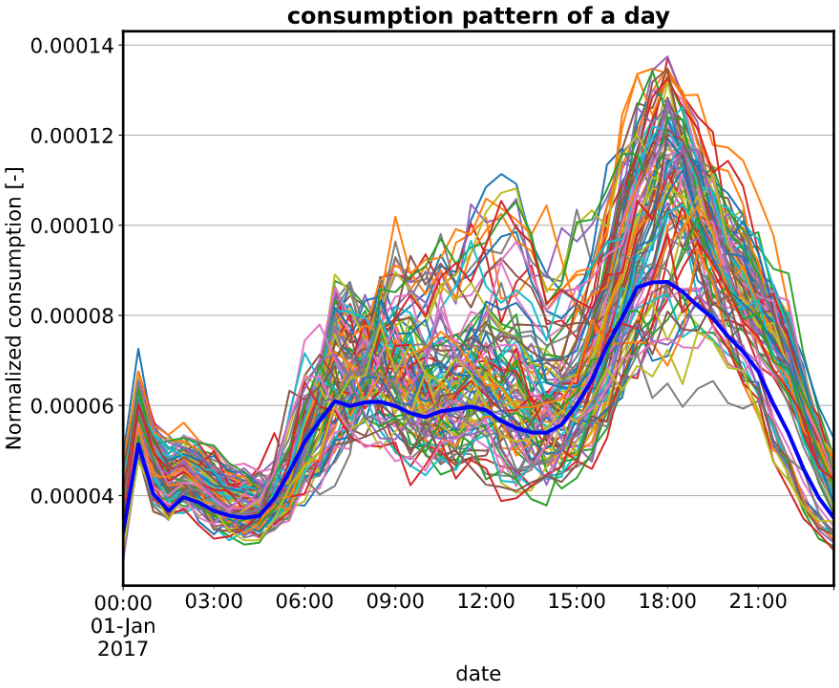


FIGURE A.6: Figure that shows the seasonality of the electrical load during the day.

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

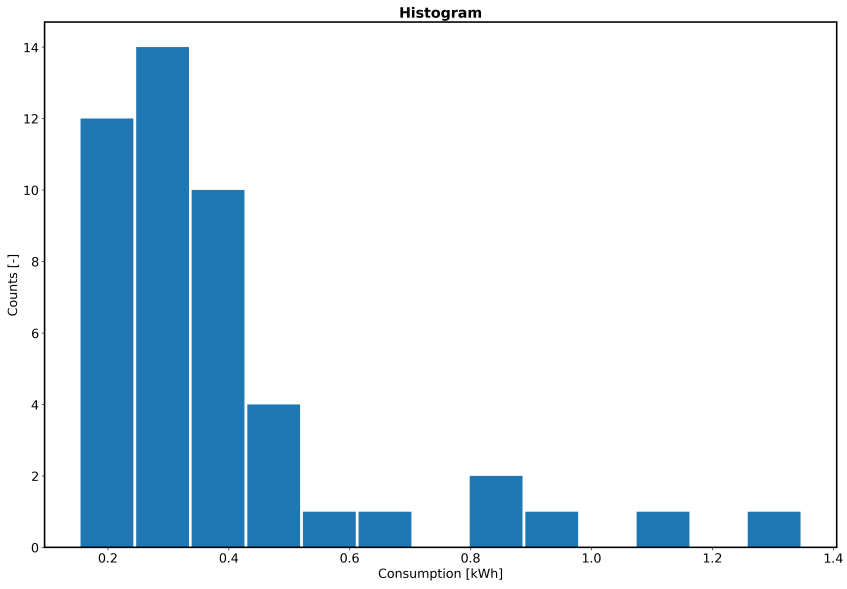


FIGURE B.1: An example histogram of the consumption in [kWh] versus count [-] used during MAPE forecast.

