Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

Eilert Henriksen

Development of an XAI-Based Residential Load Forecasting Model

Master's thesis in Energy and Environmental Engineering
Supervisor: Ümit Cali
Co-supervisor: Ugur Halden
June 2022

NTNU
Kunnskap for en bedre verden

Eilert Henriksen

Development of an XAI-Based Residential Load Forecasting Model

Master's thesis in Energy and Environmental Engineering
Supervisor: Ümit Cali
Co-supervisor: Ugur Halden
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

NTNU

Kunnskap for en bedre verden

Preface

This master's thesis is the second part of the work on this topic performed for the Department of Electrical Power Engineering at NTNU. The first part was a specialization project [1], with a mainly theoretic approach. It is therefore assumed that the reader is familiar with underlying concepts of power engineering and machine learning. The master's is a natural continuation, where the theory is applied to a household in the south of Norway.

I want to thank my classmates and family for their support throughout the period. I also want to thank my supervisor, Ümit Cali, who was always available with meaningful insights. Finally, I want to extend my gratitude to my co-supervisor, Ugur Halden, who motivated me and pointed me in the right direction.

Eilert Henriksen, Trondheim 2022

Abstract

The ever-increasing complexity in the power system has introduced a higher demand for forecasting to keep the grid stable. Load forecasting has been an integral part of planning and maintenance by power system operators for both short and long horizons. Due to lacking technology, load forecasting has mainly been applied at the regional level. However, the revolution in sensor technology and data processing for machine learning has also enabled the investigation of residential load forecasting. The current practice within machine learning consists of black box models, which are highly complicated, giving little insight and reliability. *Explainable artificial intelligence* aims to solve this by allowing domain experts and others to understand the choices of the model.

This thesis aims to develop an hour-ahead load forecasting model for a residential home using explainable artificial intelligence. Multiple LSTM and CNN-LSTM models were proposed with a foundation in theory. During the development phase, the explainable artificial intelligence tool *SHapley Additive exPlanations* was used to investigate and increase performance by feature evaluation. Additionally, anomalies and outliers were examined in hopes of obtaining insight into the behavior and a greater understanding of the model environment. It was found that the use of explainable artificial intelligence significantly improved the model's performance and gave an indication of which features to include and omit. Surprisingly, the LSTM model outperformed the CNN-LSTM hybrid models. Moreover, including regional load forecasting further enhanced the model. Finally, it was found that including too many features limited performance.

The improvements and increased insights provided by explainable artificial intelligence found in this thesis suggest that there is a potential for explainable artificial intelligence to be a fundamental path toward trustworthy artificial intelligence. However, to reach its potential, more research is needed.

Sammendrag

Den stadig økende kompleksiteten i kraftsystemet har introdusert et større behov for prognoser for å holde nettet stabilt. Lastprognoser har vært en avgjørende del av planlegging og vedlikehold gjort av kraftsystemoperatører, på både kort og lang sikt. På grunn av manglende teknologi har lastprognoser i hovedsak blitt brukt på regionalt nivå. Imidlertid har revolusjonen innen sensorteknologi og databehandling for maskinlæring også muliggjort utviklingen av lastprognoser for boliger. Dagens praksis innen maskinlæring består av *black box* modeller, som er svært kompliserte og gir lite innsikt og pålitelighet. Forklarlig kunstig intelligens har sett en økt i utvikling, slik at domeneeksperter og andre kan forstå valgene til modellen.

Denne oppgaven tar sikte på å utvikle en lastprognose-modell for strømforbruk en time frem i tid, for et bolighus ved å bruke forklarbar kunstig intelligens. Flere LSTM- og CNN-LSTM-modeller ble foreslått basert på et teoretisk grunnlag. Under utviklingsfasen ble det forklarbare kunstige intelligensverktøyet *Shapley additive explanations* brukt til å undersøke og øke ytelsen ved funksjonsevaluering. I tillegg ble anomalier og feilvurderinger undersøkt i håp om å få innsikt i atferden og en større forståelse av modellen og dens omgivelser. Det ble funnet at bruken av forklarbar kunstig intelligens forbedret modellens ytelse betydelig og ga innsikt om hvilke funksjoner som skulle inkluderes og utelates. Overraskende nok presterte LSTM-modellene bedre enn CNN-LSTM hybridmodellene. Dessuten ble modellen ytterlige forbedret ved å inkludere regionale lastprognoser. Det ble og konkludert ved hjelp av forklarbar kunstig intelligens at de best ytende modellene var de med et færre antall innputvariabler.

Forbedringene og økt innsikt gitt av forklarbar kunstig intelligens funnet i denne oppgaven antyder at det er et potensial for forklarbar kunstig intelligens til å være en grunnleggende vei mot *pålitelig kunstig intelligens*. For å nå sitt potensiale vil det være nødvendig med mer forskning innen temaet.

Table of Contents

List of Figures vii

List of Tables ix

1 Introduction 1

2 Background Theory 4

2.1 Electric Loads and Generation 4

2.1.1 Industrial and Regional Loads 4

2.1.2 Power Production 6

2.1.3 Local Loads 6

2.1.4 Different Categories of Electrical Household Appliances 7

2.2 Statistical Methods 8

2.2.1 Autocorrelation 9

2.2.2 Stationarity 9

2.2.3 Linear Regression 10

2.3 Artificial Intelligence and Machine Learning 10

2.3.1 Supervised Learning for Regression 11

2.3.2 Artificial Neural Networks and MultiLayer Perceptron 12

2.3.3 Activation Functions 14

2.3.4 Back-propagation 16

2.3.5 Optimizers for Gradient Descent 17

2.3.6 Vanishing Gradient 19

2.4 Recurrent Neural Networks 19

2.5 Long Short-Term Memory 20

2.6 Convolutional Neural Networks 22

2.7 Explainable Artificial Intelligence 22

2.7.1 Local Interpretable Model-agnostic Explanations 23

2.7.2 SHapley Additive exPlanations . . . 24
2.8 Energy Forecasting . . . 25
2.8.1 Wind Power Forecasting . . . 25
2.8.2 Photo-Voltaic Power Forecasting . . . 26
2.8.3 Electrical Load Forecasting . . . 27
2.9 Forecast Performance . . . 28
2.9.1 Root Mean Square Error . . . 28
2.9.2 Mean Absolute Error . . . 29
2.9.3 Mean Absolute Percentage Error . . . 29
2.9.4 R-squared . . . 29

3 Methodology **31**
3.1 Experimental Setup . . . 33
3.2 Data Collection . . . 33
3.2.1 Residential Load Data . . . 33
3.2.2 Regional Load Data . . . 35
3.2.3 Numerical Weather Prediction . . . 37
3.2.4 Data Visualization . . . 38
3.2.5 Data Changes . . . 40
3.3 Data Preparation . . . 40
3.3.1 Quantitative Variables . . . 40
3.3.2 Qualitative Variables . . . 41
3.4 Model Development . . . 42
3.4.1 Models with Load and NWP . . . 44
3.4.2 Models with Qualitative Variables . . . 44
3.4.3 Models with Regional Prediction Data . . . 45
3.5 SHAP Development . . . 45
3.6 Choice of Performance Measures . . . 46

4 Results and Discussion **48**
4.1 Load and NWP . . . 48
4.2 Qualitative Variables . . . 50
4.2.1 SHAP Explanations . . . 53
4.3 Regional Prediction Data . . . 54
4.3.1 SHAP Explanations . . . 56
4.4 Local Explanations . . . 58

4.4.1 Single Day Explanation 58
4.5 Discussion of Results 61

5 Conclusion and Future Work **63**

Bibliography **65**

Appendix **72**

A Model Hyperparameters **73**

List of Figures

1.1 Relationship between interpretability and accuracy for different statistical and machine learning models [12]. . . . 3

2.1 Net energy consumption in Norway, 2020. Excluding raw materials [5]. . . . 5

2.2 Different load types over time [27]. . . . 8

2.3 Different ML classes and their area of use. . . . 11

2.4 Three linear regression models with polynomial features creating three different functions [38]. . . . 12

2.5 Illustration of connections between neurons in an ANN with weights and biases. . 14

2.6 Some of the most commonly used activation functions. . . . 15

2.7 Visual example of how RNNs store sequence data for each time step. . . . 20

2.8 LSTM cell with gates, activation functions and states, inspired by [54]. . . . 21

3.1 The proposed methodology for the thesis. . . . 32

3.2 Energimerking of the detached house. Y-axis is energy grade and X-axis is heating grade. Calculated from [79]. . . . 34

