technische universität dortmund

Master Thesis

Analyzing Weather Effects in Short-Term Load Forecasting Using Feed-Forward Neural Networks

Öcal Kaptan 230914

October 2, 2024

Reviewer:
Prof. Dr. Florian Ziel
Prof. Dr. Christoph Hanck

fakultät statistik

Technische Universität Dortmund
Fakultät Statistik
Lehrstuhl Ökonometrie
https://econ.statistik.tu-dortmund.de

Abstract

Forecasting electricity load is essential in capacity planning, power system scheduling, and maintenance. It also plays a crucial role in informing end-consumers about their consumption behavior and bills on time. Short-term load forecasting (STLF) is critical to unit commitment, power distribution, and load dispatch. STLF more carefully considers daily and weekly patterns, along with weather influence. This thesis aims to concentrate on the impact of weather on short-term load forecasting, using feed forward neural networks (FFNN), in France's electricity market.

In order to fulfill this purpose, data has been collected from ENTSOE. Together with weather data from DWD, the load data has been used to create a neural network model - a feed forward neural networks (FFNN). FFNN is a type of deep neural network model describing the load day ahead based on a number of explanatory variables. These variables are mainly temperature and calendar variables, such as the hour of the day, day of the week, and month of the year. The models also take into account the load of the days before, the rolling sum, the rolling standard deviation, the rolling average, and the temperature of hours before by including the lagged values as explanatory parameters.

The models show a significant relation between the load and temperature-based variables. Furthermore, the lagged values and rolling load variables also seem to be necessary. The weather independence model was also used, showing lower accuracy than the weather-dependent model. This thesis will discuss the problems with load modeling and prediction and how it can be improved.

Keywords : Short Term Load Forecasting (STLF), Load Modelling, Load Forecasting, Feed Forward Neural Networks (FFNN), Weather Effect, Temperature Effect.

Contents

List of Figures

List of Tables

List of Abbreviations

STLF - Short-Term Load Forecasting

FFNN - Feed-Forward Neural Networks

ARIMA - Autoregressive Integrated Moving Average

HDD - Heating Degree Days

CDD - Cooling Degree Days

ANN - Artificial Neural Network

RNN - Recurrent Neural Networks

CNN - Convolutional Neural Networks

MAPE - Mean Absolute Percentage Error

SHAP - Shapley Values

ELU - Exponential Linear Unit

ReLU - Rectified Linear Unit

BN - Batch Normalization

LN - Layer Normalization

SVM - Support Vector Machine

MAE - Mean Absolute Error

MSE - Mean Squared Error

Adam - Adaptive Moment Estimation

RMSpropRoot - Mean Square Propagation

Chapter 1

Introduction

This chapter will introduce the background of the thesis and description of the problem.

1.1 Background

Capacity planning, power system scheduling, and maintenance require forecasting electricity load because their accuracy resolves disputes over the amount of consumption experienced by end consumers. Electricity load forecasting is divided into short, medium, and long term forecasting. Short-term load forecasting (STLF) is from an hour up to several days ahead and several weeks up to several months in the case of medium-term load forecasting (MTLF). Long-term load forecasting (LTLF) is concerned with predicting load from the period of one year to several years. STLF is crucial in unit commitment, power distribution, and load dispatch.

In recent years, there has been an increase in the application of machine learning techniques in STLF. Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are preferable due to their ability to capture the nonlinear relationship between load and input variables. Classical forecasting methods such as ARIMA models and other regression methods are useful in this content. However, they cannot capture the complex relationship between electricity consumption patterns as electricity load is influenced by many factors. Economic conditions, consumer behavior, and weather conditions are some of the factors that influence the electricity load.

Load forecasting models take into consideration the characteristics of daily, weekly, and yearly patterns, as well as holidays. Most of the modes use temperature information as one of the data inputs because most of the time, there is a clear relationship between temperature and load [1].

Many researchers have explored ways to incorporate weather data into STLF models to improve forecast accuracy. They used humidity, dew point, temperature, and wind speed. Temperature is the most common weather variable, as it has a direct impact on the usage of heating and cooling systems. Temperature is a crucial driver of electricity demand in countries with hot climates (e.g., Southern Europe), particularly during summer [2].

In this thesis, the impact of weather variables is investigated on Short-Term Load Forecasting using Feed-Forward Neural Networks. In order to prove the effect of weather conditions, the electricity load (day-ahead) data of France and the most representative weather variable, temperature, are employed.

1.2 Research Objectives

The aim of this thesis is to find the impact of weather information, namely temperature, in short-term load forecasting using FFNN in the context of the electric power sector of France.
Specific Objectives:

- Establish how the temperature changes; the demand for electricity also changes in France

- Create an FFNN that will be used to perform short-term load forecasting while considering temperature as one of the essential inputs.
- Assess the effectiveness of the FFNN model, including the temperature data as compared to its corresponding temperature-independent version.
- Evaluate the suitability and utility of the FFNN model in the application of short-term demand forecasting

1.3 Scope and Limitations

Scope of the Study:

- Geographical Scope: This thesis is carried out in the context of the French electricity market and employs load data specific to day-ahead for France.
- Temporal Scope: The focus of the study also aims to have a accurate short-term load forecasting with a time span of predicting electricity load 8 days (192 hours) ahead.
- Technical Scope: The work concerns with the creation of a Feed-Forward Neural Network based model which uses temperature data to assess electricity load. It tests how this model performs relative to other model, which do not take the temperature into account.

Limitations:

- Data Limitations: The study relies on the accuracy of available historical load day ahead and temperature data as the source of information. Any inaccuracies or omissions in the data would effect the models.
- Weather Variables: Except for temperature, which is the focus of all consideration as the primary weather factor, other elements such as humidity and wind speed are left out; therefore, these may also affect electricity load.
- Model Constraints: The FFNN may not be able to capture some sudden changes such as policy changes, pandemics, etc.

1.4 Significance of the Study

This thesis is an academic contribution to the field by showing how temperature information is implemented in a Feed-Forward Neural Network and improved short-term load forecasting. Thus, it contributes to the scope of applications of machine learning methods to energy forecasting and establishes a pathway for the integration of weather factors into forecasting models.

1.5 Outline

The outline for this thesis is:

- Chapter 2: This chapter explains the type load forecasting, load profiles, factors that influence the load, short term load forecasting techniques, and related studies.
- Chapter3: This chapter provides a description of the models that has been used.
- Chapter4: This chapter provides a description of the data that have been used and justification of the model design and the used model parameters in this thesis.
- Chapter5: The results from fitting the models, predictions is discussed. The chapter also review some problems with accurate load prediction and how the models and predictions can be improved in the future.
- Chapter 6: This chapter provides a short summary of the thesis together with some conclusions of the results.

Chapter 2

Literature Review

2.1 Overview of Electricity Load Forecasting

Electricity load forecasting is one of the most important tasks in the planning and operating of power systems. It involves estimating the electricity demand, enabling utility companies and grid operators to make informed decisions regarding capacity planning, system scheduling, maintenance activities, and financial management. Accurate load forecasts ensure efficient energy production, grid stability, and economic optimization of resources [3].

2.1.1 Types of Load Forecasting

Load forecasting can be divided into three categories: short-term forecasts, medium-term forecasts and long term forecasts.

- **Short-Term Load Forecasting (STLF)**: Forecasts are usually from one hour to one week. STLF is very important for real-time operations, including unit commitment, economic dispatch, and demand response programs [4].
- **Medium-Term Load Forecasting (MTLF)**: Forecasts are ranging from one week to one year. MTLF supports maintenance scheduling, mid-term planning, and budgeting.
- **Long-Term Load Forecasting (LTLF)**: Forecasts ranging from one year into the future to twenty years or even more. Strategic planning, infrastructure development, and policymaking are done based on LTLF.

Each forecasting horizon serves different operational and planning needs within the power system.

2.1.2 Challenges in Load Forecasting

Accurately predicting electricity load is challenging due to the multitude of influencing factors, which often involve significant uncertainty and non-linear relationships with the load. The electric load is inherently a non-linear and non-stationary process that can experience rapid fluctuations in response to weather conditions, seasonal patterns, and macroeconomic variations [5]. Key challenges include:

- **Weather Conditions**: Among many factors affecting the electricity load, one of the crucial ones is weather conditions, where temperature plays a significant role in heating and cooling requirements [2].
- **Economic Factors**: Economic growth, industrial activity, and employment rates influence overall energy demand [6].
- **Consumer Behavior**: Changes in consumer habits, such as the adoption of energy-efficient appliances or shifts in work patterns, can affect demand [7].

- **Data Quality and Availability**: Accurate forecasting relies on high-quality historical data. Missing, incorrect, outliers, or inconsistent data can reduce the accuracy of the forecast.
- **Non-linear and Non-stationary Behavior**:The complex pattern of the electricity load data makes it difficult for traditional linear models to capture the true patterns of it.

2.2 Load Profiles and Patterns

Characterization of electricity load and load patterns are essential topics in energy systems and management. They show variations in electricity demand over time and are, hence, critical for the planning, operation, and expansion of capacity resources. It is very important for utility operators and grid managers to ensure a reliable power supply, optimize generation assets, and anticipate future infrastructure requirements [8].

2.2.1 Load Profiles

Load profile shows how electricity demand changes over time. It reflects consumer behavior and is influenced by many factors, which will be explained in section 2.3.

2.2.2 Types of Load Profiles

Load profiles can be defined under three categories: residential, commercial, industrial load profiles.

Residential Load Profiles: Residential load profiles [9] shows electricity consumption of the households. It usually follow peaks in the morning and evening, lower consumption during the day and higher usage during weekend and holidays.
Commercial Load Profiles: Commercial load profiles [10] consist of office buildings, retail stores, restaurants, and hotels, which are business-related load profiles. This load profile mostly follows a 09:00 am to 05:00 pm load pattern. Commercial load profiles show high electricity demand during weekdays and reduced electricity consumption during weekends and holidays.
Industrial Load Profiles: Industrial load profiles [5] contain factories, manufacturing plants, and other large scale production facilities. In industrial load profiles, there is a constant and substantial need for the electricity during the day. Machines and equipments operate continuously. This results in a steady, high electricity demand. Industrial load pattern shows a significant demand spikes during startup and shutdown times.

2.3 Factors Influencing Short Term Load Forecasting

Electricity demand is influenced by many factors. Weather conditions, economic activity, demographic changes, technological progress, policy interventions, and others are the factors that have an impact on the demand. These factors might vary across countries, regions, cities, and time, which makes the electricity load forecasting task challenging.

2.3.1 Weather Effects

Temperature: Temperature is the most important weather variable which has direct impact on electricity demand due to heating and cooling needs. During the winter, use of heating systems increases with the low temperature, which leads increasing electricity demand . On the other hand, when the temperature is high (in warmer months), then electricity demand increases due to need for cooling.
Humidity: Like temperature, humidity has also direct effect on electricity demand during warmer months. Because the perceived temperature increases with high humidity, which leads to use of cooling systems such as air conditioning.
Wind Speed: Wind speed affects electricity demand by influencing the perceived temperature, known as wind chill [11]. Also, wind speed influences renewable energy generation, affecting the balance between supply and demand.

Solar Radiation: Solar radiation has also impact on electricity demand by influencing indoor temperatures and lighting usage.

2.3.2 Calender Effects

Hour of the Day: The demand for electricity varies throughout the day and in most cases, it is low in the morning and at night while it is high in the time between. These characteristics are relevant for managing loads as well as for the stability of the grid.
Day of the Week: Most commercial and industrial activities are less active during the weekends leading to lower electricity demand than weekdays.
Holidays: Public holidays influence the electricity load demand due changes in industrial, commercial and residential activities. Electricity consumption nearly always drops during holidays like Christmas among others.

2.3.3 Autoregressive Effect

The autoregressive effect of load is where the demand at a given time is partly determined by the load in previous hours, days, or weeks. Including lagged load variables captures the tendency of electricity demand.

Although electricity demand is greatly affected by some of the factors such as economic activity, population growth, technological advancements, electricity prices, government policies, demographic patterns, behavioral changes, grid conditions, emergencies, renewable energy adoption, and market dynamics also influence electricity demand. These factors are not covered in this thesis.

2.4 Short Term Load Forecasting Techniques

2.4.1 Statistical Methods

The STLF has benefited from the incorporation of many statistical methods due to their mathematical foundation and interpretability. These methods mostly assume that it is possible to forecast electricity consumption with the help of explanatory variables and the historical consumption patterns.

Autoregressive Integrated Moving Average (ARIMA) Models

AutoRegressive Integrated Moving Average (ARIMA) [12] models are the most used time series model due to their ability to capture autocorrelation pattern in the data. Autoregressive (AR) component capture the relationship between past (previous lags) values and current values. Integrated (I) term involve making non-stationary data to a stationary by taking the difference. Unlike AR, moving average (MA) terms capture the relation between the current observations and lagged residuals. ARIMA models have been applied successfully in load forecasting but are limited in capturing non-linear relationships and handling multiple input variables.