B.2 Parameter Search

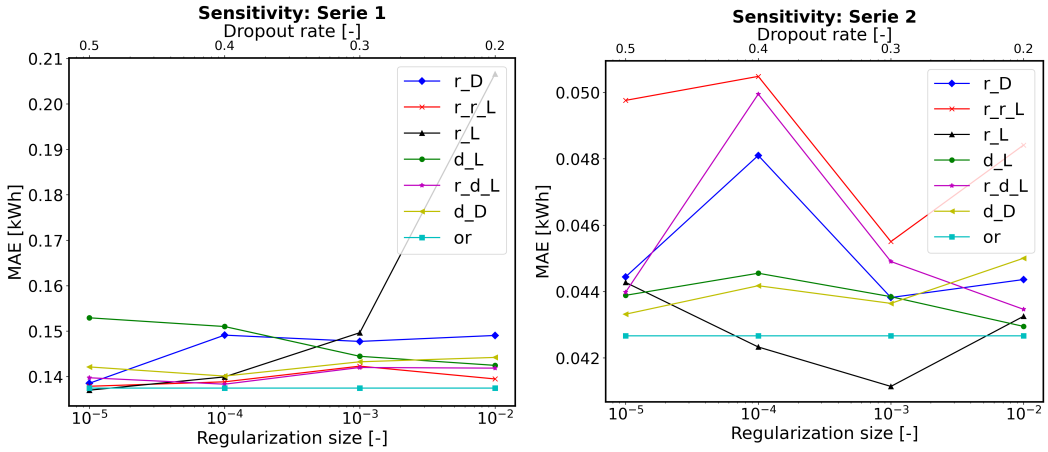
B.2.1 Model 2

Model 2: Stateless (flatten layer)				
Chosen parameter	**Value**	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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 corresponding parameter value is used, normalized by the biggest error of the possible values of one parameter and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst value for one parameter for each serie during phase 1 of the parameter search.

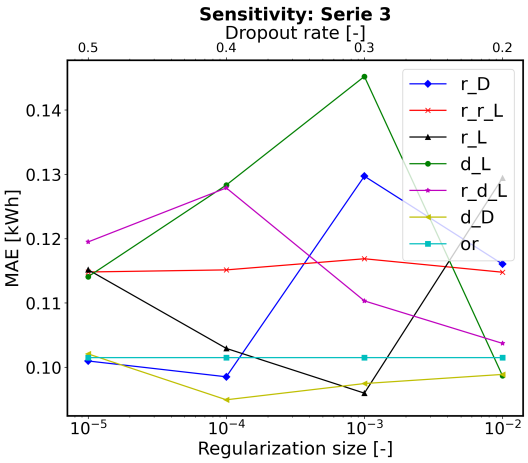
Model 2: Stateless (flatten layer)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states LSTM	20	50	50
layers LSTM	3	3	3
Lag value	96	48	96
Learning rate	0.001	0.0001	0.001
MAE error 1	0.137	0.0426	0.0973
MAE error 2	0.139	0.0430	0.107
MAE error 3	0.136	0.0424	0.100

Table B.2: The values of the parameters with the lowest average MAE on the validation set over three runs.



