Universitetet
i Stavanger

**FACULTY OF SCIENCE AND TECHNOLOGY**

# MASTER'S THESIS

| | |
|---|---|
| Study programme/specialisation:<br><br>Computer Science | Spring semester, 2019<br><br>Open |
| Author: Redjol Resulaj | ..... (signature of author) |

Supervisor: Mina Farmanbar

Co-Supervisors: Antorweep Chakravorty, Aida Mehdipourpirbazari

Title of master's thesis:

**Smart Meter Based Load Forecasting for Residential Customers Using Machine Learning Algorithms**

Credits: 30

| | |
|---|---|
| Keywords:<br><br>Smart meters, load forecasting, machine learning, artificial neural network, random forest | Number of pages: 65<br><br>+ supplemental material/other: 12<br><br>Stavanger, 12/06/2019 |

# *Abstract*

The focus of this thesis is the use of machine learning algorithms to perform next step short term load forecasting on fifty five households in Stavanger, Norway. A dataset containing electricity consumption data for more than one year is used to train and evaluate a Feedforward Neural Network model and a Random Forest model. Weather data, atmospheric data and calendric variables are also used to aid the forecasting task.

First, the implementation of the two models is introduced. Their architectures are given and the rationale behind the design principles are explained. Then, for every household, a separate neural network and random forest model are trained using the training dataset. The models are tested using the testing dataset, to evaluate the models' accuracy. The models were trained and tested on three different but equivalent datasets. The difference between them was the time resolution of the data. These resolutions are 1 hour, 15 minutes and 1 day.

The implemented models achieved various levels of accuracy depending on the household and the data resolution. Generally, the implemented Neural Network achieved higher accuracy than its Random Forest counterpart. It was also discovered that the resolution has a big influence on the outcome of the next step short term load forecasting task.

# *Acknowledgements*

I dedicate this thesis to my beloved parents Xhevit and Andonika, my wonderful sister Borana and my lovely unborn niece, whose name I don't know yet. I would like to thank them for the great support and encouragement they have given to me. Special thanks and my deepest gratitude go to Maria, for her love, support, criticism and suggestions.

In addition I would like to thank my supervisors Mina, Aida, Antorweep and everyone else who have helped to write this thesis.

# Contents

# Abbreviations

| | |
|---|---|
| **ANN** | **A**rtificial **N**eural **N**etwork |
| **RF** | **R**andom **F**orest |
| **ReLU** | **Re**ctified **L**inear **U**nit |
| **MSE** | **M**ean **S**quared **E**rror |
| **MSLE** | **M**ean **S**quared **L**ogarithmic **E**rror |
| **MAE** | **M**ean **A**bsolute **E**rror |
| **MAPE** | **M**ean **A**bsolute **P**ercentage **E**rror |
| **SGD** | **S**tochastic **G**radient **D**escent |
| **SOM** | **S**elf **O**rganizing **M**aps |
| **PCP** | **P**rincipal **C**omponents **P**ursuit |
| **CART** | **C**lassification **A**nd **R**egression **T**ree |
| **CNN** | **C**ombinatorial **N**eural **N**etwork |
| **DNN** | **D**eep **N**eural **N**etwork |
| **RBM** | **R**estricted **B**oltzmann **M**achine |
| **SNN** | **S**hallow **N**eural **N**etwork |
| **ARIMA** | **A**utoregressive **I**ntegrated **M**oving **A**verage |
| **DSHW** | **D**ouble **S**easonal **H**olt **W**inters |
| **RNN** | **R**ecurrent **N**eural **N**etwork |
| **LSTM** | **L**ong **S**hort **T**erm **M**emory |
| **SVR** | **S**upport **V**ector **R**egression |
| **NNETAR** | **N**eural **Ne**twork **A**uto**r**egression |
| **NARX** | **N**onlinear **A**uto**r**egressive E**x**ogenous model |
| **SARIMA** | **S**easonal **A**utoregressive **I**ntegrated **M**oving **A**verage |
| **MLP** | **M**ultiple **L**inear **R**egression |
| **GA** | **G**enetic **A**lgorithm |

| | |
|---|---|
| **FS** | **F**eature **S**election |
| **SVM** | **S**upport **V**ector **M**achine |
| **SARMA** | **S**easonal **A**uto**r**egressive **M**oving **A**verage |
| **FFNN** | **F**eed**f**orward **N**eural **N**etwork |

# Symbols

| Symbol | Name | Unit |
|---|---|---|
| $P$ | Power | W ($Js^{-1}$) |
| $P_{avg}$ | Average power | W ($Js^{-1}$) |
| $E$ | Energy | J |
| $t$ | Time | s |
| $x$ | Independet variable | |
| $\bar{x}$ | Mean of x | |
| $y$ | Dependent variable | |
| $\hat{y}$ | Forecast of dependent variable | |
| $r$ | Correlation | |
| $T$ | Temperature | °C |
| $D$ | Dew point | °C |
| $C$ | Cloud cover | no unit |
| $H$ | Humidity | no unit |

# Chapter 1

# Introduction

## 1.1 Introduction to load forecasting

Electrical load forecasting is the prediction of the load (power) demand that an electricity consumer will have in the future. Load forecasting is very important for utilities and electricity distribution companies, which must ensure uninterrupted electricity supply to their customers, while maintaining minimal costs in the energy production and transmission process.

Load forecasting is becoming increasingly easier, from a technical point of view, as the utilization of smart meters is becoming more and more common. The surge of smart meter installations allows utility and distribution companies to collect an abundant amount of electricity consumption data about their customers, which was not possible in the past. This rich profusion of data opens many doors of opportunities in electricity data analysis. Load forecasting is one such subfield of smart meter data analytics and it is of great significance.

Load forecasting can be classified based on how far ahead in time the load prediction is attempted. Based on this criteria, load forecast can be categorized in three groups [1]:

- Short term load forecast (a few minutes to 1 day ahead).
- Medium term load forecast (1 day to 1 year ahead).
- Long term load forecast (more than 1 year ahead).

This thesis will be focused on the next step short term load forecast. Specifically, the 15 minutes ahead, 1 hour ahead and 1 day ahead load forecasts will be studied.

## 1.2 Motivation

Electricity is a very special type of commodity. It cannot be easily stored for later use. The only storage strategy for electricity are batteries which can convert deposited chemical energy to electrical energy. However, batteries are suitable for small scale energy requirements and are rarely used by utilities to meet the high demand of large geographical areas, because of their high cost. Although, such large scale batteries are not unheard of, with the most well known example being the Hornsdale Power Reserve Battery in South Australia built by Tesla Inc [2]. This giant battery has a voltage of approximately 100 megawatts (MW) and a storage capacity of 129 megawatt hours (MWh), which is enough to power about 30,000 homes for 1 hour [3]. The construction cost of this battery has been estimated at 50 million USD [4].

The lack of means to store electrical energy signifies that the production companies constantly need to struggle with the necessity to match the supply with the demand. The motivation behind short term load forecasting is the goal to provide these companies with an accurate prediction of future energy demands by their costumers, so that their production is planned accordingly.

Failing to match the supply with the demand leads to two possible situations: undersupply or oversupply. Undersupply is the situation where the consumers require more energy than it is being produced at the given moment. Obviously, it leads to a power outage and a lot of consequences that come with it. Oversupply is the situation where more electrical energy is being produced than it is needed. This excess energy has to be disposed, or sold at a lower price. Both actions imply financial loss.

In the future, if technologies like Hornsdale Power Reserve Battery start to catch up, load forecasting may lose its importance. Undersupply can be solved by using the stored energy in batteries, while overupply can be overcome by storing the excess energy. However, until such technologies become ordinary, short term load forecasting has a powerful role in the industry.

## 1.3 Problem description

Short term load forecasting is the prediction of the load demand that specific consumers will have in the near future. The near future may be the next few minutes, the next hour, or even the next day. The load demand can be predicted by analyzing historical load demand data and other data, such as weather and atmospheric parameters, calendric variables, etc.

There are many slight variations to the problem of load forecasting. This thesis is concerned with the *next step* average load forecast. A *time step*, or *step*, is defined as a specific time interval whose duration is equal to the time resolution of the available data. For example, if the time resolution at hand is 1 hour, the *next step* is the next hour from now. So, in other words, this thesis is concerned with the task of forecasting the average electric load demand that a specific consumer will have in the next time interval from now.

## 1.4 Research question

The problem described above is relatively broad. There are many techniques that can be used to perform a load forecast, which will also be discussed later on. The Feedforward Neural Network and Random Forest are supervised machine learning techniques that are very common for regression analysis. These techniques implemented and evaluated here using real data at different resolutions. The aim of this thesis can be summarized by the following questions:

- Are the Feedforward Neural Network and Random Forest models capable of performing accurate load demand forecasts?
- How well do these two machine learning models perform at different data resolutions?
- Which machine learning model forecasts the next step load demand more accurately: Feedforward Neural Network or Random Forest?

## 1.5 Planned contribution

Part of the goal of this research thesis is to implement a highly accurate load forecasting model. However, regardless of the achieved accuracy, the evaluation of the forecasting capabilities of the implemented Feedforward Neural Network and Random Forest, in general, is also important.

In addition, the models under consideration will be tested and evaluated using three different data resolutions: 15 minutes, 1 hour and 1 day resolutions. This is important to understand how the load demand patterns are manifested at different resolutions.

## 1.6 Outline of the thesis

This thesis has 7 chapters. In the first chapter, an introduction to the problem of short term load forecasting is given and the research questions of this thesis are stated.

In the second chapter, a few basic scientific concepts that are essential to understand the thesis are explained. These concepts are related to machine learning and the electric grid.

Next, some related works to smart meter data analytics are discussed, with a special focus on load forecasting.

The Solution Approach chapter is the most important part, where the case study is presented, along with the techniques used to solve the problem of short term load forecasting in this case study. Data preprocessing and the machine learning models implementations are discussed in detail.

Chapter 5 demonstrates the conducted experiments and the results yielded by these experiments. Graphs and forecast errors are given, to analyze the accuracy of the implemented models in chapter 4.

The Discussion chapter analyzes the experimental results in detail and elaborates on key points of these results.

The last chapter highlights all the conclusions of this thesis and presents a few interesting topics that can be pursued to extend the research.

# Chapter 2

# Basic Theory

## 2.1 Introduction to the electric grid and basic electricity terms

The electrical grid, or power grid, is a giant system of electrical components that spans large geographical regions. Its purpose is to distribute electrical energy to consumers. The energy is produced by generators. So, if stripped of all the complex details, the grid's most basic components are:

- Electrical energy generators.
- Energy consumers.
- Transmission lines.

The energy generators are devices that can convert mechanical or solar energy to electrical energy. Common examples include solar panels, water turbines, windmills, etc. Energy consumers are components that utilize the energy produced by the generators and convert it to mechanical energy, heat, light, etc. Common examples are washing machines, ovens, light bulbs or any other electric appliance. Transmission lines are the cables that transmit the electrical energy from the generators to the consumers.

Energy consuming components are also called **loads** [5]. When a load is turned on, it consumes energy at a constant rate. The rate at which the load is consuming this energy is called power and it is measured in watts (W). The formula for power is given in equation 2.1:

$$P = \frac{\Delta E}{\Delta t} \tag{2.1}$$

where P is the power, $\Delta E$ is the consumed energy and $\Delta t$ is the time interval at which the energy is consumed. 1 watt is equal to 1 joule/second. So, every load has a specific power at which the energy is consumed, or the power can be adjustable. Nonetheless, each of them consumes energy at a specific rate in a given moment. That is why the word term *load*, or *load demand* are often used as synonyms for power in literature. In this thesis, these terms will also be used interchangeably to refer to the rate at which energy is consumed. Though, strictly speaking, load is the equipment that is consuming the energy.

In a household, there may be many electrical devices running simultaneously. The load demand of a household at any given moment is the sum of the load demands of all the running devices at that moment. This sum of power values is the load demand value of the household measured by smart meters and is the objective of the forecast.

The graph of the load demand of the household against time, over a specific time interval, is called a *load profile* or a *load curve*. It is an important characteristic of a household, as it represents its energy consumption behaviour. Figure 2.1 depicts a random daily load profile of some random house.

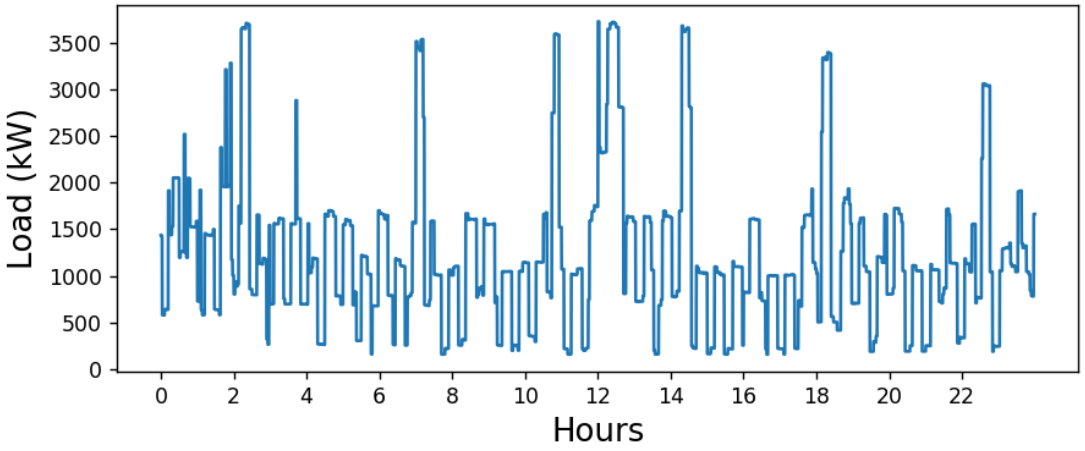



**Figure 2.1:** Load profile of a random household

## 2.2 Artificial Neural Networks

Artificial Neural Networks are very powerful modelling systems that try to mimic the processing capabilities of the human brain. Presently, they are one of the most popular machine learning algorithms, having received a lot of attention from the scientific community and the industry in the recent years.

Historically, there was a rising interest on neural networks in the early 1940's with the development of the *artificial neurons* by Pitts and McCulloch [6]. The neurons that they

designed were electrical components in a circuit that could perform computational tasks based on binary inputs. However, the neuron model of Pitts and McCulloh was only able to execute computational tasks and it was not able to "learn". The next major development in this area was done by Rosenblatt in 1958, which introduced the *perceptron* [7]. The perceptron could accept any real number as input and unlike the classical neuron of Pitts and McCulloh, the inputs were weighted. This allowed for more flexible modelling capabilities.

However, by the late 1960's the interest and funding on neural networks began to disappear, only to re-emerge again in the early 1980's when some prominent results were achieved [6]. The most notable achievement, that fueled the awoken interest on neural networks, was the discovery of the backpropagation algorithm by Werbos [8].

Today, neural networks are able to solve problems that conventional logic based programs cannot. Some of these fields where they have achieved substantial success are pattern recognition, computer vision, natural language processing, etc. Artificial neural networks have some advantages over traditional programming [9]:

- Neural networks are capable of performing any non-linear mapping. For that reason, their implementation is relatively easy.
- They are inherently easy to parallelize. This allows for intensive tasks to be executed much faster than their serial counterparts.
- They can be adapted to perform supervised learning and unsupervised learning tasks as well.

This section will be dedicated to the theoretical foundation behind artificial neural networks. Their general architecture, algorithms and mathematical properties will be introduced. This theoretical part will be useful to understand the logic behind the forecasting model introduced later in this thesis.

### 2.2.1 The neuron

The neuron is the building block of neural networks. It is the most basic computational unit in a network. A neuron has an arbitrary number of inputs, a specific weight for each input, a function applied on the inputs and one output. A visual representation of the neuron is depicted in figure 2.2.

As it is obvious from the illustration, the output of the neuron is equal to the value produced by the $f$ function applied to the weighted sum of the inputs. This operation is mathematically expressed by the following equation:

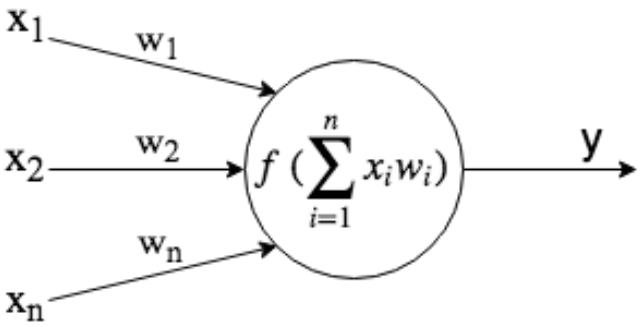



**Figure 2.2:** The neuron

$$y = f(\sum_{i=1}^{n} x_i w_i) \tag{2.2}$$

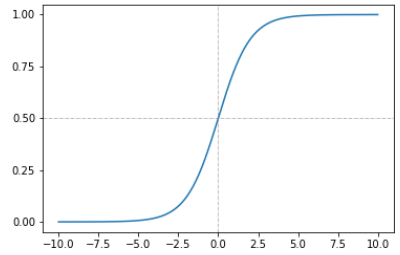
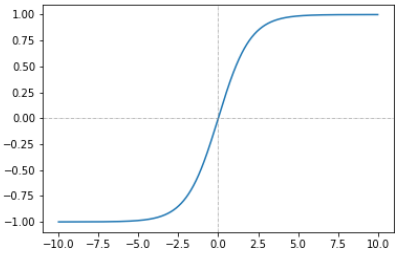
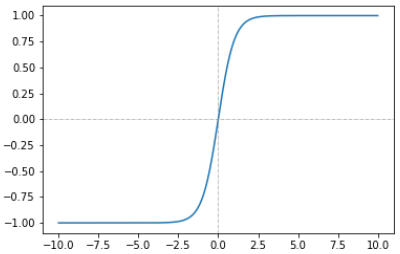
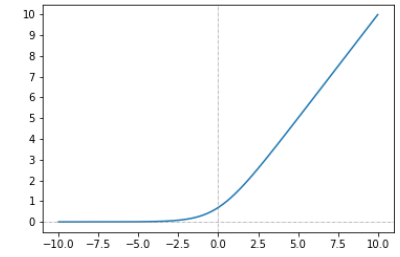
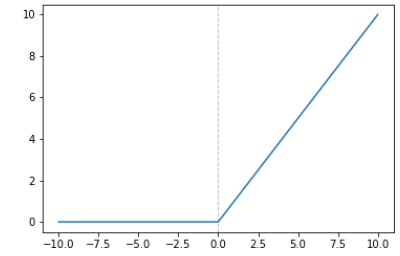
where $y$ is the output of the neuron; $x_1, x_2, ..., x_i$ are the inputs; $w_1, w_2, ..., w_i$ are the respective weights of each input, $f$ is the aforementioned function and $n$ is the number of inputs. The $f$ function is called the *activation function* and it is a very important aspect of neural networks.

The weights of the neuron are not static, they change over time. In fact, it is this property of the neurons that enables the neural network to "learn". Later on, it is shown that tuning the value of the weights is what the model training actually is.

### 2.2.2 The activation function

The activation functions are an essential part of artificial neural networks. The role of the activation function is to convert an input signal to an output one. However, the real purpose of these functions is to establish non-linear relationships between the input and output [10]. This is very important for neural networks if they are needed to perform non-linear mappings, which as stated at the introduction of this section, is a very powerful property of ANNs.

There are many activation functions such as the unipolar sigmoid function, bipolar sigmoid function, hyperbolic tangent function, step function, rectified linear unit function (ReLU), etc. In theory, any activation function can be used to train an ANN. There is no mathematical or logical rule that justifies the preference of one over the others. However, empirical evidence shows that in some applications, specific activation functions provide better accuracy and faster "learning" than the others [10][11]. Table 2.1 depicts some common activation functions used in practice.

| **Function name** | **Function** | **Graph** |
|---|---|---|
| Unipolar Sigmoid | $f(x) = \frac{e^x}{1+e^{-x}}$ |  |
| Bipolar Sigmoid | $f(x) = \frac{1-e^{-x}}{1+e^{-x}}$ |  |
| Hyperbolic tangent | $f(x) = tanh(x)$ |  |
| Softplus | $f(x) = \log(1 + e^x)$ |  |
| Rectified Linear Unit | $f(x) = \begin{cases} x & \text{if x} \geq 0 \\ 0 & \text{if x} < 0 \end{cases}$ |  |

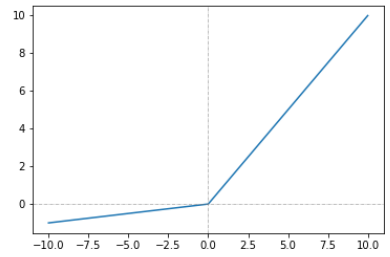
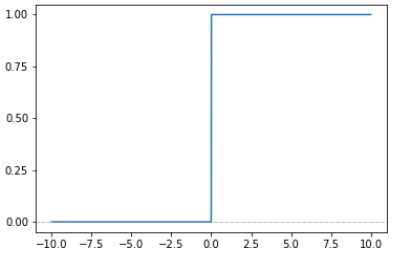
| | | |
|---|---|---|
| Leaky ReLU | $f(x) = \begin{cases} x & \text{if x} \geq 0 \\ \alpha x & \text{if x} < 0 \end{cases}$ |  |
| Step | $f(x) = \begin{cases} 1 & \text{if x} \geq 0 \\ 0 & \text{if x} < 0 \end{cases}$ |  |

**Table 2.1:** Common activation functions

### 2.2.3 Architecture of Artificial Neural Networks

As mentioned previously, the neuron is the building block of artificial neural networks. In theory, a neural network can consist of a single neuron, but that would not achieve much. That's why, in practice, real networks comprise many neurons that are organized in relatively complex architectures.

Neurons are grouped in layers and a neural network consists of a single *input layer*, at least one *hidden layer* and a single *output layer*. Each layer can have any number of neurons and, in principle, there is no upper or lower limit. The layers are stacked together and all neurons of a given layer are connected with all neurons of neighboring layers. A simple four layer artificial neural network is depicted in figure 2.3.

This neural has three inputs, two outputs and two layers with 4 neurons each. It is obvious that the network is fully connected. Every connection is called a synapse and every synapse has a weight, as explained in subsection 2.2.1. The input layer receives inputs during the training phase and during the forecasting phase as well. The input is fed to the first hidden layer, whose neurons calculate the output based on equation 2.2. The output of each neuron in that layer is fed to the next layer and so forth. The values produced by the *output layer* is the output of the neural network. This type of neural network is called a feed forward neural network, because the information travels only in one direction, as shown by the arrows in the synapses.