Multiple Linear Regression (MLR)

Multiple Linear Regression (MLR) [13] models use the independent variables linearly to predict the target variable. In STLF, temperature, humidity, and a hour of the day variables can be included as a predictor. Despite the fact that MLR is straightforward and easy to understand, it usually does not capture the nonlinear relationships between variables.

Exponential Smoothing Methods

Exponential smoothing [14] is statistical forecasting techniques that past observations are smoothed using exponentially decreasing weight to moving averages of past observations. Exponential smoothing is similar to simple moving average method in that it uses a moving average but gives more weights to recent observations.

2.4.2 Machine Learning Approaches

Machine learning models are very useful techniques and have gained popularity in last years due to their ability to capture complex relationships without having strict assumptions about the data.

Support Vector Machines (SVM)

Support Vector Machines (SVM) is very effective for high-dimensional spaces. It is used for classification and also for regression tasks. SVM maps data to high dimentional space to categorize data points. Then, a separator between data points is found, and the data is transformed in a way that the separator can be drawn as a decision boundary.

Decision Trees and Ensemble Methods

- **Decision Trees**: Decision Tree is a simple and interpretable model that split the data based on the values of the features. Decision Trees are simple but prone to overfitting.
- **Random Forests**: Random Forests is an ensemble of decision tree model where each tree in the model is trained on a bootstrap sample with random feature selection, improving generalization.
- **Gradient Boosting Machines (GBM)**: Gradient Boosting Machines creates additive models to optimize a loss function, offering high predictive accuracy..

2.4.3 Neural Networks in Load Forecasting

Artificial Neural Networks (ANNs) [15] are a type of machine learning method which is particulary suited for modelling complex and non-linear relationships.

- **Feedforward Neural Networks (FFNN):** In artificial neural networks, FFNN [15] is the common type where information passes through the units in non-cyclic way. More details will be given in Section 3.1.
- **Recurrent Neural Networks (RNN):** RNNs are designed to handle sequential data by maintaining a hidden state that captures information from previous time steps.
- **Convolutional Neural Networks (CNN):** CNNs are mostly used in image processing tasks. Recent years they have been applied to time series data by treating the data as a one-dimensional image.

Advantages of Neural Networks: ANNs can capture the non-linear relationship well and can learn automatically from raw data. Therefore, new data can be trained to adapt to changing pattern.

Challenges: ANNs require large amounts of data for the training and training deep networks can be resource-intensive. Lastly, risk of overfitting if the model is too complex for the available data.

2.5 Review of Related Studies

FFNNs are very powerful methods in STLF due to their ability to capture complex, non-linear relationships in the data. One of the earliest FFNN studies in STLF, proved that a simple FFNN could outperform traditional methods in daily electricity load forecasting [16].Another study laos examined FFNN in STLF, and he highlighted the advantage of using FFNN models in terms of accuracy compared to statistical methods [4].

More recently, two studies used an FFNN based model by integrating several external variables into model to predict short term electricity demand. The results of this study showed that better accuracy results are obtained with FFNN model when there is a non-linear relationships between variables [17],[18].

Weather variables play an important role in STLF. One study used temperature in an FFNN model for STLF and MTLF [19]. This study showed that incorporating temperature information into the model significantly improved

the model's performance in STLF, particularly during periods of extreme weather. Same study also explored the impact of multiple weather variables, including temperature, humidity, and wind speed, on STLF using an FFNN model. The authors found that the inclusion of other weather variables, in addition to temperature, improved the accuracy compared to using temperature alone. Their work highlights the importance of considering a range of weather factors when developing load forecasting models.

In addition to the studies mentioned above, Table 2.1 presents several short-term load forecasting studies.

Table 2.1.: Previous Studies on Short-Term Load Forecasting

Ref.	Market	Method	Prediction Horizon	Used Features
[20]	UK	Neural Network with Weather Ensemble Predictions	Day-ahead	Weather ensemble predictions
[21]	Japan	Neural Network	1 hour ahead	Historical load, temperature
[22]	Egypt	Artificial Neural Networks	24 hours	Temperature, humidity, historical load
[23]	Japan	Weather-Adaptive Neural Network	Short-term	Dynamic weather information
[24]	USA	ANNSTLF (Neural Network Forecaster)	Hours to days	Multiple weather variables, historical load
[25]	Australia	Neural Networks and Wavelet Transforms	Day-ahead	Weather data, historical load
[26]	China	Adaptive Hybrid Method	1 day	Weather variables, historical load
[27]	Japan	Hybrid of SOM and SVM	Short-term	Weather data, historical load
[28]	USA	Deep Neural Networks	Hourly and daily	Temp., humidity, wind speed, historical load
[29]	China	Deep Learning with Multiple Weather Variables	Hours to days	Temp., weather conditions, historical load
[30]	Iran	Deep Feedforward Neural Networks	1 day	Weather data, historical load
[31]	China	Variational Bayesian GRU Networks	1 hour to 1 day	Weather data, historical load
[32]	China	Time Series Analysis	Short-term (weekly)	Historical load, weather variables
[33]	South Korea	Deep Neural Networks	1 day	Weather data, historical load
[34]	Australia	Multi-Layer Perceptron Neural Network	1 hour ahead	Weather data, historical load
[35]	China	Bi-LSTM with Attention Mechanism	Hours to days	Weather data, historical load
[36]	France	Artificial Neural Networks	Short-term	Temperature, solar radiation, historical load
[37]	France	Neural Networks with Weather Forecasts	Day-ahead	Weather forecasts, historical load

Chapter 3

Methodology

This chapter introduces the method used in this thesis. First, the aimed method, Feed-Forward Neural Networks (FFNN), will be explained. Then, the Autoregressive (AR) model, which is used as a benchmark model in this thesis, will be explained. Then, the evaluation metrics, Mean Absolute Percentage Error (MAPE), and Shapley Values (SHAP) will be introduced.

3.1 Feed Forward Neural Networks

Feed forward neural networks (FFNN) [38], also known as deep feed forward neural networks, are fundamental models in deep learning. These networks are called feed-forward because information flows in one direction. The model has no feedback connections that loop the outputs back into the system.

A **neuron** is a core building block of the artificial neural networks. It is responsible for executing operations inside of the layers. As depicted in Figure 3.1, the first neuron $h_1^{(1)}$ in the hidden layer, which is painted with blue color, receives inputs, denoted as $X_{t,1}, \ldots, X_{t,n}$, from the input layer, which is painted with orange color. Each input is multiplied by weights $w_{1,1}, \ldots, w_{1,n}$, a crucial factor in the process, and then summed. Here, **weights** controls the strength of the connection between two neurons. In other words, when inputs are transmitted between neurons, the weights are applied to them along with an additional value. We call this additional value as **bias**. A bias is a component in neural networks, serving as a neuron with a constant value. Subsequently, a bias, b_1^0, is added to a weighted sum of the input layer and can be represented as

$$w_{1,1}X_{t,1} + w_{1,2}X_{t,2} + \ldots + w_{1,n}X_{t,n} + b_1^{(0)} = \sum_{i=1}^{n} w_{1,i}X_{t,i} + b_1^{(0)}. \tag{3.1}$$

The equation (3.1) resembles a linear regression equation. We should remember that (3.1) represents the calculation for the first neuron in the hidden layer. Consequently, this bias-added weighted sum, denoted as (3.1), undergoes pre-processing in neurons. It is reasonable to assume that there is a post-processing step in neurons known as the activation function.

3.1.1 Activation Functions

An **activation function** [39] transforms the weighted sum of the inputs of every neuron at the input side to output, and these output values are distributed as inputs to neurons of the next layer. Activation functions allow neural networks to model complex non-linear relationships. Figure 3.2 shows what is happening on the hidden neuron's side. It takes the input features; they are summed, and a bias is added. Σ represents this summation here. Then, a bias-added weighted sum goes to the activation function, represented with σ. From there, it goes to output. As mentioned above, if the activation function is linear, then we have a linear regression model. Therefore, the choice of the activation function is crucial. Here are the most popular activation functions used in time series forecasting.

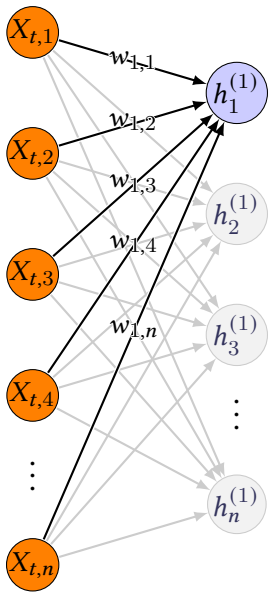

Figure 3.1.: Neural network diagram with input, hidden, and output layers. The input layer nodes are labeled $X_{t,1}$ to $X_{t,n}$, and the hidden layer nodes are labeled $h_1^{(1)}$ to $h_n^{(1)}$.

A. Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) activation function is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

The Rectified Linear Unit (ReLU) is a widely used activation function in ANNs. Its effectiveness stems from its ability to address the vanishing gradient problem by avoiding saturation in the positive input range. This results in improved training times and better convergence rates. Furthermore, its straightforward mathematical formulation facilitates efficient computational processing, making it a popular choice in practical applications. ReLU was first introduced to improve the convergence of deep networks [40].

B. Exponential Linear Unit (ELU)

The Exponential Linear Unit (ELU) activation function is defined as:

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

where α is a hyperparameter. One of the advantages of ELU is that it was shown to outperform ReLU in certain deep learning tasks [41]. This has made ELU a popular choice for researchers and practitioners in deep learning.

C. Softmax

The Softmax function is typically used in the output layer for multi-class classification problems. It converts logits into probabilities:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

The Softmax function, a type of activation function used in neural networks [42]. It is commonly used to convert a vector of real numbers into a probability distribution over multiple classes.

D. Softplus

The Softplus function is a smooth approximation of the ReLU function and is defined as:

$$\text{Softplus}(x) = \ln(1 + e^x)$$

The Softplus function is directly related to the Sigmoid function, as it is the derivative of the Sigmoid function [43].

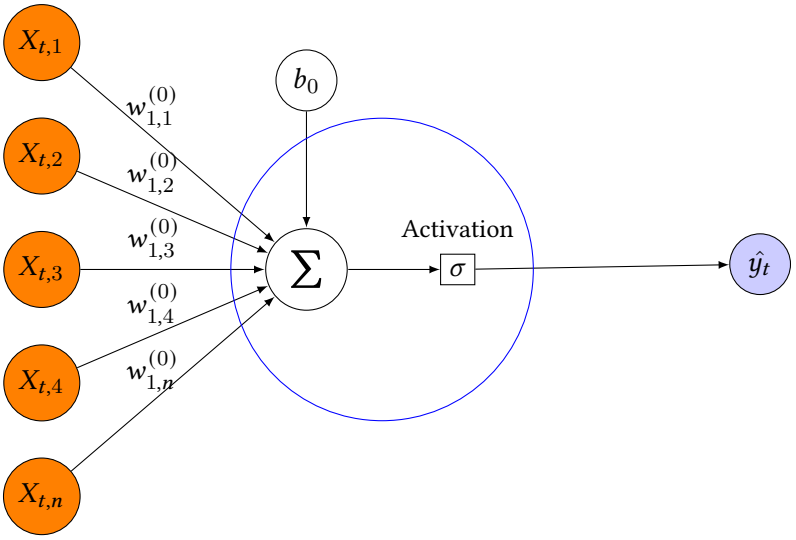

Figure 3.2.: Architecture of a single neuron in a FFNN. The inputs $X_{t,1}, X_{t,2}, \ldots, X_{t,n}$ are multiplied by corresponding weights $w^{(0)}_{1,1}, w^{(0)}_{1,2}, \ldots, w^{(0)}_{1,n}$, summed along with a bias term b_0, and then passed through an activation function σ to produce the output $\hat{y}_t$.

E. Hyperbolic Tangent (Tanh)

The Tanh function [44] is a scaled version of the Sigmoid function and is defined as:

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The outputs values range from -1 to 1.

F. Sigmoid

The Sigmoid function is defined as:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

It outputs values between 0 and 1, making it useful for binary classification problems. The Sigmoid function was widely used in early neural networks and was featured in the foundational backpropagation papers [45].