3.3 Electrical consumption with different time spans. . . . 35

3.4 Overview of transmission lines in Norway. Adopted from [83]. . . . 36

3.5 Regional loads by price zone. . . . 37

3.6 Standardized regional and residential load for week in February. Load imply residential load, whilst NO1 the load of price zone NO1. . . . 38

3.7 Average consumption each hour in 2020. . . . 39

3.8 An excerpt of temperature, humidity and pressure from a week in August. . . . 39

3.9 Average monthly residential load, regional load in NO1 and temperature, All have been standardized between 0 and 1. . . . 40

4.1 Comparison of the best and worst predicted weeks according to MAPE. . . . 49

4.2 Performance per month for the second generation of models. . . . 50

4.4 Summary plot of model LSTM-B1. . . . 52

4.3 Comparison of LSTM-B1 model and CNN-LSTM-B1 model for second generation models. . . . 52

4.5 Summary plot of model CNN-LSTM-B1. 54
4.6 Regional forecast compared to actual load, from January 2019 - December 2021. . 55
4.7 Excerpt from results using model C2. 56
4.8 Summary plot of LSTM-C1. 57
4.9 Summary plot of LSTM-C2. 58
4.10 Decision plot for B1 models and load profile for December 9^{th}. 59
4.11 Force plot for 12^{th} December 22:00. 61

List of Tables

3.1 Statistics of the residential load data. 35

3.2 Excerpt of NWP data format an afternoon in January. 38

3.3 Example of how seasons are one-hot encoded. 42

3.4 Hyperparameter space for all proposed models. the red color signify the CNN layer which is only used for the hybrid models. 43

4.1 Performance measures for first generation models. The best performing measures are highlighted. 48

4.2 Performance of LSTM-A1 for each weekday. 49

4.3 Performance measures for second generation models. 50

4.4 Performance of LSTM-B1 and CNN-LSTM-B2 for each weekday. 51

4.5 Performance measures for third generation models. 55

4.6 Monthly results for C2. 55

A.1 Chosen hyperparameters for an assortment of models. Be aware due to a error some models were lost. 74

List of Abbreviations

ACF	Autocorrelation Function
ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ALM	Appliance Load Monitoring
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
CNN	Convolutional Neural Network
DLT	DayLight Time
DNN	Deep Neural Network
DNV	Det Norske Veritas
DSO	Distribution System Operator
ETS	Emissions Trading Systems
FMI	Finnish Meteorological Institute
FVU	Fraction of Variance Unexplained
GSHP	Ground Source Heat Pump
ILM	Intrusive Load Monitoring
LCOE	Levelized Cost Of Energy
LIME	Local Interpretable Model-agnostic Explanations
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MEPS	MetCoOp Ensemble Prediction System
MET	Norwgian Meteorological Institute
MetCoOp	Meteorological Cooperation on Operational
ML	Machine Learning
MLP	MultiLayer Perceptron
MSE	Mean Square Error
NAG	Nesterov Accelerated Gradient
NHO	Næringslivets Hovedorganisasjon - The Confederation of Norwegian Enterprise
NILM	Non-Intrusive Load Monitoring
NVE	Norges vassdrags- og energidirektorat - The Norwegian Water Resources and Energy Directorate
NWP	Numerical Weather Prediction
OLS	Ordinary Least Square
PACF	Partial Autocorrelation Function
PV	Photo-Volatic
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
R^2	Coefficient of Determination
RSS	Residual Sum of Squares
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanations
sMAPE	Symmetric Mean Absolute Percentage Error
SMHI	Swedish Meteorological and Hydrological Institute
STD	Standard Deviation
SVM	Support Vector Machine
SVR	Support Vector Regression
TSO	Transmission System Operator
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

The constant progression in the field of electric power engineering and leaps made in computer science has not only led to new exciting ideas, but new challenges have surfaced as well. Furthermore, it has become evident with global warming that new, intelligent solutions are crucial to complete the green shift. The electricity demand is expected to increase by 2050 [2], with Norwegian power consumption estimated to increase by 23 TWh [3]. The new energy sources penetrating the market, such as wind and solar, are weather dependent, meaning they are more unreliable. An increasingly loaded power system supplied by renewable energy will lead to vulnerabilities. Statnett, the Norwegian Transmission System Operator (TSO), has in their investments plans for 2030 an estimated $60 - 100$ *billion NOK* earmarked for grid investments [4]. The Norwegian government has also created a new operating cost model, taking effect on July 1^{st}. This model aims to incentivize distributing the load throughout the day, counteracting peaks. Power peaks are unfortunate as they increases the stress, leading to higher losses and wear to components [1].

In Norway, households are responsible for about 22% of the total energy use, comprised mostly of electricity [5]. The aggregated power consumption of households is especially susceptible to power peaks due to people having similar living patterns. Formerly, little planning was done in regards to power consumption in households. This is because electricity is viewed as readily available at the "push of a button", and changes use of electricity should not burden the consumer. However, introducing "smart technology" and automatization, can aid planning and peak shaving. This is apparent with the recent Electric Vehicle (EV) evolution. If every car owner were to charge their car simultaneously, the grid would be overloaded. However, by spreading the charging throughout the day, peaks are removed. This is also possible for other high demanding equipment, like heaters and dishwashers. On the production side, new compact solar and wind technology has made it relevant for consumers to become prosumers– namely, both producers and consumers.

With the aforementioned changes and challenges, there is an increasing need for forecasting to control production and consumption optimally. Today, forecasting is used actively by producers, TSOs and Distribution System Operators (DSOs), and power market operators, among others. However, due to multiple challenges, relatively little is done in residential load forecasting. First of all, local load forecasting demands accurate sensors due to the relatively small fluctuations in consumption becoming significant. Secondly, data storage and processing have improved massively over the last couple of years. And finally, for residential load forecasting to be useful, it needs to be scalable, which has until recently been expensive.

A comprehensive review by Debnath et al. [6] explored different types of forecasting methods used for energy forecasting. Historically, statistical models such as AutoRegressive Integrated Moving Average (ARIMA) have been important. ARIMA is still a valid option due to its simplicity compared to Deep Learning (DL) models. DL models utilizing neural networks usually outperform statistical methods, with Artificial Neural Networks (ANN) being used in 13.2 % of the models in the review [6]. The literature also mentions Support Vector Machines (SVM) and Long Short-Term Memory (LSTM), among the most used Artificial Intelligence (AI) methods [6], [7]. Hybrid models are becoming increasingly popular, often consisting of a statistical method and a DL method or multiple different DL methods. Aside from model type, selecting input variables, also called features, is crucial for performance. The choice of features is heavily affected by the forecasting horizon. For short-term load forecasting, such as hour-ahead predictions are reliant on historical loads data. Furthermore, Numerical Weather Predictions (NWP) and categorical data is often utilized in forecasts [7].

DL models, such as LSTM and ANN, are complex, with multiple interconnected layers. Introducing numerous features further complicates the model, making it impossible to understand the underlying decisions made by the model. Models which humans do not understand are called *black box* models. Due to not being explainable, black box models are controversial in medicine and other fields due to a lack of trust and responsibility [8]. Explainable Artificial Intelligence (XAI) partly solves this problem. It can be seen as a way of getting knowledge about the model, turning the black box into a *glass box* model. Figure 1.1 depicts different DL and statistical methods plotted against the accuracy and interpretability of the model. Red circles show the various methods, while the arrows and green dots indicate how XAI is expected to improve the models. XAI is one of four pillars of Trustworthy AI, an increasingly used measure for the usefulness of AI. The three other pillars are Responsible AI, Privacy Preserving AI, and Valid AI [8]. Within the energy field, XAI has been utilized for frequency control in power systems [9]. Zhang et al. [10] used the XAI library called SHapley Additive exPlanations (SHAP) on a reinforcement model for emergency control on power systems. A model for Photo-Voltaic (PV) forecasting was explained using different types of XAI-tools by Kuzlu et al. [11]. The paper compared Local Interpretable Model-agnostic Explanations (LIME), SHAP, and ELI5. The paper did not suggest any improvements to the model, nor did any other papers found in the literature.