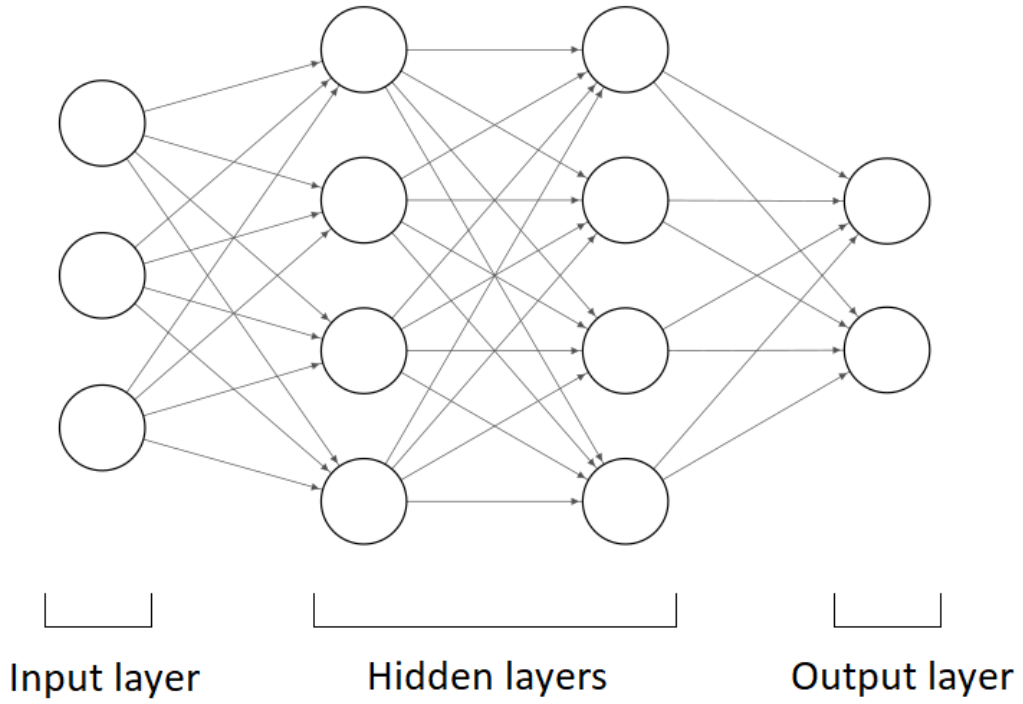



**Figure 2.3:** ANN architecture

### 2.2.4 Training a neural network

Artificial neural networks fall into the supervised learning algorithms category (though they can be adapted to be used in unsupervised learning). This means that to "teach" a neural network, we must show it a collection of observed inputs and the associated observed outputs. After the network has seen enough input-output associations, it can predict an output from a new input that it has never seen before. A simple training algorithm for neural networks is depicted in figure 2.4 and explained in more detail in the upcoming paragraphs.

The first step in ANN training is the initialization of all the weights. It is important to initialize weights to different values. If the weights are the same, all the neurons belonging to the same layer will output the same value and thus the model will fail to learn. The most common practice is to randomly initialize the weights based on a uniform distribution, or some other distribution. However, there are some complex methods that enable the network to learn faster, such as the Delta Rule [12], the SCAWI method [13], etc.

In step 2 and 3, an input from the observations (the input list) is fed to the network. This signal is forward propagated, which means that the neurons are activated one layer after the other. A neuron is activated by applying the equation 2.2. The series of neuron activation leads to the activation of the neurons of the output layer. This produces the output of the neural network $\hat{y}$.

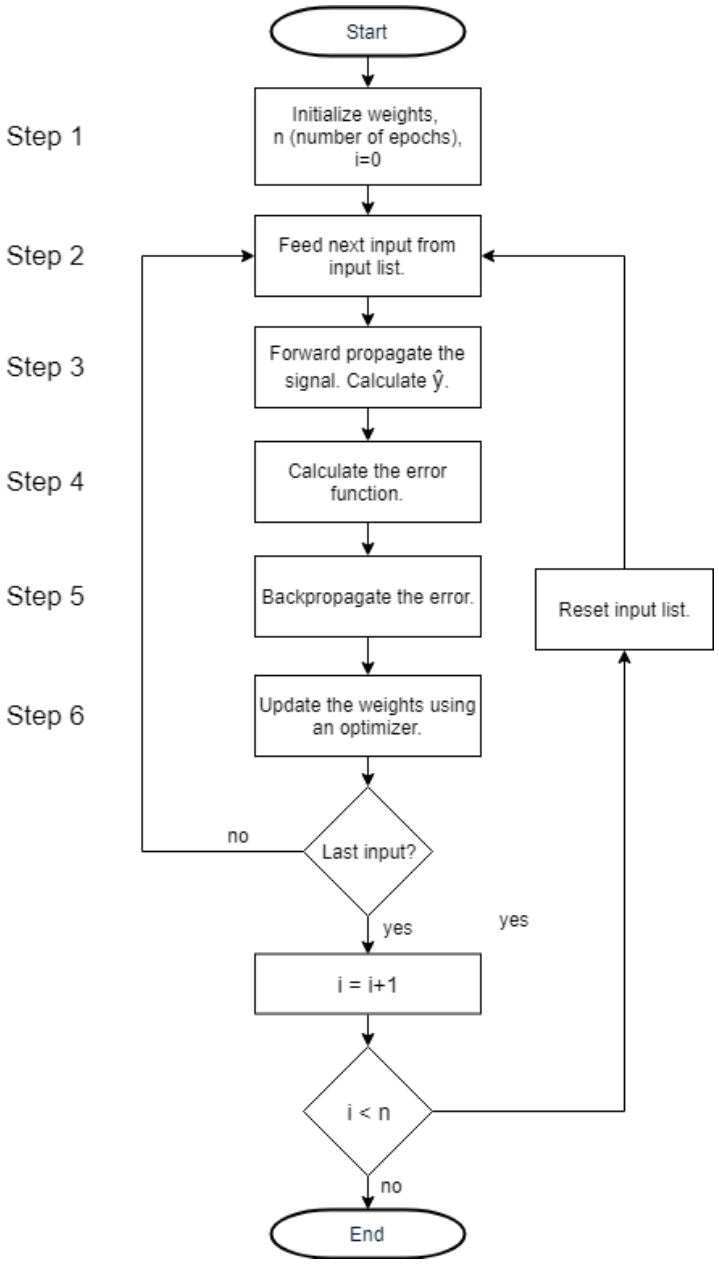



**Figure 2.4:** ANN training algorithm

In step 4, the generated output $\hat{y}$ is compared to the observed output $y$ associated to the input that is fed in step 1. The error between these two values is calculated. This error is called the loss function (also called cost function, or error function). The purpose of the whole training process is to minimize the error by adjusting the weights. There are many loss functions used in practice and some of the most popular are the Mean Squared Error (MSE), Mean Squared Logarithmic Error (MSLE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE).

In step 5, Backpropagation is performed. Backpropagation is the practice of determining the gradient of the weights based on the error function. In simpler terms, it calculates

how much every weight is responsible for the error. It does so by calculating the partial derivative of the error with respect to the specific weight, which is called gradient.

In step 6, an *optimizer* updates the weights of the network based on the gradients that were calculated in step 5. The optimizer adjusts the weights with the goal to minimize the loss function in the next iteration. The most popular optimizing algorithm is Stochastic Gradient Descent (SGD) [14]. Other existing optimizers are usually variants of SGD [15], such as Adam, RMSProp, Adagrad, etc.

Steps 2 to 6 are repeated until the list of inputs is exhausted, that is until every input is fed to the network. Feeding the whole list of inputs to the network consists of one epoch. Usually, more than one epoch is carried out. The number of epochs depends on the convergence speed of the network. This parameter is very important, because a high value may lead to overfitting of the data, whereas a low value might lead to a poor prediction from the model.

The presented algorithm here is a very general case of feed forward neural networks. There are many other variants that are not discussed here. However, this section is a good introduction to neural networks and a required background for the presented algorithm in the upcoming chapters.

## 2.3 Random Forest

Random forests are ensemble machine learning models that are very popular for classification and regression tasks. An ensemble machine learning model is one in which many independent basic models are trained separately and their prediction results are combined together for a more accurate result [16]. Random forests are compiled by training many decision trees and aggregating their results [17]. In this section, the decision tree will be discussed firstly and then random forests will be explained.

### 2.3.1 Decision Trees

Decision trees are a machine learning model used originally for classification, but can also be modified for regression tasks. A decision tree can be thought of as a graph in the shape of a tree, where each branch represents a condition or rule that has to be met [18].

A visual representation of a decision tree is given in figure 2.5. This decision tree is a model that forecasts whether the university's basketball team will play outdoors, or train in the gym for a given day. The forecast is made based on three input parameters: air

humidity, weather outlook and whether it is a windy day or not. So, in a given day it can be predicted whether the team will play outdoors by following the conditions (branches) in every node, until a leaf node is reached, whose label is the forecast whether the team will play or not. This is a very simple classification example and usually practical problems require much more complex trees.

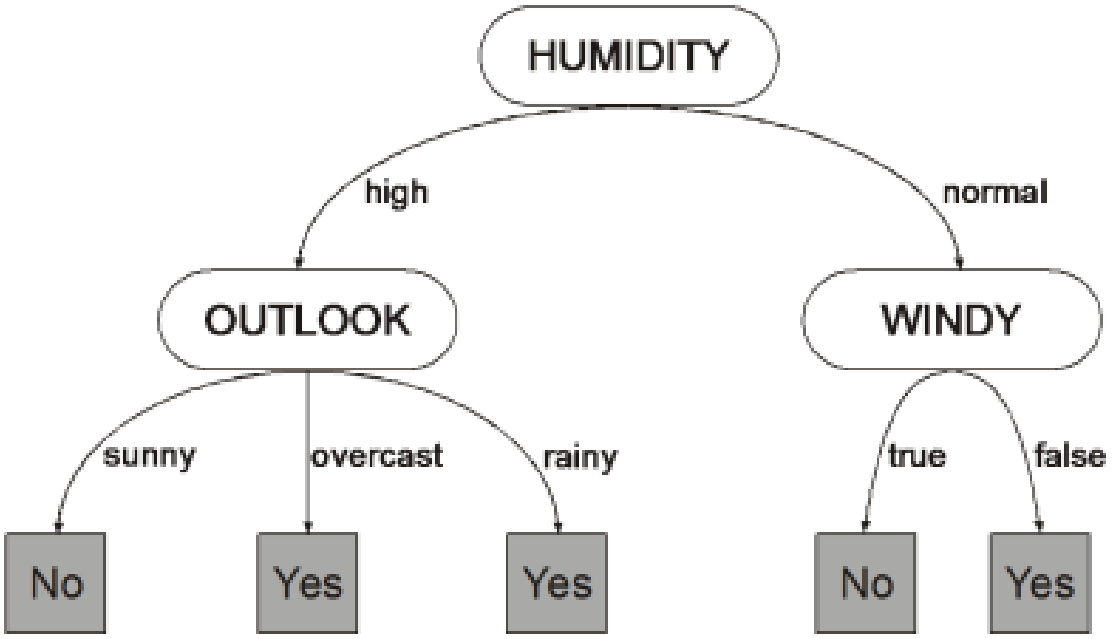



**Figure 2.5:** Decision Tree for playing outdoors [19]

In order to create a forecast model, a decision tree has to be built based on the available data. This process can be considered as model training, but is more often referred to as *tree induction*. Tree induction is the process of building the tree by starting with a root node and recursively adding branches to the tree by analyzing the input and output variables. In theory, the induction of the optimal decision tree is an NP hard problem, which makes it infeasible for practical purposes [20]. However, there are many greedy algorithms that are able to construct a tree very fast, though suboptimal [18].

The Hunt's algorithm is a typical such algorithm, which is the basis for many others. In this algorithm the tree is built recursively by splitting the dataset in smaller ones and associating every smaller dataset with a new node. The algorithm for building a decision tree is as follows [18]:

**Definitions:**

1. Let $X = \{x_1, x_2, ..., x_c\}$ be the set of attributes (independent/input variables) in the dataset. The features may be numerical or categorical values.

2. Let y be the dependent variable (output variable) in the dataset. In regression this variable would be a numerical variable, whereas in classification it would be a categorical variable.

3. Let $Dn = \{(x_{11}, x_{21}, ..., x_{c1}, y_1), (x_{12}, x_{22}, ..., x_{c2}, y_2), ..., (x_{1n}, x_{2n}, ..., x_{cn}, y_n)\}$ be the dataset of n observations.

**Algorithm:**

1. Associate the dataset $D_n$ with a new node. If the dataset satisfies the stopping condition, stop the algorithm and consider the node as a leaf node.

2. Select one of the attributes from $X$ that the set will be split on. The attribute that produces the best split must be chosen.

3. Create a branch for every outcome of the selected attribute.

4. For every branch, split the dataset based on the outcome of every branch.

5. Recursively apply the algorithm.

As it is obvious, this algorithm is too general. For instance, it doesn't specify the stopping condition (step 1). Furthermore, it is not obvious how to select the attribute that produces the best split(step 2). There are many ways to do these, and they are implementation specific. Moreover, there are many mathematical ways to determine the best split, which also differ between regression and classification. These issues will not be explained any further.

During the prediction, if the decision tree is a classification tree, when following the branches of the tree, the outcome of the classification is the class with the majority number in the leaf node. If it is a regression tree, the outcome is the average of the dependent variables present in the leaf node.

### 2.3.2 Random forests

Decision trees are simple and intuitive machine learning models, but they have a major drawback. Decision trees usually overfit training data, by trying to accommodate outliers and peculiar cases [21]. This property usually leads to bad predictive results in the test set. However, many decision trees can be trained on the same set and the aggregated result is usually much more accurate than the result of a single tree [22].

In principle, training a random forest is to induce many decision trees using the available dataset. However this presents a problem: the tree induction process on the same dataset always yields the same tree. In order to avoid inducing the same tree over and over, randomness is introduced in the training phase. The algorithm for training a random forest is as follows [23]:

**Definitions:**

1. Let $X = \{x_1, x_2, ..., x_c\}$ be the set of attributes (independent/input variables) in the dataset.
2. Let $n_{tree}$ be the number of trees in the forest
3. Let $n_{att}$ be a predefined number that satisfies the condition: $0 < n_{att} < c$.

**Algorithm:**

1. Randomly select $n_{tree}$ bootstrap samples from the dataset. A bootstrap sample is a sample equal in size to the whole dataset, but the drawing of the observations is made with replacement. Optionally, the whole dataset can be used as is, instead of bootstrap sampling.
2. From every bootstrap sample, induce a decision tree using the Hunt's algorithm, but with a small difference. When branching the tree, instead of selecting the best split among all the attributes (see step 2 in the Hunt's algorithm), select the best split among $n_{att}$ randomly chosen attributes.

Predicting new data is achieved by feeding the inputs to every decision tree and aggregating the result. For a classification task, the aggregation is the majority of votes by the trees. In a regression task, the average of the values produced by the trees is considered as the result of the whole forest.

## 2.4 Load forecasting input parameters

The accuracy of any forecast model depends on the selected inputs and the influence that they have on the output. Poor choice of input parameters can lead to the degradation of the forecast accuracy, whereas carefully chosen input parameters can make a very powerful model.

Usually, the choice of model inputs has been made based on intuition and human expertise, because there is no exact science in determining the best inputs [24]. However, there

exist mathematical tools that you can apply on the the measured output variable and a candidate input variable, that suggest whether the two variables are related to one another somehow. This section will be dedicated to the presentation of a few common input parameters employed in the load forecasting research domain as well as a mathematical tool that is used to imply whether two variables are dependent.

### 2.4.1 Historical load data

Previous load demand data are the most important and influential features in forecasting models. When trying to predict the load demand in the next time step, specific load demand values of the past have proven to affect it at a high degree [24]. The load variables used in forecast models depend on the resolution of the dataset, however there are some common variables that will be discussed here. They are:

- **Previous $n$ time steps**. The load demand value in a specific time step is closely related to the load values in the recent time steps [25]. For example, if the resolution of the data is one hour, the load value at 13:00 is closely related to the load values at 12:00, 11:00, ... and so on. However, this is a delicate issue, because using too many recent time step values may negatively affect the prediction. This may happen because
- **Previous day, same time step**. Electricity consumers in households tend to have a daily routine, which affects their consumption behaviour [26]. For that reason, it is beneficial for the model to have as input the load demand value of the previous day, at the exact same time step. For example, if the resolution of the data is one hour and the model is trying to predict the load at 9 April 2019 13:00, the input variable in question would be the load value at 8 April 2019 13:00.
- **Previous week, same day, same time step**. Just like there is a daily routine in households, there is usually a weekly routine as well. This practical routine is generally indicative of an underlying pattern in the data that is repeated weekly. That's why the load demand value of the one week ago time step is commonly used as an input variable for load demand prediction [27]. For example, if the resolution of the data is one hour and the model is trying to predict the load at 9 April 2019 13:00, the input variable in question would be the load value at 2 April 2019 13:00.
- **Average load in the previous day**. The average load demand of the previous day may prove to be very influential in some specific cases. However, it is not as common as the previously mentioned features.

There are many other features that can be extracted from historical load data because the load time series are highly auto-correlated [25]. Their influence rate varies from case to case and careful analysis is necessary to determine if they qualify as input variables for load forecast models. Some other worth mentioning features are average load of previous week, standard deviation of load values in the previous day, maximum load value of the previous day, etc.

### 2.4.2 Date and time attributes

Date and time attributes are features that are related to the date or time when a specific load measurement was taken. Along with the actual load values that were introduced in the previous subsection, these variables help to uncover the underlying patterns in the data. Some common date and time attributes are :

- **Daily offset** of the time step when the measurement was taken is an important input variable [28]. This is an ordinal number. For example, if the resolution of the data is one hour and the measurement was taken at 3 AM, this number would be 3 (or 2, if the counting starts from 0).
- **Day of week**. Intuitively, people tend to behave differently at different days of the week. For instance, one family might be doing laundry on Thursdays, whereas some other individual might be baking a pie every Monday. That's why it is important to include the day of the week as an input variable in the form of an ordinal number: 1 to 7 (or 0-6) [28].
- **Working day**. This is a boolean value indicating if the day being considered is a working day or not. The value would be 1 if the day is a working day and 0 if it is a day off (or the other way around). This variable is very important, because the work schedule greatly affects the electricity consumption behaviour [24]. Intuitively, in households, the load demand is higher in days off, whereas in offices and factories the load demand is higher in working days.
- **Holiday**. This is a boolean value indicating if the day being considered is a holiday or not. The value would be 1 if it is a holiday and 0 if it is not (or the other way around). Usually, holidays are days off, but they affect the consuming behaviour based on the festive nature of the day. For example, it is customary for Norwegian families to make the so called "pinnekjøtt" for Christmas. Pinnekjøtt is a salty lamb dish cooked for several hours, which is a cause for a high electricity consumption for Christmas. That is why, the *holiday* variable may have a big impact on prediction for those specific days [29].

Other date and time features that capture more specific patterns are used. These features also depend on the culture, geography and climate of the population being studied.

### 2.4.3 Weather variables

There is a strong relation between various weather parameters and the load demand. It has been continuously proven and demonstrated that meteorological conditions greatly affect the electricity consumption [30] [31] [32]. A few weather parameters that may affect the load demand will be discussed below.

- **Dry bulb temperature**. Dry bulb temperature is the measured temperature of the air that is not affected by the sun radiation and air humidity [33]. It is the common temperature that is usually reported by weather forecasts. This weather parameter has proven to affect energy consumption the most [34].
- **Dew point temperature**. Dew point temperature is the temperature at which the moisture in the temperature condenses [33]. It is closely related to dry bulb temperature and the humidity level of air and it is very important because it contributes to the so-called "real feel temperature". That is why dew point is an important parameter that affects the electricity consumption [24].
- **Relative humidity**. In layman's terms, relative humidity is the ratio of the mass of water vapour found in the air over the maximum mass of vapour that the air can keep[33]. Just like dew point temperature, relative humidity affects the "real feel temperature" and can be used in forecast models instead of dew point or together with it [34].
- **Cloud cover**. Cloud cover is the ratio of the sky that is covered by clouds [33]. During the day, cloud cover affects the received sunlight, which is why it may encourage people to increase the usage of electric light during the day [35].

There are many other weather parameters that can be used in load forecast models, but the aforementioned ones are the most common. However, when utilizing weather parameters as inputs in a load forecast model, an accurate weather dataset and weather forecast is crucial for the accuracy of the load forecast model.

### 2.4.4 Correlation

Correlation is a technique that measures how much two variables are related to one another [36]. The correlation of the variables is measured by computing the correlation coefficient,

which is in the range of $-1$ to $+1$. The value 0 suggests that the variables are not related at all. The value $+1$ indicates that there is a strong positive relation between the variables, whereas the value $-1$ suggests a strong negative relation between the variables.

There are a few correlation coefficients, but the most common is the Pearson correlation [36]. The Pearson correlation tests the existence of a linear relationship between the variables. It is usually denoted by $r$ and the equation is:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.3}$$

where $n$ is the number of the observations, $x$ and $y$ are the variables being tested for correlation, $\bar{x}$ is the mean of the $x$ variable and $\bar{y}$ is the mean of the $y$ variable.

# Chapter 3

# Related Work

## 3.1 Smart Meter Data Analytics

Smart meters have seen widespread use in the past decade by the electrical distribution companies. They are quickly replacing all the traditional analog electricity meters. In 2009, only 4.7% of the households in the US had them installed [37]. That number grew to almost 50% in 2016 [38]. Optimistic figures are estimated for Europe as well. The EU's Third Energy Package obligates the member countries to replace 80% of old meters with smart ones by the end of 2020 [39]. According to The Norwegian Water Resources and Energy Directorate (NVE), every Norwegian household has a smart meter installed starting from January 2019 [40].

Data gathered from smart meters has opened the path to a lot of research opportunities. According to [41], the three main areas of research in smart meter data science are *Load Analysis*, *Load Management* and *Load Forecasting*. Other research areas that have not received as much attention are *Connection Verification*, *Outage Management*, *Data Compression* and *Data Privacy* [41]. In this chapter, a brief overview of *Load Analysis* and *Load Management* will be covered. Then, a more in-depth literature review of *Load Forecasting* will be presented.

## 3.2 Load Analysis

Load analysis is the practice of inspecting the load data gathered by the smart meters in order to gain insightful information about the consumer and its behaviour. It is also related to the analysis of the data quality, with a special focus in bad data detection.

Load profiling is the practice of classifying consumers in large groups with certain properties based on their behaviour. Often, this process first requires the detection of such groups, using clustering techniques. The properties of these groups are related to different factors such as weather, geography, consumer attributes, etc [42].