Equation 3.1 represents a bias added to a weighted sum of input variables. This bias-added weighted sum then goes through to an activation function, which is depicted as σ, which can be any activation function described in section 3.1.1. Then equation 3.1 becomes

$$\sigma\left(\sum_{i=1}^{n} w_{1,i} X_{t,i} + b_1^{(0)}\right) \tag{3.2}$$

Equation 3.2 is just the calculation of one neuron inside of the first hidden layer. If we consider all the neurons or nodes in the hidden layer in figure 3.1, then the equation 3.2 becomes

$$\begin{pmatrix} h_1^{(1)} \\ h_2^{(1)} \\ \vdots \\ h_m^{(1)} \end{pmatrix} = \sigma\left(\begin{pmatrix} w^{(0)}_{1,1} & w^{(0)}_{1,2} & \cdots & w^{(0)}_{1,n} \\ w^{(0)}_{2,1} & w_{2,2} & \cdots & w^{(0)}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w^{(0)}_{m,1} & w^{(0)}_{m,2} & \cdots & w^{(0)}_{m,n} \end{pmatrix} \begin{pmatrix} X_{t,1} \\ X_{t,2} \\ \vdots \\ X_{t,n} \end{pmatrix} + \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_m^{(0)} \end{pmatrix}\right). \tag{3.3}$$

The vector $h_1^{(1)}, \ldots, h_m^{(1)}$ is the output of the hidden neurons, which will be input to another hidden layer or output of the network. If we assume we have a network with two hidden layers, then equation 3.3 becomes

$$\begin{pmatrix} h_1^{(2)} \\ h_2^{(2)} \\ \vdots \\ h_k^{(1)} \end{pmatrix} = \sigma\left(\begin{pmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & \cdots & w_{1,n}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & \cdots & w_{2,n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1}^{(1)} & w_{k,2}^{(1)} & \cdots & w_{k,n}^{(1)} \end{pmatrix}\begin{pmatrix} h_1^{(1)} \\ h_2^{(1)} \\ \vdots \\ h_m^{(1)} \end{pmatrix} + \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_m^{(1)} \end{pmatrix}\right). \tag{3.4}$$

Graphical illustration of equation 3.4 is shown in Figure 3.4. Lastly, to get the predictions, 3.4 goes through another layer known as the output layer. The output layer is illustrated in Figure 3.5 with green-colored neurons. With the output layer, a model can be represented as

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \sigma\left(\begin{pmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} & \cdots & w_{1,n}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} & \cdots & w_{2,n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1}^{(2)} & w_{k,2}^{(2)} & \cdots & w_{k,n}^{(2)} \end{pmatrix}\begin{pmatrix} h_1^{(2)} \\ h_2^{(2)} \\ \vdots \\ h_m^{(2)} \end{pmatrix} + \begin{pmatrix} b_1^{(2)} \\ b_2^{(2)} \\ \vdots \\ b_m^{(2)} \end{pmatrix}\right) \tag{3.5}$$

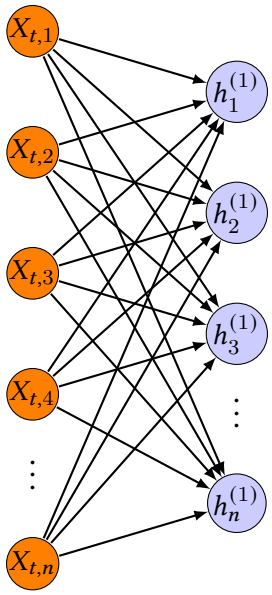

Figure 3.3.: Input Layer to First Hidden Layer in a Feed-Forward Neural Network (FFNN). This figure illustrates the connections between the input layer and the first hidden layer in a feed-forward neural network. Each input $X_{t,1}, X_{t,2}, \ldots, X_{t,n}$ is fully connected to every neuron in the first hidden layer, denoted as $h_1^{(1)}, h_2^{(1)}, \ldots, h_n^{(1)}$.

The output layer in Figure 3.5 produces the final output, which could be a single value (for regression) or multiple values. Getting the predictions y in the output layer (Figure 3.5) and the process above is known as forward-propagation. To improve the model's performance, the weights have to be updated. This procedure is known as **backpropagation**.

3.1.2 Backpropagation

Backpropagation (BP) [46] is a learning algorithm used in neural networks to reduce error by adjusting the weights and biases. A loss function is required to achieve this. The loss function is the difference between the actual and predicted values, enabling the neural network to monitor its overall performance. The choice of loss

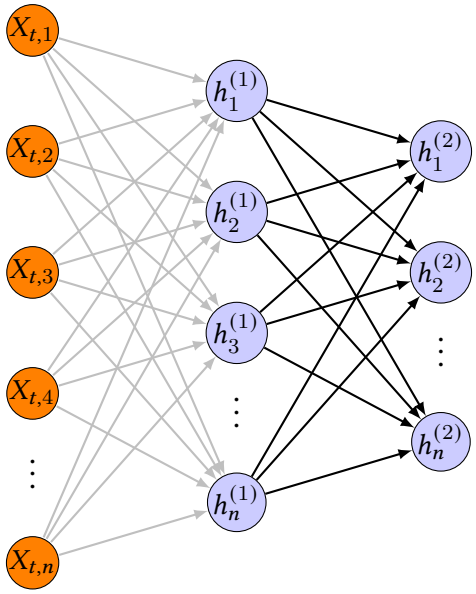

Figure 3.4.: Full Feed-Forward Neural Network Architecture. This figure represents the architecture of a multi-layer Feed-Forward Neural Network (FFNN) with two hidden layers. The inputs $X_{t,1}, X_{t,2}, \ldots, X_{t,n}$ are fully connected to neurons in the first hidden layer, $h_1^{(1)}, h_2^{(1)}, \ldots, h_n^{(1)}$, and then to neurons in the second hidden layer, $h_1^{(2)}, h_2^{(2)}, \ldots, h_n^{(2)}$. The network computes weighted sums, applies activation functions at each hidden layer, and ultimately produces outputs $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$.

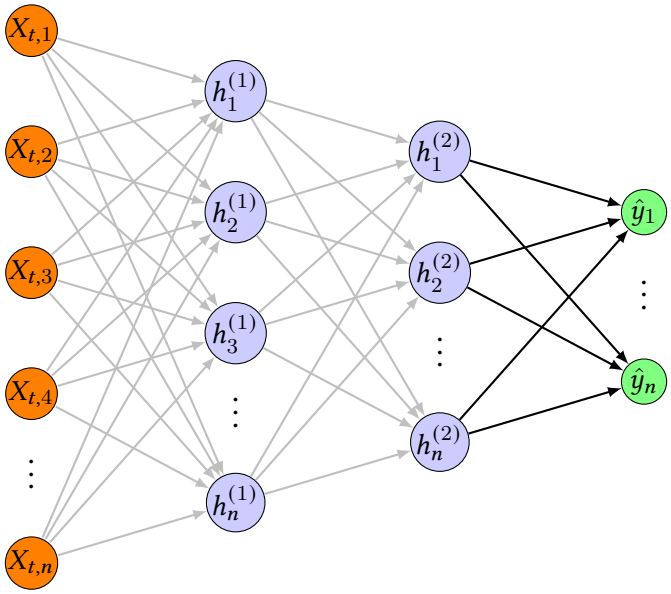

Figure 3.5.: Simplified Feed-Forward Neural Network Showing Focus on Output Layer. This figure displays a simplified version of a Feed-Forward Neural Network (FFNN). The first hidden layer's connections are shown in a lighter shade to highlight the connection between the second hidden layer $h_1^{(2)}, h_2^{(2)}, \ldots, h_n^{(2)}$ and the output layer $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$.

function depends on the specific task. **Mean Squared Error (MSE)**, **Mean Absolute Error (MAE)**, and **Mean Absolute Percentage Error (MAPE)** are commonly used loss functions in regression models [47]. These loss

functions are defined as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.6}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.7}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{3.8}$$

where y_i are the actual values, $\hat{y}_i$ are the predicted values, and N is the number of data points. In general we use the notation below for the loss functions:

$$\mathcal{L} = \mathcal{L}(Y, \widehat{Y}) = Error(Y, \widehat{Y}) \tag{3.9}$$

The goal of back propagation (BP) is straightforward: to adjust each weight in the network in proportion to its contribution to the overall error. By iteratively reducing the error associated with each weight, the network eventually converges to weights that produce accurate predictions. BP leverages optimization algorithms to minimize the error, with modern methods like the Adaptive Moment Estimation (Adam) optimizer being particularly effective.

Adaptive Moment Estimation (Adam) [48] is a widely used optimization algorithm in deep learning that combines the advantages of Adagrad [49] and RMSprop [48]. Adam maintains individual adaptive learning rates for each parameter by computing estimates of the first and second moments of the gradients. It adjusts the learning rates based of the gradients, which enables faster convergence than other algorithms. Adam also includes bias correction terms to account for initialization biases in the moment estimates of the gradients. Steps of Adam is explained in Appendix A.1.

3.1.3 Layer Normalization

Layer normalization (LN) [50] is a technique for normalizing the activations of neurons across the features within a layer. It is alternative to Batch normalization (BN) which is found to be computational more demanding than LN, LN stabilize and improve the training process. Formula for LN is defined as:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{3.10}$$

where x_i is the input, μ is the mean of the inputs in the layer, σ^2 is the variance, and ϵ is a small value added for numerical stability. After Equation 3.10, layer normalization also includes learnable scaling γ and shifting β parameters as shown in the Equation 3.11, which allow the model to adapt the normalized values to better fit the data.

$$y_i = \gamma\hat{x}_i + \beta. \tag{3.11}$$

3.1.4 Regularization

ANNs are very powerful machine learning models. Like every other model, ANNs also have disadvantages. One of the biggest problems is overfitting [51]. Overfitting occurs when a machine learning model becomes too complex and learns not just the underlying patterns in the training data but also the noise or irrelevant details. Therefore model performs very well on the training data but needs to generalize to new, unseen data. There are couple of methods to prevent or to reduce overfitting.

A. Dropout

Dropout [52], a regularization technique, aims to reduce over-fitting in neural network models during training. Dropout randomly drops the units from the network. This choice is random and depends on the chosen drop rate. When dropout is used, all the units' connections (incoming and outgoing) are also dropped. As shown in Figure, $X_{t,1}$ and $X_{t,3}$ are dropped from the network. Their connections are also dropped from the model.

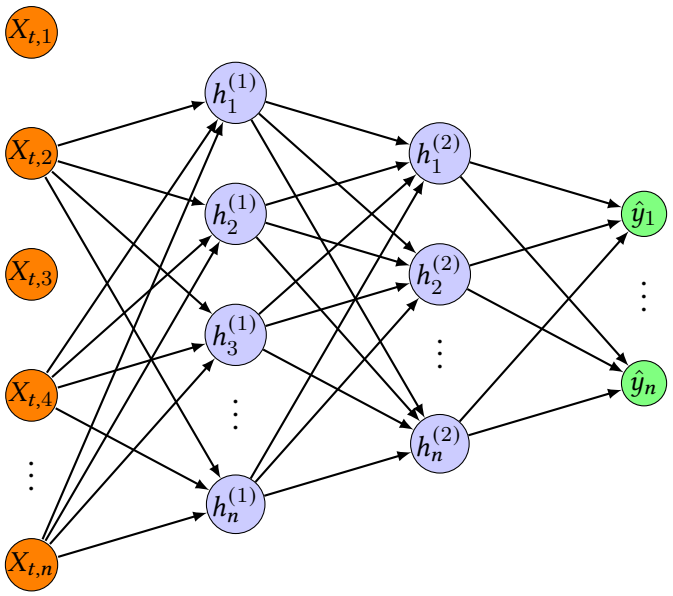

Figure 3.6.: Neural network diagram with input, two hidden, and output layers after dropout. $X_{t,1}$ and $X_{t,3}$ inputs are dropped.

B. L1 Regularization

L1 regularization, also known as Lasso (Least Absolute Shrinkage and Selection Operator) [53], is a type of regularization technique used in machine learning models. L1 regularisation reduces overfitting problem. L1 adds a regularization term to the loss function, which is proportional to the sum of the absolute values of the model parameters. L1 regularization can shrink some model coefficients to exactly zero, effectively performing feature selection by eliminating less important features. This makes L1 regularization especially useful when working with high-dimensional datasets where many features may be irrelevant.

A **kernel regularizer** [54] is applied to the neural network's weights. The model adds a penalty equal to the sum of the absolute values of the weights when using L1 regularization. The critical effect is that it encourages sparsity in the model's weights, which means some weights may become exactly zero, eliminating some features and promoting simpler models.

Opposite to the kernel regularizer, **activity regularizer** [54] applies a penalty to the layer. It can be used to prevent the activations from growing too large or too complex, which could lead to overfitting. The network produces more regular and generalizable outputs by penalizing large activations.

Both regularization techniques are useful for reducing overfitting and improving the model's performance on unseen data by controlling different parts of the neural network. The total loss function of the neural network

with both L1 kernel regularization and L1 activity regularization is defined as:

$$\mathcal{L} = \text{Loss}(y, \hat{y}) + \lambda_1 \sum_{i=1}^{n} |\theta_i| + \lambda_2 \sum_{j=1}^{m} |a_j|,$$

where $\mathcal{L}$ is the total loss, $\text{Loss}(y, \hat{y})$ is the data loss, θ_i are the model's weights (kernel), and λ_1 controls the strength of L1 kernel regularization, a_j are the activations of the neurons, and λ_2 is the rate of activity regularization.