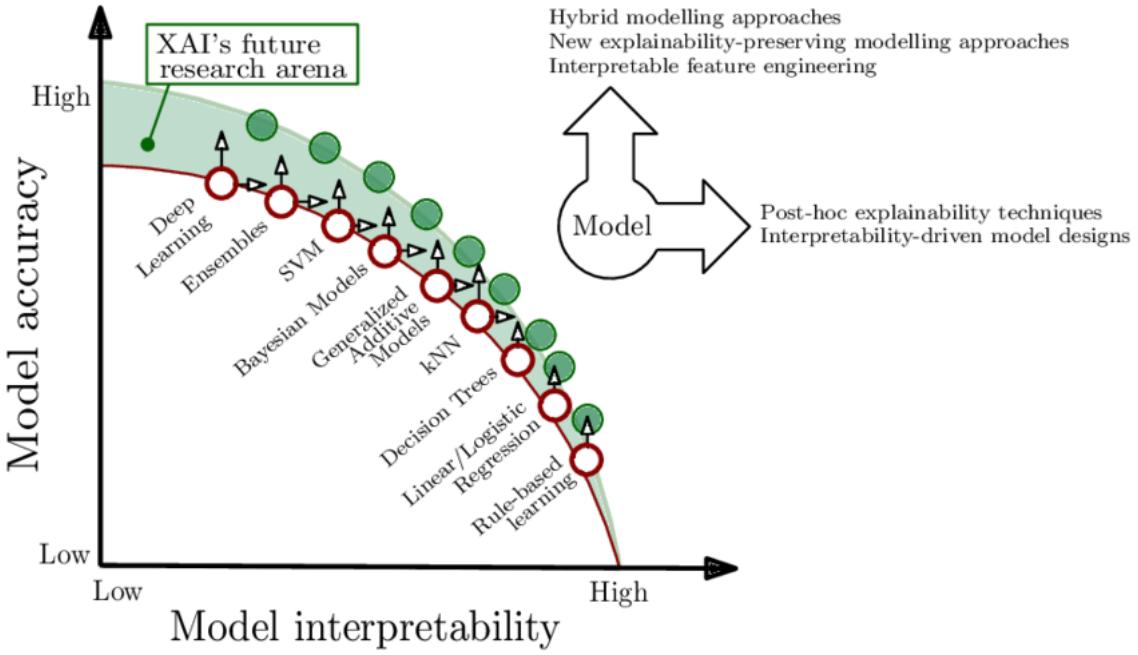


Figure 1.1: Relationship between interpretability and accuracy for different statistical and machine learning models [12].

With an increasing demand for load management, the use of forecasting will become critical in all stages of the power system. The uncertainty of black box models is one of the leading pitfalls of today's technology. Obtaining clarity can lead to higher reliability, and subsequently, it can be applied more extensively. This thesis sets out to explore the possibilities of XAI within energy forecasting and will try to answer: **how can the application of XAI-tools improve and gain insights into the inner workings of black box models as a step toward trustworthy AI?**

This master's thesis is structured into three main parts. The first part consists of researching the current environment of relevant topics. Additionally, data is collected and investigated during this segment to gain insight ahead of the modeling stage. The second part involves developing an hour-ahead load forecasting model for a residential house using different measures. Two different approaches are used. The first type is an LSTM model, and the second is a hybrid approach consisting of a Convolutional Neural Network (CNN) hybridized with an LSTM model. The third part of the thesis concentrate on implementing SHAP to the different models in hopes of unlocking a deeper understanding of the models. This will, in turn, be used to try to improve the model further. Specifically, the thesis attempts to contribute the following:

- Development of hour-ahead electrical load forecasting models for a Norwegian detached house using LSTM and CNN
- Perform explanatory analysis with XAI-tools such as SHAP to gain insight and improve the forecasting model
- Conduct performance evaluation to attain the best performing model
- Obtain a greater understanding of the modeling environment for future forecasting problems

Chapter 2

Background Theory

2.1 Electric Loads and Generation

This section is inspired by the specialization work [1] and covers the electric loads from a regional to household perspective, in addition to a short section covering electric power production.

2.1.1 Industrial and Regional Loads

The long history of the process industry in Norway was and is heavily reliant on hydropower production. According to Støa et al. [13], the process industry uses about 35 TWh of annual hydropower production. The metallurgy industry and aluminum production are among the leading causes of emissions in the industry, releasing close to 6 million tons of $CO_2 - eq$, with the entire sector being responsible for 18% of Norway's emissions in 2017 [13]. To decrease these emissions, the aim is to become net zero by 2050 through direct and indirect electrification. Among the measures are more electric heating and hydrogen in electrolysis. Most of the electricity going into the oil production today is created on-site by gas turbines. However, during the last decade, the introduction of underwater cables providing electricity from onshore production has become more regular. It is expected that by 2025 7.5 TWh of energy use on Norwegian oil rigs will be covered by sea cables, equating to about five percent of the total power consumption of Norway [14]. It is expected that this will only increase towards 2040 as well.

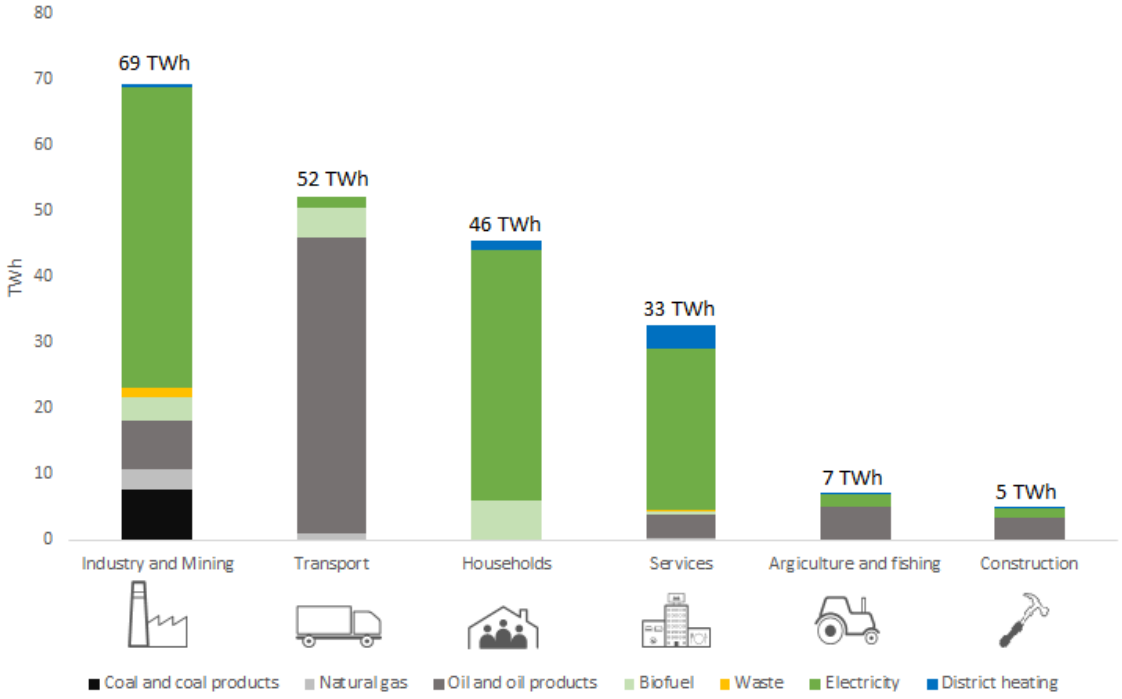


Figure 2.1: Net energy consumption in Norway, 2020. Excluding raw materials [5].