Three popular clustering techniques were compared in [43]. These investigated techniques were k-means, k-medoid, and Self Organising Maps. According to the results, SOM proved to be the more appropriate technique for consumer load data clustering. Following this result, SOM was used to cluster the consumers, yielding ten unambiguous consumer profile classes. Then, the consumers were classified using a multi-nominal logistic regression, which yielded a value that represents the level of association of the consumer to every class. The highest association value suggests that the consumer belongs to the corresponding profile class.

A k-medoid based clustering algorithm was proposed by [44]. In the proposed technique, the Hausdorff distance was used as the distance metric between the data points and the medoids. The algorithm proved to be very efficient, but has a major drawback. It requires a predefined number of seeds. The paper does not propose any method for seed selection.

A data cleaning method for load curve data was proposed in [45], which detects outliers caused by inaccurate measurements, network transmission errors, malicious attacks, etc [45]. In addition, the algorithm is able to complete missing data to some extent. This method uses a Principal Components Pursuit (PCP) based algorithm. The algorithm exploits the sparsity of the outliers and the low-intrinsic dimension of the load curves.

An important research area in smart meter data analysis is electricity theft detection. An electricity theft detection technique was proposed by [46]. This technique makes use of the measurable power loss in the distribution network caused by technical reasons (called technical loss). However, the network also suffers from the so-called "non-technical loss", which in a nutshell is the electricity consumed but not billed. Among the reasons for non technical loss is also theft detection. Non technical loss is calculated by substracting the measured technical loss from the overall network loss. The authors of this paper developed a predictive model based on these network losses that takes into account the network's resistance dependence on temperature. They demonstrated that the model has a very good performance in theft detection.

This was a very short review of the Load Analysis research. A lot more research has been conducted in this field, but it is not the focus of this thesis, so they will not be further explored.

## 3.3 Load Management

Load management is a set of actions and measures undertaken by the electricity distributors to affect the consumers behaviour [47]. The goal is to level out the daily load curve of every consumer. The benefit of doing this is the reduction of the peak load demand, by evenly distributing the demand throughout the day. There is a big economic benefit related to that, since high load demands may impel the energy companies to invest a lot of money in the upgrade of the distribution network or build new energy plants.

In load management it is important to correctly classify load profiles to specific classes, such as commercial, residential, industrial, etc. A classification method based on the Fourier Transform of the smart meter data was proposed in [48]. Smart meter data are time series data that can be treated as discrete time signals, therefore they can be easily represented in the frequency domain by using Fast Fourier Transform algorithms. The load profile is converted to its frequency domain representation, thus yielding a set of frequencies that identify the load profile. The coefficients of these frequencies are used as inputs to a classification model called Classification and Regression Tree (CART). The proposed method proved to be a very efficient classification technique.

As implied, peak load is a very important aspect of load management. Determining when the peak load is going to be and how much load there will be at that specific moment is crucial. A short term load forecasting technique (STLF) with peak load prediction capabilities was introduced in [49]. The authors in this paper have used a modified version of the Auto Regressive Integrated Moving Average (ARIMA) algorithm to build a regression model, which has yielded very good forecasting results.

In order to avoid an electric overload during peak hours, electricity costumers have to be encouraged to refrain from consuming electricity during those hours. However, electricity is a commodity that has been long taken for granted and distributors cannot just interrupt the supply for certain customers. A good solution that is frequently used in practice is the introduction of dynamic electricity prices. In deregulated energy markets, the electricity distributors may increase its price for specific time intervals and keep it lower for time intervals when the consumption is low. This has proven to be a very effective technique for load management in practice [50]. The dynamic prices technique yields very interesting socio-technological results, but on the other hand, it introduces a new problem: setting the actual price. One aspect that would help electricity providers to plan the prices is the ability to predict the market prices. If the power company is able to predict the price fluctuations to some extent, it can make better decisions.

It has been demonstrated that Artificial Neural Networks can be used to perform accurate price forecasts [51]. The devised model is a classical feed forward neural network. According

to the experiments, this model was able to predict the price with an error less than €0.01 in 85% of the cases and less than €0.075 in 70% of the cases. In long term forecasting, the model was capable of predicting the price with an error less than €0.05 in 50% of the cases.

A highly accurate price forecasting model was introduced in [52]. This model was based on Combinatorial Neural Networks (CNN). The training of this CNN was done by a improved design of the chemical reaction optimization (CRO) algorithm. CRO, is a more efficient training algorithm compared to the classical Stochastic Gradient Descent, because it is able to explore the solution space in many directions at the same time, thus avoiding the common "local minima" trap. The devised model has been extensively tested with real data from the electric markets of Maryland US, Pennsylvania US, New Jersey US and Spain.

## 3.4 Load Forecasting

Short term load forecast is the focus of this thesis, that is why the literature related to load forecasting will be more comprehensive. Load forecasting is divided in three categories: short term load forecasting, medium term load forecasting and long term load forecasting. All three will be discussed here, but with a special focus on short term load forecast.

Load forecasting techniques are commonly divided in four categories [53]:

- Machine Learning Techniques
- Statistical or Probabilistic Techniques
- Hybrid Techniques
- Rule based Systems

### Machine Learning Techniques

In one of the earliest works conducted in the field of load forecasting [54], the authors propose two Artificial Neural Network based methods for 24 hours ahead prediction of the load. The first method is a static method, because it forecasts the next 24 hours at the same time. The second method is considered dynamic, because it forecast the next 24 hours incrementally, using the prediction of one hour load as input for the prediction of the next hour. The methods were tested with real world data provided by a South Korean electrical company. According to the tests, both methods yielded adequate results, with a

similar forecasting error of 2%. However, the dynamic method proved to be faster in the training phase and it also gave better results in the forecast of peak hours.

A Deep Neural Network based short term load forecasting framework was introduced in [55]. This framework defines the methodology to pre-process the dataset, train the model and use the model to predict the load demand for the next 24 hours. The training data included historical weather data, date and costumer consumption data of forty industrial entities. Two types of DNNs were built and compared: a pre-trained Restricted Boltzmann Machine (RBM) and a normal untrained DNN with ReLU as the activation function. Both models proved to be roughly equally accurate in prediction. However, when compared to Shallow Neural Network (SNN), ARIMA and double seasonal Holt Winters (DSHW) models the DNN models proved to be more accurate with an average improvement of MAPE up to 6.77% and an average improvement of RRMSE up to 11.51%.

A more complex neural network architecture was proposed in [56]. The model is a Recurrent Neural Network (RNN) based on Long Short Term Memory (LSTM). According to this paper, such a network is able to better address the problems of non-linearity, non-stationarity and non-seasonality of the electrical load time series data, compared to conventional approaches like Deep Neural Networks or simple Recurrent Neural Networks. It is also emphasized that such a model is superior to classical linear models like ARIMA, because it handles the non-stationarity of the data better. The model was trained with ten days of data and tested for the next day. Then it was compared against other forecasting methods such as SVR, NNETAR, NARX and SARIMA based on RMSE and MAPE. The LSTM model proved to be the more accurate with a MAPE equal to 0.0535 and an RSME equal to 0.0702. The next best accurate model proved to be NARX with a MAPE equal to 0.1192 and an RSME equal to 0.1446.

#### Statistical or Probabilistic Techniques

ARIMA is a statistical model that is commonly used to predict time series data. ARIMA is a generalization of the ARMA model, devised with the purpose to handle non stationary data [57], which is the case of electrical load data [58]. One of the earliest works that use ARIMA to forecast electrical load is [58]. In this work, only historical data were used to predict future load values (which is an inherent limitation of ARIMA). The model introduced yielded an average forecast error equal to 4.25%, which is a relatively good forecast accuracy.

A more advanced ARIMA based method to predict electrical load time series data was proposed in [59]. The paper presented four new algorithms, namely: SWH2A, SWHSA, SWDP2A and SWDPSA. SWHSA and SWDPSA are seasonal sliding window ARIMA

based algorithms, whereas SWH2A and SWDP2A are non-seasonal sliding window ARIMA based algorithms. The algorithms were evaluated using a data set of hourly electricity consumption for a 16 months period. The best performing algorithm of all was the Sliding Window Daily Profile ARIMA algorithm (SWDP2A) with an average MAPE of 9.047%. SWHSA yielded a similar accuracy with a MAPE equal to 9.532%. A very interesting finding in this paper is that these algorithms perform well even when the data set size is small.

Another important, yet overlooked statistical method for load forecasting is the Multiple Linear Regression (MLP) model. Multiple Linear Regression is a supervised learning method that tries to approximate (predict) a variable that is dependent on some other independent variables by defining a linear relationship between them [60]. MLP was used in [61] to perform a long term forecast of the load consumption in an electrical grid in Palawan, Philippines. There were three independent variables used in the regression model: historical load data, costumer growth and development plans of the grid. The model proved to be highly accurate, with a minimum MAPE of 0.16%. The forecast results for different years and their associated MAPE is depicted in table 3.1.

| **Year** | **Average Historical Load (kW)** | **Forecasted Average Load (kW)** | **MAPE** |
|---|---|---|---|
| 2011 | 4,380 | 4,496 | 1.25% |
| 2012 | 4,650 | 4,883 | 3.58% |
| 2013 | 5,190 | 5,270 | 0.16% |
| 2014 | 5,050 | 5,657 | 0.36% |
| 2015 | 5,340 | 6,045 | 5.97% |
| | | **Average MAPE** | **2.26%** |

**Table 3.1:** Forecast results of [61]

### Hybrid Techniques

A hybrid technique is the combination of two or more techniques with the purpose to exploit the individual advantages of each, or to dampen the negative effects of the drawbacks of a specific technique. In the load forecast research literature, it is most common to find artificial neural networks in combination with other techniques like fuzzy logic algorithms, evolutionary algorithms, regression algorithms, etc [24].

A hybrid method that combines an artificial neural network model with a genetic algorithm (GA) was introduced in [62]. In this paper, a conventional artificial neural network was used to predict the load demand of specific consumers for the future 24 hours, using historical load data, weather data and stock market index data. The genetic algorithm was

employed to discover the most important and influencing input parameters. The hybrid model was trained and tested with three distinct datasets. The smallest MAPE achieved in the test phase was about 2.053%. The results of the conducted experiments with the different datasets and methods are summarized in table 3.2. As seen from the table, the usage of the genetic algorithm for feature selection always improves the accuracy of the model.

| **Dataset #** | **GA feature selection** | **Number of features** | **MAPE** |
|---|---|---|---|
| 1 | Yes | 16 | 2.053% |
| | No | 32 | 2.284% |
| 2 | Yes | 22 | 2.549% |
| | No | 33 | 2.645% |
| 3 | Yes | 13 | 1.935% |
| | No | 30 | 2.178% |

**Table 3.2:** Forecast results of [62]

Another hybrid method for short term load forecasting was introduced in [63]. In this paper, a combination of Wavelet Transform (WT), a Gram-Schmidt based Feature Selection algorithm and Support Vector Machine (SVM) were used together to predict the next hour load demand. WT was utilized to represent the load data (treated as signals), using the Coiflet wavelet. Temperature in addition to the components of the decomposed signal are considered as input. Then, the Feature Selection algorithm was applied to identify the most influencing inputs. These selected inputs were used as inputs in a SVM model. Based on the test results, this hybrid technique achieved a minimum value of MAPE equal to 1.26%. Table 3.3 shows summarized results of the forecasting accuracy for different week days and different seasons.

| | **Spring** | **Summer** | **Autumn** | **Winter** |
|---|---|---|---|---|
| **Saturday** | 1.76% | 1.18% | 3.02% | 1.67% |
| **Sunday** | 1.09% | 1.99% | 2.33% | 1.28% |
| **Monday** | 1.68% | 1.04% | 1.49% | 0.88% |
| **Tuesday** | 1.44% | 1.31% | 1.54% | 1.32% |
| **Wednesday** | 1.23% | 1.65% | 2.16% | 1.9% |
| **Thursday** | 1.24% | 0.99% | 1.19% | 2% |
| **Friday** | 0.91% | 0.67% | 1.45% | 1.5% |
| **Average** | 1.33% | 1.26% | 1.89% | 1.51% |

**Table 3.3:** Average MAPE values for WT+FS+SVM [63]

Then, the same technique described above was tested using an artificial neural network (ANN) instead of SVM. The experiments concluded that the SVM based method was more accurate, except for summer days, where the ANN based method provided a slightly better accuracy. Then, both SVM based and ANN based methods were tested with and without the application of Wavelet Transform. The usage of WT proved to usually increase the accuracy of the model, but not every time. The results of the comparison of the four mentioned methods is summarized in table 3.4.

| | **Spring** | **Summer** | **Autumn** | **Winter** |
|---|---|---|---|---|
| **ANN + FS** | 4.189% | 2.436% | 3.98% | 3.166% |
| **SVM + FS** | 3.26% | 2.531% | 2.98% | 2.38% |
| **WT + ANN + FS** | 2.126% | 2.24% | 4.26% | 4.72% |
| **WT + SVM + FS** | 1.35% | 1.26 % | 1.89% | 1.51% |

**Table 3.4:** Average MAPE values for different methods [63]

### Rule Based Systems

Rule based systems are systems modelled by humans that are experts of the specific domain [53]. Based on the extensive knowledge that these people might have in their domain, they are able to devise a series of rules that can handle many scenarios of the problem in question. Loosely speaking, rule based systems are sequences of *if...else...* instructions.

A rule based system was introtuced in [29] for short term load forecasting in special days, like holidays or other days that exhibit a non typical load profile. This rule based system was trained and tested with a dataset that contained French load data for a period of nine years. The first step in this rule based system was the categorization of the special days, based on their load profile by the specific domain experts. In the French dataset, seven special day categories were obtained. Then, for each category a different Seasonal Autoregressive Moving Average (SARMA) model was trained. So, when performing a forecast for a special day, the day is classified into one of the seven categories and then the corresponding trained SARMA model is used for the prediction. According to the test results, the rule based SARMA was a much more accurate model than simple SARMA.

# Chapter 4

# Solution Approach

## 4.1 Introduction

After the presentation of the theoretical foundation behind neural networks, random forests and the intuition of load forecast models, a concrete case study will be introduced. First, the case study will be described with a little background detail. Then, the dataset used in this research will be introduced and described. The dataset is of little use in its raw form, that's why data preprocessing is applied. At last, the Feedforward Neural Network and Random Forests architectures are discussed, which are the most important part of this chapter.

## 4.2 Case study

This chapter will be dedicated to the case study of the electricity consumption behaviour of a set of households located in Stavanger, Norway. More than one year worth of load demand data and weather data will be analyzed. The intention is to build predictive machine learning models that will be able to forecast the next step load demand of respective households.

## 4.3 Dataset description

The original, unprocessed dataset contains load demand data of several households in Stavanger, Norway from 7 February 2017 to 19 April 2018. The data are recorded measurements taken by smart meters installed in every household. Every measurement taken by the meters has an epoch timestamp associated to it that indicates the time when

the measurement was taken. The timestamp is the number of milliseconds that have passed since 1st of January 2017 midnight up to when the measurement was taken. The timestamp is timezone independent and refers to UTC time. This dataset was provided by a local electrical utility company in Stavanger.

The resolution of the data is ten seconds, i.e. the meters have taken measurements every ten seconds. This means that in a normal measuring day, the meters have collected about 8640 data samples in a day. This is important to highlight, because for various reasons, certain days might have collected much less data. The total number of measurements taken is $131,916,982$.

A measurement that a meter records is the total load demand of the household at that specific time, or in other words: it is the electrical power being consumed by the household at that moment. It is measured in watts ($W$) and is the rate at which the electrical energy is being consumed.

Another dataset that is complementary to the main one is also available. This dataset is a collection of weather measurements taken in the time interval from 7 February 2017 to 19 April 2018. It contains different weather parameters such as dew point, apparent temperature, dry bulb, wind speed, humidity, cloud cover, etc. The resolution of the data is one hour. This dataset was obtained using the the public API Dark Sky [64].

## 4.4 Data preprocessing

Data preprocessing is the practice of transforming and modifying the existing data for better utilization and performance. The purpose of data preprocessing is twofold: data purification and data formatting.

Data purification, or data cleaning, is the practiec of detecting bad or missing data and removing or replacing them with other data with the purpose to improve the overall quality of the dataset [65]. The data may be missing or "bad" for many reasons, such as technical errors during measurement, formatting errors, transmission errors, etc.

Data cleaning is crucial to the accuracy of a machine learning model. As already stated, the model is supposed to learn from past observations and if those observations are erroneous, the model is learning things wrong, thus rendering the model utterly useless.

In the three following subsections, three data purification techniques will be discussed:

- Data constraints imposition

- Outlier detection
- Missing data

Data formatting is the process of transforming the available data in a format that can be easily utilized by the machine learning model. In order to keep the training time low and keep the complexity of the program small, every independent variable and the associated dependent variable must be defined explicitly beforehand. The data format will be explained in detail in section 4.4.4.

### 4.4.1 Load demand dataset constraints

A common practice in data cleaning is the definition of a set of constraints that the data needs to satisfy. If some data points do not comply these constraints, they can either be deleted, or modified. When they are modified they can be approximated, based on the other data points. A few constraints that are relevant to this thesis' data set will be discussed in the next section.

The load demand dataset was tested against various constraints. However, not all of them will be mentioned here, but only those that are relevant and which the data failed to satisfy. Below, these constraints will be presented and the measures that were taken to deal with them.

#### Non negative load values constraint

From a physics point of view, in an electric circuit, the power of a passive component (a component that consumes power), is always positive [66]. If the value of power in the component was negative, that would make it an active component (power generator). Considering the fact that none of the smart meters was installed in a power generator plant, negative readings indicate an incorrect measurement of the load.

An analysis on the dataset that tests for negative values concluded that the dataset contains a considerable amount of negative values. More precisely, it contains $277,652$ negative load measurements out of 131 million measurements.

In order to prevent the loss of data, one might consider to approximate these incorrect values. The approximation of a value can be done either by equating it to the value of one of the adjacent data points, or by taking the average of both adjacent data points.

However, a closer inspection revealed that all these negative values are, in fact, continuous temporal ranges. This means that approximating the values based on adjacent data points

would produce negative values again for most of them. In this situation, the best strategy would be to delete all the negative values in the dataset.

#### Non duplicate timestamps constraint

It is logical to expect only one load demand value for a household in a specific moment. Double measurements in the same timestamp would either indicate that the smart meters recorded the measurements more than once by mistake, or the electrical appliances in the household are in a quantum superposition, thus yielding two different valid load values at the same time. The latter is highly unlikely, so duplicate measurements will be considered mistakes.

A careful inspection of the dataset revealed that it contains 20 such duplicates. Apparently it is not a common error and does not pose any complication in the data. It is obvious that the only way to deal with this issue, is by deleting the duplicates. There are no values to approximate, so deletion is the only logical action to take.

#### Timestamps inside valid interval constraint

As previously mentioned, the smart meters have been active in the time frame 7 February 2017 to 19 April 2018. The dataset is expected to contain only timestamps that fall in this time frame and those that fall outside of it, are errors made by the smart meters, either in the measurement process or during the transmission process.

A simple analysis showed that there are ten measurement instances in the dataset whose timestamps fall outside the aforementioned time frame. These timestamps all belong to the date of 1 January 1970, which is clearly a technical mistake made during the recording/transmission process. The reason why this specific pattern of mistakes exists, is unclear.

These measurements can be dealt with in the same ways that were explained at the start of this section: deletion or approximation. Deleting these faulty data points does not pose any practical issues, because they are only a tiny portion of the whole dataset. Thus, even if they are gotten rid of, the overall quality of the data is not affected.

A better way to deal with them is by evaluating the actual timestamp based on the adjacent data points. The timestamps are stored in a strictly increasing fashion. Furthermore, as it has already been established, the resolution of the data is 10 seconds. This means that the faulty timestamp is supposed to be ten seconds less than the timestamp of its

next adjoining data point. In this thesis, this was the technique used to deal with invalid timestamps.

### 4.4.2 Outlier detection

An outlier is an observation whose value differs greatly from the rest of the observations [36]. The existence of outliers may be attributed to measurement errors, or to the high volatility of the data. The latter is a normal state of the data, so removing such outliers is wrong. On the other hand, if an outlier attributed to an error is detected, it should be removed.

There are many outlier detection techniques applicable to time series data. However the load demand data are by nature very volatile. In order to prevent the removal of false positives during the outlier detection process, it was decided to refrain from this practice.

### 4.4.3 Missing data

Missing data usually usually means the absence of values in some observations. However, in time series data, it may take another meaning. In time series data, it may mean that there is no observation at a specific moment when it is expected an observation to exist. The latter is considered here, because there are no missing values in the measurements made by the smart meters.

In our dataset, an observation is expected every ten seconds. If two observations are twenty seconds apart or more, a missing observation has occurred in between. For example, two observations are missing in the following load forecast series:

$$(t_1 = 100, p = 2050), (t_2 = 110, p = 2150), (t_3 = 140, p = 2000), (t_4 = 150, p = 2000)$$

The two missing observations were supposed to be between timestamps $t_2$ and $t_3$.

Data may be missing for many reasons. Some of them may have even been removed during the constraint checking procedure. Whatever the reason, in this particular case, missing observations are not a problem if it is a rare occurrence. However, if there are large continuous blocks of missing data, that would be big problem.

Careful examination of the dataset revealed that there do exist large continuous blocks of missing observations. A simple but effective strategy was employed to tackle this problem.