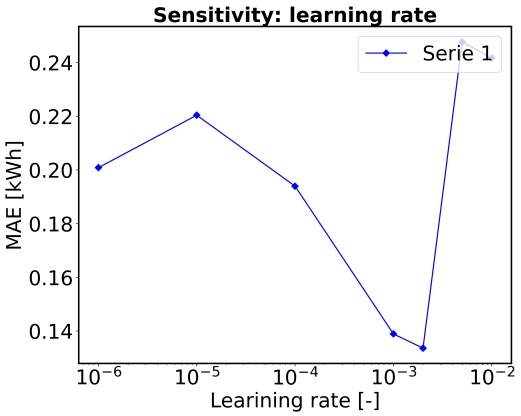
(A) Model 2

(B) Model 2

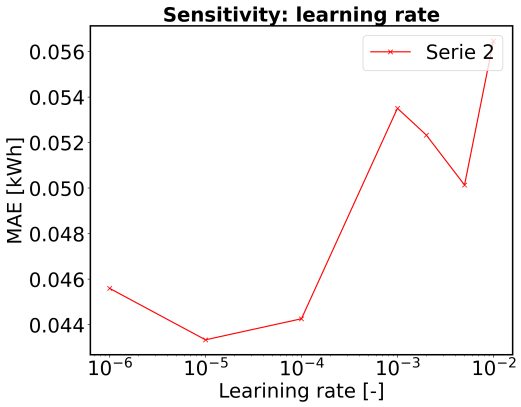


(C) Model 2

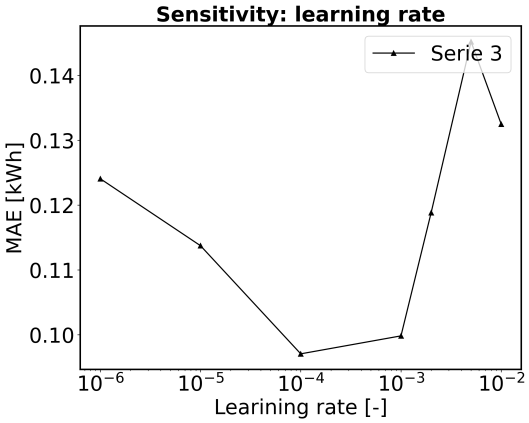
FIGURE B.2: Results of the sensitivity analysis on the size of the regularization parameter and the dropout rate according to MAE.(Legend: *r_D*: regularization size of weights of DENSE layer, *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM, *d_L*: dropout rate of inputs LSTM, *r_d_L*: dropout rate of hidden states LSTM, *d_D*: dropout rate of DENSE layer, *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.

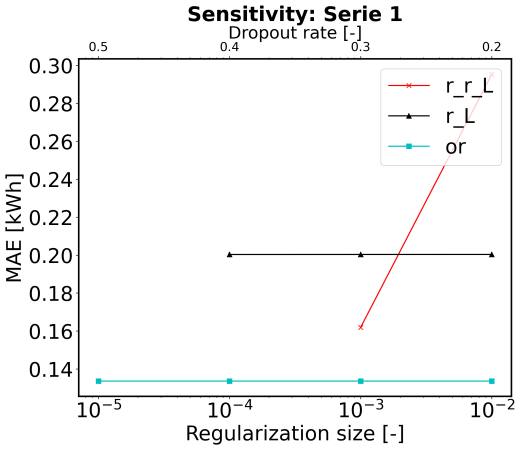
B.2.2 Model 3

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

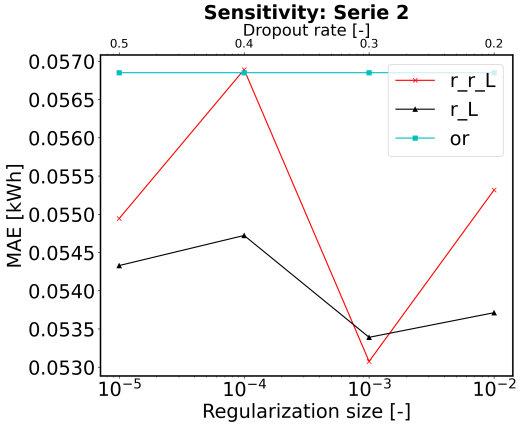
TABLE B.3: Each value in this table shows the average error when the corresponding parameter value is used, normalized by the biggest error of the possible values of one parameter and finally subtracted by one. Therefore, each value shows a percentage of improvement with respect to the worst value for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

Model 3: Stateful (1 time step)			
Parameters	**Serie** 1	**Serie** 2	**Serie** 3
Hidden states 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

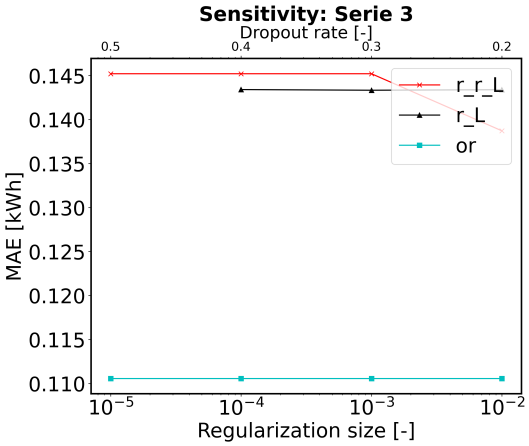
TABLE B.4: The values of the parameters with the lowest average MAE on the validation set over three runs.