Sintef has, in collaboration with the Confederation of Norwegian Enterprise (NHO), proposed a road map for probable future industries in Norway [13]. Among these industries are battery and hydrogen production. Furthermore, the Norwegian Water Resources and Energy Directorate (NVE), in their report on *Expected electricity consumption towards 2040* [3], indicated that data centers can amass 5 TWh in Norway by 2040. Another study, performed by Det Norske Veritas (DNV) for the Norwegian government, indicated that producing $10,000$ *tons* $H_2/year$ would require 0.5 TWh [15]. As of today $225,000$ *tons* of grey hydrogen is produced in Norway. In addition, DNV estimated that by 2030 the maritime sector and transport would need an accumulated 46.9 *tons* of hydrogen annually, which translates to 2.3 TWh of electricity for hydrogen production [15]. Regarding battery production, as of 2021, multiple factories are planned for large-scale battery production. Among these are Morrow batteries in Arendal [16] and Freyr in Mo i Rana [17]. Large-scale battery production is energy-intensive. In a study by Kurland [18], producing 1 kWh of battery capacity requires $50 - 60$ kWh, excluding previous steps in the supply chain. Introducing battery production to Norway will establish a great need for energy [3].

From Figure 2.1, it is observed that the transport sector is mainly run on oil and oil products. In total, the transport sector is accountable for 60 % of emissions in the sectors not subject to regulation through the Emissions Trading System (ETS) [14]. There is currently a significant change to both personal transport as well as heavy duty transport. The Norwegian government has decided that by 2025 all new passenger cars, vans, and city busses will be electric or run on other clean sources [14]. By 2030 50 % of trucks and all heavy vans will be electric [14]. As of 2019, only 1 % of all transport consumption came from electric sources [5]. There are different predictions about much power on land transport will require in 2050, but NVE estimates about 14.5 TWh, whilst Energy Norway estimates upwards of 20 TWh [19].

Furthermore, the maritime and aviation industries are headed towards more electric consumption. There is work on short-distance electric ferries, and already some electric ferries in operation

[20]. Electrification of long-distance ships is, however, not feasible as of right now. A more viable solution is using other clean energy sources, such as the aforementioned hydrogen [15]. The aviation industry has similar struggles, where short distance flights are deemed feasible, but longer flights are challenging to accomplish. Avinor intends that all domestic flights in Norway are electric by 2040 [21].

The third largest energy consumer in Norway is households. Contrary to the transport sector, homes are mostly powered by electricity. In 2017, 83 % of the 47.6 TWh total energy use came from electricity. The primary consumers are heating, lighting, and electrical appliances. Over the last couple of years, the percentage of electricity used in this has increased. The two main reasons for this are increased electricity use for heating and more electrical appliances used in the home [5]. Even though the percentage is rising, NVE reports that the total consumption in the sector will remain stable, even decline by up to 1 TWh by 2040 [3]. Although the number of buildings in Norway is increasing, improved energy efficiency will counteract the need for more energy to buildings. Especially heating demand will most likely decrease as a result of improved insulation and warmer outdoor temperatures.

2.1.2 Power Production

Apart from hydropower, Norwegian generation consists of wind, thermal and solar. About 10 % of Norwegian production stems from wind energy. Today, wind energy is harvested on land by large-scale turbines. However, in the future, offshore wind power will most likely be a viable solution. As of today, the technology has not reached this point, but it is estimated that production will be ready by 2030 [22], [23]. The rest of the energy production stems from thermal and solar, covering 2 % and a negligible amount, respectively. According to NVE, solar power plants will be the cheapest option based on Levelized Cost Of Energy (LCOE) by 2030. The expected production price of PV panels will most likely drop, making it a viable option, even in Norway [24]. Furthermore, PV panels have the advantage of being scalable, making it versatile, from large solar plants to be roof mounted. A future power grid with an increased share of unreliable power sources will increase the need for better planning and use of available power.

2.1.3 Local Loads

NS3031 [25] is the Norwegian standard for rules and regulations for the calculation of energy performance of buildings. In this, the gross energy need for a household is divided into six categories:

1. Heating
    (a) Room heating
    (b) Ventilation heating
2. Hot water
3. Cooling
    (a) Room cooling
    (b) Ventilation cooling

4. Fans and pumps
    (a) Fans
    (b) Pumps
5. Lighting
6. El-specific equipment

The main contributor to the energy usage of Norwegian households is heating. About 78% of the energy demand is due to heating and hot water [2]. Norway, being situated far north, is heavily affected by seasonal changes. Especially during the winter, the need for heating is essential. Furthermore, in the winter, the number of sunlight hours diminishes, with some parts of Norway not seeing the sun for a short period of the year. This increases the need for lighting. With improved technology, the efficiency of heating and lighting has drastically reduced the energy demand. Where heating formerly was mainly provided from fireplaces and oil boilers, today, the major share of heating is supplied by electricity [26]. With the summers being temperate, there is little need for cooling or heating in residential buildings. Generally, the use of electrical appliances is increasing. This is particularly relevant for the car fleet as the number of EVs penetrating the market grow, which in turn increases the residential loads and peaks.

2.1.4 Different Categories of Electrical Household Appliances

The load profile of electrical appliances is affected by multiple parameters. One of the most predominant ones is user pattern, e.g., domestic hot water use is heavily influenced by the length of showers taken by the individual. Extensive research has been done to understand how different household appliances affect the general load of the resident. Among the most used Appliance Load Monitoring (ALM) methods are Non-Intrusive Load Monitoring (NILM) and Intrusive Load Monitoring (ILM). The difference between the methods is where and how many sensors are placed. NILM relies on a single measuring point, usually located on a smart meter. As this measurement will be the net consumption, one can subsequently define it as,

$$P(t) = \sum_{i=1}^{n} p_i(t). \tag{2.1}$$

From Equation (2.1), it is understood that each term in the sum represents an active electrical load at time *t*. NILM aims to provide an accurate breakdown of *P(t)*. This is accomplished through load signatures. A load signature is the distinct behaviour different appliances exhibit during use. For example, a heater will behave differently to an EV charger. These states can be assigned into four different types, as done by [27]. The following four states are (and depicted in Figure 2.2):

Type I: ON/OFF Type I appliances operate at only one load level, which can easily be recognized when used. Examples of ON/OFF equipment are non-dimmable lights.

Type II: Finite State Machines Finite state machines are similar to Type I appliances. However, finite state machines can operate at multiple different load levels based on different settings on the machine. A washing machine is an example of a Type II load in a household. During a

wash, it will require different amounts of power depending on what mode it is on and where in the washing cycle it is.

Type III: Continuous Variable Devices Equipment that draws different amounts of power depending on the operating state is recognized as continuous variable devices. These load signatures can be difficult to identify as they do not have a preset number of operating conditions, making them unpredictable. Charging of batteries falls into this category.

Type IV: Permanent Consumer Devices This category is self-explanatory as these devices are constantly on. Among these is equipment that draws small amounts of power, such as alarm sensors.

ILM, on the other hand, relies on multiple sensors located at different spots in the house. There are three different types of intrusive sensors depending on the location [28].

1. Submeters at circuit breaker level dividing the house into zones
2. Plug level sensors that monitor loads connected to each plug
3. Embedded sensors in appliances

An advantage to ILM is with reliant sensors, one can obtain more accurate measurements. On the other hand, ILM solutions are more expensive and require more overhead as sensors need to be connected to desired locations.

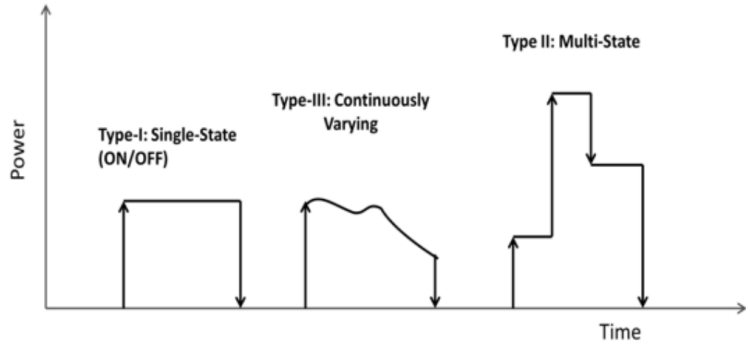


Figure 2.2: Different load types over time [27].

2.2 Statistical Methods

Statistical methods have been one of the drivers of advancements within computer science. At the same time, computer improvements have driven statistical methods to be faster and able to process more data. Statistics and statistical models provide insights and generalizations of large data sets. This section is adopted from the specialization project [1] due to being underlying math, which requires no new information.