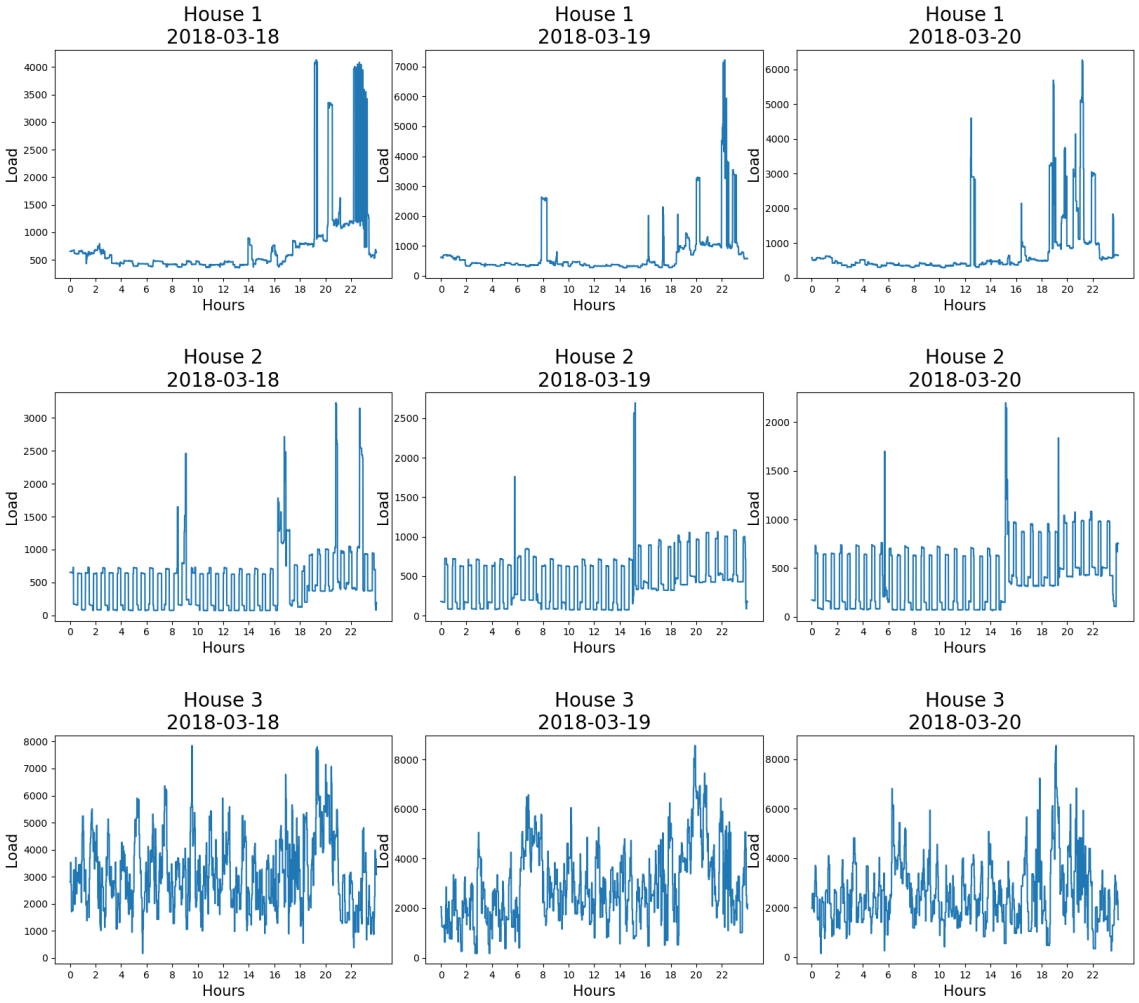



**Figure 4.1:** Random daily load profiles

First, the data were divided in groups, where the measurements of each group belong to the same date. Every group is expected to contain about 8640 measurements, because that's how many measurements can be made in a day with a ten seconds resolution.

Then, every group was inspected to see how many measurements each actually contained. If a group comprised less than 80% of the expected number of observations, the whole list of values of that group is replaced with the list of values of the previous day group. So, if the group contains less than 6912 measurements, the values of that day are replaced with the values of the previous day. There are 818 such groups that contain less than 6912 measurements.

This is a heuristic technique based on the fact that consumers tend to have very similar daily load profiles in consecutive days. Picture 4.1 supports this allegation. In this graph, the load profiles of three random houses in three random consecutive dates are depicted. It is very clear, that the load profiles of each house are very similar in consecutive days.

### 4.4.4 Data formatting

Data formatting is the final step of data preprocessing. In a sense, this process generates the net result of all the processing stages, yielding a high quality dataset that can be easily used by many machine learning models.

This stage also involves the change of the resolution of the data. The original resolution of ten seconds is quite small, making it extremely difficult to forecast future demand because the load is very volatile at such a small interval. More suitable resolutions like fifteen minutes, 1 hour or several hours are more suitable. The increase of the resolution does not only affect the accuracy of the model, but it also substantially reduces the size of the dataset, thus decreasing the training time.

The change of resolution requires the average power consumption during the new time intervals to be calculated. So, if the resolution needs to be changed from ten seconds to one hour, every hour would contain about 360 measurements. The average of these 360 measurements needs to be calculated to determine the load demand of that specific hour.

#### Average load demand

The average load demand of the new resolution is not the arithmetic mean of all the values, and designating it as such is scientifically wrong. The average power being consumed in a time interval is equal to the the integral of the power in that time period divided by the time interval in seconds [66]. The formula is :

$$P_{avg} = \frac{1}{T} \int_{t_0}^{t_0+T} p(t)dt \tag{4.1}$$

where $p(t)$ is the load demand at time $t$, $t_0$ is the starting time of the interval and $T$ is the duration of the time interval in seconds.

However, the above equation is only suitable for continuous load functions. Our dataset contains discrete load measurements, thus equation 4.1 is not applicable. In this scenario, the integral should be approximated. The approximation method used for our dataset is the trapezoid method. The trapezoid method is a very simple and efficient method to approximate the definite integral of discrete values.

Consider the following load measurements taken at time $t_1$, $t_2$, $t_3$, $t_4$, $t_5$ as depicted in the graph of figure 4.2. According the trapezoid method, the approximation of integral of the load values is the area under the curve (which is also the geometrical definition of the integral) [67]. The area under the curve at figure 4.2 is the sum of the areas of all the

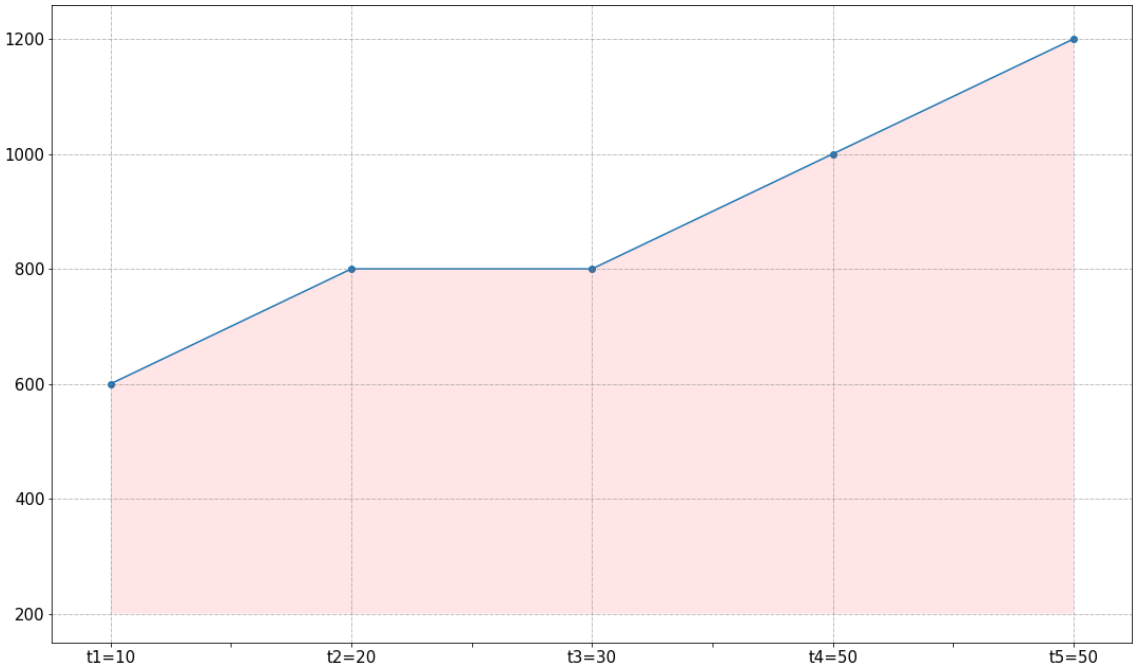



**Figure 4.2:** Trapezoid method

trapezoids formed in intervals $[t_1, t_2]$, $[t_2, t_3]$, $[t_3, t_4]$, $[t_4,t_5]$. The total are under the curve is calculated by the following equation:

$$I = \sum_{i=1}^{N-1} \frac{p(t_i) + p(t_{i+1})}{2}(t_{i+1} - t_i) \tag{4.2}$$

where I is the integral, N is the number of load measurements and $p(t_i)$ is the i-th load measurement. So, to find the average load demand in interval T the following equation is used:

$$P_{avg} = \frac{1}{T} \sum_{i=1}^{N-1} \frac{p(t_i) + p(t_{i+1})}{2}(t_{i+1} - t_i) \tag{4.3}$$

where $t_1 = 0$ and $t_N = T$.

### Data formating

When formatting the data, the new resolution of the data must first be decided. After choosing the resolution, the average load for every time interval is calculated. This is the dependent variable that models need to forecast. Then every other variable must be calculated to fit the new resolution. For example, if the new data resolution is one hour, the hourly average temperature, dew point, humidity, etc. must be calculated. The

| day | hour | temp | humid | p(h-1) | p(h-2) | p(d-1,h) | p(w-1,d,h) | load |
|---|---|---|---|---|---|---|---|---|
| 2 | 19 | 10.63 | 0.89 | 835.12 | 1247.83 | 1087.27 | 1999.57 | 1087.27 |
| 2 | 20 | 10.20 | 0.93 | 1087.27 | 835.12 | 945.44 | 1670.43 | 945.44 |
| 2 | 21 | 9.18 | 0.94 | 945.44 | 1087.27 | 937.98 | 1751.65 | 937.98 |
| 2 | 22 | 9.41 | 0.91 | 937.98 | 945.44 | 1236.15 | 1935.90 | 1036.15 |
| 2 | 23 | 9.28 | 0.93 | 1036.15 | 937.98 | 837.55 | 1715.80 | 930.01 |
| 2 | 0 | 9.01 | 0.92 | 930.01 | 1036.15 | 1057.53 | 1856.56 | 984.12 |
| 2 | 1 | 8.54 | 0.90 | 984.12 | 930.01 | 1482.48 | 18467.80 | 920.20 |
| 3 | 2 | 8.20 | 0.89 | 920.20 | 984.12 | 937.78 | 1465.63 | 702.00 |
| 3 | 3 | 7.10 | 0.89 | 702.00 | 920.20 | 1038.66 | 1450.12 | 807.55 |

**Table 4.1:** Formatted dataset example

calculated average hourly values are compiled together in a table format as depicted in the fictious table 4.1.

In this table, *load* is the dependent variable and the other columns are the independent variables. This is just a small piece of the whole dataset and not all of the available variables have been included. The tabular format is very easy and efficient to utilize, as the input and output features can be easily fetched.

In this thesis, three different equivalent datasets with different resolutions will be investigated. These resolutions are fifteen minutes, one hour and one day.

## 4.5 Inputs selection

As mentioned, there are many independent variables in the dataset that can be used as inputs in the machine learning models. However, more is not always merrier. The input variables should be carefully chosen in a way that irrelevant variables are left out, in order to not swamp the models with trivial data.

The main technique used in the selection of input variables is the correlation analysis as presented in section 2.4.4. The strategy is to select as input the variables with the highest correlation to the load demand.

Since every house in the dataset has its own load demand pattern, it may be useful to see how each variable affects the load demand of specific houses. That's why, it is useful to calculate the correlation coefficients between independent variables and the dependent variable for each house separately.

Table 4.2 depicts the top ten correlation coefficients between every independent variable and the load demand variable for the 1 hour resolution dataset. By "top ten" it is meant

the ten largest correlation coefficients for each specific variable among all the correlation coefficients of all the houses.

The variable names in table 4.2 have been shortened to fit the table. Their full names and descriptions are given in table A.1 of appendix A.1.

| **d** | **w** | **h** | **T** | **D** | **C** | **H** | **p(h-1)** | **p(h-2)** | **p(h-3)** | **p(d-1,h)** | **p(w-1,d,h)** | **p(d-1)** | **p(w-1,d)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.18 | 0.17 | 0.45 | -0.77 | -0.73 | -0.11 | -0.33 | 0.99 | 0.98 | 0.98 | 0.98 | 0.94 | 0.98 | 0.94 |
| 0.14 | 0.17 | 0.33 | -0.72 | -0.68 | -0.1 | -0.22 | 0.86 | 0.86 | 0.82 | 0.8 | 0.78 | 0.8 | 0.72 |
| 0.1 | 0.11 | 0.33 | -0.7 | -0.68 | -0.1 | -0.2 | 0.84 | 0.79 | 0.78 | 0.8 | 0.77 | 0.78 | 0.69 |
| 0.09 | 0.11 | 0.33 | -0.69 | -0.65 | -0.1 | -0.17 | 0.84 | 0.78 | 0.74 | 0.79 | 0.74 | 0.76 | 0.69 |
| 0.09 | 0.11 | 0.32 | -0.68 | -0.64 | -0.09 | -0.17 | 0.83 | 0.77 | 0.74 | 0.78 | 0.67 | 0.76 | 0.68 |
| 0.08 | 0.1 | 0.31 | -0.67 | -0.62 | -0.09 | -0.16 | 0.82 | 0.76 | 0.74 | 0.76 | 0.67 | 0.76 | 0.66 |
| 0.08 | 0.1 | 0.31 | -0.66 | -0.62 | -0.07 | -0.16 | 0.82 | 0.76 | 0.72 | 0.74 | 0.67 | 0.75 | 0.65 |
| 0.08 | 0.08 | 0.3 | -0.64 | -0.6 | -0.06 | -0.14 | 0.81 | 0.74 | 0.71 | 0.72 | 0.66 | 0.73 | 0.64 |
| 0.07 | 0.08 | 0.3 | -0.63 | -0.59 | -0.06 | -0.14 | 0.81 | 0.74 | 0.7 | 0.71 | 0.66 | 0.73 | 0.63 |
| 0.03 | 0.03 | 0.23 | -0.56 | -0.55 | -0.04 | -0.12 | 0.79 | 0.66 | 0.59 | 0.67 | 0.59 | 0.66 | 0.57 |

**Table 4.2:** Correlation coefficients for the 1 hour resolution data

As mentioned in section 4.4.4, there are two more available datasets with different resolutions, namely the 15 minutes resolution dataset and the 1 day resolution dataset. The correlation coefficients are computed separately for these datasets because some specific variables might be more important for some resolutions than the others. Another reason is the fact that in some resolutions, a few extra independent variables may be used that don't make sense for the others.

Table 4.3 depicts the top ten correlation coefficients between every independent variable and the load demand variable for the 15 minutes resolution dataset. Table 4.4 depicts the top ten correlation coefficients for the 1 day resolution dataset.

| **d** | **w** | **s** | **T** | **D** | **C** | **H** | **p(s-1)** | **p(s-2)** | **p(s-3)** | **p(d-1, s)** | **p(w-1, d, s)** | **p(d-1)** | **p(w-1, d)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.16 | 0.15 | 0.4 | -0.74 | -0.7 | -0.1 | -0.31 | 0.98 | 0.98 | 0.98 | 0.97 | 0.93 | 0.97 | 0.93 |
| 0.15 | 0.15 | 0.31 | -0.63 | -0.59 | -0.09 | -0.18 | 0.92 | 0.84 | 0.79 | 0.69 | 0.7 | 0.73 | 0.67 |
| 0.1 | 0.11 | 0.3 | -0.61 | -0.58 | -0.08 | -0.15 | 0.89 | 0.84 | 0.76 | 0.68 | 0.66 | 0.73 | 0.63 |
| 0.09 | 0.1 | 0.3 | -0.61 | -0.57 | -0.07 | -0.14 | 0.88 | 0.8 | 0.75 | 0.67 | 0.63 | 0.73 | 0.62 |
| 0.08 | 0.1 | 0.3 | -0.6 | -0.56 | -0.06 | -0.13 | 0.88 | 0.79 | 0.75 | 0.64 | 0.62 | 0.67 | 0.62 |
| 0.07 | 0.1 | 0.28 | -0.59 | -0.55 | -0.06 | -0.13 | 0.85 | 0.78 | 0.74 | 0.63 | 0.61 | 0.67 | 0.6 |
| 0.07 | 0.08 | 0.28 | -0.58 | -0.55 | -0.06 | -0.13 | 0.85 | 0.78 | 0.73 | 0.6 | 0.6 | 0.67 | 0.59 |
| 0.06 | 0.07 | 0.26 | -0.58 | -0.55 | -0.06 | -0.13 | 0.85 | 0.75 | 0.72 | 0.59 | 0.6 | 0.66 | 0.58 |
| 0.06 | 0.07 | 0.26 | -0.58 | -0.54 | -0.06 | -0.12 | 0.85 | 0.75 | 0.71 | 0.59 | 0.58 | 0.66 | 0.58 |
| 0.06 | 0.07 | 0.26 | -0.58 | -0.52 | -0.05 | -0.12 | 0.85 | 0.74 | 0.71 | 0.59 | 0.58 | 0.65 | 0.57 |

**Table 4.3:** Correlation coefficients for the 15 minutes resolution data

The variable names in tables 4.3 and 4.4 have been shortened to fit the table. Their full names and descriptions are given in tables A.2 and A.3 of appendix A.1 respectively.

Following the results of correlation analysis, three types of ANN and RF models differing on their input variables were built: one for each data resolution. The variables with the highest correlation to the load demand were chosen as inputs.

For the 1 hour resolution models these variables are chosen as inputs:

| d | w | T | D | C | H | min(p(d-1)) | max(p(d-1)) | stdev(p(d-1)) | p(d-1) | p(d-2) | p(d-3) | p(w-1,d) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.26 | 0.25 | -0.95 | -0.9 | -0.16 | -0.44 | 0.97 | 0.98 | 0.97 | 0.99 | 0.97 | 0.97 | 0.95 |
| 0.22 | 0.24 | -0.92 | -0.87 | -0.16 | -0.23 | 0.93 | 0.86 | 0.79 | 0.95 | 0.94 | 0.93 | 0.89 |
| 0.21 | 0.23 | -0.91 | -0.86 | -0.16 | -0.2 | 0.93 | 0.82 | 0.72 | 0.94 | 0.92 | 0.89 | 0.88 |
| 0.19 | 0.21 | -0.9 | -0.83 | -0.14 | -0.19 | 0.9 | 0.82 | 0.72 | 0.94 | 0.92 | 0.89 | 0.87 |
| 0.16 | 0.19 | -0.9 | -0.83 | -0.12 | -0.19 | 0.9 | 0.82 | 0.69 | 0.94 | 0.89 | 0.88 | 0.86 |
| 0.16 | 0.19 | -0.89 | -0.83 | -0.1 | -0.18 | 0.88 | 0.82 | 0.69 | 0.93 | 0.89 | 0.86 | 0.84 |
| 0.16 | 0.18 | -0.89 | -0.83 | -0.1 | -0.17 | 0.88 | 0.82 | 0.67 | 0.92 | 0.89 | 0.86 | 0.84 |
| 0.15 | 0.17 | -0.89 | -0.82 | -0.09 | -0.16 | 0.87 | 0.82 | 0.64 | 0.92 | 0.88 | 0.86 | 0.82 |
| 0.14 | 0.16 | -0.87 | -0.82 | -0.08 | -0.16 | 0.86 | 0.82 | 0.64 | 0.91 | 0.88 | 0.85 | 0.82 |
| 0.13 | 0.15 | -0.86 | -0.81 | -0.07 | -0.16 | 0.86 | 0.81 | 0.6 | 0.91 | 0.88 | 0.85 | 0.82 |

**Table 4.4:** Correlation coefficients for the 1 day resolution data

- **Hour of day** - h
- **Temperature** - T
- **1 hour ago load** - p(h-1)
- **2 hours ago load** - p(h-2)
- **3 hours ago load** - p(h-3)
- **1 day ago load** - p(d-1, h)
- **1 week ago load** - p(w-1, d, h)
- **Yesterday daily load** - p(d-1)
- **Last week daily load** - p(w-1, d)

For the 15 minutes resolutions models these variables are chosen as inputs:

- **Step of day** - s
- **Temperature** - T
- **1 step ago load** - p(s-1)
- **2 steps ago load** - p(s-2)
- **3 steps ago load** - p(s-3)
- **1 day ago load** - p(d-1, s)
- **1 week ago load** - p(w-1, d, s)
- **Yesterday daily load** - p(d-1)
- **Last week daily load** - p(w-1, d)

For the 1 day resolution models these variables are chosen as inputs:

- **Temperature** - T
- **Minimum load previous day** - min(p(d-1))
- **Maximum load previous day** - max(p(d-1))
- **Standard deviation of load previous day** - stdev(p(d-1))
- **Yesterday daily load** - p(d-1)
- **2 days ago load** - p(d-2)

- **3 days ago load** - p(d-3)
- **Last week daily load** - p(w-1, d)

## 4.6 Feedforward Neural Network implementation

One of the models that was built to forecast the load demand is the Feed Forward Neural Network. FFNN is one of the simplest neural networks in supervised machine learning, but it is a very powerful predictive tool nonetheless. In this section, the architecture of the implemented Feedforward Neural Network will be discussed. The architecture is similar to the one introduced in section 2.2, but more specific details on the architecture will be presented.

### 4.6.1 General structure

The architecture of this Feedforward neural network is relatively simple. It contains one input layer with several neurons, which will be discussed shortly. The output layer contains only one neuron, which outputs the load demand. Two hidden layers are used. Various experiments demonstrated that increasing the number of hidden layers does not affect the forecasting accuracy of the model (but those results are outside the scope of this thesis).

Increasing the number of hidden layers doesn't necessarily lead to the increase of the model's performance. In fact, according to the Universal Approximation Theorem, a feed forward neural network is capable of approximating any given nonlinear function using only one hidden layer [68].

In order to determine the optimal number of hidden layers in the network, an experiment was conducted. The dataset of a household from the 1 hour resolution dataset was picked and different neural networks with various hidden layers were trained 10 times times. The MAPE test results were recorded and the average MAPE values for each ANN were compared. The ANN with two hidden layers yielded the smallest MAPE value of all, equal to 12.9%. These results are illustrated in figure 4.3, where the test MAPE values are plotted against the number of hidden layers that the neural network contained.

A few experiments were conducted, testing different numbers of neurons in the hidden layers. The configuration with the best accuracy was 50 neurons for every hidden layer. The results of these experiments are not the focus of this thesis, so they will not be presented here, as to not overwhelm the reader.

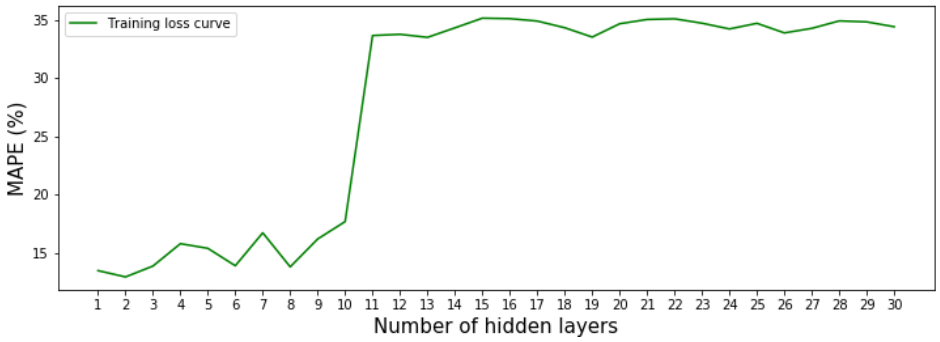



**Figure 4.3:** MAPE depending on number of hidden layers

### 4.6.2 Activation function

The activation function is a very important aspect of the Artificial Neural Network architecture. The wrong choice of the activation function may negatively affect the training speed of the network, or its learning capabilities. In this thesis, the hidden layers use a different activation function from the output layer. They will be discussed in this section.