(A) Model 3

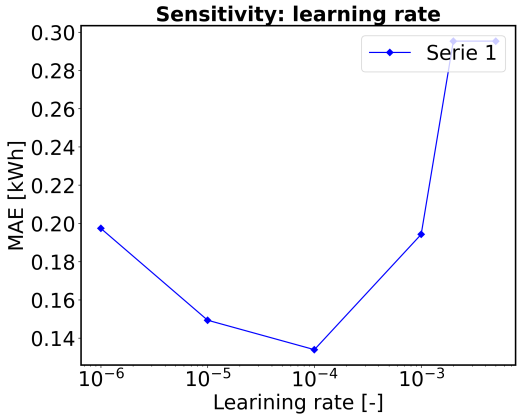


(B) Model 3

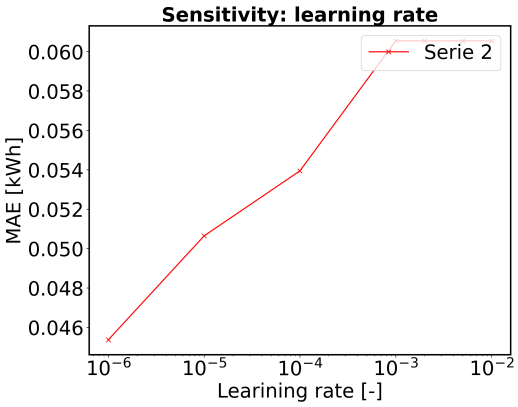


(C) Model 3

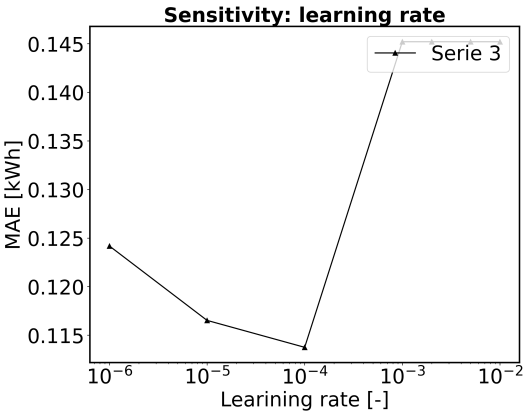
Figure B.4: Results of the sensitivity analysis on the size of regulation parameter and the dropout rate with respect to the mean absolute error.(Legend: *r_r_L*: regularization size of recurrent weight of LSTM, *r_L*: regularization size of input weights of LSTM and *or*: best performing serie from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