2.2.1 Autocorrelation

Autocorrelation gives an indication of correlation between a dependent variable in a time series at two different intervals. Comparing a variable with a lagged version of itself gives insight into whether data has a degree of randomness. In time series modeling, the observed data is usually taken from one contributor rather than one data point from multiple contributors [29]. An advantage of autocorrelation is that it reveals trends and seasonality aspects of a time series model. Using the AutoCorrelation Function (ACF) one can measure the linear relationship between a value y_t and a value y_{t-k}, which lags the time t with k steps. The ACF is given by,

$$r_k = \frac{\sum_{t=k+1}^{n}(y_t - \hat{y})(Y_{t-k} - \hat{y})}{\sum_{t=1}^{n}(y_t - \hat{y})^2} \tag{2.2}$$

where $\hat{y}$ is the predicted value. Each factorial $(y_t - \hat{y})$ represents the residual or estimate of the error in the model, denoted e_t. The resulting autocorrelation coefficient r_k will be in the interval $[-1, 1]$, where high positives represent a high degree of correlation. By plotting a series of autocorrelation coefficients based on different degrees of lag, one obtains a correlogram that clearly depicts trends [30].

Another way of understanding the relationship between current values and lagged versions of itself is with the Partial AutoCorrelation Function (PACF). Autocorrelation includes the influence of the values between the present and the *kth* lagged value. This can make the actual correlation between y_t and y_{t-k} hard to identify. PACF, on the other hand, removes the effects of the lags $[1, k-1]$ [30]. The general expression of the PACF at lag k can be expressed as,

$$\phi_{kk} = Corr(y_t, y_{t-k}|y_{t-1}, y_{t-1}, ..., y_{t-k+1}). \tag{2.3}$$

This requires that the time series is normally distributed.

2.2.2 Stationarity

A time series model with the same joint distribution for $y_{t_1}, y_{t_2}, ..., y_{t_n}$ and $y_{t_1-k}, y_{t_2-k}, ..., y_{t_n-k}$ for all points in time and all lags k is called strictly stationary [31]. Weak stationarity, on the other hand, implies that a stochastic process $\{y_t\}$ has a constant mean function $\mu(t)$ and that the autocovariance $\gamma(t, h)$ is independent of time and all lags h. It is essential to know that strong stationarity does not imply weak stationarity. Forecasting relies on stationarity as many statistical tools require that the statistical properties not change over time. However, in the real world, stationarity is seldom. Among the most normal deviations from stationary models are trends and seasonality. Initially, it is important to get a sense of what kind of deviations are relevant for the given model. As mentioned in Section 2.2.1, a correlogram can be used to find trends and seasonality. An easy way of removing trends is by calculating the difference between subsequent observations, giving the change in observations. In some cases, it is also necessary to take the change of the change, resulting in second-order differencing. Removing seasonalities can be done similarly. However, instead of finding the difference in consecutive values, one finds the difference $y'_t = y_t - y_{t-m}$, where m is the number of lags to get the same season at a previous time [30].

2.2.3 Linear Regression

Linear regression models describe linear approximation of independent variables, x_i, and their relationship with the outcome. The goal of regression analysis is to create a regression model that predicts an outcome based on the independent variable and is often used in forecasting and projections. The result of this prediction is called the dependent variable and is denoted by $\hat{y}$. A simple linear regression model is written as $\hat{y} = \beta_0 + \beta_1 x + \epsilon$. The scalar value, β_0, is the intercept, and β_1 describes the slope of the regression line [32]. When having multiple independent variables for a scenario, one can create the following general formula

$$\hat{y} = \mathbf{X}\beta + \epsilon. \tag{2.4}$$

Here ϵ is the error term, which considers the influence of noise and other factors. When fitting a linear regression model, the goal is to minimize the error term, meaning $\epsilon = y - \mathbf{X}\beta$. It is important not to mix y and $\hat{y}$ as they are the actual and expected values, respectively. There are multiple ways to estimate the β, and it is therefore not necessarily a unique solution to the issue. Ordinary Least Squares (OLS) are among the estimators often used for linear regression. OLS takes advantage of the Residual Sum of Squares (RSS),

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - \hat{y}(x_i))^2. \tag{2.5}$$

Rewriting Equation (2.5) using the vector from Equation (2.4) one obtain the following equation,

$$RSS(\beta) = (y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta). \tag{2.6}$$

By minimizing Equation (2.6) it can be shown through some extra calculations (such as in Hastie et al. [33]) that one can obtain the unique solution,

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y. \tag{2.7}$$

2.3 Artificial Intelligence and Machine Learning

Artificial Intelligence originates from logic and biology, among others, and has become an integral piece of computer science over the last decade. The resurgence after the AI winter (A period with stagnated research on the topic) is highly owed due to the improved computational power and storage [34]. AI uses computers to replicate natural intelligence, i.e., human and animal-like intelligence. Machine Learning (ML) and AI are found in everything from finance to surveillance. AI is usually an umbrella term for all programs inspired by natural thinking, whereas ML utilizes statistics and different algorithms. DL is often seen as a subset of ML utilizing neural networks.

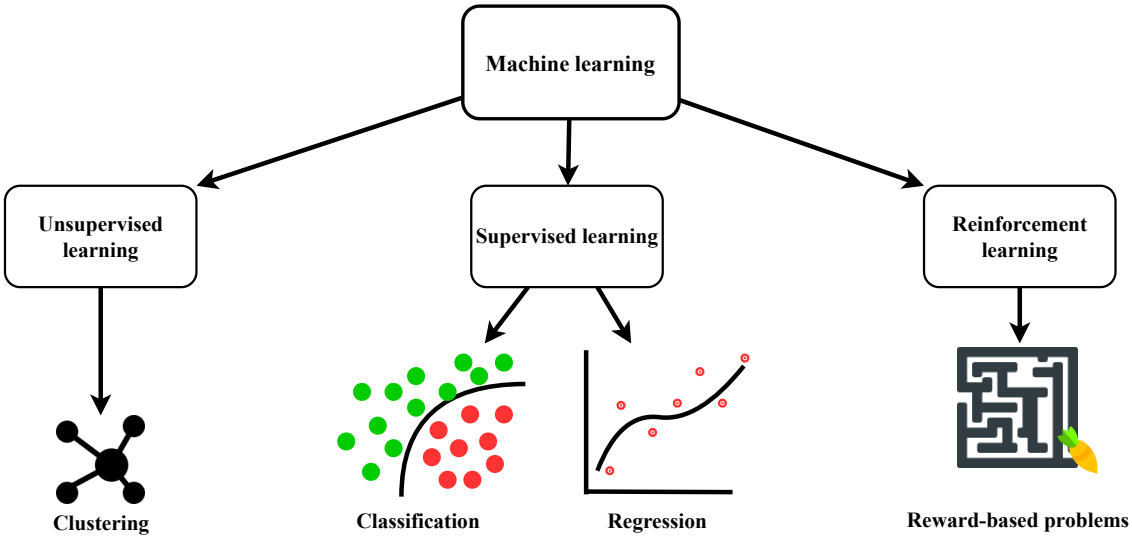


Figure 2.3: Different ML classes and their area of use.

Different ML problems require different solutions. For example, training an intelligent vacuum would need a different approach compared to a forecasting problem. The four predominant learning techniques in ML are supervised, unsupervised, semi-supervised, and reinforcement learning [35]. Supervised learning problems are given labeled data, often consisting of input and output, to train the model to replicate the desired output. In Figure 2.3, it is seen that supervised learning generally consists of classification and regression problems, where classification problems consist of sorting data into given categories. Unsupervised learning excels at pattern recognition without the help of an expert and is often used to sort data into multiple groups or clusters. Semi-supervised learning combines the techniques above [35]. Reinforcement learning, or reward-based learning, rewards or punishes the intelligent agent based upon how well it performs. For example, the intelligent vacuum could be given positive feedback for cleaning well. The following section will cover Supervised learning as this is the method used for forecasting problems.