#### Activation function in hidden layers

The activation function that was chosen for the Feedforward Neural Network in this thesis is the Leaky ReLU for the hidden layers. The Leaky ReLU is a modified version of the the Rectified Linear Unit (ReLU) function. Leaky ReLU is not zero for arguments less than 0, unlike ReLU.

The Leaky ReLU activation function has the following equation:

$$f(x) = \begin{cases} x & \text{if x} \geq 0 \\ \alpha x & \text{if x} < 0 \end{cases} \tag{4.4}$$

where $\alpha$ is a small value close to 0. In the models presented here, $\alpha$ is equal to 0.01. Figure 4.4 depicts the graph of the activation function in question. As seen from the graph, it has a tiny slope for arguments less than 0.

Leaky ReLU has a relatively big advantage compared to ReLU, but first the advantages of ReLU will be discussed, which are inherent to Leaky ReLU as well. The advantages are listed and explained below. Then, the "Dying Relu" problem is explained, which is the reason why Leaky ReLU is a better choice than ReLU.

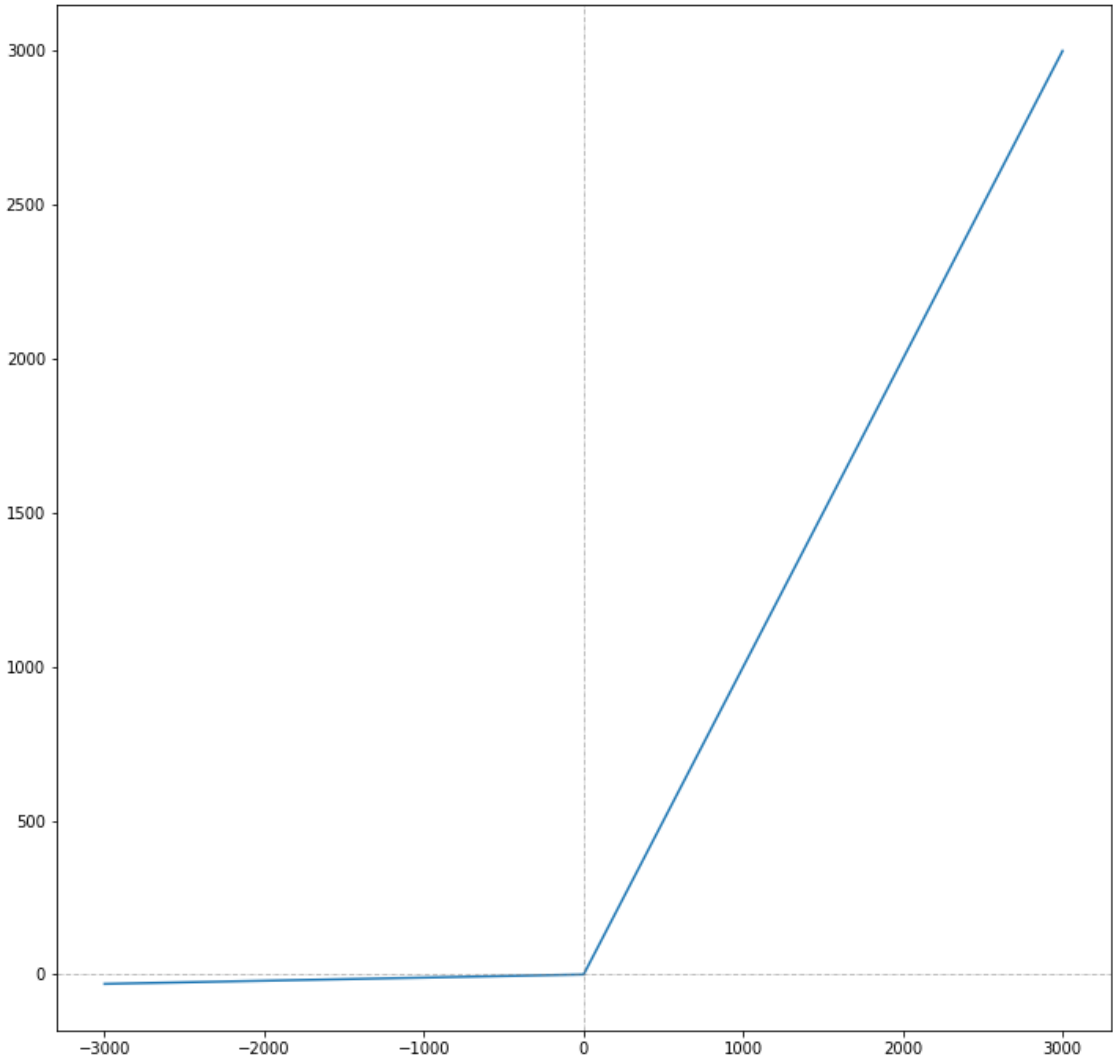



**Figure 4.4:** Leaky ReLU with $\alpha = 0.01$

- **Computation efficiency**

  ReLU is a very efficient function computation wise. The reason is that ReLU only needs to pick the maximum value among 0 and the argument and return it. No computationally expensive operations are required like in other common activation functions, such as sigmoid, hyperbolic tangent, softplus, etc [69]. Leaky ReLU is slower than ReLU in this regard, however much faster than the others.

  In order to support this argument, a simple experiment was conducted. The purpose of the experiment is to reveal how much time is spent on the execution of the activation function during the training process of the ANN model, when it is trained on the 1 hour resolution dataset.

  During the training process of the ANN with the 1 hour resolution dataset, the neurons of the hidden layers in total get activated approximately 894252000 times. That's because there are 2 hidden layers, 30 neurons in each of them, about 40 epochs and

372605 observations. The experiment measures the time that is spent to execute each activation function 894252000 times. It was conducted in a normal personal computer and the results are depicted in table 4.5.

| **Activation function** | **Execution time (s)** |
|---|---|
| Softplus | 1834.47 |
| Unipolar sigmoid | 1579.02 |
| Bipolar sigmoid | 1555.14 |
| Hyperbolic tangent | 630.05 |
| Leaky ReLU | 150.76 |
| ReLU | 136.32 |

**Table 4.5:** Activation functions execution time

It is very obvious that ReLU and Leaky ReLU are far more superior than the other activation functions when it comes to computation speed. ReLU is slightly faster than Leaky ReLU.

- **Vanishing gradient and exploding gradient problems**

The vanishing gradient problem is a problem faced in the training process of Artificial Neural Networks using backpropagation and gradients. In simple terms, during backpropagation, the gradients are multiplied up the layer stack and the gradients in the initial layers of ANN may become very small along this process [70]. This causes the weights of those layers to be updated by a very small amount. For that reason, in some cases, the training of the network may come to a halt. The exploding gradient is the opposite problem, where the gradients take very large values and make the network unstable.

The positive thing about ReLU is that it completely avoids the vanishing gradient and exploding gradient problems [71]. The reason is that the derivative of ReLU is either 1 (for positive input) or 0 (for negative input) [71]. So, the gradient is backpropagated without causing increasingly small or very large gradients up in the layer stack.

These problems are usually present in deep neural networks. They may not be prevalent in the networks discussed here, which are composed of only two hidden layers. However, in theory it is always possible that it occurs even in these situations.

- **Sparsity**

Sparsity refers to a state of an artificial neural network where a relatively large amount of neurons in the network don't fire, i.e. they yield a value equal to 0 [72]. Sparsity is a desirable property of neural networks, because they decrease the calculation complexity of the network [72].

According to [71], the ReLU activation function is quite capable of achieving a state of sparsity in the ANN. The reason for that, is that when the weighted sum of the input weights of a neuron is negative, the function generates a value of 0, which means that the neuron is deactivated. The gradient equal to 0 for negative values also contributes to keeping these neurons deactivated [71].

- **Dying ReLU**

Regardless of the many advantages of ReLU, an artificial neural network that uses ReLU in its hidden layers runs the risk of "dying". The sparsity may increase so much that most of the neurons may become inactive, thus greatly affecting the ability of the network to learn [73].

Leaky ReLU is able to prevent the dying state of the neural network, because its gradient is not 0, but $\alpha$, for negative input [71]. This strategy negatively affects sparsity, but nonetheless, it is a good trade-off between sparsity and a dying state.

It is clear that Leaky ReLU is a very appropriate activation function, as it inherits all the benefits of the regular ReLU, while eliminating the dying ReLU problem.

#### Activation function in the output layer

The activation function used in the output layer is a simple linear function of the form $f(x) = x$. The reason lies in the fact that load forecast is essentially regression. Regression output values should be unbounded, in order to account for every possible outcome. That's why Leaky ReLU, ReLU or any other common activation function cannot be used in the output layer of regression models.

### 4.6.3 Loss function

The loss function is an important part of the neural network. It is a crucial part of the training process because it serves as a metric of how "bad" the model is performing. The minimization of this metric is the ultimate goal of the whole training process.

The choice of the loss function affects the ability of the model to learn during the training phase. There are many heuristic approaches that suggest the best loss function to be used in given situations, based on the distribution of the data, the degree of outliers, etc. However, the only bulletproof method to guarantee the selection of the best performing loss function in a model, is to test and compare many of them. Then, the function that yields the smallest error, can be selected to be used in the model.

The loss functions were tested only on a subset of the one hour resolution dataset. They were compared based on the generated test MAPE values. The experiment involved these loss functions:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Square Logarithmic Error (MSLE)
- Huber loss
- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)

These functions yielded various level of accuracy. Table 4.6 shows the error metric values values yielded by the neural network trained on the 1 hour resolution data of a random house. The testing error metric used is MAPE. The least accurate ANN was trained using the Mean Squared Logarithmic Error function. The best result was achieved by Root Mean Squared with a test MAPE equal to 11.17%. That's why all the neural network models in this thesis are trained using RMSE as a loss function.

| **Loss function** | **Test MAPE** |
|---|---|
| Mean Squared Error | 20% |
| Root Mean Squared Error | 11.17% |
| Mean Square Logarithmic Error | 95% |
| Huber Loss | 15.4% |
| Mean Absolute Error | 12.4% |
| Mean Absolute Percentage Error | 13% |

**Table 4.6:** Test MAPE of a neural network trained on different loss functions

### 4.6.4 Optimization algorithm

The optimization algorithm is the algorithm that is used to minimize the loss function in Artificial Neural Networks. It is a crucial part of the neural network and there are many considerations to take into account when choosing one. Some of these considerations are:

- Computational efficiency
- Memory
- Rate of convergence

- Convexity
- Local Minima vs Global Minima

A relatively new and very well known optimization algorithm is Adam (adaptive moment estimation) [74]. This algorithm is gradient based. Some of the reported advantages of Adam are computational efficiency, ability to deal with sparse gradients, low memory requirements during the execution, suitable for non convex loss functions, etc [74]. Research has also shown that Adam outperforms the classical optimizer Stochastic Gradient Descent, in terms of rate of convergence and error, when it is applied in load forecasting problems [75].

Theoretical and empirical studies were encouraging to accept Adam as the optimizer of choice in the neural networks in this thesis.

## 4.7 Random Forest implementation

In addition to the Feedforward Neural Network implementation, a Random Forest model is built. In this section, its implementation details and design choices will be introduced. These choices will be justified by empirical evidence drawn from experiment conducted with the dataset. In order to keep it simple, the experiments were conducted on the 1 hour resolution data of only a few households. The design choices discussed here are: number of trees in the forest and whether bootstrap sampling should be performed before training the model.

### 4.7.1 Number of trees in the forest

The most important factor in the accuracy of Random Forests is the number of trees in the forest. Intuitively, the accuracy increases as the number of trees increases. This is, of course, bounded by the maximum accuracy that a decision tree can achieve. However, the more trees there are in the model, the longer it takes to train the whole forest.

A small experiment was devised to decide an adequate number of trees. One of the households in the dataset was chosen and its 1 hour resolution dataset was used to train 100 different models. These models differed in the number of trees that they contained. The test MAPE value were recorded and then plotted against the number of trees. The plot is depicted in figure 4.5:

As it can be seen from the graph, the accuracy doesn't change much. It ranges from 10.98% with 350 trees to 12.06% with 10 trees in the forest. Normally, one would consider

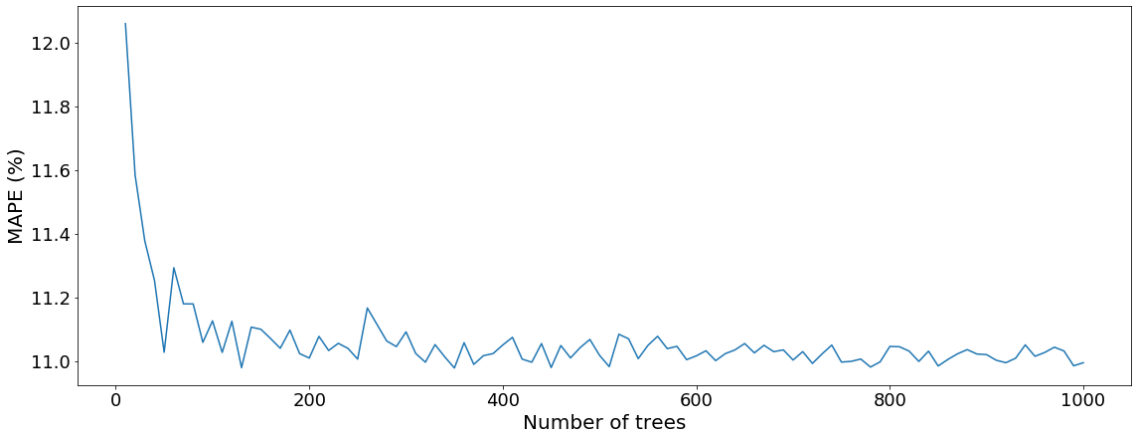



**Figure 4.5:** MAPE values of Random Forest with different number of trees

350 as the optimal number of trees. However, by taking a close look at the results, there are a few other configurations that give almost the same result as the 350 trees option. Table 4.7 depicts 10 configurations with the lowest MAPEs.

| Number of trees | MAPE (%) |
|---|---|
| 350 | 10.9800 |
| 130 | 10.9810 |
| 450 | 10.9815 |
| 780 | 10.9830 |
| 510 | 10.9843 |
| 850 | 10.9864 |
| 990 | 10.9870 |
| 370 | 10.9914 |
| 720 | 10.9944 |
| 1000 | 10.9967 |

**Table 4.7:** 10 lowest MAPE values of Random Forests with different number of trees

As seen from table 4.7 the forest with 130 trees yields an almost exact accuracy as the forest with 350 trees. Since the training and testing time of the random forest increases with the number of trees, it was decided to use 130 trees in the model of this thesis.

### 4.7.2 Bootstrap sampling

Bootstrap sampling is the practice of generating a sample of elements, from an existing set by randomly picking elements with replacement [76]. In the case of Random Forests, bootstrapping is used to generate a training dataset for each tree in the forest. Usually the size of the samples is chosen to be the same size as the original dataset. It is obvious that during sampling, some records may be included more than once and some may not be included at all in the training set.

In Random Forest, bootstrap sampling is used to introduce more randomness during training. However, it can optionally not be used at all and the exact original training set can be used to induce all the trees. This would mean that the only randomness introduced in the algorithm is the random selection of attributes that are used to choose the best split on, as presented in the second step of the algorithm of section 2.3.2.

In order to decide whether the bootstrap sampling technique will be used in the training process of the random forest, a simple experiment was conducted. Five random households were chosen and random forests with and without bootstrap sampling were trained. Their test MAPE errors were calculated to compare the two techniques.

| **Random household** | **Bootstrap MAPE(%)** | **No Bootstrap MAPE (%)** |
|---|---|---|
| 1 | 11.14 | 17.67 |
| 2 | 19.77 | 31.16 |
| 3 | 13.71 | 18.80 |
| 4 | 16.05 | 22.53 |
| 5 | 20.35 | 28.15 |

**Table 4.8:** Bootstrap sampling vs no bootstrap sampling in Random Forest training comparison

Table 4.8 depicts the average test MAPE errors of the households when the Random Forest was trained with and without bootstrap sampling. As seen from the results, in all cases, the error of the Random Forest model was much lower when the bootstrap sampling technique was used. Based on these results, it was decided to train all the Random Forest models of this thesis by using the bootstrap sampling method.

## 4.8 Training, testing and final remarks

As previously mentioned, in data preprocessing, three datasets with different resolutions were generated. Also, each dataset contains data for more than 50 houses. The set of data from one dataset belonging to one house will be called a subset.

A new model is trained for each house. The reason is because every house has its own energy consumption pattern and separate trained models are needed to capture those patterns. Furthermore, in order to study the impact of the data resolution in the forecast, a separate model for each resolution is trained for each house. This is an overwhelming number of trained models, however the highlights of the results will be given, in order not to swamp the thesis with a large amount of crude information.

Before training an ANN or a RF, the corresponding subset is split in a training subset, validation subset and a testing subset. The testing subset comprises 20% of the whole

subset, whereas the validation subset and training subset consist of 16% and 64% of the subset respectively.

A model is trained on a training subset. Thereafter, its accuracy is evaluated by feeding the testing subset, which is a portion of the data that the model hasn't seen before. The purpose behind this method is to check whether the model has memorised the training data (overfitting), or has actually learned its underlying pattern. If the model has a low training error and a high testing error, it indicates that the model has overfitted the data.

The validation subset has a specific purpose as well. Training a model is costly and takes a lot of time. Before delving into lengthy model training procedures it is crucial to know that the model is actually learning. A very simple way, is to take a small subset (validation subset) and use it to validate the model after each epoch. The validation procedure is the same as the testing procedure, where the test subset is fed to the network and a loss value is calculated for that subset. The only difference is the fact that the validation set is fed after every epoch.

If the training loss and validation loss are plotted in a graph against the epoch number, some useful insight on the learning process is provided. If the training loss curve and the validation loss curve converge, then it means that the model is learning. On the other hand, if they diverge, it suggests that the model training is not effective.

Figures 4.6, 4.7, 4.6 depict the validation loss and training loss curves of one random training model from each dataset. As it can be seen from these graphs, the curves are converging, which means that the model is able to learn from the data.

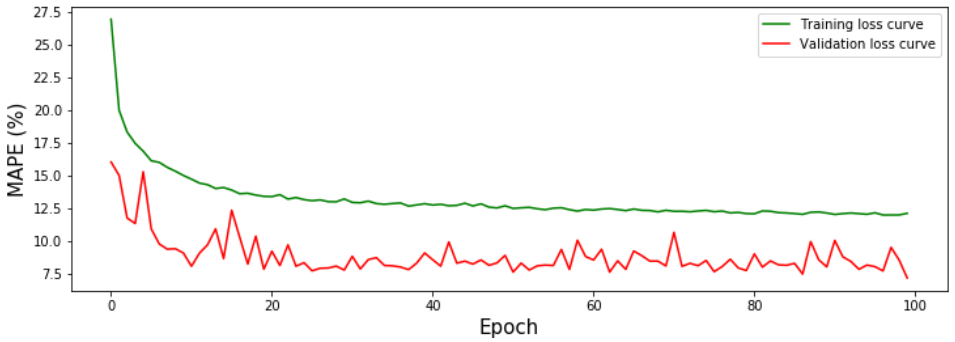



**Figure 4.6:** Training loss and validation loss curves of a household using the 1 hour resolution dataset

In machine learning, it is common to shuffle the data before splitting to prevent overfitting. In time series data, the data should not be shuffled before splitting, because the test data points must have more recent timestamps than those in the training subset. however, after splitting, the training dataset can be shuffled.

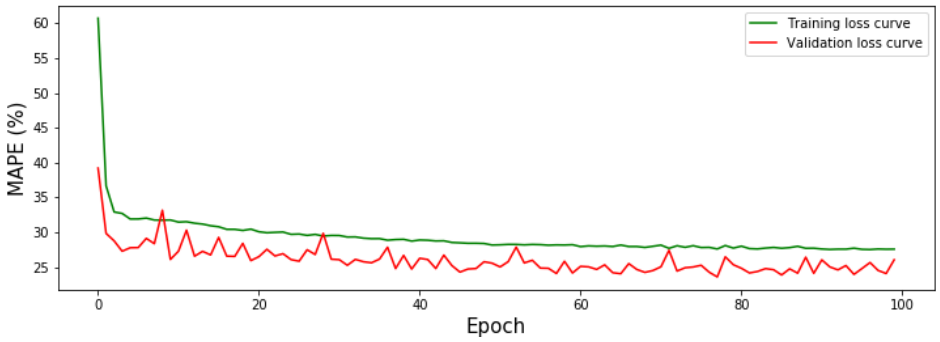



**Figure 4.7:** Training loss and validation loss curves of a household using the 15 minutes resolution dataset

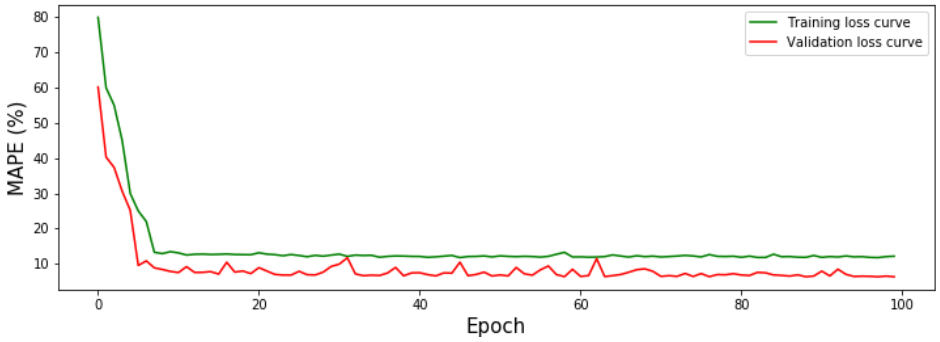



**Figure 4.8:** Training loss and validation loss curves of a household using the 1 day resolution dataset

Before feeding the data to the neural network, the variables are scaled first. Scaling is the process of transforming the variables to fit in a particular range of values. It is used because variables tend to have different ranges of values. Variable scaling stops the larger variables from dominating the smaller ones. When the features are scaled to fit in the same range, all of them have the same influence.