C. Early Stopping

Early stopping [55] is a technique to prevent from overfitting in ANN. It is used to stop training of the model when the performance of the model stops improving. The key concept involves monitoring the validation error throughout training and stopping when it saturates. This approach typically uses "patience" parameter, which defines a minimum number of epochs to wait before deciding that the validation error has truly stabilized and is no longer decreasing. As shown in figure 3.7, after 500 iterations, validation errors increase. Here, the 'patience' parameter can be setted up to finish the iteration. Overfitting can be reduced, and unnecessary computational time can be saved.

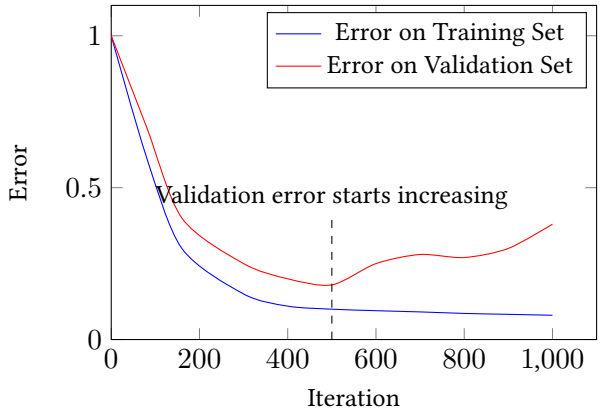

Figure 3.7.: Training and validation error as a function of iterations. A vertical dashed line marks the point (iteration 500) where the validation error starts to increase, indicating potential overfitting.

3.2 Autoregressive (AR) Models

Autoregressive (AR) models are a subclass of ARIMA models where the future value of a variable is assumed to be a linear function of its past values. An AR(p) model is defined as:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t$$

where Y_t is the value at time t, c is a constant, ϕ_i are the model parameters, p is the order of the model, and ε_t is white noise [56]. AR models have been widely used in short-term load forecasting due to their simplicity and effectiveness in capturing linear temporal dependencies [13].

3.3 Model Evaluation

To evaluate the performance of the model, following evaluation methods are used.

3.3.1 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is a measure of prediction accuracy in a forecasting model [47]. MAPE shows the average absolute difference between the predicted values and the actual values as a percentage of the actual values. MAPE is defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{3.12}$$

where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of data points. A lower MAPE value indicates that a better fit of the model to the data.

3.3.2 Shapley values (SHAP)

Shapley values (SHAP) are inspired by the game theory concept [57], which helps evaluate the impact of each feature on predicting outcomes in machine learning models [58]. Unlike evaluation metrics that often overlook feature dependencies, SHAP feature importance provides a comprehensive analysis of individual and combined feature impacts while considering collinearities in the model's predictions [59].

Kernel SHAP [60] is a technique that does not rely on any model and works to estimate Shapley values for any machine learning models by using weighted linear regression methods to distribute the model predictions among different features as if they were players in a coalition game aiming for fair payouts based on their contributions. The Shapley value (ϕ_i) of a feature i is calculated as follows;

$$\phi_i(f,x) = \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!}\left(f(S\cup\{i\}) - f(S)\right) \tag{3.13}$$

where S is a subset of features excluding feature i, $|S|$ is the size of the subset S, $|N|$ is the total number of features, $f(S)$ is the model prediction using the subset S of features, $f(S\cup\{i\})$ is the model prediction using the subset S with feature i included. Direct computation of the Shapley value for all subsets of features is computationally expensive, especially for large models. Kernel SHAP addresses this by approximating Shapley values using a weighted linear regression approach.

The Shapley values are approximated as the solution to the following weighted least squares problem:

$$\min_{\phi}\sum_{S\subseteq N}\left(f(S) - \phi_0 - \sum_{i\in S}\phi_i\right)^2 \cdot w(S) \tag{3.14}$$

where $w(S) = \frac{1}{|S|(|N|-|S|)}$ is the weighting kernel, which assigns higher weights to subsets closer in size to the full set of features, ϕ_0 is the base value (i.e., the expected output if no features are included), ϕ_i is the Shapley value for feature i

3.4 Tools and Software Used

Due to technical limitations, the author was unable to use their personal computer for coding.In this thesis, all programming tasks were carried out using Google Colab[61]. The software programs Python [62] with the libraries [63], [64], [65], [66], [67], [68], [69], [70] and R [71] with the libraries [72], [73] have been used to perform tasks.

Additionally, this thesis utilizes OpenAI ChatGPT-4o [74] to enhance language and readability by providing rephrasing suggestions.

Chapter 4

Model Training: Data Preparation and Parameter Optimization

In this chapter the methods that are used in this thesis will highlighted. This involves explaining the steps for collecting and preprocessing data, analyzing and creating features, training and tuning the model.

4.1 Data Collection

The dataset covers a period of five years and starts from 01.08.2019 to 01.08.2024. The electricity load data is collected from the ENTSOE [75]. The weather data is collected from the German Meteorological Service DWD [76]. Figure 4.1 shows the electricity day ahead and the actual temperature series.

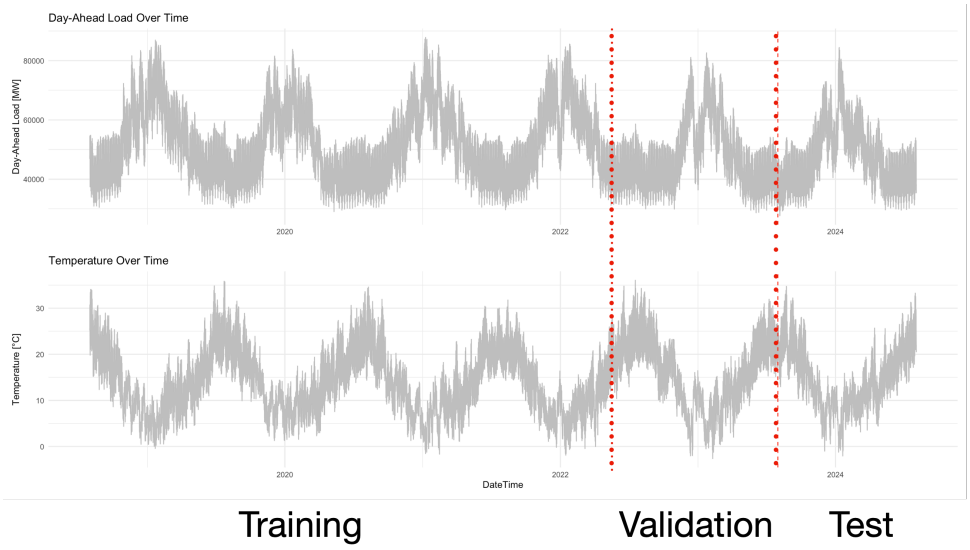

Figure 4.1.: Time series plots of the considered data.

The temperature forecasts are also collected from same database DWD [76]. For temperature data (actual and forecasts), hourly averages of ten most populated cities in France used, those cities are illustrated in Figure 4.2. The weather forecasts are published every hour for the next ten days.

The electricity load of France shows seasonal patterns. In Figures 4.1, 4.3 these seasonal (daily, weekly, and annually) patterns are illustrated. In Figure 4.1, there is lower energy consumption during summer and higher energy consumption during the winter. In Figure 4.3, we illustrated the weekly mean of the load profile. It can be seen that higher consumption on weekdays and lower consumption on weekends. Also reduced consumption at night compared to daytime. This also indicates that there is a high autocorrelation in the most recent lags, as well as

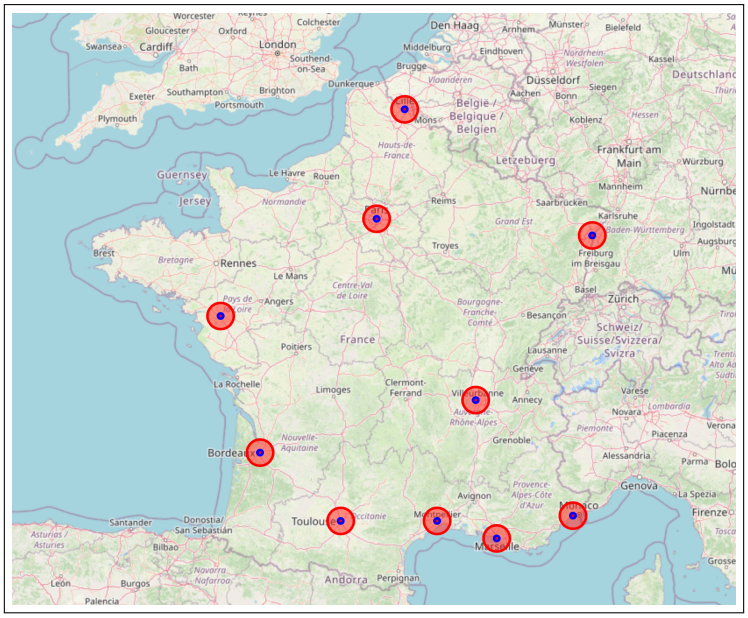

Figure 4.2.: Location of the considered cities for the weather data in France. The map is generated in Python using the library Folium

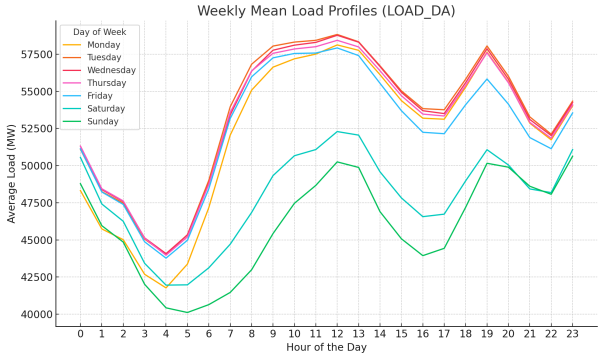

Figure 4.3.: Weekly average electiricity load. Days are presented in different colors.

for lags of the neighboring hours on the previous day and the previous week [1]. Similar to load data,the temperature shows clear seasonal patterns (such as weekly and annual cycles) as shown in Figure 4.3 and in Figure 4.4. The temperature impacts the electricity load in France due to high dependency on electricity heating and cooling systems [1]. A small change in temperature value can lead to a significant change in the consumption [77]. Thus, the use of electric heating has a direct impact on electricity consumption. The relationship between load and temperature is non-linear, and this relationship is well-established in the literature. It is confirmed that temperature changes directly influence electricity load, with particularly strong effects during winter [77], [78]. The non-linear relationship between load and temperature can seen in Figure 4.5.

4.2 Data Preprocessing

Load day-ahead and temperature datasets are checked for missing values and not detected any of them. Then Day Light Saving (DST) change is applied to both datasets. Missing hours during the day light change are filled with linear interpolation.
To prepare the data for modeling and improve the performance of the machine learning algorithms, **Robust Scaler**

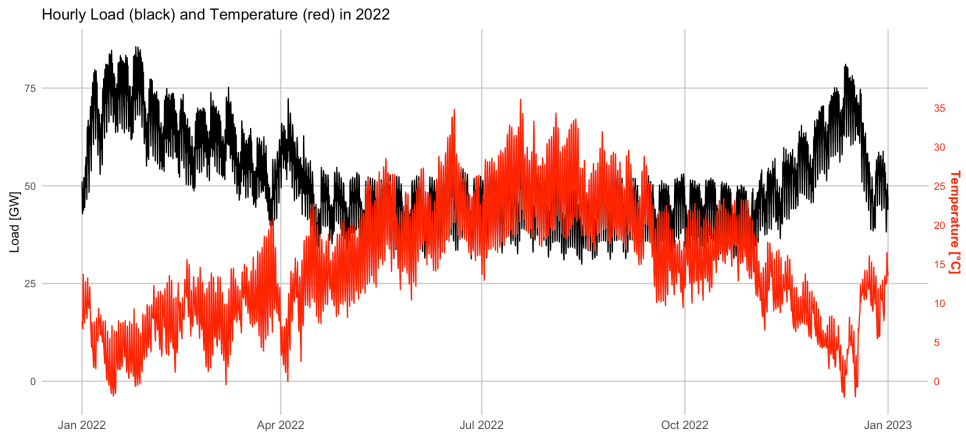

Figure 4.4.: Hourly load (black) and temperature (red) in 2022.