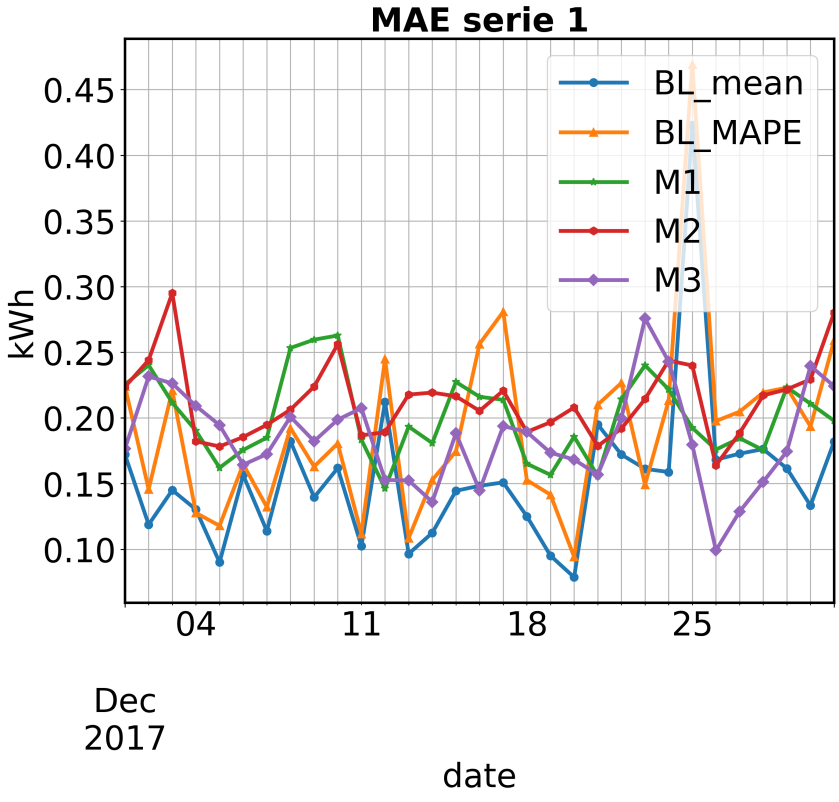


FIGURE C.1: The MAE performance for the different days in the test set for Serie 1.

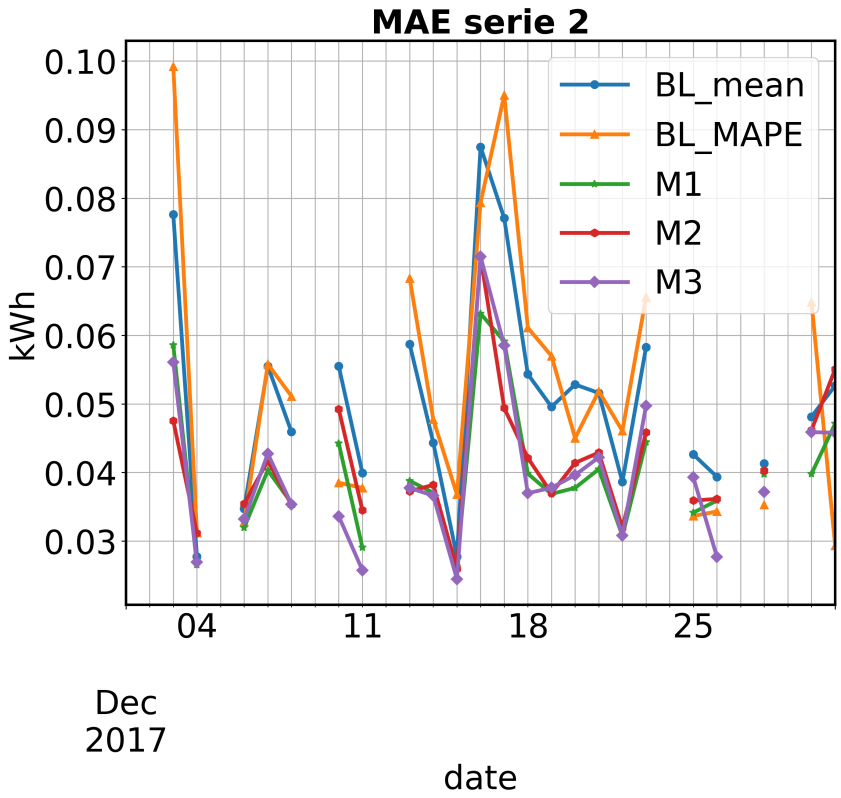


Figure C.2: The MAE performance for the different days in the test set for Serie 2.