2.3.1 Supervised Learning for Regression

The labeled data used in supervised learning is split into a training, a test, and often a validation set. The model will use the training set to learn different relationships between input and output. An unbiased opinion can be formed during tuning using a validation set. And finally, after tuning, the performance of the best model is checked with a test. There are some guidelines to how one chooses to divide the sets. **(1)** The test set needs to represent the general notion of the entire data set [36]. For example, for a regression problem over ten years, if there are significant changes in the behavior after the initial five years, it would not be wise to train for only those years. One way to get a realistic distribution is to shuffle the data before splitting. **(2)** DL algorithms require large data sets to learn. There are no distinct answers to how large the different sets need to be, but they need to be representative and statistically sound [37]. If there is a lot of available data, it is customary to use an 80/20 ratio for the training and test set. The validation set usually is a small portion of the training data. Less data typically indicate that a larger percentage of the data should be used for testing [37].

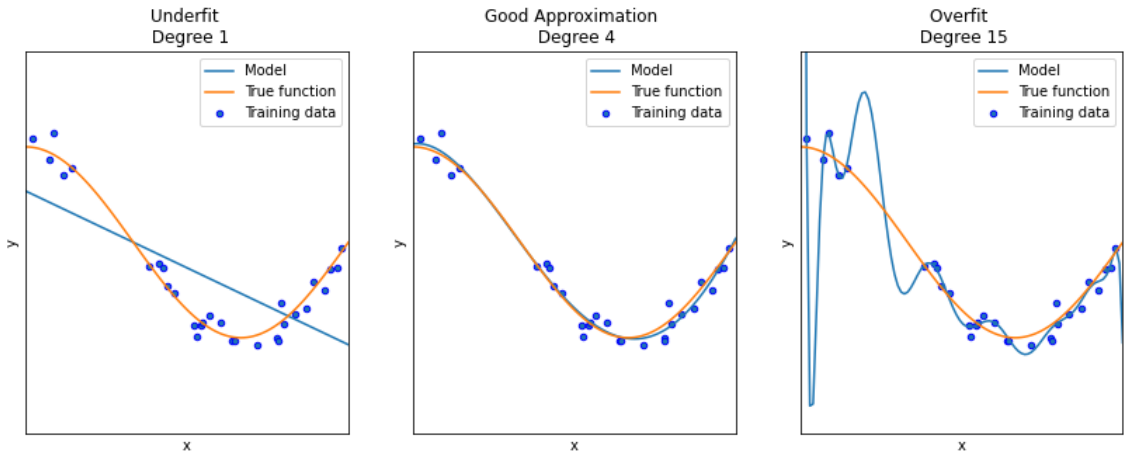


Figure 2.4: Three linear regression models with polynomial features creating three different functions [38].

Input and output can be paired into a set of, $[(x_1, y_1), ..., (x_n, y_n)]$. The input variables, also called features, can create a hypothesis space, $\mathfrak{H}$, with multiple functions h_i. Each of these functions resembles the unknown function $y = f(x)$. The supervised learning algorithm tries to find the function $h(\mathbf{x}; \theta) \approx f(\mathbf{x})$ [35]. However, even though an accurate model is realizable, it could be too advanced. This is often associated with overfitting, which is a model not resembling a natural pattern but rather an obscure model perfectly fitting the model. Such an advanced model will often result in poor performance in testing and use due to the testing set most likely not being identical to the training set. The opposite of an overfit model is an underfit model, which poorly fits the model. The two different versions and a good approximation are shown in Figure 2.4. Choosing the most important features is often challenging and requires a domain expert to structure the data.

- Neural networks
- Naive Bayes
- Regression
- SVM
- K-nearest neighbour
- Random forest

Above is a list of the most used supervised learning techniques [39]. For intricate regression problems, SVM and neural networks are highly used. In the following sections, two subsets of neural networks are described, namely MultiLayer Perceptron (MLP) and Recurrent Neural Networks (RNNs).

2.3.2 Artificial Neural Networks and MultiLayer Perceptron

The human brain is composed of millions of interconnected neurons, firing signals between each neuron. How strong each connection is and the signals passed in between are processed into

information flow. ANNs are built with the same structure. A neural network consists of multiple neurons or nodes[1] connected in multiple layers. Typically, neural networks are composed of three types of layers: an input layer, hidden layer(s), and an output layer. In vanilla ANNs, all neurons in neighboring layers are connected through directed links called edges. All edges have a weight, $w_{i,j}$, representing how strong the connection between nodes i and j is. This notation will be used in the following sections and superscripts denoting the current layer. The following description is of MLP, a popular subcategory of ANNs [40]. The structure of an MLP is divided into two stages, **(1)** forward propagation and **(2)** backward propagation [41]. The features are fed into the input layer during the first step, where each neuron corresponds to a feature. Following the input layer, the signals are sent through their edges into the first hidden layer. This process can be mathematically written as,

$$z_j^{(1)} = b + \sum_{i=0}^{n} w_{i,j} x_i. \tag{2.8}$$

Here, $z_j^{(1)}$ is the input into node j in the first hidden layer as indicated by the superscript. b is the bias vector, which can be compared to the constant in a linear model since it can move the output either left or right. Meanwhile, x_i is the output from the input layer. Before the signals are fed forward to the next layer, an activation function, $\sigma^k(z_j^k)$, is applied to map the linear function in Equation (2.8) into a non-linear function with the output, $a_j^{(k)}$ [37], [42]. This is illustrated in Figure 2.5 and Equation (2.9). An example of how a signal is sent through a network with three layers is $f(\mathbf{x};\theta) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$ where $f(\mathbf{x};\theta)$ is a mapping of the actual problem $\mathbf{y}$ using the weights and biases, θ [43]. Activation functions are elaborated in more detail in Section 2.3.3.

$$a_j^k = \sigma^k(z_j^k) \tag{2.9}$$

Usually, the same activation function is used on all neurons within the same layer, a_j^k is multiplied with the subsequent weights and fed to the next hidden layer. This process repeats itself for all hidden layers, meaning that all previous layers affect the current layer. Finally, the model is sent through the output layer, where it is transformed into understandable data, $\hat{y}_m$.

$$\hat{y}_m = \sigma_{out}(z_m^L) \tag{2.10}$$

Neural networks can easily be rewritten from neural form, shown in Figure 2.5 into matrix form shown in Equations (2.11a) and (2.11b) [41]. This formulation is easier to interpret for computers.

$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^k + \mathbf{b}^k) \tag{2.11a}$$

$$\mathbf{O} = \mathbf{H}\mathbf{W}^L + \mathbf{b}^L \tag{2.11b}$$

Here, $\mathbf{H}$ is the hidden layer matrix, $\mathbf{H} \in \mathbb{R}^{n \times h}$, n is understood as the number of examples in a *minibatch*, and h is the number of hidden neurons in the given layer. The input matrix is denoted as $\mathbf{X} \in \mathbb{R}^{n \times d}$, where d is features. Finally, the weight and bias matrices are dependent on whether it is in a hidden layer or the output layer. Bias can be written mathematically as $\mathbf{b} \in \mathbb{R}^{1 \times m}$, with m being the number of neurons, making $m = h$ for the hidden layers. The weights are given by a

[1]These will be used interchangeably

real number $\mathbf{W} \in \mathbb{R}^{u \times v}$, where u and v represent neurons in the previous and current layer. For the input layer $u = d$, and for the output layer v equals the number of output nodes.

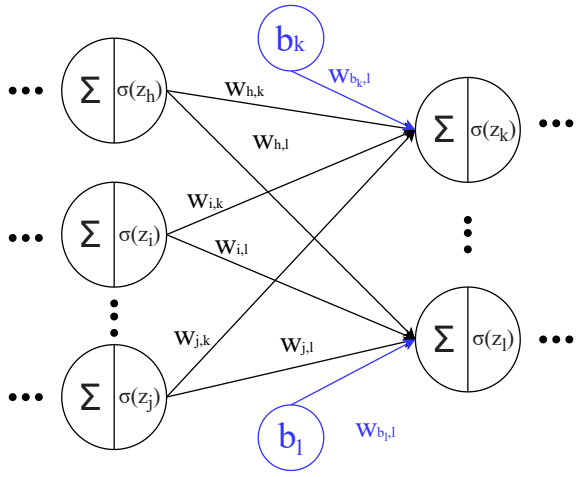


Figure 2.5: Illustration of connections between neurons in an ANN with weights and biases.

2.3.3 Activation Functions

From Equation (2.8), it is understood that the linearity would be kept in a network without activation functions. Furthermore, the output from each neuron would be in the range of $[-\infty, \infty]$. This could create instability in how much power each neuron would hold and is often restricted to given limits. Activation functions are introduced to remove these issues. Activation functions can aid in "activating" important data and contain less critical information by moving the output either to the left or right [44].