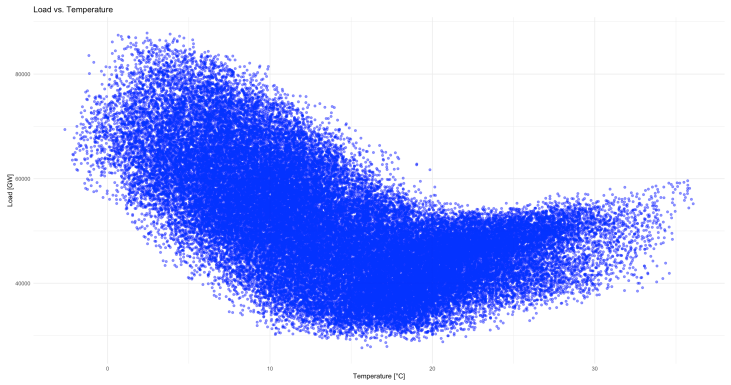

Figure 4.5.: Load (Y-axis) vs temperature(X-axis).

transformation was applied. The Robust Scaler [66] is a scaling method that removes the median and scales the data according to the interquartile range (IQR). Models trained on scaled data often converge faster and can achieve better performance due to the reduced influence of outliers and feature value disparities. The robust scaler is defined as

$$x_i^{\text{scaled}} = \frac{x_i - \text{Median}(X)}{\text{IQR}(X)}$$

where $Median(X)$ is the median of the feature. $IQR(X) = Q_3 - Q_1$ is the interquartile range. Q_1 and Q_3 are the 25th and 75th percentiles of the feature, respectively.

4.3 Feature Selection and Engineering

To account for seasonal variations, variables representing hour of the day (ranging from 0 to 23), day of the week (ranging from 1 to 7), and month of the year (ranging from 1 to 12) were created. These variables help to capture daily, weekly, and yearly cycles. Additionally, the last 30 and 15 days of average, cumilative sum , and standard deviation of the load day ahead were created to account for fluctuations in usage patterns, trends, and seasonal patterns. We shifted those variables by 8 days to prevent the use of last 8 days. To account for autoregressive effect, we included lags of the load 192 (8 days) hour, 336 hours (2 weeks) and 504 hours (3 weeks) as a input features. For the temperature inputs, we included the temperature lags by 1 ,2, 3, 24, 48 hours, the mean temperatures over the past 6, 12, and 24 hours. Additionally, we included minimum and maximum temperatures from the last 24 hours and Heating Degree Days (HDD) and Cooling Degree Days (CDD). HDD refers to the heating energy requirements of buildings, and CDD refers to the cooling requirements of buildings. We use a base temperature

of 18℃ for calculating HDD and 21℃ for calculating CDD [79].
The forecast horizon is 192 hours (8 days) ahead. The in-sample period covers the hourly range from August 1, 2019, to August 1, 2023. The out-of-sample period begins on August 1, 2023, and ends on August 1, 2024. The in-sample and the out-of-sample periods are shown in Figure 4.1 with dashed red lines. A rolling window study is employed for the models. It is a widely used and standard procedure in the electricity literature [80].

4.4 Network Architecture

The neural network designed for electricity load forecasting is structured to capture the complex relationships between the input variables and the target output. The architecture uses input layer, two hidden layers, layer normalization after every hidden layer and output layer. As shown in Figure 4.6:

- **Input Layer** $(X_{t,1}, \ldots, X_{t,24})$**:** All the input variables that are created in 4.3.
- **Hidden Layers (depicted with blue neurons):** Consists of multiple hidden layers to model nonlinear patterns in the data. For instance, the network includes two hidden layers, each with the neurons that would be obtained from hyperparameter tuning. These layers apply activation functions to introduce non-linearity, enabling the network to learn complex features.
- **Output Layer (depcited with green neurons,** $\hat{y}_1, \ldots, \hat{y}_{192}$**):** The output layer produces 192 outputs which means 8 days ahead forecasts. A linear activation function is used in the output layer since the goal is to predict a continuous target variable (electricity load).
- **Loss Function:** Mean Absolute Error (MAE) is chosen as the loss function.
- **Optimizer:** The Adam optimizer is utilized for training due to its adaptive learning rate capabilities and efficiency in handling sparse gradients.
- **Batch Size:** A batch size of 32 is selected, meaning the network updates its weights after processing every 32 samples.
- **Epochs:** The model is trained over 1000 epochs, allowing multiple passes through the entire training dataset to adjust the network weights incrementally.
- **Early Stopping:** Early stopping is implemented based on the validation loss with a patience of 50 epochs.

4.5 Hyperparameter Optimization

Optuna [63] is used for the hyperparameter tuning. Optuna is an open source hyperparameter optimization framework designed to automate the process of optimizing hyperparameters in machine learning models. It utilizes a Bayesian optimization algorithm to find the optimal parameters. By focusing on minimizing the MAE, the optimization process ensures that the model's predictions are as close as possible to the actual values in terms of absolute differences, which is critical for practical applications in energy management.

The hyperparameters are obtained during the tuning run for each month in the training set. For example, the data from 01.08.2019 to 31.07.2023 is used to get the parameters for August 2023. This process was repeated up to August 2024, which resulted in 12 hyperparameter tunings. The hyperparameter process consists of 256 iterations.The tuned parameters are reported below.

- **Input features:** 24 input features that are described in 4.3. 24 hyperparameters.
- **Dropout layer:** Whether to use the dropout layer after the input layer. 1 hyperparameter.
- **Dropout rate:** If dropout yes, then the rate parameter is drawn from (0, 1). 1 hyperparameter.

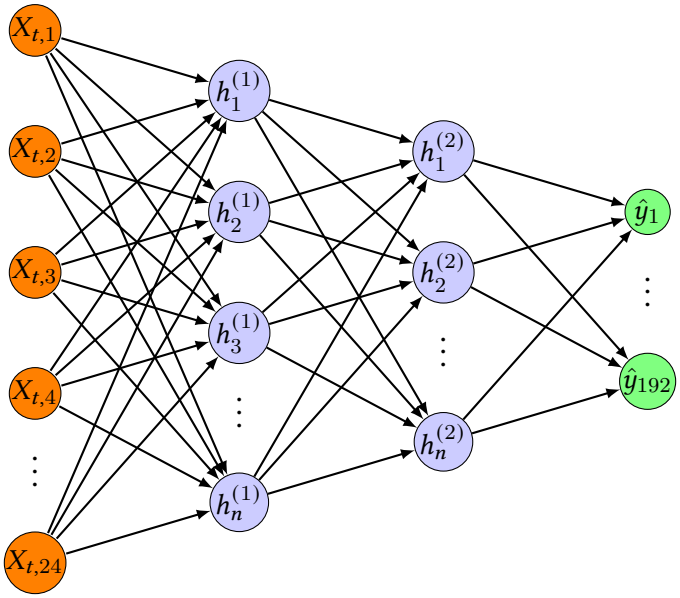

Figure 4.6.: Neural network diagram with input, two hidden, and output layers. The input layer nodes are labeled $X_{t,1}$ to $X_{t,n}$, the first hidden layer nodes are labeled $h_1^{(1)}$ to $h_n^{(1)}$, the second hidden layer nodes are labeled $h_1^{(2)}$ to $h_n^{(2)}$, and the output layer nodes are labeled y_1 to y_{192}.

- **Number of neurons in the hidden layers:** The values are chosen from integers from [16, 256] interval. 1 hyperparameter per layer.
- **Activation functions:** Elu, relu, sigmoid, softmax, softplus, and tanh. 1 hyperparameter per layer.
- **L1 regularization for hidden layers:** Whether to use the L1 regularization and their weights in the hidden layers. 2 hyperparameters per layer.
- **Rate of L1 regularization:** If L1 regularization yes, then the rate is drawn from $(10^{-5}, 10)$ on a log-scale. 2 hyperparameters per layer.
- **Learning rate:** Learning rate of the Adam algorithm. It chosen from $(10^{-5}, 10^{-1})$ on a log-scale. 1 hyperparameter.

The model is retrained daily, ensuring that they are always up-to-date with the most recent set of observations, thereby enhancing the reliability of our predictions. And the predictions are made every day at 08:30. We use the recent data up to 08:30 and additionally updated the weather inputs for the next 10 days using the temperature forecasts. The steps from hyperparameter tuning to model predictions are summarized below.

- Obtain hyperparameters for a specific month using the data up to that month. We roll the window by one month for the hyperparameter tuning of the next month.
- Use those parameters to train the model.
- Use the weather forecasts in the test set and make predictions for next 192 hours.
- make predictions everyday at 08:30 for next 192 hours which starts from hour 09:00 am.
- Repeat that process until reaching end of the out-of-sample dataset.

4.6 AR Model

The AR parameter p is tuned by minimizing the Akaike information criterion (AIC) with $pmax = 24x30 = 720$. This is done using the function $auto.arima()$, the forecast package in R [73].

Chapter 5

Result and Discussion

5.1 Results of Hyperparameter Tuning

The FFNN hyperparameter tuning conducted 14 times[1]. In Figure 5.1, the choice frequency of input features is presented. This figure is obtained using the best of 12 hyperparameter tuning results. Day of the week, hour of the day, load lag 336, temperature actual, temperature lag 48 and temperature max are the most important inputs according to this figure.

The average of neuron numbers in first hidden layer is 177 and 171 in the second hidden layer. Figure 9 presents

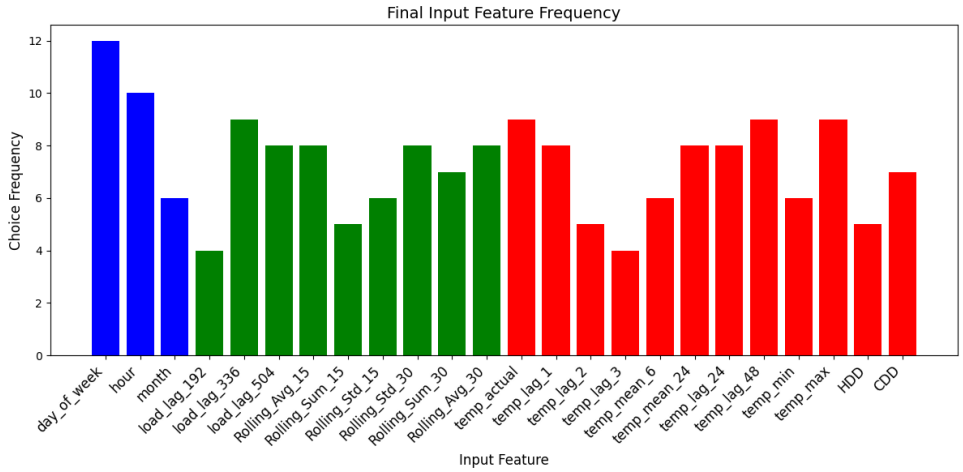

Figure 5.1.: Input choice frequency in the best 12 FFNN hyperparameter sets.

the choice of neurons in the hidden layers.

Figure 5.3 shows the validation errors of the best results from 12 hyperparameter tuning runs. The lowest errors are achieved during the first two months and the final month. Interestingly, these months correspond to configurations where the total number of neurons in the hidden layers was lower compared to other months[2]. Figure 5.4 shows the tuning result of the activation functions in each layer. In the first hidden layer, relu was chosen with a 58 % rate, and in the second hidden layer, elu, softmax, and softplus were equally chosen with a 25 % rate. Figure 5.5 reports the tuning result of regularization methods.In most cases, a dropout rate of 76% has not been selected, and this result is consistent with findings from several studies [54], [81], [82]. The $L1$ regularization of layer one was used with a 43 % rate and almost not chosen in the second layer.

[1] 12 times for data with all input features, one time for data with no temperature input, and one time for data with holiday input.

[2] The total number of neurons for the optimal hyperparameters in the first month is 201, in the second month is 228, and in the last month is 165

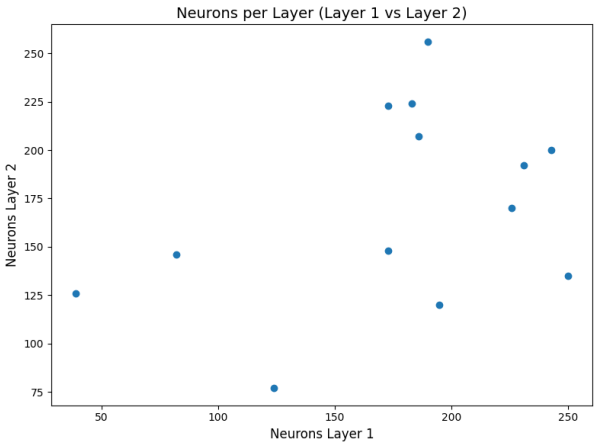

Figure 5.2.: Neurons per layer in the best 12 FFNN hyperparameter runs.