There are many possible scaling formulas that can be used, but here a very simple and intuitive one is used. It is called the Minimum Maximum Scaler. It normalizes the features in the 0 to 1 range. The formula of the Minimum Maximum Scaler is given in equation 4.5.

$$\hat{x_i} = \frac{x_i - min(x)}{max(x) - min(x)} \tag{4.5}$$

where $\hat{x_i}$ is the scaled value, $x_i$ is the real value, $min(x)$ is the minimum value of the variable and $max(x)$ is the maximum value of the variable.

After a neural network model has been trained, its ability to forecast future load demand is tested using the testing subset. Each observation in the testing subset is fed to the trained ANN as input. The output is the forecasted load demand for that specific observation. The forecasted values are compared to the true values (ground truth) and the forecast error is assessed. The error used in this thesis is MAPE. The formula for MAPE is depicted in equation 4.6.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i} \tag{4.6}$$

In the formula above, MAPE is the error as a percantage value, n is the number of observations in the test subset, $y_i$ is the i-th ground truth and $\hat{y}_i$ is the i-th forecast value.

# Chapter 5

# Experimental Evaluation

## 5.1 Experimental Setup

Many experimental studies were conducted with the purpose to assess the performance of the machine learning models presented in chapter Solution Approach and to understand their limitations. In principle, three types of experiments were conducted for each model:

- 1 hour resolution experiment
- 15 minutes resolution experiment
- 1 day resolution experiment

The purpose of these distinct experiments is to study the way load demand patterns are manifested at different resolutions and the ability of the Feed Forward Neural Network and Random Forest to predict the next step load demand, by learning these patterns. The results of the experiments are compared and a conclusion is made based upon them. The experiments were conducted with all the available house subsets. However, the results of only a few houses are given here, for space considerations and easier presentation reasons.

The next subsection will be dedicated to the results of every experiment. In each of them, the test subset of every house is compared to the ground truth. The MAPE metric is calculated to measure the forecast error. Also the ground truth and the forecast values are visualized using graphs, for a more comprehensive and intuitive report of these results.

## 5.2 Experimental Results

The datasets contain data for 55 houses, meaning that there are 55 subsets. Some houses have a more defined load demand pattern than others, which is exactly what machine learning models need, to discover hidden regularities in the data. Some other houses have a very volatile, "chaotic" consumption behaviour and with no clear patterns, which makes it rather difficult to predict the load demand.

When trying to forecast the next step load demand, the MAPE value is a good indicator of the model's ability to predict. A 0% MAPE indicates that the predictor is perfect: it can forecast the next step load exactly as it is. The higher the value is, the worse the forecast is.

However, MAPE is not a bulletproof metric that speaks the absolute truth regarding the model's precision. As it will be obvious later on, it is not able to capture a few intrinsic details of the forecasting process. Nonetheless, it is a useful tool that gives some insight.

There is a wide range of MAPE values yielded in experiments here. In the one hour resolution dataset, the trained models yielded MAPE values ranging from 10% to 85%. In the 15 minutes resolution dataset, this range was from 9% to 82%. In the 1 day resolution dataset, the accuracy was more optimistic, with a range of MAPE from 6% to 45%.

Tables 5.1, 5.2 and 5.3 show 5 houses with the highest MAPEs produced by ANN and RF respectively, from each dataset. Judging from the depicted values in these tables, the machine learning models presented here, have a poor performance in these specific houses, in the respective resolutions. The error yielded in these cases is objectively very high.

| House ID | MAPE (%) |
|---|---|
| H19 | 90.2% |
| H32 | 85.2% |
| H8 | 76.3% |
| H10 | 66% |
| H11 | 60.7% |

(a) Artificial Neural Network

| House ID | MAPE (%) |
|---|---|
| H11 | 80.7% |
| H23 | 49.3% |
| H19 | 49.1% |
| H10 | 47.8% |
| H32 | 47.6% |

(b) Random Forest

**Table 5.1:** Houses with high error values in the 1 hour resolution dataset

In this thesis, the focus will be laid on the instances with the lowest MAPE values, i.e. the models that are the most accurate in forecasting the next step load demand. Tables 5.4, 5.5 and 5.6 depict the forecast errors of the houses that ANN produced the smallest error for. Next to the ANN MAPE error, the corresponding MAPE values yielded by RF, for the specific houses are given. So, in simple terms, the ANN models are used as the baseline for comparing ANN to RF.

| House ID | MAPE (%) |
|---|---|
| H19 | 85.5% |
| H10 | 82.4% |
| H8 | 76.6% |
| H34 | 68% |
| H44 | 65.8% |

(c) Artificial Neural Network

| House ID | MAPE (%) |
|---|---|
| H11 | 72.1% |
| H19 | 70.8% |
| H10 | 54.5% |
| H8 | 53% |
| H23 | 48.5% |

(d) Random Forest

**Table 5.2:** Houses with high error values in the 15 minutes resolution dataset

| House ID | MAPE (%) |
|---|---|
| H11 | 49% |
| H46 | 45% |
| H23 | 44% |
| H19 | 43% |
| H44 | 41% |

(e) Artificial Neural Network

| House ID | MAPE (%) |
|---|---|
| H19 | 42.6% |
| H11 | 36.1% |
| H23 | 34.7% |
| H46 | 33.9% |
| H44 | 33.8% |

(f) Random Forest

**Table 5.3:** Houses with high error values in the 1 day resolution dataset

| House | ANN MAPE (%) | RF MAPE (%) |
|---|---|---|
| H41 | 10.5% | 11.1% |
| H4 | 12.4% | 13.7% |
| H49 | 13.7% | 15.6% |
| H9 | 14.5% | 16.1% |
| H2 | 17.4% | 17.9% |

**Table 5.4:** Houses with low error values in the 1 hour resolution dataset

| House | ANN MAPE (%) | RF MAPE (%) |
|---|---|---|
| H28 | 9.2% | 17% |
| H37 | 10.5% | 12.9% |
| H13 | 10.8% | 40.4% |
| H4 | 11.8% | 15.1% |
| H29 | 13% | 18.5% |

**Table 5.5:** Houses with low error values in the 15 minutes resolution dataset

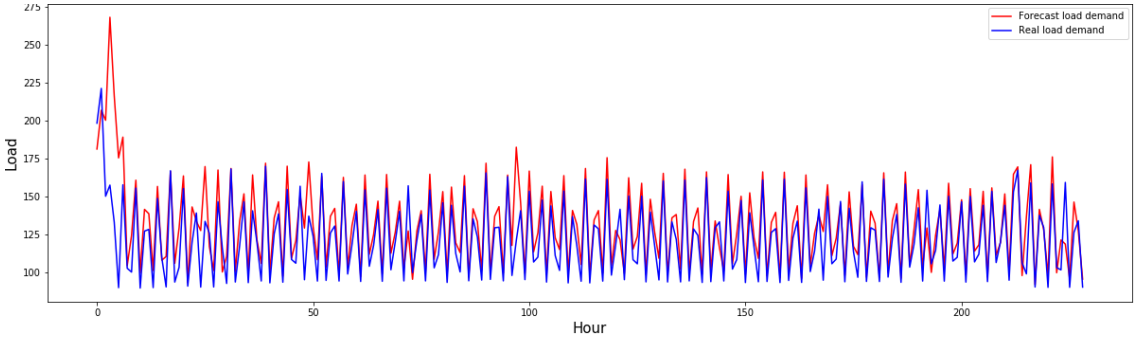
It is obvious that the forecasts of the models trained on the 1 hour and 15 minutes resolution datasets are the least accurate. The yielded errors are comparable. Surprisingly, the best forecasting accuracy was achieved by the models trained in the 1 day resolution dataset.

The results of table 5.4 are partially depicted in figure 5.1. This figure illustrates the forecast results of the Artificial Neural Network and Random Forest on the 1 hour resolution dataset for the household with ID **H41**, which is the lowest result achieved by ANN in this resolution. The blue lines represent the true load demand of the household. The red line represents the forecasted load demand by the ANN and RF. These graphs show the

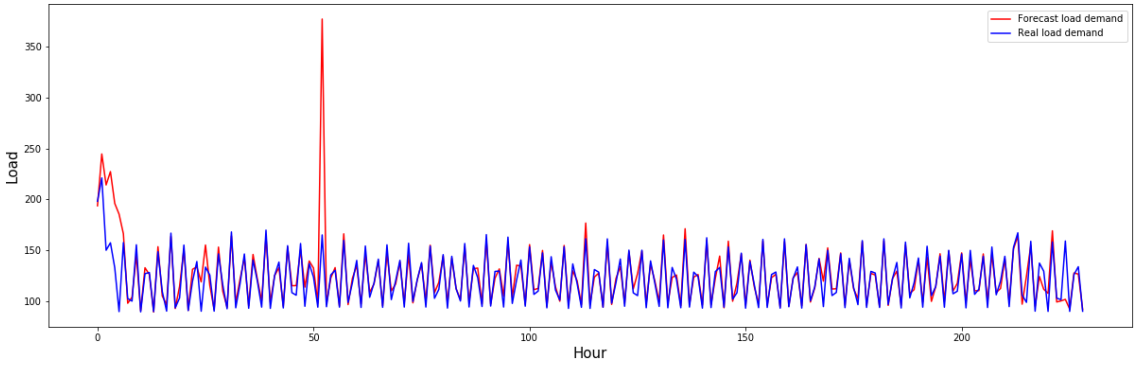
| House | ANN MAPE (%) | RF MAPE (%) |
|---|---|---|
| H4 | 6.2% | 9.6% |
| H28 | 7.1% | 10.2% |
| H51 | 7.3% | 18.5% |
| H9 | 7.7% | 11.1% |
| H21 | 8.8% | 11.9% |

**Table 5.6:** Houses with low error values in the 1 day resolution dataset

predicted load curve and real load curve for a few hundred hours that were not included in the train set. They are only part of the test set, which means that the models have never seen these values. The rest of the results of table 5.4 are presented in the appendix in figures A.1, A.2, A.3 and A.4.




(g) ANN: MAPE 10.5%

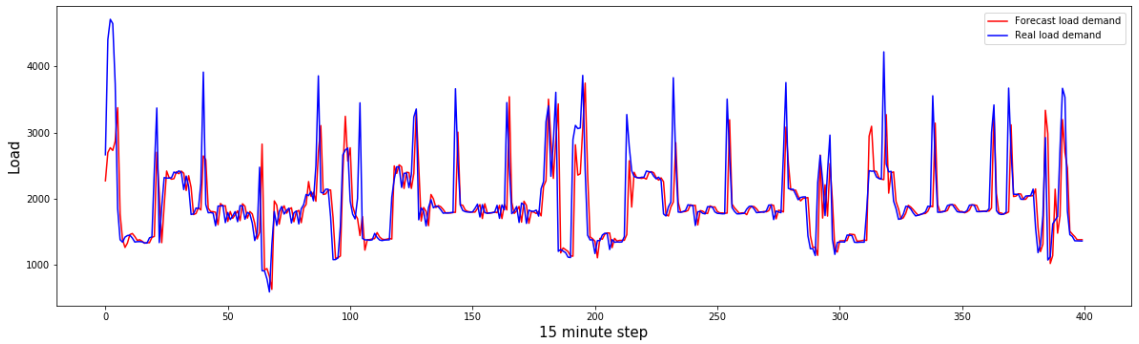



(h) RF: MAPE 11.1%

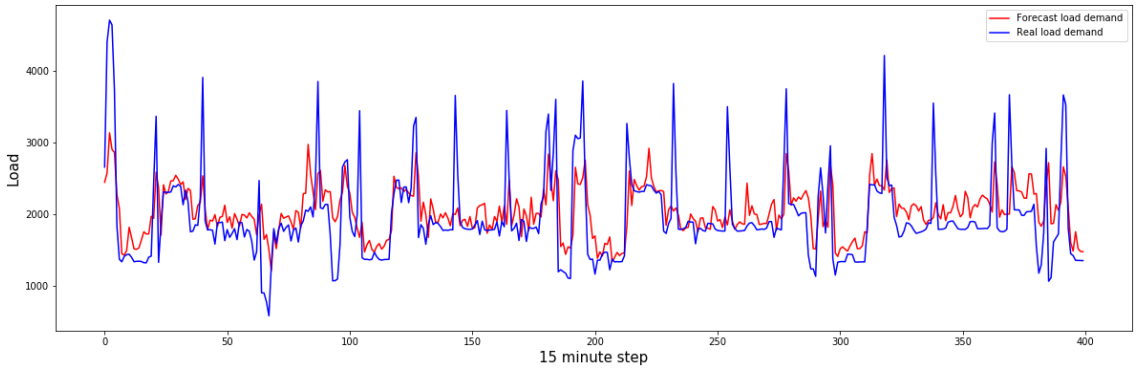
**Figure 5.1:** Forecasting results of models trained on the 1 hour resolution dataset of household **H41**

In the graphs, it can be seen that the forecasted values (red lines) are usually able to follow the pattern of the actual load to some extent. However, this appears to not be true in the case of house **H4**, which fails to detect a pattern.

Another characteristic of the graphs is that the red lines tend to not follow the blue line in peak values, i.e. when the load changes abruptly. In practice, this means that it is not able to forecast load demand in peak hours. This problem is very obvious in pictures A.2, A.3 and especially A.4.




(a) ANN: MAPE 9.2%



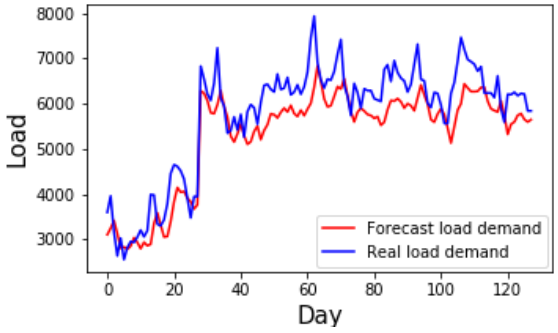

(b) RF: MAPE 17%

**Figure 5.2:** Forecasting results of models trained on the 15 minutes resolution dataset of household **H28**

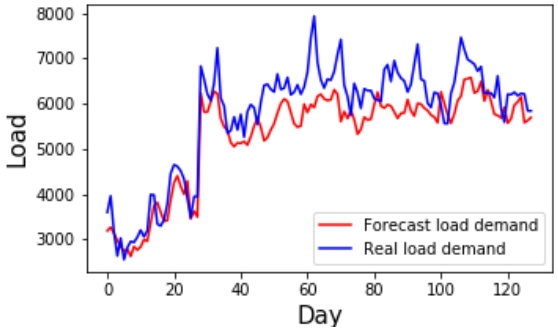
The partial results of table 5.5 are depicted in figure 5.2. The figure illustrates the forecasted load curve (red line) by ANN and RF for the 15 minutes resolution dataset of household **H28**. The true load curve (blue line) is depicted as well for comparison. The rest of the results of table 5.5 are presented in the appendix in figures A.5, A.6, A.7 and A.8.

In the 15 minutes ANN resolution graphs, it appears that the forecasted values are capable of following the pattern of the real load quite well. The lower MAPE values of this resolution compared to the 1 hour resolution MAPE values are also confirmed by the graphs. Intuitively, it is obvious that the neural network is capable of learning the load patterns more easily in the 15 minutes resolution dataset, compared to the 1 hour resolution dataset. However, this is not true for the RF model. The RF model's accuracy seems to be lower than that of ANN for the 15 minutes resolution, but also it is also generally lower than the accuracy of RF in the 1 hour resolution. Furthermore, it can be seen that also in this resolution, the neural network is not capable to predict peak load demands.

The partial results of table 5.6 are also graphically depicted in figure 5.3. These graphs visualize the forecasted load curve by ANN and RF trained on the 1 day resolution dataset of household **H4**. As in the previosuly mentioned graphs, the true load demand curve




(a) ANN: MAPE 6.2%




(b) RF: MAPE 9.6%

**Figure 5.3:** Forecasting results of models trained on the 1 day resolution dataset of household **H4**

is drawn for comparison against the forecasts. The rest of the results of table 5.6 are visualized in the appendix in figures A.9, A.10, A.11 and A.12.

Just like in the previous graphs, the forecasted values are capable, to some extent, to detect the load demand pattern. Also, these models fail to predict the peak values of the load demand.

# Chapter 6

# Discussion

In chapter 5 the results of the conducted experiments were presented. Some interpretation of the results was given, but generally it was decided to refrain from a comprehensive analysis. This chapter will be dedicated to an in depth discussion and interpretation of the results given previously.

As witnessed in Experimental Evaluation, the Feedforward Neural Network and Random Forest presented in the Solution Approach chapter, have achieved varying degrees of success in the next step load demand forecast task. The degree of success depends on three factors:

- The machine learning model used, i.e. Feedforward Neural Network or Random Forest
- The specific household for which the machine learning models were trained on.
- The resolution of the data that the models were trained on.

All the results reported in this thesis indicate that the Feedforward Neural Network is more capable of predicting the next step load demand forecast than the Random Forest is. In all the resolutions and all the households, ANN succeeded to yield a lower forecast error and sometimes with substantial differences. The depicted graphs also suggest that ANN predicted curves follow the real load curves more closely than the RF predicted curves. The reason is difficult to interpret, but these empirical pieces of evidence strongly suggest that ANN is a superior machine learning model to RF, for the given task of load demand forecast.

The issues caused by the second factor are, to some extent, out of the researcher's control. Some households simply have a very unpredictable and chaotic electricity consumption

behaviour. A very high forecast error is yielded when the machine learning models attempt to predict the next step load demand of such households, as witnessed in tables 5.1, 5.2 and 5.3. Refinement of the models' architectures, or even more complex machine learning models will probably do very little to improve the accuracy in these cases.

The resolution of the data has proven to be quite influential in the next step load demand forecast. It turns out that the consumers' behaviour patterns are manifested differently at different resolutions. This leads to various forecasting capabilities for models trained on data of different resolutions.

The next day forecast (1 day resolution data) is evidently easier to perform. The prediction error is considerably lower than that of the other resolutions. There are three hypothetical reasons that might explain this result:

- At such a large resolution the load demand volatility is smaller. Usually, there are no abrupt changes in load demand from one day to the other.
- People in a household exhibit a daily behaviour cycle, imposed by their lifestyle. This cycle tends to repeat every day, thus causing a daily electricity consumption pattern. The benefit of the 1 day resolution is that it doesn't matter when a specific action of the consumer happened throughout the day, but it matters that it happened. At a daily aggregate level, the electricity consumption is pretty much the same. For example, it doesn't matter whether the consumer cooked dinner one hour later than the previous day, it matters that he/she cooked dinner. The energy consumed to cook the dinner is roughly the same as the energy consumed to cook dinner the previous day. In this simple example, despite the exact time, the consumer sticks to his/her daily routine anyway. This *time invariant* pattern is not exhibited at the lower resolutions studied here.
- Temperature affects daily load demand more than it affects load demand in smaller resolutions. This assertion is sustained by the correlation coefficients depicted in tables 4.2, 4.3 and 4.4. The correlation between the load demand and the temperature is much higher (in absolute terms) in the 1 day resolution dataset than it is in the other resolutions. The higher correlation of this variable means that its effect is higher, thus easier for the ANN model to to capture its impact.

The next 15 minutes forecast by ANN is the second most accurate forecast. It's accuracy is somehow comparable to the accuracy of the next hour forecast by ANN and RF. However, as seen from the graphs of both resolution forecasts, the 15 minutes ANN resolution forecast follows the real load demand pattern more closely than the 1 hour resolution forecast can, as is evident by comparing graphs. A potential reason for the higher accuracy

is the fact that electricity consumption is more volatile in the 15 minutes resolution, thus easier to predict by ANN. The RF next 15 minutes forecast results, on the other hand, are much less accurate.

A common problem for all the models is the disability to forecast peak load demands, i.e. load demands that change significantly from the load demand of the previous step. This is obviously not a problem related to the resolution or at all, but a deeper one. There are two potential explanations to this issue:

- The presented machine learning models here are not ideally suited to detect such peak load demands. More complex architectures may be required to tackle this problem. These architectures must be able to learn more complicated patterns hidden in the data that the ANN and RF presented here cannot.
- A peak load demand, in some cases, may be totally random and, as a result, unpredictable. Despite the assumption that people exhibit electricity consumption patterns, that pattern is easy to break. For example, there is no way to predict the random urge of some person to turn on the clothes dryer in the middle of the night, and consequently causing a huge spike in the load demand curve at that moment.

In this thesis, the average load demands for the specific time steps are used as input and output variables. An alternative would be to use the total load demand of that time step, i.e. the sum of electrical load. This approach has not been explored here and it is unknown how it would affect the forecast accuracy. However, there should not be much difference, as in essence, the average load and total load in a specific time interval convey basically the same information. This assertion is based on the fact that the total load is derived by simply multiplying the average load by the time interval (see equation 4.4).

Lastly, one thing to be considered is the fact that the next day load forecast accuracy can be substantially improved with more data. About one year worth of load demand data have been used in this research. At least two years of data are required to study the yearly patterns of electricity consumption.

# Chapter 7

# Conclusions and future work

## 7.1 Conclusions

This thesis explores the problem of next step load forecasting in residential customers using Feed Forward Neural Networks, which are one of the most basic architectures of artificial neural networks, and Random Forests, which are on of the most common ensemble machine learning models. The architectures presented in the Solution Approach chapter are trained on three equivalent datasets but with different resolutions: 1 hour, 15 minutes and 1 day resolutions.

The datasets contain load demand data for 55 households for a time period of about 14 months. The data of a specific household in a specific dataset is called a subset. A separate neural network and random forest model is trained on every subset.

These machine learning models yielded various levels of forecast accuracy, with MAPE values ranging from as low as 6.2% up to a staggering 90.2%. The households with high margins of error are considered to have a very volatile load demand, thus almost impossible to attain decent forecast results for. The focus is shifted towards the subsets for which the ANN yields the lowest MAPE values.

The models trained on the 1 day resolution dataset yielded the highest accuracy, with the 5 highest test MAPE values equal to 6.2%, 7.1%, 7.3%, 7.7% and 8.8%, generated by ANN. The RF model achieved its highest accuracy on the 1 day resolution dataset too, but not as accurate as ANN. The potential reasons why the models trained on 1 day resolution data are more accurate, as explained in the Discussion chapter, are:

- Low volatility of load demand in this resolution.

- The existence of daily electricity consumption patterns in households.
- High correlation between temperature and next step load demand.

The models trained on the 15 minutes and 1 hour resolution datasets, yielded comparable results, with a couple of exceptions for the Random Forest in the 15 minutes resolution dataset.