One of the most straightforward step functions is the **binary step**,

$$\sigma(z) = \begin{cases} 0 \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{cases} .$$

There are multiple issues with the binary step function. First of all, the derivative is 0 if $x \neq 1$ and undefined if $x = 0$. This can cause issues during back-propagation covered in Section 2.3.4. Additionally, it is often limited to binary classification problems. A more widely used activation function is introduced with the **sigmoid function** [42].

$$\sigma_{sigmoid}(z) = \frac{1}{(1 + \exp^{-z})} \tag{2.12}$$

In addition to being non-linear, the sigmoid is a bounded differentiable real function defined for all real inputs [45]. The output is between 0 and 1, as seen in Figure 2.6a, making it suitable for calculating probabilities [46]. The downfall of being in this range is it will never create a non-negative output. This can cause the output to propagate left and right. Furthermore, even though the sigmoid has a smooth gradient, the significant range is short, which leads to gradient saturation. The sigmoid struggles especially with deep networks, as presented by Nwankpa et al. [45]. To counteract the issues of the sigmoid the **Hyperbolic Tangent Function**, **Tanh** was

introduced.

$$\sigma_{tanh}(z) = Tanh(z). \tag{2.13}$$

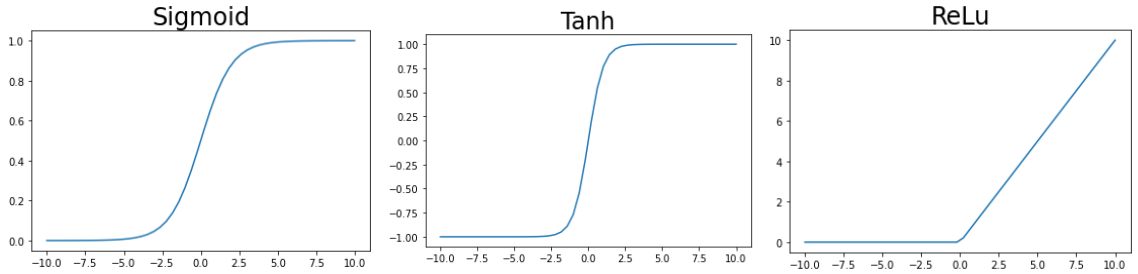
From Figure 2.6a and Figure 2.6b, it can be observed that they both possess the S-shape, but the tanh function is zero-centered. This aids during back-propagation. It has been found to improve training for neural networks with multiple layers [45]. Both the sigmoid and tanh functions struggle with a vanishing gradient. The vanishing gradient problem is a common issue for neural networks and will be covered in Section 2.3.6. There are activation functions counteracting this, and arguably one of the most used activation functions is the **Rectified Linear Unit function** or **ReLu** introduced by Nair et al. [47].

$$\sigma_{ReLu}(z) = max(0, z) \tag{2.14}$$

Initially, from Figure 2.6c, the ReLu can easily be misinterpreted as linear, which it is not. For values larger than 0, it does, however, behave linearly. Because of the simplicity of the ReLu, the computational time is quick compared to heavy mathematical operations found in sigmoid and tanh [46]. Nodes where input values are below zero will have an output of zero, as seen in Equation (2.14). This causes an issue called *the dying ReLu problem*. With the output of a neuron being 0 and the gradient being undefined, the node is at the chance of being deactivated. The weight accompanying that node will not update during back-propagation, leading to no update in the learning capacity of that node. **Leaky ReLu** was introduced in 2013 to fix the dying Relu problem. By including a small constant, α, turning the equation to,

$$\sigma_{LReLu}(z) = \alpha z + z = \begin{cases} z & if\ z > 0 \\ \alpha z & if\ z \leq 0 \end{cases}. \tag{2.15}$$

This change removes gradients from being zero. In the output layer, different activation functions are often used to fit the problem at hand better. Classification models usually use some variation of a sigmoid function or a Softmax function. Regression problems, on the other hand, use a linear activation function to get an unbounded value that represents the target value.



(a) The sigmoid function. (b) The tanh function. (c) The ReLu function.

Figure 2.6: Some of the most commonly used activation functions.

2.3.4 Back-propagation

During forward propagation, the network uses the weights and biases to compute the model's output. For regression problems in supervised learning, the goal consists of getting as close as possible to the actual value. Formulating this into an optimization problem, one tries to minimize the model's aggregated error. This is done using a cost function, also called the error function [48].

$$C = \frac{1}{2n} \sum_{x} (\|y(\mathbf{x}) - \hat{y}(\mathbf{x}; \theta)\|^2 \tag{2.16}$$

The cost function in (2.16) is often referred to as quadratic cost function or Mean Square Error (MSE). x is individual training examples in the total set $\mathcal{X}$ with n number of samples. For back-propagation, two assumptions are needed: **(1)** The cost function needs to be written as an average over all individual training samples, $C = \frac{1}{n} \sum_x C_x$. **(2)** The cost function needs to be able to be written as a function of outputs *cost* $C = C(a^k)$ [48]. MSE satisfies both, as seen below,

$$\textbf{(1)}\ C_x = \frac{1}{2}\|y - \hat{y}\|^2,$$

$$\textbf{(2)}\ C = \frac{1}{2}\|y - \hat{y}\|^2 = \frac{1}{2} \sum_{j} (y_j - \hat{y}_j)^2.$$

Before introducing back-propagation, an understanding of gradient descent is needed. Optimization problems usually involve finding either the maxima or the minima of the objective function. In this case, the objective function is the cost function. Getting as close to the actual value implies finding the minima of the cost function. Taking the derivative of a function at a point finds the slope of the given function at that point. For multi-variable functions partial derivatives, $\frac{\partial}{\partial \theta_i} f(\theta)$ finds the change in regards to x_i. Vectorizing the partial derivative over all variables gives the gradient. Mathematically this is denoted with $\nabla_\theta f(\theta)$. The gradient of a given point θ returns the steepest ascent. To find the minima, one has to move away from this point. This is done by iteratively calculating the gradient and then taking a short step in the opposite direction.

$$\theta' = \theta - \eta \nabla_\theta f(\theta) \tag{2.17}$$

In Equation (2.17), θ' is the new point after moving slightly away from the previous. The step distance is decided by the learning rate, η, and is usually a small constant. By increasing the learning rate, the model moves quicker, but it is at risk of missing the minima. A challenge with gradient descent methods that they do not necessarily find the global minima since they will only find the closest minima.

During forward propagation, each neuron impacts the final output. However, as they are not perfect, each of them introduces a small error. Node i in the k^{th} layer will have the error $\delta_i^k = \frac{\partial C}{\partial z_i^k}$. Back-propagation calculates the gradient of the cost function with respect to weights and biases [43]. Back-propagation calculates the gradient using the chain rule. The complete derivation of back-propagation will not be covered here but is thoroughly covered by Goodfellow et al. [43].

One way to formulate the four equations of back-propagation is as the following [48],

$$\delta^L = \nabla_a C \odot \sigma'(z^L), \tag{2.18a}$$

$$\delta^k = ((w^{k+1})^T \delta^{k+1}) \odot \sigma'(z^k), \tag{2.18b}$$

$$\frac{\partial C}{\partial b_i^k} = \delta_i^k, \tag{2.18c}$$

$$\frac{\partial C}{\partial w_{i,j}^l} = a_j^{k-1} \delta_i^k. \tag{2.18d}$$

As the back-propagation is calculated recursively, the first equation (Equation (2.18a)) calculates the error given by the output layer, δ_i^L. The partial derivative, $\frac{\partial C}{\partial a_j^L}$ is a measure of the rate of change in cost as a function of the i^{th} activation function in the output layer. $\sigma'(z_i^L)$ is the derivative of the activation function for input into node i in the output layer, z_i^L. Equation (2.18b) is then applied to calculate the error terms in the previous layer, which is done recursively through all prior layers. Equation (2.18a) is used for the penultimate layer, and then the results from (2.18b) are used for the subsequent layers. Taking the transpose of the weights in layer $k+1$, $(w^{k+1})^T$, and applying the error from the same layer one can think of it as moving the error backward in the system [48]. The rest of Equation (2.18b) is the same as in Equation (2.18a), namely the Hadamard product of $\sigma'(z^k)$. Equation (2.18c) describes the rate of change with respect to potential biases in node i in the k^{th} layer. It is observed that this error is already calculated in Equations (2.18a) and (2.18b). Finally, using Equation (2.18d), the impact of the weights regarding the rate of change in cost is calculated. Earlier, it was presented on how to calculate a_j^{k-1} and δ_i^k. Consequently, the gradient of the cost function is given by Equations (2.18c) and (2.18d).