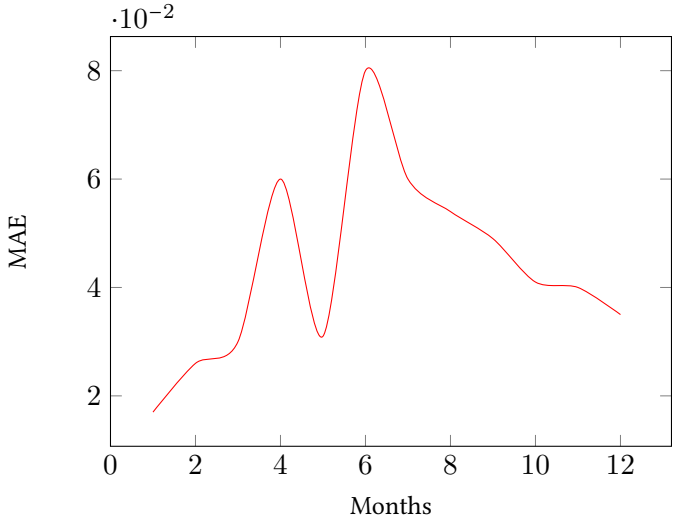

Figure 5.3.: Validation errors (in MAE) during parameter tuning. X-axis represents the months which starts from August (1) 2024 and ends on July (31) 2024.

5.2 Performance on Test Data

FFNN and AR models are used to predict the load for 365 days from 01 August (09:00 am) 2023 to 08:00 am 01 August (08:00 am) 2024. The predictions for the FFNN model obtained using temperature forecasts. The prediction results for model AR and FFNN are shown in Table 5.1. As seen on this table, the FFNN model yields better predictions than AR model. The Average MAPE of FFNN is 0.08 % and 0.13 % for the AR model. We can also see that the MAPE score of the AR model is 0.126 %, which is 0.015 % better than the FFNN model since the AR model is straightforward but handles the autoregressive effect a way better than the FFNN model. Since the aim is to prove the impact of the weather variables in short term load forecast, we also employed another FFNN model, which doesn't account for temperature data. We call this model as FFNN-wt (FFNN without temperature). The hyperparameter tuning of the FFNN-wt model is employed only in January. Since the temperature is low during the winter and the load is high in this month, it makes sense to show the temperature effect of this month. For simplicity, we took the MAPE of the first eight days ahead of the predictions of January, which corresponds to days between 01.01.2023 09:00 am to 09.01.2024 08:00 am. The average MAPE for this prediction is reported as 0.11 %. The MAPE score for the same period of FFNN with temperature inputs is reported as 0.085. In Figure 5.6, we can see the predictions of the corresponding week with the FFNN model and the FFNN-wt. The model FFNN-wt seems to have a problem with underestimation of the higher load values. The model FFNN performs better than the model FFNN-wt. Until now, we did not account for the public holiday or special days impact on

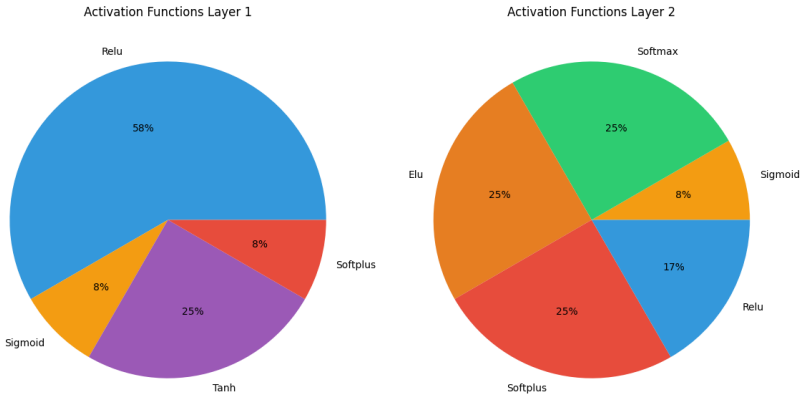

Figure 5.4.: Activation functions in the best 12 FFNN hyperparameter runs.

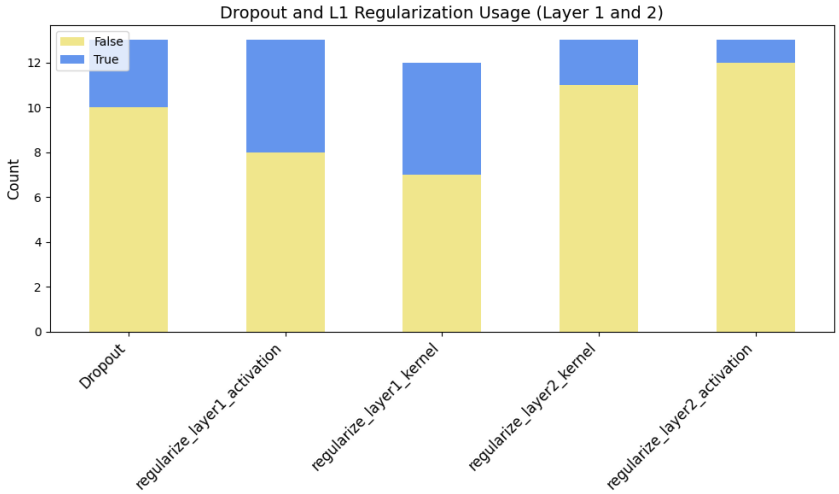

Figure 5.5.: Regularization frequency in the best 12 FFNN hyperparameter runs.

the load. To account for that, a dummy variable is included to capture the new year's impact on electricity load. A hyperparameter tuning was made, but this time, the new year dummy was chosen as a mandatory variable in the model, but other variables were tuned. Using the tuned parameters from this model, we obtain the prediction for the first week of January as we did for the models above. The FFNN with the new year dummy MAPE is reported as 0.081, which improved the predictions of the FFNN model, decreasing the MAPE by 0.004.

To understand how the load is affected by the different explaining variables i.e. the relation between the response and the input variables, Shapley Values (SHAP) are employed. In Figure 5.7 the model FFNN of first 8 days ahead predictions of January is shown. This figure is a summary plot of SHAP (in section 3.3.2), which provides insight into the impact of input variables on the predictions. The X-axis represents the SHAP. It shows the contribution of each feature on the predictions. Positive SHAP (right side) increase the predicted load, whereas negative SHAP values (left side) decrease the predicted load. In Figure 5.7, rolling sum of last 15 days has a large and consistent effect on the predictions, as indicated by the wide spread of SHAP. Higher values (red dots) tend to have positive SHAP, which means that high rolling sum of last 15 days values increase the predicted load. For the temperature lag 1, higher values (red) are associated with lower predictions (negative SHAP), indicating that higher lagged temperatures decrease the predicted load. Rolling sum of last 15 days, mean of last 24 hours of temperature, and rolling average of last 30 days have a strong impact on the load prediction. Some features, such as actual temperature and HDD, day of the week, CDD seem to have a more localized impact around 0, suggesting they contribute less.

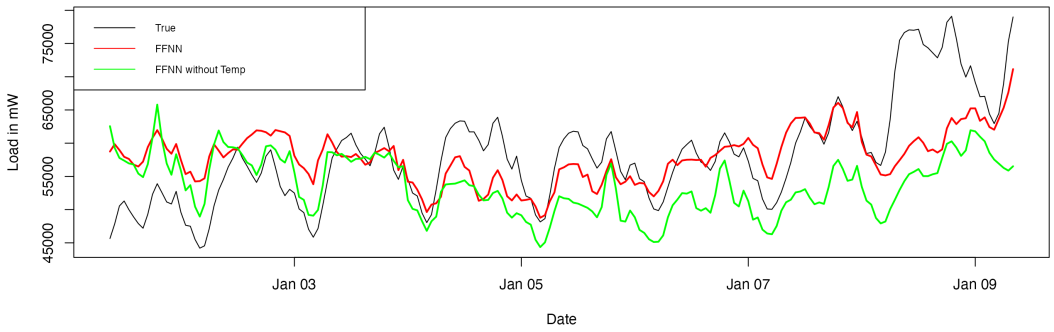

Figure 5.6.: 8 Days Ahead Predictions Starting from 01.01.2024 09:00 AM. The true load values are shown in black, predictions from the Feed-Forward Neural Network (FFNN) model are in red, and predictions from the FFNN model without incorporating temperature are shown in green.

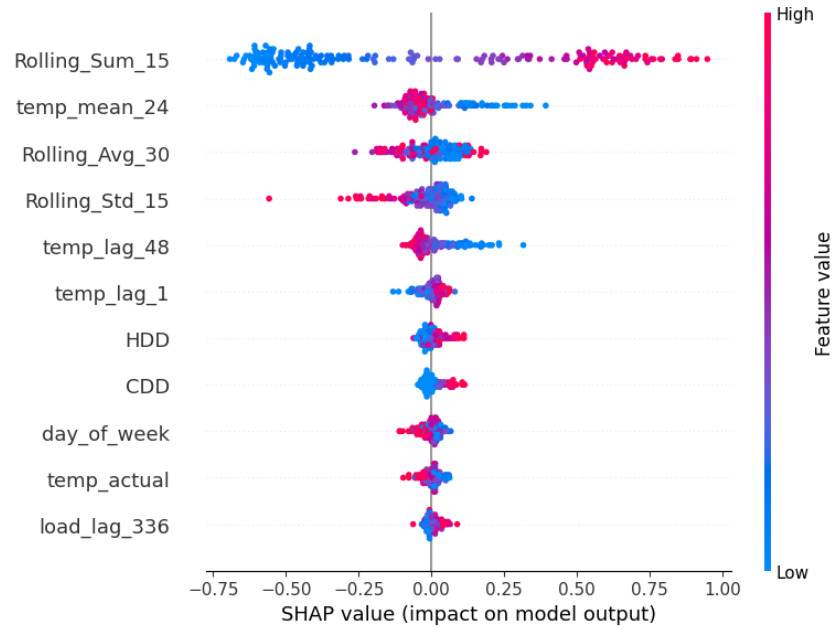

Figure 5.7.: SHAP Summary Plot for Feature Importance for January 2024

Table 5.1.: Average MAPE for Each Month

Month	FFNN	AR
2023-08	0.087	0.175
2023-09	0.071	0.166
2023-10	0.081	0.154
2023-11	0.061	0.114
2023-12	0.087	0.106
2024-01	0.088	0.187
2024-02	0.052	0.137
2024-03	0.093	0.119
2024-04	0.087	0.121
2024-05	0.126	0.111
2024-06	0.084	0.115
2024-07	0.091	0.120
Average	0.08	0.13

Several temperature related features (mean of last 24 hours of temperature, temperature lag 48, temperature lag 1 hour, actual temperature, HDD, and CDD) are highlighting the model's sensitivity to temperature on the predictions.
Figure 5.7 visually confirms the importance of rolling statistics (rolling sum of last 15 days, rolling average of last 30 days) and temperature features in predicting electricity load in first 8 days of January 2024. SHAP figures of other months are reported in Appendix A.2. Looking at those figures, we can draw several conclusions about the common features influencing electricity load prediction model. The rolling statistical features and lagged load variables are the key predictors that they reinforces the fact that electricity consumption is highly dependent on recent usage patterns and trends. Temperature is a critical feature, with colder temperatures driving higher load due to heating requirements. The impact of temperature lags further highlights that recent weather conditions can continue to influence future consumption. Seasonal factors like day of the week and hour of the day are crucial, indicating that human behavior (working hours, weekends, holidays) plays a significant role in electricity consumption patterns.

5.3 Discussion

This thesis aims to explain how weather variables impact short-term electricity load forecasts. In the short-term electricity price forecasting literature, weather effects are usually not incorporated directly [**?**]. We use the weather forecasts of temperature directly in this thesis. The temperature was of high interest to capture. The electricity consumption is higher in the colder months than during summertime. This is due to the need for heating in winter and cooling in summer months. Also, more electricity is used in the winter due to longer periods of darkness. The relation between other weather variables is not considered. The use of other weather variables could improve the prediction results.

The event of holidays, the day before a holiday, and the day after a holiday can impact the loads. For example, the load differs a lot between weekdays and weekends. Furthermore, not only the public holidays will affect the load. Also, school holidays, summer holidays, Christmas holidays, etc., will impact the load. In conclusion, further work with the holiday effect is important to satisfactorily predict the loads.

Parts of the autoregressive effect is reduced by adding the lags of the load and temperature from the previous hours. However, there is still some correlation. This is probably because lags of the first 8 days of the load are not included in the model. Perhaps the autocorrelation could be handled by a more advanced model like AR-FFNN, AR-GAM in [1].

The one model predictions or predictions of one trial hyperparameter tuning are not appropriate. There needs to be more than one trial tuning, or there needs to be ensemble predictions to get more stable predictions. Using an ensemble of models with different hyperparameters would reduce overfitting and increase the prediction ac-

curacy.

The rolling, load lags, and temperature variables are essential for the models. So, the choice of input variables is critical. The hidden layer sizes for each month are close, but the lowest MAPE scores are obtained when the hidden layer sizes are smaller. Relu seems to be the best activation function for the first hidden layer. The dropout is not chosen often, and similarly, the regularizations. This indicates that we might have an overfitting problem. This could be handled by increasing the trial number of the hyperparatmer tuning.

Chapter 6

Conclusion

This thesis aimed to investigate the impact of weather on short-term electricity load forecasting using feed forward neural networks (FFNN). The models described in this thesis focus on forecasts for eight days ahead.