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.4 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

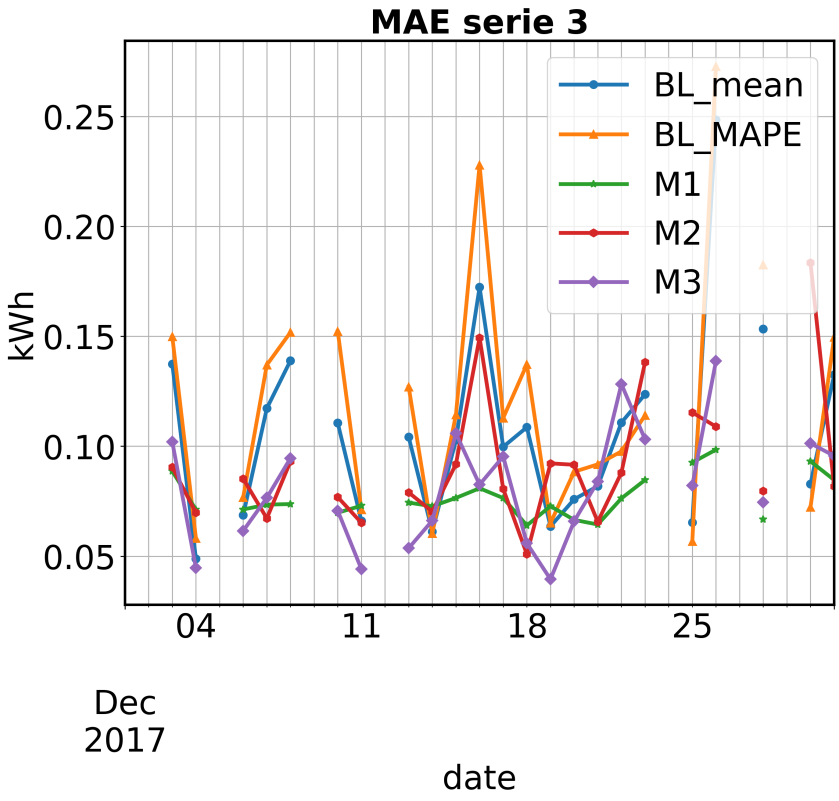


FIGURE C.3: The MAE performance for the different days in the test set for Serie 3.

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 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

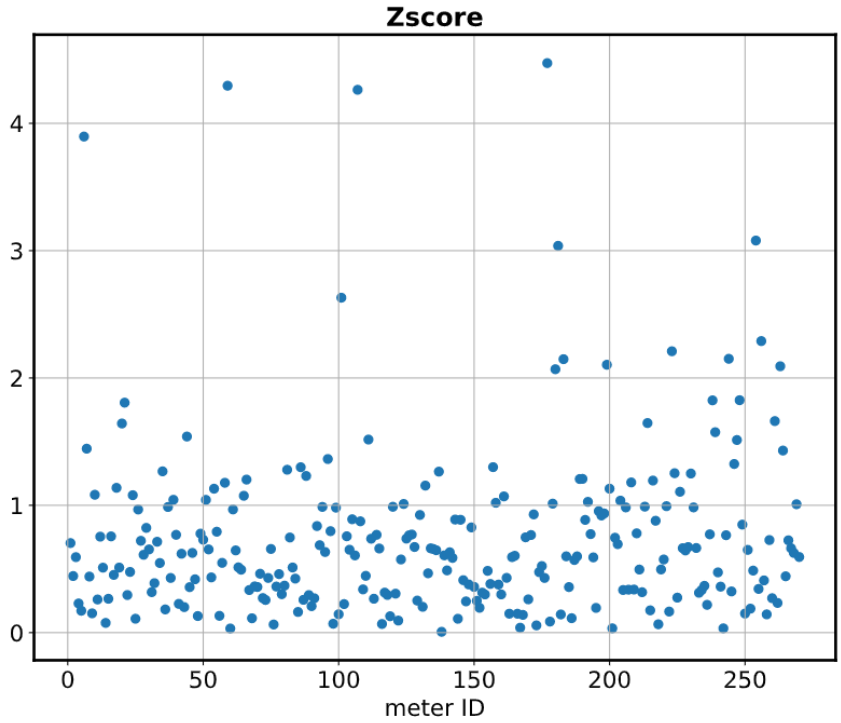


Figure C.4: Z-scores calculated from the yearly consumptions.

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] N. Srivastave, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting Nitish. *Joernal of Machine Learning Research 15*, 2014.

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

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

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