The reported results strongly indicate that the Feedforward Neural Network is more effective than Random Forest for forecasting the next step load demand. The reported ANN error values were smaller than those of RF, for every household. This claim is also attested by the graphs, were it is seen that the forecasted load curve by ANN follows the real load demand curve more closely than the load curve forecasted by RF.

Generally, the forecasted load demand curve follows the pattern of the real load demand curve nicely, with some exceptions. However, in almost all the cases the models fail to predict peak load demands, i.e. load demands that change significantly from the load demand of the previous step. There are two potential reasons for that:

- The Feedforward Neural Network is not well suited to detect peak load demands.
- In some cases, the peak load demands are totally random, thus impossible to predict.

As a final remark, it is concluded that the implemented Feedforward Neural Network and Random Forest achieve some degree of success in forecasting the next step load demand. However, the attained level of accuracy is not enough for practical purposes. More complex machine learning models are required to overcome the problems of high load demand volatility and the prediction of peak load demands.

## 7.2 Future work

There is a wide range of pathways to be followed in the load forecast research field. However, there are a few special topics of interest that can be pursued, in order to complement the research done here and improve the results.

First, a more in depth analysis is required for the forecast models that yielded large MAPE values, in order to better understand the reason for the low forecast accuracy. A thorough understanding of the reason (or maybe reasons), may lead to the devise of more advanced techniques that are able to perform highly accurate load demand forecasts for the specific households.

Secondly, more studies should be conducted to improve the prediction capabilities of the peak load demand values. More complex architectures may prove to be fruitful. In addition, the Feedforward Neural Network and Random Forest presented here can be used in conjunction with other separate machine learning models that are capable of predicting solely the peak load demand values. The results can be unified, thus providing a better overall forecast accuracy.

Finally, the implementation of a special type of artificial neural network called Recurrent Neural Network (RNN) may be of interest. RNNs have memory, i.e. they have the ability to store previous input values in their internal state. In theory, this is beneficial for load forecasting, as it was shown that load demand values from several previous time steps are highly correlated to the next step load demand.

# List of Figures

# List of Tables

# Appendix A

# Appendix

## A.1 Independent variables

In this section, all the available independent variables in all three generated datasets with different resolutions will be listed. The shortened names given here have been used in tables throughout the thesis.

| Variable | Full name | Description |
|---|---|---|
| d | Day of week | Ordinal number from 0 to 6 representing the day of week. |
| w | Weekend | Boolean value representing whether the day is a weekend or not. |
| h | Hour of day | Ordinal number from 0 to 23 representing the hour of day. |
| T | Temperature | Temperature in degrees Celcius. |
| D | Dew point | Dew point in degrees Celcius. |
| C | Cloud cover | Percentage of sky covered in clouds. |
| H | Humidity | Relative humidity. Value ranging from 0 to 1. |
| p(h-1) | 1 hour ago load | Average load demand of previous hour. |
| p(h-2) | 2 hours ago load | Average load demand of 2 previous hours. |
| p(h-3) | 3 hours ago load | Average load demand of 3 previous hours. |
| p(d-1, h) | 1 day ago load | Average load demand of the same ordinal hour in the previous day. |
| p(w-1, d, h) | 1 week ago load | Average load demand of the same ordinal hour in the previous week. |
| p(d-1) | Yesterday daily load | Average load demand of the whole previous day. |
| p(w-1, d) | Last week daily load | Average load demand of the whole day in the previous week. |

**Table A.1:** One hour resolution dataset variables

| Variable | Full name | Description |
|---|---|---|
| d | Day of week | Ordinal number from 0 to 6 representing the day of week. |
| w | Weekend | Boolean value representing whether the day is a weekend or not. |
| s | Step of day | Ordinal number from 0 to 95 representing the 15 minute step of the day. |
| T | Temperature | Temperature in degrees Celcius. |
| D | Dew point | Dew point in degrees Celcius. |
| C | Cloud cover | Percentage of sky covered in clouds. |
| H | Humidity | Relative humidity. Value ranging from 0 to 1. |
| p(s-1) | 1 step ago load | Average load demand of previous step. |
| p(s-2) | 2 steps ago load | Average load demand of 2 previous steps. |
| p(s-3) | 3 steps ago load | Average load demand of 3 previous steps. |
| p(d-1, s) | 1 day ago load | Average load demand of the same ordinal step in the previous day. |
| p(w-1, d, s) | 1 week ago load | Average load demand of the same ordinal step in the previous week. |
| p(d-1) | Yesterday daily load | Average load demand of the whole previous day. |
| p(w-1, d) | Last week daily load | Average load demand of the whole day in the previous week. |

**Table A.2:** 15 minutes resolution dataset variables

| Variable | Full name | Description |
|---|---|---|
| d | Day of week | Ordinal number from 0 to 6 representing the day of week. |
| w | Weekend | Boolean value representing whether the day is a weekend or not. |
| T | Temperature | Average daily temperature in degrees Celcius. |
| D | Dew point | Average daily dew point in degrees Celcius. |
| C | Cloud cover | Average daily percentage of sky covered in clouds. |
| H | Humidity | Average daily relative humidity. Value ranging from 0 to 1. |
| min(p(d-1)) | Minimum load previous day | Minimum hourly load of the previous day. |
| max(p(d-1)) | Maximum load previous day | Maximum hourly load of the previous day. |
| stdev(p(d-1)) | St dev of load previous day | Standard deviation of hourly loads of the previous day. |
| p(d-1) | Yesterday daily load | Average daily load demand of the previous day. |
| p(d-2) | 2 days ago load | Average daily load demand of 2 days ago. |
| p(d-3) | 3 days ago load | Average daily load demand of 3 days ago. |
| p(w-1, d) | Last week daily load | Average daily load demand one week ago. |

**Table A.3:** 1 day resolution dataset variables

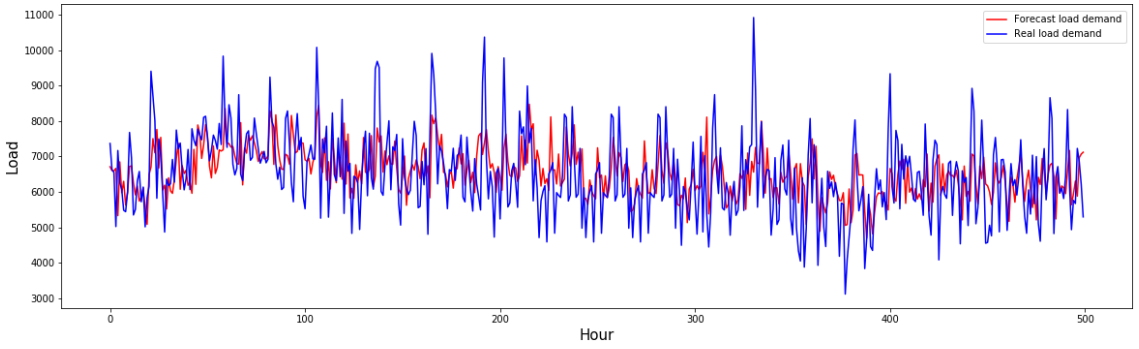
## A.2 Load forecast results

In this section, the remaining results of tables 5.4, 5.5 and 5.6 will be illustrated with graphs. Only the results that were not illustrated in Chapter 5, will be presented here.