The incorporation of temperature significantly improved forecasting accuracy. The temperature effect on the load is higher due to heating needs during the cold months. Also, high temperatures in the summer months lead to the use of cooling. The findings highlight that temperature lags and rolling statistical features, such as the rolling average of prior days' load, are critical predictors of electricity consumption.

This study extends the use of weather variables in load forecasting, where the temperature forecasts are directly used in the predictions, which is not common in the literature. Moreover, improved load forecasting helps energy operators control electricity demand much better, enhancing operational efficiency in power grids.

This thesis also identifies some fundamental limitations. The model fitting and predictions need refining. Reliance on single model approaches with a fixed set of hyperparameters can lead to instability in predictions. This has the potential for overfitting specific parameters, which results in inconsistent forecasts. The ensemble forecast could be a promising solution to handle this problem.

Adding a dummy variable to account for the impact of the New Year event enhanced the accuracy of predictions. It highlighted the significance of adjusting for calendar effects (such as public holidays, school breaks, and summer vacations) in energy demand forecasting.

While the results are promising for further enhancement, there are still chances to improve them by considering factors like other weather variables (humidity, wind speed, and solar radiation) and using more advanced models like Autoregressive Neural Networks (AR-FFNN) or Generalized Additive Models (AR-GAM) to handle the autoregressive effect in the data.

In summary, this work demonstrates the significant impact of weather on electricity load in the short term and provides a roadmap for integrating weather information into electricity load forecasting models. This is beneficial for system operators, energy management systems, and operational planning of power systems. It leads to the improvement of energy forecasting techniques, making them more precise and efficient, and thereby reinforcing the value of this work in the energy industry.

Bibliography

[1] M. Zimmermann and F. Ziel, "Efficient mid-term forecasting of hourly electricity load using generalized additive models," *Chair of Environmental Economics, esp. Economics of Renewable Energy, University of Duisburg-Essen*, 2018.

[2] V. M. A. Pardo and E. Valor, "Temperature and seasonality influences on spanish electricity load," *Energy Econ.*, vol. 24, no. 1, pp. 55–70, 2002.

[3] S. Fan and R. Hyndman, "Short-term load forecasting based on a semi-parametric additive model," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 134 – 141, 2012.

[4] C. E. P. H. S. Hippert and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, Feb. 2001.

[5] E. A. Feinberg and D. Genethliou, *Load Forecasting*. Boston, MA: Springer US, 2005, pp. 269–285. [Online]. Available: https://doi.org/10.1007/0-387-23471-3_12

[6] P. Narayan and R. Smyth, "Electricity consumption, employment and real income in australia evidence from multivariate granger causality tests," *Energy Policy*, vol. 33, no. 9, pp. 1109–1116, 2005. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:enepol:v:33:y:2005:i:9:p:1109-1116

[7] L. G. Swan and V. I. Ugursal, "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 8, pp. 1819–1835, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032108001949

[8] J. Sharp, "Comparative models for electrical load forecasting: D.h. bunn and e.d. farmer, eds.(wiley, new york, 1985) [uk pound]24.95, pp. 232," *International Journal of Forecasting*, vol. 2, no. 2, pp. 241–242, 1986. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:intfor:v:2:y:1986:i:2:p:241-242

[9] J. V. Paatero and P. D. Lund, "A model for generating household electricity load profiles," *International Journal of Energy Research*, vol. 30, no. 5, pp. 273–290, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.1136

[10] H. Cai and F. Li, "Distribution network planning considering demand response and reliability," *IEEE Transactions on Power Systems*, vol. 26, no. 2, pp. 1113–1121, 2011. [Online]. Available: https://doi.org/10.1109/TPWRS.2010.2059057

[11] L. Lin, M. Luo, T. O. Chan, E. Ge, X. Liu, Y. Zhao, and W. Liao, "Effects of urbanization on winter wind chill conditions over china," *Science of The Total Environment*, vol. 688, pp. 389–397, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969719327160

[12] C. Tarmanini, N. Sarma, C. Gezegin, and O. Ozgonenel, "Short term load forecasting based on arima and ann approaches," *Energy Reports*, vol. 9, pp. 550–557, 05 2023.

[13] H. Hahn, S. Meyer-Nieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," *Eur. J. Oper. Res.*, vol. 199, pp. 902–907, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:34005461

[14] R. G. Brown, "Statistical forecasting for inventory control," 1960. [Online]. Available: https://api.semanticscholar.org/CorpusID:153020798

[15] R. Uhrig, "Introduction to artificial neural networks," in *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, vol. 1, 1995, pp. 33–37 vol.1.

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

[17] J. W. H. S. Y. C. G. Li, K. Sun and X. Zhu, "Short-term load forecasting based on the similarity of weather forecast trajectory," *Appl. Energy*, vol. 162, pp. 941–951, 2016.

[18] S. Sharif and J. Taylor, "Short-term load forecasting by feed-forward neural networks," vol. Al Ain, United Arab Emirates, 05 2000.

[19] N. Bashiri Behmiri, C. Fezzi, and F. Ravazzolo, "Incorporating air temperature into mid-term electricity load forecasting models using time-series regressions and neural networks," *Energy*, vol. 278, p. 127831, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544223012252

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

[21] K. U. T. Senjyu, H. Takara and T. Funabashi, "One-hour-ahead load forecasting using neural network," *IEEE Transactions on Power Systems*, vol. 17, no. 1, pp. 113–118, Feb. 2002.

[22] F. M. M. Kandil, S. Anis and S. El-Debeiky, "An efficient approach for short-term load forecasting using artificial neural networks," *Electric Power Systems Research*, vol. 78, no. 2, pp. 321–328, Feb. 2008.

[23] H. Mori and M. Nakayama, "A weather adaptive short-term load forecasting system using neural networks," in *1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, Oct. 1998, pp. 2140–2145.

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

[25] S. M. M. M. A. Haidar and D. Sutanto, "Short-term load forecasting using neural networks and wavelet transforms," in *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, Jul. 2008, pp. 1–6.

[26] S. Fan and L. Chen, "Short-term load forecasting based on an adaptive hybrid method," *IEEE Transactions on Power Systems*, vol. 21, no. 1, pp. 392–401, Feb. 2006.

[27] H. Mori and M. Nagai, "A hybrid intelligent approach for short-term load forecasting using self-organizing map and support vector machine," in *2007 IEEE Power Engineering Society General Meeting*, Jun. 2007, pp. 1–6.

[28] K. A. D. L. Marino and M. Manic, "Building energy load forecasting using deep neural networks," in *2016 IEEE IECON*, Oct. 2016, pp. 7046–7051.

[29] D. J. H. F. L. W. Kong, Z. Y. Dong and Y. Xu, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[30] S. Koochaki and S. Mirsaeidi, "Short-term load forecasting using deep neural networks for residential loads," in *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, Jun. 2018, pp. 1–5.

[31] P. L. Y. Chen and S. P. Chatʒis, "Short-term load forecasting using variational bayesian GRU networks," *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 163–173, Jan. 2020.

[32] M. G. X. Wang and K. Li, "Short-term load forecasting using time series analysis: A case study in china," *IEEE Access*, vol. 7, pp. 123 123–123 132, Aug. 2019.

[33] J. N. S. Ryu and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, p. 3, Jan. 2017.

[34] M. H. A. Rahman, A. S. K. Chowdhury and N. Hosseinzadeh, "Short-term load forecasting using multi-layer perceptron neural network model for an institutional building," *Energies*, vol. 13, no. 10, p. 2642, May 2020.

[35] X. Z. Y. Li, J. Li and Y. Zhang, "Short-term load forecasting based on attention mechanism and Bi-LSTM," in *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, May 2019, pp. 2091–2095.

[36] H. M. P. Bacher and H. A. Nielsen, "Online short-term solar power forecasting," *Solar Energy*, vol. 83, no. 10, pp. 1772–1783, Oct. 2009.

[37] C. T.-A. V. Dordonnat and F. Dejou, "Short-term electricity load forecasting with generalized exponential smoothing and functional spline models," *International Journal of Forecasting*, vol. 24, no. 4, pp. 566–587, Oct. 2008.

[38] A. Aggarwal and J. S. Vitter, "The input/output complexity of sorting and related problems," *Commun. ACM*, vol. 31, no. 9, pp. 1116–1127, 1988.

[39] H. Pothina and K. V. Nagaraja, "Artificial neural network and math behind it," in *Proc. Smart Trends Comput. Commun.*, C. S.-I. Y.-D. Zhang, T. Senjyu and A. Joshi, Eds. Singapore: Springer Nature, 2023, pp. 205–221.

[40] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[41] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2016.

[42] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 1990, pp. 211–217.

[43] F. B. C. N. C. Dugas, Y. Bengio and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2001, pp. 472–478.

[44] Y. B. Y. LeCun, L. Bottou and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[46] B. M. Wilamowski and H. Yu, "Improved computation for levenberg–marquardt training," *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 930–937, 2010.

[47] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *eurocomputing*, 2016.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:6628106

[49] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[50] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016. [Online]. Available: https://arxiv.org/abs/1607.06450

[51] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks," *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pp. 173–177, 2017.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[53] I. Nusrat and S.-B. Jang, "A comparison of regularization techniques in deep neural networks," *Symmetry*, vol. 10, no. 11, 2018. [Online]. Available: https://www.mdpi.com/2073-8994/10/11/648

[54] G. Marcjasz, M. Narajewski, R. Weron, and F. Ziel, "Distributional neural networks for electricity price forecasting," *Energy Economics*, vol. 125, p. 106843, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140988323003419

[55] C. Corneanu, M. Madadi, S. Escalera, and A. Martinez, "Explainable early stopping for action unit recognition," pp. 693–699, 2020.

[56] N. Citroen, M. Ouassaid, and M. Maaroufi, "Long term electricity demand forecasting using autoregressive integrated moving average model: Case study of morocco," *2015 International Conference on Electrical and Information Technologies (ICEIT)*, pp. 59–64, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:15075331

[57] R. I. Hamilton and P. N. Papadopoulos, "Using shap values and machine learning to understand trends in the transient stability limit," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 1384–1397, 2024.

[58] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2017, pp. 4768–4777.

[59] G. Hooker, L. Mentch, and S. Zhou, "Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance," *Statistical Computing*, vol. 31, no. 82, 2021.

[60] S. L. Chau, R. Hu, J. Gonzalez, and D. Sejdinovic, "Rkhs-shap: Shapley values for kernel methods," *arXiv preprint arXiv:2110.09167*, 2022.

[61] "Google colaboratory: Frequently asked questions," https://research.google.com/colaboratory, accessed: [From July 2024 to August 2024].

[62] P. S. Foundation, "Python3: A programming language for general purpose programming," https://www.python.org/, 2023.

[63] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019. [Online]. Available: https://arxiv.org/abs/1907.10902

[64] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[65] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous systems," https://www.tensorflow.org/, 2015.

[66] Scikit-learn Developers, *RobustScaler — scikit-learn 1.5.2 documentation*, 2024, available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

[67] F. Chollet *et al.*, "Keras: The python deep learning library," https://keras.io, 2015.

[68] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," pp. 357–362, 2020.

[69] pandas development team *et al.*, "pandas-dev/pandas: Pandas," Feb 2020. [Online]. Available: https://pandas.pydata.org/

[70] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[71] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2023. [Online]. Available: https://www.R-project.org/

[72] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. [Online]. Available: https://ggplot2.tidyverse.org

[73] R. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen, *forecast: Forecasting functions for time series and linear models*, 2024, r package version 8.23.0. [Online]. Available: https://pkg.robjhyndman.com/forecast/

[74] OpenAI, "Chatgpt (version 4.0)," https://chat.openai.com, 2023, accessed: [Insert Date].

[75] ENTSO-E, "Entso-e transparency platform api documentation," https://transparency.entsoe.eu/content/static_content/Static%20content/web%20api/Guide.html.

[76] Deutscher Wetterdienst, "[title of the specific data or service]," https://www.dwd.de/.

[77] L. Dubus, "Weather sensitivity of electricity demand in france," *Electric Power Systems Research*, vol. 80, no. 11, pp. 1367–1374, 2010.

[78] A. Boissonnade and S. Heid, "The impact of temperature on french residential electricity consumption," *Energy Policy*, vol. 129, pp. 1251–1261, 2019.

[79] S. Pangsy-Kania, J. Biegańska, A. Sokół (Sokol), and F. Flouros, "Heating and cooling degree-days vs climate change in years 1979-2021. evidence from the european union and norway, economics and environment 1 1(88) • 2024," vol. 1, pp. 1–25, 03 2024.

[80] G. Marcjasz, M. Narajewski, R. Weron, and F. Ziel, "Distributional neural networks for electricity price forecasting," *Energy Economics*, vol. 125, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.eneco.2023.106843

[81] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, pp. 448–456. [Online]. Available: http://jmlr.org/proceedings/papers/v37/ioffe15.pdf

[82] J. Berrisch, M. Narajewski, and F. Ziel, "High-resolution peak demand estimation using generalized additive models and deep neural networks," *Energy and AI*, vol. 13, p. 100236, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546823000083

Chapter A

Appendix

A.1 Adam Optimization Algorithm

The Adaptive Moment Estimation (Adam) [48] algorithm updates the model's parameters by maintaining and utilizing running averages of both the first moment and the second moment. The following steps outline the algorithm:

Steps of the Adam Algorithm:

1. Initialization:

- Parameters: Initialize the parameters (weights) θ_0.
- First moment vector: $m_0 = 0$ (a moving average of the gradients).
- Second moment vector: $v_0 = 0$ (a moving average of the squared gradients).
- Hyperparameters:
  - Learning rate: $\alpha = 0.001$
  - Exponential decay rates: $\beta_1 = 0.9$, $\beta_2 = 0.999$
  - Small constant for numerical stability: $\epsilon = 10^{-8}$

2. Iteration (for each time step $t = 1, 2, \ldots$):

- Compute the gradient of the loss function $J(\theta)$ with respect to the parameters at step t:

$$g_t = \nabla_\theta J(\theta_{t-1})$$

- Update the biased first moment estimate:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

- Update the biased second moment estimate:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

- Compute bias-corrected first moment estimate:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

- Compute bias-corrected second moment estimate:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- Update the parameters:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

3. **Repeat:** Continue the iteration until the algorithm converges or a stopping criterion is met.

A.2 SHAP Figures

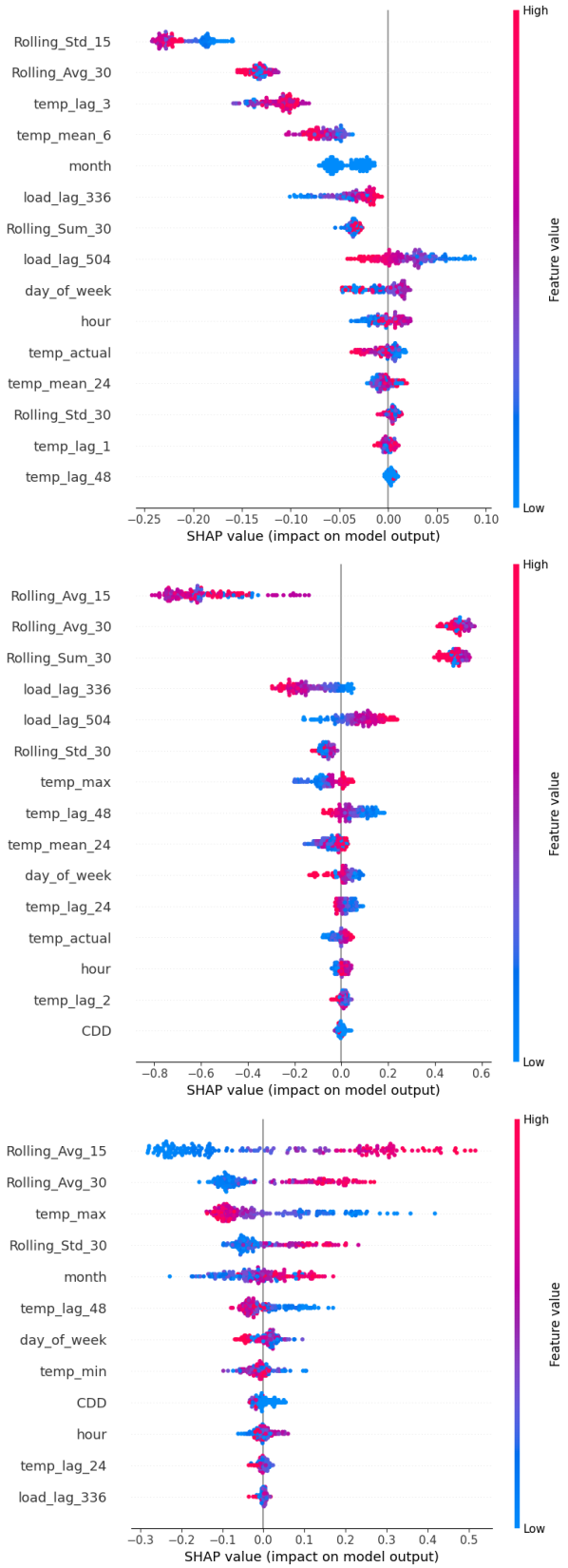

Figure A.1.: SHAP VALUES for September(up), October(mid), November(down)

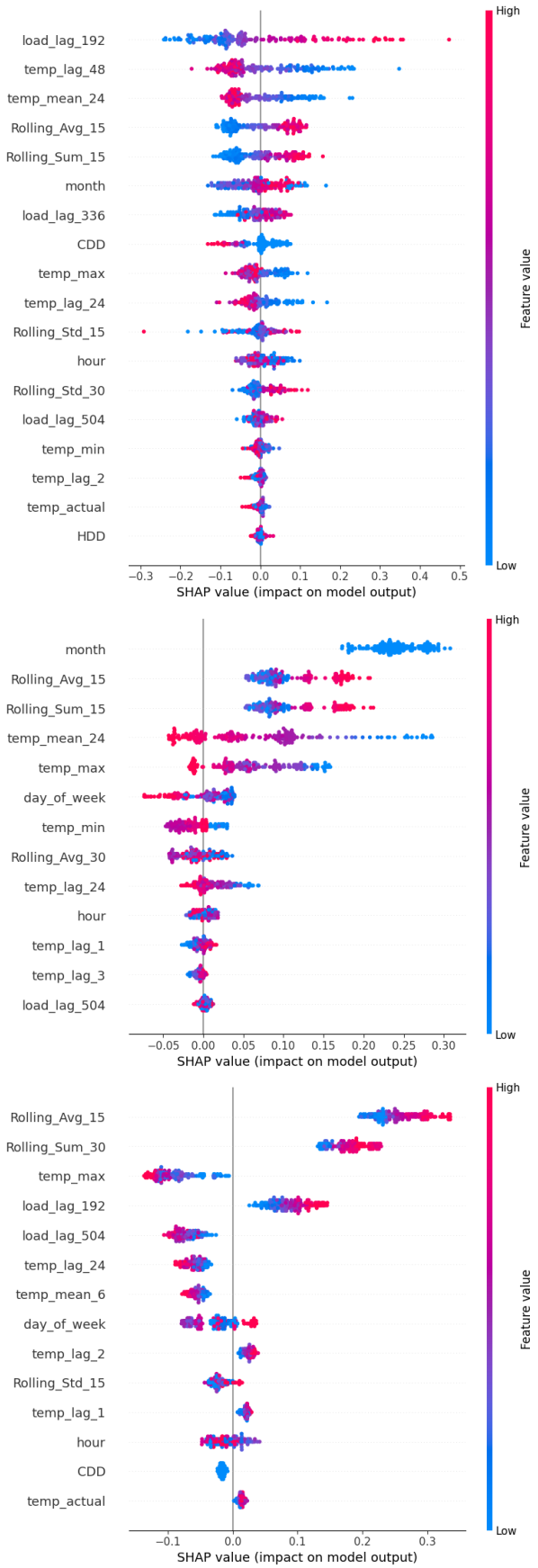

Figure A.2.: SHAP VALUES for December(up), January(mid), February(down)

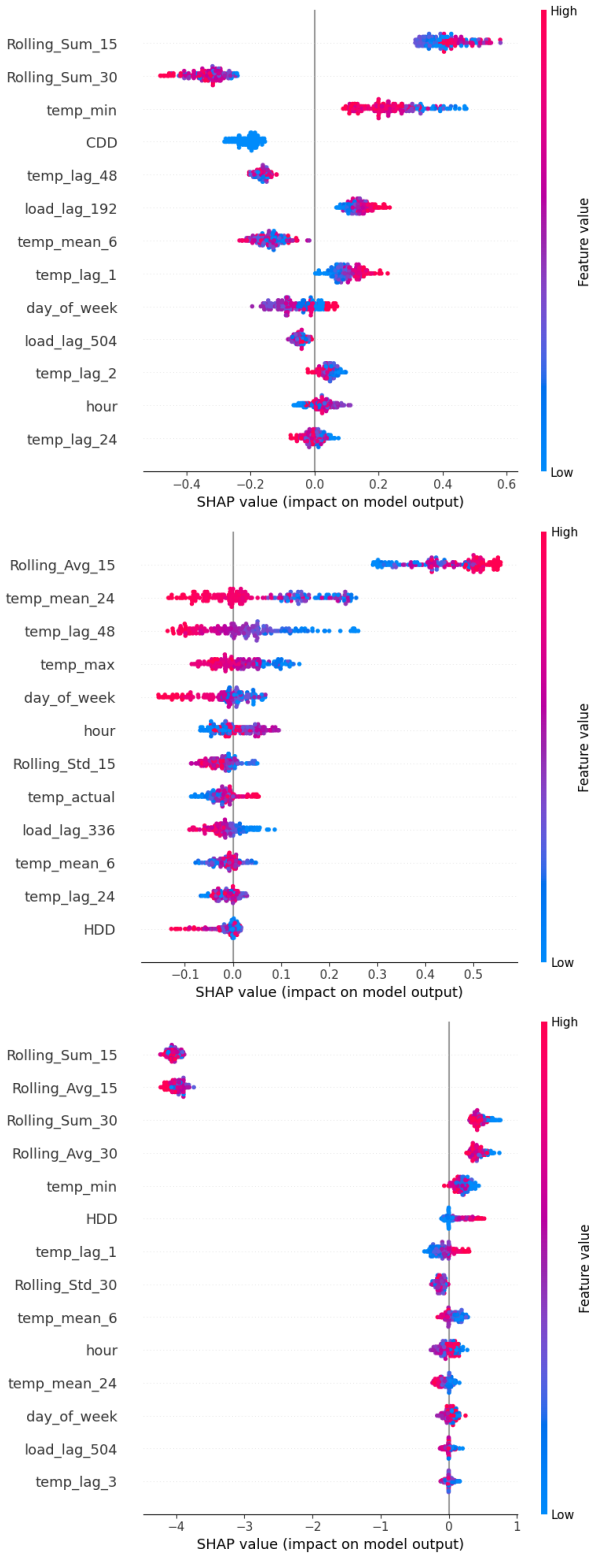

Figure A.3.: SHAP VALUES for March (up), April(mid), May(down)

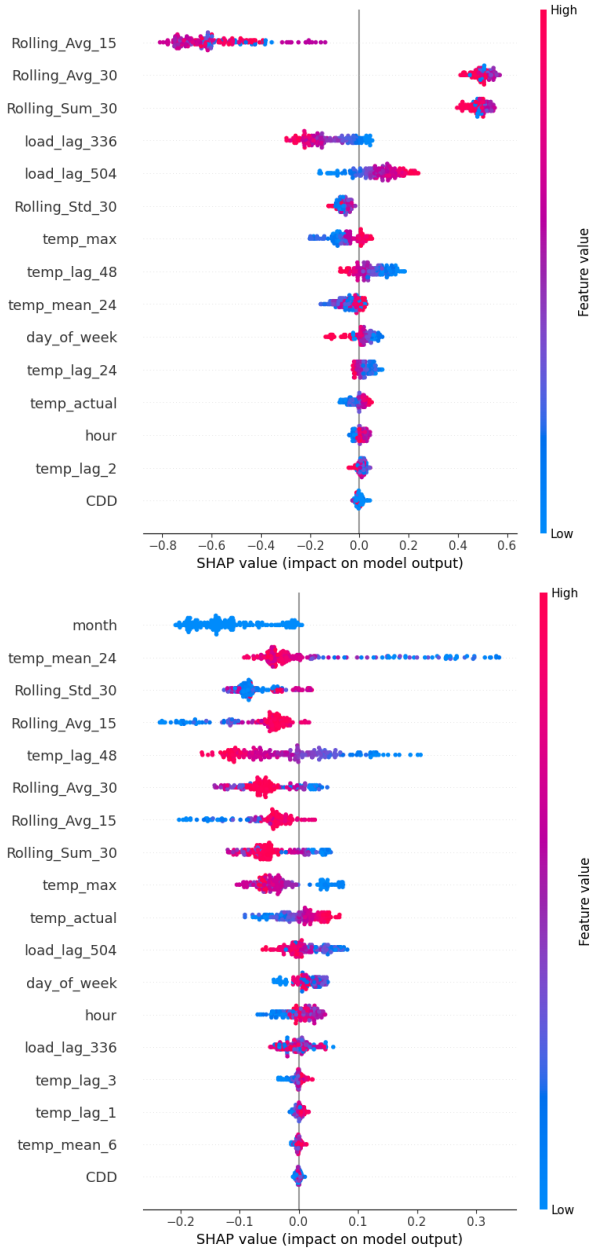

Figure A.4.: SHAP VALUES for June (up), July(down)