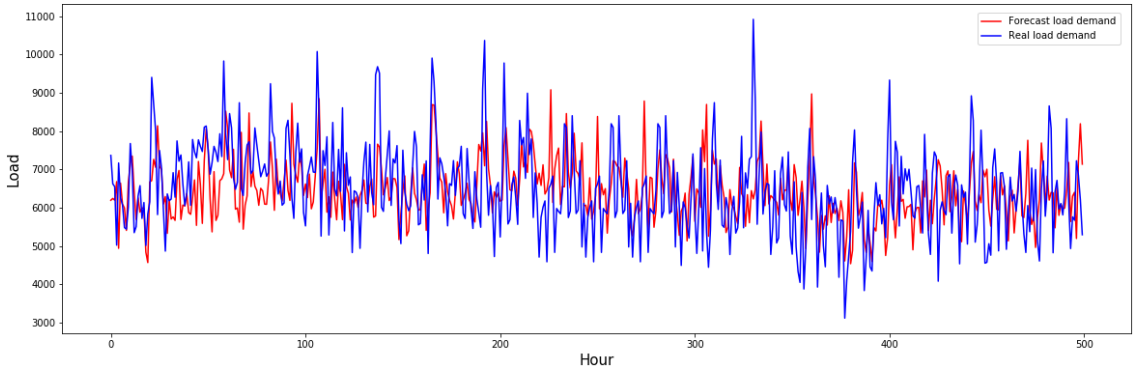
### A.2.1 1 hour resolution results

### A.2.2 15 minutes resolution results

### A.2.3 1 day resolution results

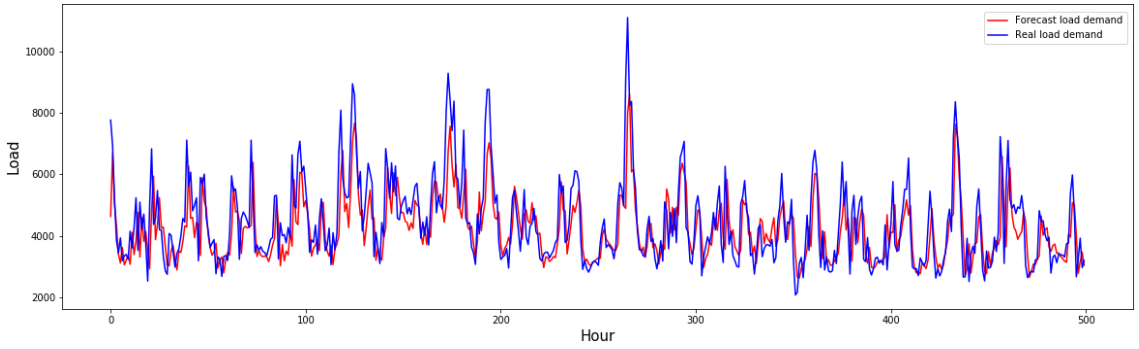



(a) ANN: MAPE 12.4%

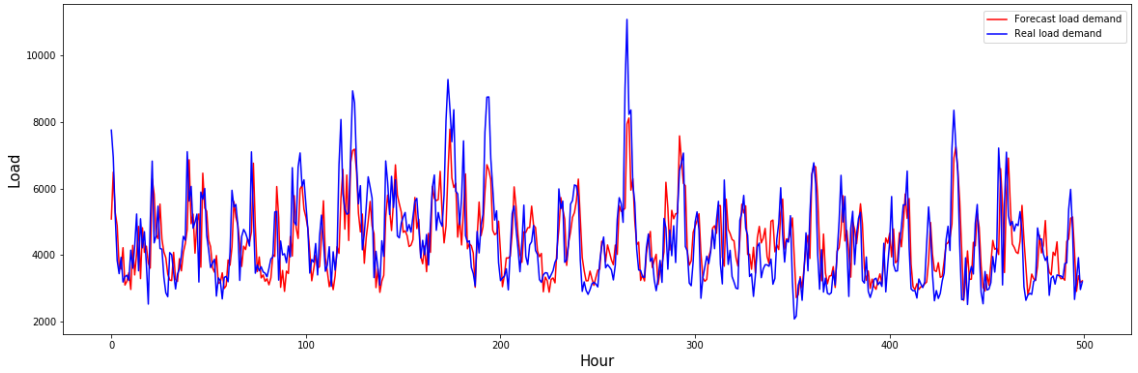



(b) RF: MAPE 13.7%

**Figure A.1:** Forecasting results of models trained on the 1 hour resolution dataset of household **H4**

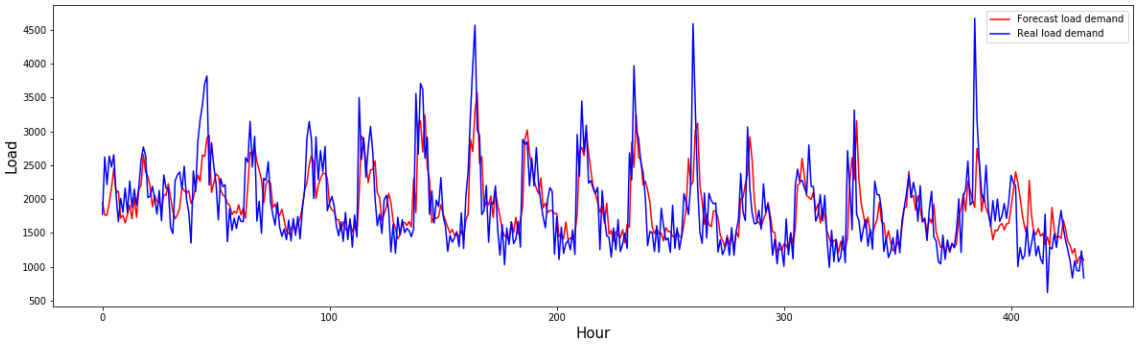



(a) ANN: MAPE 13.7%

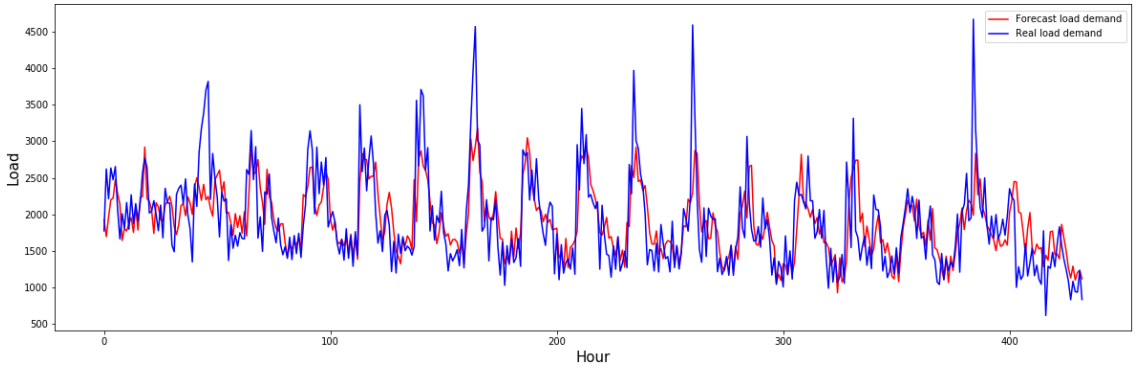



(b) RF: MAPE 15.6%

**Figure A.2:** Forecasting results of models trained on the 1 hour resolution dataset of household **H49**

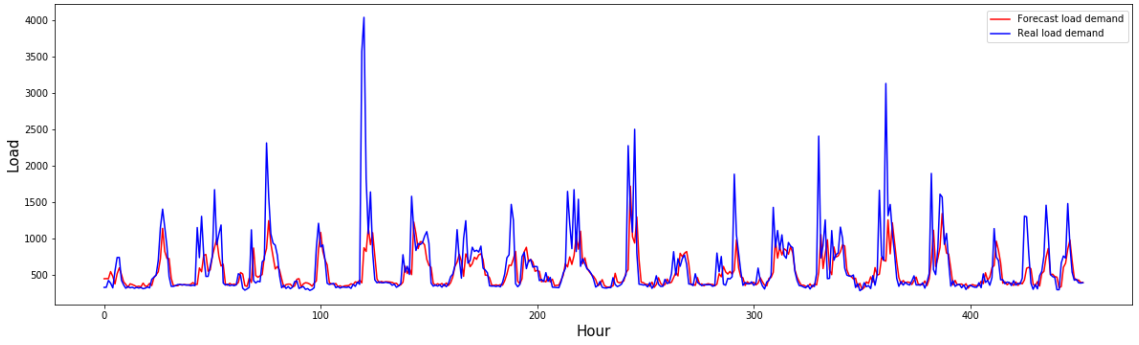



(a) ANN: MAPE 14.5%

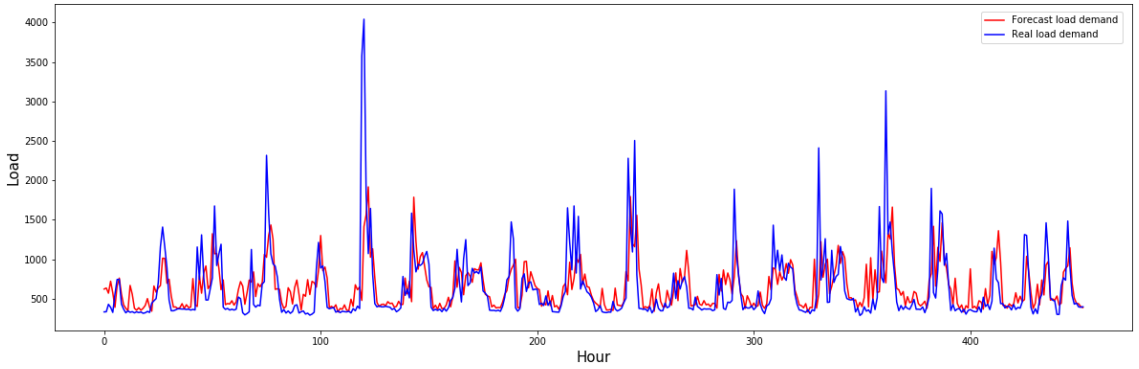



(b) RF: MAPE 16.1%

**Figure A.3:** Forecasting results of models trained on the 1 hour resolution dataset of household **H9**

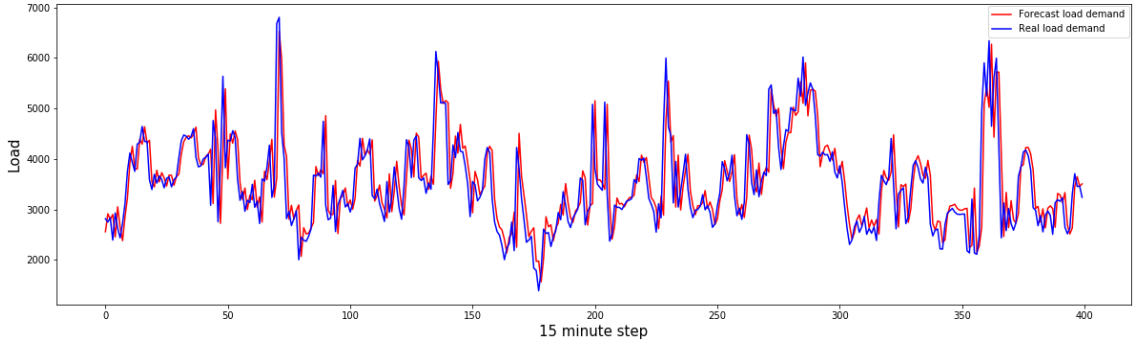



(a) ANN: MAPE 17.4%

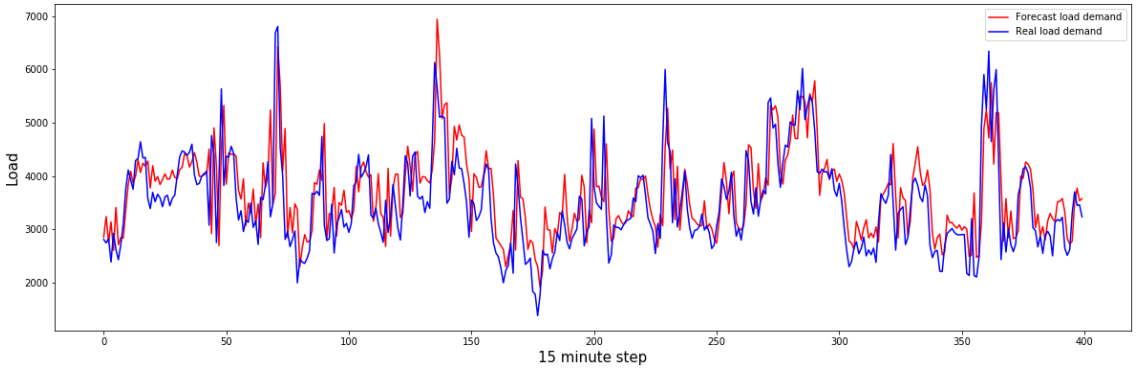



(b) RF: MAPE 17.9%

**Figure A.4:** Forecasting results of models trained on the 1 hour resolution dataset of household **H2**

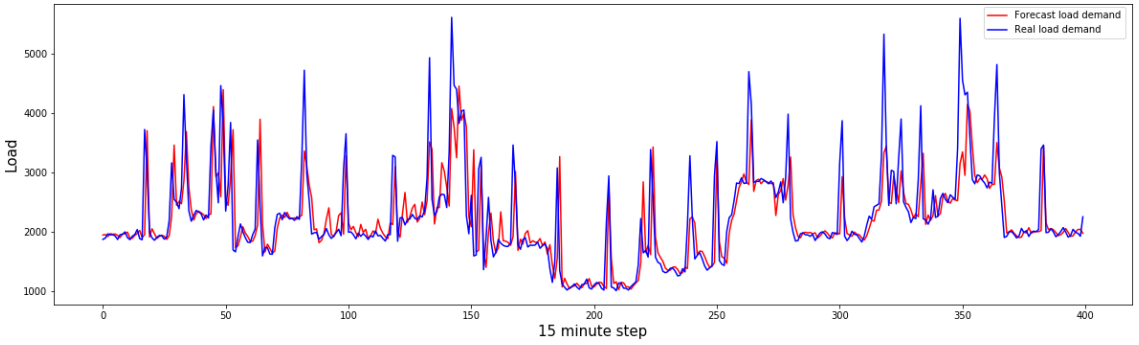



(a) ANN: MAPE 10.5%

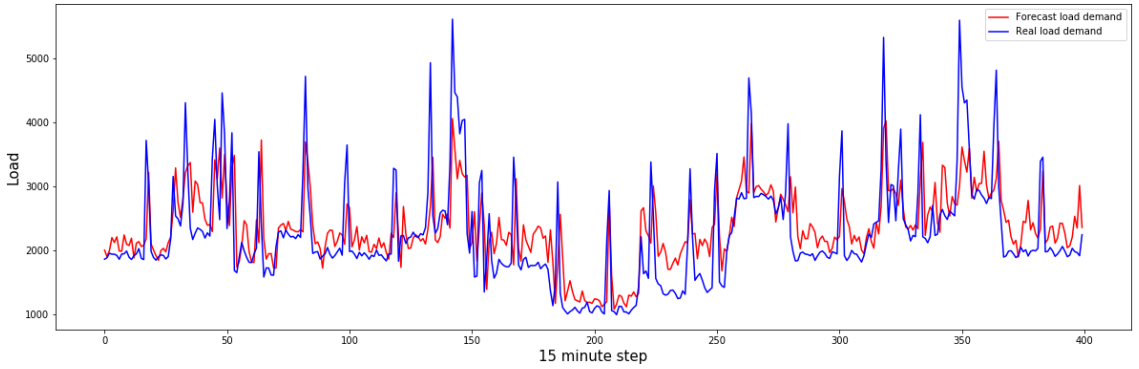



(b) RF: MAPE 12.9%

**Figure A.5:** Forecasting results of models trained on the 15 minutes resolution dataset of household **H37**

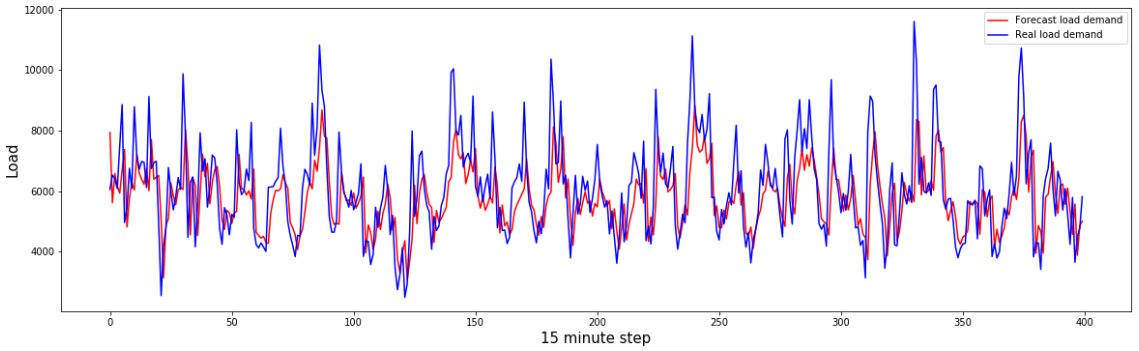



(a) ANN: MAPE 10.8%

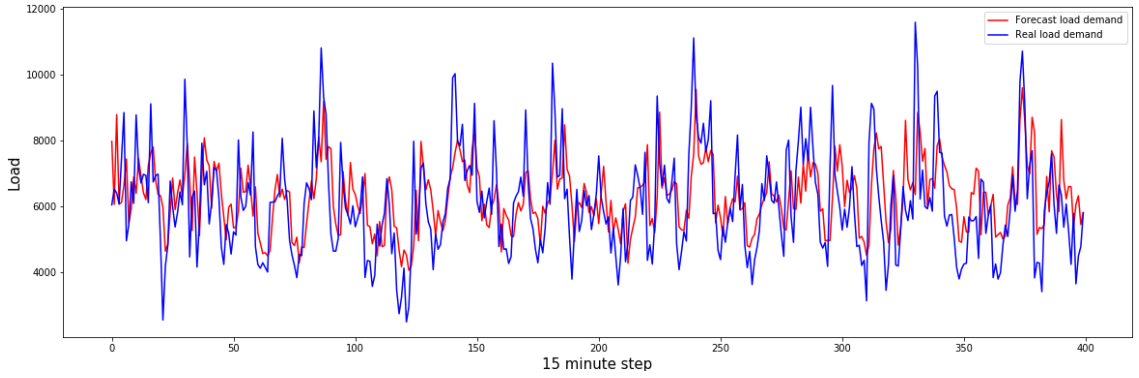



(b) RF: MAPE 40.4%

**Figure A.6:** Forecasting results of models trained on the 15 minutes resolution dataset of household **H13**

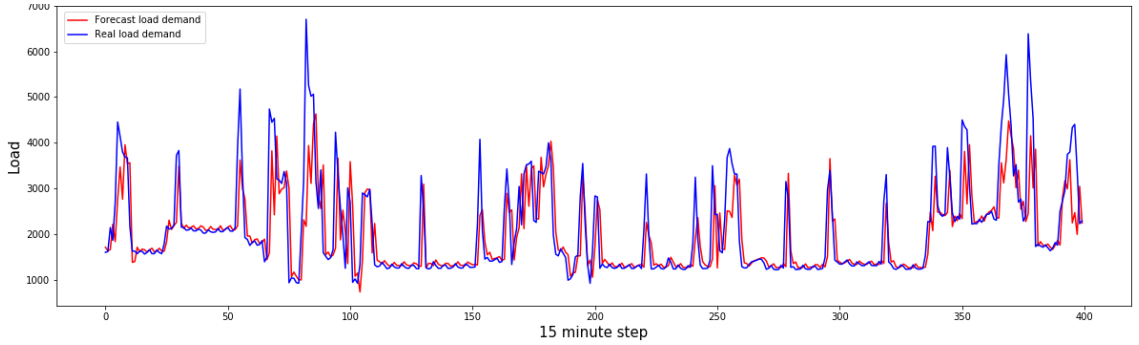



(a) ANN: MAPE 11.8%

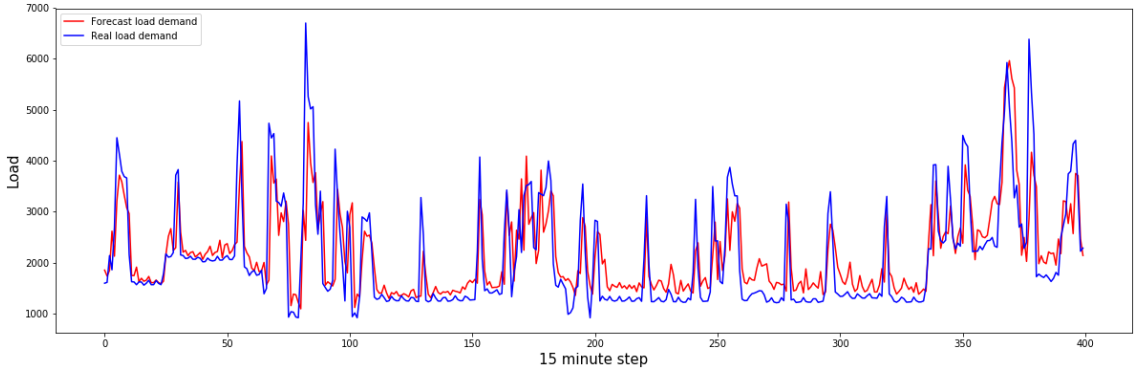



(b) RF: MAPE 15.1%

**Figure A.7:** Forecasting results of models trained on the 15 minutes resolution dataset of household **H4**

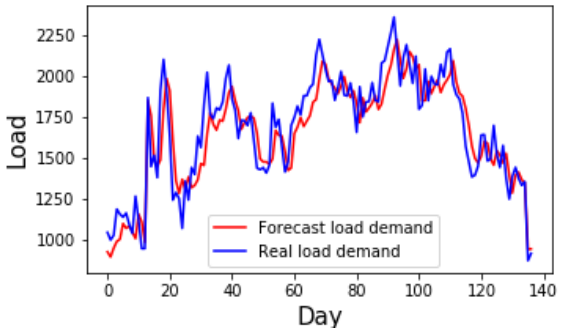



(a) ANN: MAPE 13%

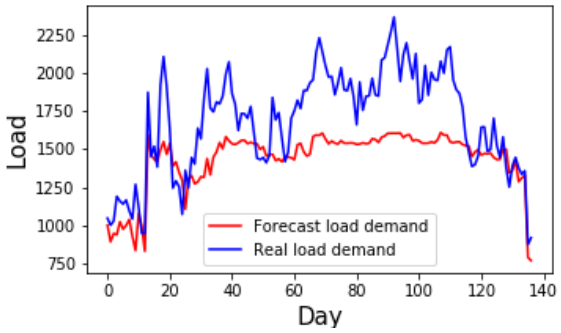



(b) RF: MAPE 18.5%

**Figure A.8:** Forecasting results of models trained on the 15 minutes resolution dataset of household **H29**

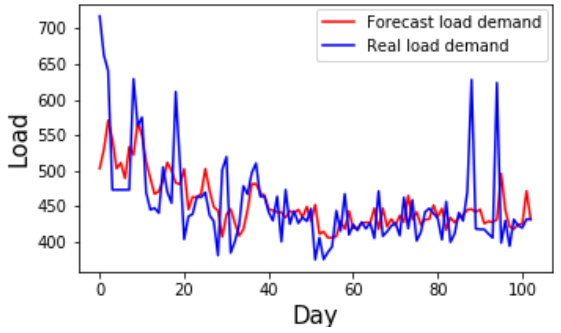



(a) ANN: MAPE 7.1%

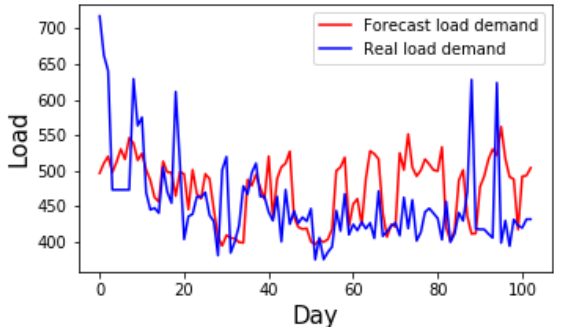



(b) RF: MAPE 10.2%

**Figure A.9:** Forecasting results of models trained on the 1 day resolution dataset of household **H28**

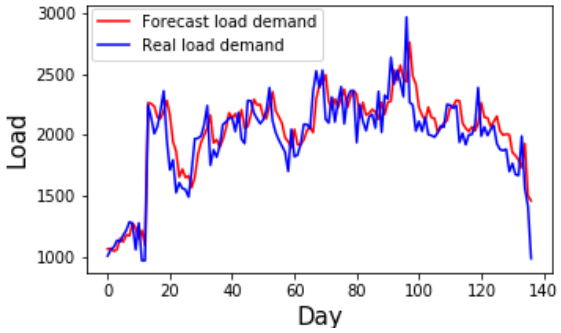



(a) ANN: MAPE 7.3%

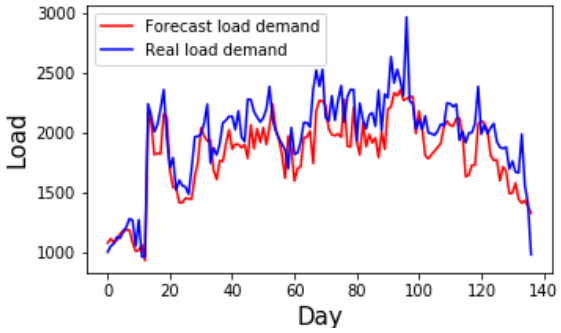



(b) RF: MAPE 18.5%

**Figure A.10:** Forecasting results of models trained on the 1 day resolution dataset of household **H51**

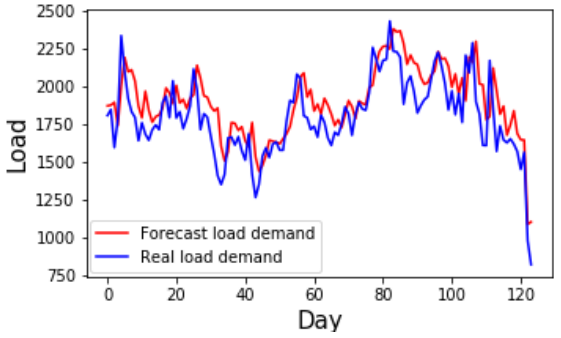



(a) ANN: MAPE 7.7%

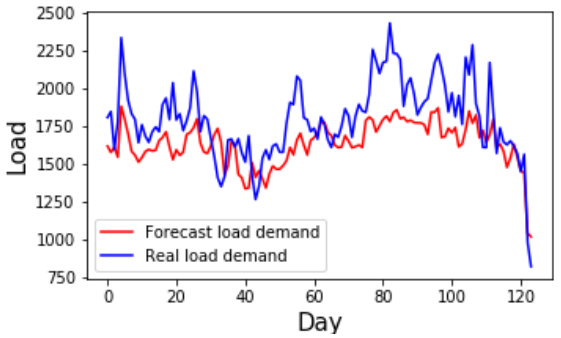



(b) RF: MAPE 11.1%

**Figure A.11:** Forecasting results of models trained on the 1 day resolution dataset of household **H9**




(a) ANN: MAPE 8.8%




(b) RF: MAPE 11.9%

**Figure A.12:** Forecasting results of models trained on the 1 day resolution dataset of household **H21**

# Bibliography


[1] Hesham K Alfares and Mohammad Nazeeruddin. Electric load forecasting: literature survey and classification of methods. *International journal of systems science*, 33(1): 23–34, 2002.

[2] Australian Energy Market Operator. Initial operation of the hornsdale power reserve battery energy storage system. 2018.

[3] Anil Arya and AL Sharma. Polymer nanocomposites: synthesis and characterization. *arXiv preprint arXiv:1807.03540*, 2018.

[4] Tesla completes its giant australian powerpack battery on time. https://www.engadget.com/2017/11/23/tesla-australia-powerpack-100-day-bet/?guccounter=1&guce_referrer=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnLw&guce_referrer_sig=AQAAAM4ECegXACnFFu-89V3maVuHEdMexoRHpMjFgRlmfSNcshJT7Cjpk2AZwXDi6sct10mPk5vGF33tQebxYgIXqKZ4Q96-QPxUWx_UAfLj3eqq6NOUBijILQUQXFOz57WoxVa-bp85glTlWI_ZTXpX. Accessed: 2019-05-03.

[5] Steven W Blume. *Electric power system basics for the nonelectrical professional*. John Wiley & Sons, 2016.

[6] Ben Kröse, B Krose, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks. *J Comput Sci*, 48, 01 1993.

[7] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[8] Paul Werbos and Paul J. (Paul John. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 01 1974.

[9] Robert C Sass. Neural networks: Capabilities and applications. Technical report, 1990.

[10] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[11] Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35 – 62, 1998. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(97)00044-7. URL http://www.sciencedirect.com/science/article/pii/S0169207097000447.

[12] Gang Li, Hussein Alnuweiri, Yuejian Wu, and H Li. Acceleration of back propagation through initial weight pre-training with delta rule. In *IEEE International Conference on Neural Networks*, pages 580–585. IEEE, 1993.

[13] Gian Paolo Drago and Sandro Ridella. Statistically controlled activation weight initialization (scawi). *IEEE Transactions on Neural Networks*, 3(4):627–631, 1992.

[14] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[15] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[16] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[17] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[18] Pang-Ning Tan. *Introduction to data mining*. Pearson Education India, 2018.

[19] John Sukup. Using statistical modeling to predict product purchase in microsoft azure, 2018. URL https://expectedx.com/blog/using-statistical-modeling-to-predict-product-purchase-in-microsoft-azure/2018/6/26.

[20] Steven W Norton. Generating better decision trees. In *IJCAI*, volume 89, pages 800–805, 1989.

[21] Ned Horning. Introduction to decision trees and random forests. *Am. Mus. Nat. Hist*, 2:1–27, 2013.

[22] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5): 272, 2012.

[23] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[24] Muhammad Qamar Raza and Abbas Khosravi. A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, 50:1352–1372, 2015.

[25] Henrique Steinherz Hippert, Carlos Eduardo Pedreira, and Reinaldo Castro Souza. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on power systems*, 16(1):44–55, 2001.

[26] G Lambert-Torres, LE Borges Da Silva, B Valiquette, H Greiss, and D Mukhedkar. A fuzzy knowledge-based system for bus load forecasting. In *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*, pages 1211–1218. IEEE, 1992.

[27] Zainab H Osman, Mohamed L Awad, and Tawfik K Mahmoud. Neural network based approach for short-term load forecasting. In *2009 IEEE/PES Power Systems Conference and Exposition*, pages 1–8. IEEE, 2009.

[28] C-N Lu, H-T Wu, and S Vemuri. Neural network based short term load forecasting. *IEEE Transactions on Power Systems*, 8(1):336–342, 1993.

[29] Siddharth Arora and James W Taylor. Rule-based autoregressive moving average models for forecasting load on special days: A case study for france. *European Journal of Operational Research*, 266(1):259–268, 2018.

[30] Angel Pardo, Vicente Meneu, and Enric Valor. Temperature and seasonality influences on spanish electricity load. *Energy Economics*, 24(1):55–70, 2002.

[31] John Peirson and Andrew Henley. Electricity load and temperature: Issues in dynamic specification. *Energy Economics*, 16(4):235–243, 1994.

[32] Ching-Lai Hor, Simon J Watson, and Shanti Majithia. Analyzing the impact of weather variables on monthly electricity demand. *IEEE transactions on power systems*, 20(4): 2078–2085, 2005.

[33] C Donald Ahrens. *Essentials of meteorology: An invitation to the atmosphere*. Cengage Learning, 3rd edition, 2001.

[34] Ching-Lai Hor, Simon J Watson, and Shanti Majithia. Analyzing the impact of weather variables on monthly electricity demand. *IEEE transactions on power systems*, 20(4): 2078–2085, 2005.

[35] Dong C Park, MA El-Sharkawi, RJ Marks, LE Atlas, and MJ Damborg. Electric load forecasting using an artificial neural network. *IEEE transactions on Power Systems*, 6(2):442–449, 1991.

[36] Jarkko Isotalo. Basics of statistics. *Finland: University of tampere*, 2001.

[37] Eric Miller. Renewables and the smart grid. *Renewable Energy Focus*, 10(2):67–69, 2009.

[38] Nearly half of all u.s. electricity customers have smart meters. https://www.eia.gov/todayinenergy/detail.php?id=34012. Accessed: 2019-03-27.

[39] M Godoy Simões, Robin Roche, Elias Kyriakides, Abdellatif Miraoui, Benjamin Blunier, Kerry McBee, Siddharth Suryanarayanan, Phuong Nguyen, and Paulo Ribeiro. Smart-grid technologies and progress in europe and the usa. In *2011 IEEE Energy Conversion Congress and Exposition*, pages 383–390. IEEE, 2011.

[40] VaasaETT. Assessing the potential of home automation in norway. 2017. ISBN 978-82-410-1587-8. URL http://publikasjoner.nve.no/rapport/2017/rapport2017_34.pdf.

[41] Yi Wang, Qixin Chen, Tao Hong, and Chongqing Kang. Review of smart meter data analytics: Applications, methodologies, and challenges. *IEEE Transactions on Smart Grid*, 2018.

[42] Barnaby Pitt. *Applications of data mining techniques to electric load profiling*. PhD thesis, University of Manchester, 2000.

[43] Fintan McLoughlin, Aidan Duffy, and Michael Conlon. A clustering approach to domestic electricity load profile characterisation using smart metering data. *Applied energy*, 141:190–199, 2015.

[44] Charalampos Chelmis, Jahanvi Kolte, and Viktor K Prasanna. Big data analytics for demand response: Clustering over space and time. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2223–2232. IEEE, 2015.

[45] Gonzalo Mateos and Georgios B Giannakis. Load curve data cleansing and imputation via sparsity and low rank. *IEEE Transactions on Smart Grid*, 4(4):2347–2355, 2013.

[46] Sanujit Sahoo, Daniel Nikovski, Toru Muso, and Kaoru Tsuru. Electricity theft detection using smart meter data. In *2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2015.

[47] Katarina Kostková, L' Omelina, P Kyčina, and Peter Jamrich. An introduction to load management. *Electric Power Systems Research*, 95:184–191, 2013.

[48] Shiyin Zhong and Kwa-Sur Tam. Hierarchical classification of load profiles based on their characteristic attributes in frequency domain. *IEEE Transactions on Power Systems*, 30(5):2434–2441, 2015.

[49] Nima Amjady. Short-term hourly load forecasting using time-series modeling with peak load estimation capability. *IEEE Transactions on Power Systems*, 16(3):498–505, 2001.

[50] J-N Sheen, C-S Chen, and J-K Yang. Time-of-use pricing for load management programs in taiwan power company. *IEEE Transactions on Power Systems*, 9(1): 388–396, 1994.

[51] Raquel Gareta, Luis M Romeo, and Antonia Gil. Forecasting of electricity prices with neural networks. *Energy conversion and management*, 47(13-14):1770–1778, 2006.

[52] O Abedinia, N Amjady, M Shafie-Khah, and JPS Catalão. Electricity price forecast using combinatorial neural network trained by a new stochastic search method. *Energy Conversion and Management*, 105:642–654, 2015.

[53] AK Srivastava, Ajay Shekhar Pandey, and Devender Singh. Short-term load forecasting methods: A review. In *2016 International Conference on Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES)*, pages 130–138. IEEE, 2016.

[54] KY Lee, YT Cha, and JH Park. Short-term load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 7(1):124–132, 1992.

[55] Seunghyoung Ryu, Jaekoo Noh, and Hongseok Kim. Deep neural network based demand side short term load forecasting. *Energies*, 10(1):3, 2016.

[56] Jian Zheng, Cencen Xu, Ziang Zhang, and Xiaohua Li. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2017.

[57] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[58] Martin T Hagan and Suzanne M Behr. The time series approach to short term load forecasting. *IEEE Transactions on Power Systems*, 2(3):785–791, 1987.

[59] Dima Alberg and Mark Last. Short-term load forecasting in smart meters with sliding window-based arima algorithms. *Vietnam Journal of Computer Science*, 5 (3-4):241–249, 2018.

[60] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.

[61] KRM Supapo, RVM Santiago, and MC Pacis. Electric load demand forecasting for aborlan-narra-quezon distribution grid in palawan using multiple linear regression. In

*2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pages 1–6. IEEE, 2017.

[62] Joaquim L Viegas, Susana M Vieira, Rui Melício, Victor MF Mendes, and João MC Sousa. Ga-ann short-term electricity load forecasting. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pages 485–493. Springer, 2016.

[63] Adel Abdoos, Mohammad Hemmati, and Ali Akbar Abdoos. Short term load forecasting using a hybrid intelligent method. *Knowledge-Based Systems*, 76:139–147, 2015.

[64] Dark Sky API. https://darksky.net/dev. Accessed: 2 February 2019.

[65] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[66] J David Irwin and Robert M Nelms. *Basic engineering circuit analysis*, volume 900. John Wiley & Sons, 2010.

[67] Kendall E Atkinson. *An introduction to numerical analysis,* $2^{nd}$ *edition*. John Wiley & Sons, 1989.

[68] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[69] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 90–98. SIAM, 2017.

[70] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In *Advances in Neural Information Processing Systems*, pages 582–591, 2018.

[71] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on mnist classification task. *arXiv preprint arXiv:1804.02763*, 2018.

[72] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019. URL http://arxiv.org/abs/1902.09574.

[73] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[74] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[75] Daniel L Marino, Kasun Amarasinghe, and Milos Manic. Building energy load forecasting using deep neural networks. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 7046–7051. IEEE, 2016.

[76] Christopher Z Mooney, Robert D Duval, and Robert Duvall. *Bootstrapping: A nonparametric approach to statistical inference*. Number 95. Sage, 1993.