2.3.5 Optimizers for Gradient Descent

Multiple different versions of the gradient descent utilize different measures for finding the minima. One of the simplest algorithms, therefore often called vanilla gradient descent, is the **Batch gradient descent**. Batch gradient descent takes advantage of Equation (2.17), calculating the value for the training set. The obvious downfall of this is that it is slow and needs a lot of memory. **Stochastic gradient descent** (SGD) increases the speed by performing the updates for training samples instead of after each iteration. However, if the learning rate is not sufficiently low, the SGD algorithm is at the chance of overshooting [49]. Additionally, with a decreased learning rate, SGD performs similarly to Batch gradient descent.

The two previous optimizer algorithms are both naive, in the sense that they will always have the same step length. Imagine one stands near a maximum with a steep slope downwards, instead of taking meticulous and short steps, one could take advantage of the momentum and take longer strides. Mathematically this can be formulated by introducing an update vector, v_t, consisting of the step from Equation (2.17) and the update vector for the previous time step multiplied with a constant γ. The constant is often called the momentum term and is usually 0.9 [49]. The general

formulation of gradient descent methods using momentum is the following,

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta f(\theta), \tag{2.19a}$$

$$\theta = \theta - v_t. \tag{2.19b}$$

Continuing down the steep slope mentioned above, when at the bottom, one have gained a lot of momentum, meaning one would most likely continue past the minima and start "climbing" up on the other side. Therefore, it is desirable to add some way of slowing the momentum before reaching the minima. There are multiple different algorithms proposed, often building on previous versions. The **Nesterov Accelerated Gradient** (NAG) takes advantage of γv_{t-1}, when calculating the gradient to estimate the following parameters. **Adagrad** improves upon NAG by introducing automatic learning rate tuning [49]. Furthermore, Adagrad differentiates the learning rate based on the occurrence of the parameters. Other examples of optimization algorithms are **Adadelta**, **RMSProp**, **Adaptive Moment Estimation** (Adam), and **Adapg** [50].

Proposed by Kingma and Ba [51], Adam is a continuation of RMSProp, taking advantage of the first and second moment of the gradients. The first moment, m_t, is understood as the moving average of the gradient, while the second gradient, v_t is the squared gradient. Calculation of Adam can be formulated into five equations, Equations (2.20a) - (2.20e).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla f(\theta_t) \tag{2.20a}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla f(\theta_t)^2 \tag{2.20b}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.20c}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.20d}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.20e}$$

Equations (2.20a) and (2.20b) calculate the first and second moment, using two biases, β_1 and β_2. The biases control the decay rates of the respective moving average and are conventionally set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial values, m_0 and v_0, are 0, making them zero biased. Kingma and Ba corrected this by forming estimates for the two, $\hat{m}_t$ and $\hat{v}_t$, fixing the zero bias. Finally, (2.20e) shows how the new parameters are calculated, where ϵ is added to counteract division by zero.

2.3.6 Vanishing Gradient

Recalling the calculations of the gradient of the cost function in Section 2.3.4, Equation (2.18d) describes how the gradient is calculated with regards to the weights using Equations (2.18a) and (2.18b). Furthermore, it covered how the chain rule is applied throughout the network to calculate the error term, δ^k. Below is an example of a neural network with three layers (where the output layer is denoted as L).

$$\delta^0 = ((w^1)^T(((w^L)^T(\nabla_a C \odot \sigma'(z^L))) \odot \sigma'(z^1))) \odot \sigma'(z^0)$$

It is seen that the error in layer 0 will be dependent on the product of the derivatives of the activation layers. This exposes a weakness for neural networks using gradient descent algorithms and back-propagation called *vanishing gradients*. First, networks with many hidden layers will be the product of many derivatives. If one or more of these derivatives are small in scale, the resulting error term will diminish, resulting in a small weight gradient. Neurons affected by this will, in return, learn less and, in the worst case, stop learning altogether. This is further established if the activation function has a derivative with a short significant range. The sigmoid is an example of an activation function with a short significant range. When $\sigma(z_i^k)$ is close to 0 or 1, the derivative will consequently become nearly zero [48]. One solution is to change the activation function to, ReLu, which will have the following derivative,

$$\sigma'_{ReLu}(z) = \begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x > 0 \\ \text{undefined if } x = 0 \end{cases}.$$

Another issue caused by the same weakness is *exploding gradients*. As the name implies, this happens when the gradients become excessively large, making the gradient descent algorithm update with too large steps. Gradient clipping is a partial solution to exploding gradients. Introducing a threshold that is compared with the norm of the gradient, one can hinder steps larger than this threshold.

2.4 Recurrent Neural Networks

A downside to MLPs is that they were initially developed to work on static models [52]. However, most real-life problems are not in static environments. This makes MLPs less efficient when modeling systems where previous instances impact the current. An example is language recognition, where prior words in a sentence heavily influence the meaning of the sentence and meaning of the following words. Regression problems are often impacted by previous sequences, forming patterns in the problem. One way to deal with sequence data is using RNNs. The structure of RNNs is very similar to that of MLPs, but with one significant difference. In addition to forward links, an additional internal connection is added. This connection can mathematically be formulated for each hidden state, $\mathbf{h}^t$, as,

$$\mathbf{h}^t = f(\mathbf{h}^{(t-1)}, \mathbf{x}^t; \theta). \tag{2.21}$$

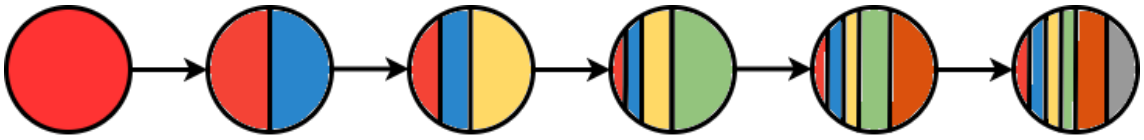


Figure 2.7: Visual example of how RNNs store sequence data for each time step.

Recalling forward propagation for MLPs (section 2.3.2), the equations can be rewritten to fit recurrence. The initial hidden state $\mathbf{h}^0$ is used to calculate the first input $\mathbf{z^1}$. Equation (2.22a) explains forward propagation for time steps $[1, \tau]$ [43]. $\mathbf{W}$ and $\mathbf{U}$ represent the weight matrices between hidden layers and input to the hidden state, respectively. $\mathbf{x}^t$ is the inputs at time step t. $\mathbf{z^t}$ is then processed through an activation function as seen in Equation (2.22b). Each time step is fed through an output layer, where the hidden state is first multiplied with an output weight matrix, $\mathbf{V}$, and a new bias, $\mathbf{c}$, is added. To get the actual output, $\hat{\mathbf{y}}^t$, a new activation is applied, as seen in Equation (2.22d).

$$\mathbf{z}^t = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^t \tag{2.22a}$$

$$\mathbf{h}^t = \sigma_h(\mathbf{z}^t) \tag{2.22b}$$

$$\mathbf{o}^t = \mathbf{c} + \mathbf{V}\mathbf{h}^t \tag{2.22c}$$

$$\hat{\mathbf{y}}^t = \sigma_o(\mathbf{o}^t) \tag{2.22d}$$

Calculating the gradient of an RNN is performed as described in Section 2.3.4. However, the calculations become more cumbersome due to recurrence and the extra weights and biases introduced with RNN. A graphical understanding of how an RNN holds information in each node is shown in Figure 2.7. Each new time step contains previous information and data for the current time t. In theory, this holds information better. However, neural networks with long sequences are challenging for RNNs as early information fades over time. This is seen in Figure 2.7, as the red in the final node is nearly gone. RNN are additionally susceptible to vanishing gradients as described in Section 2.3.6. To counteract this, Hochreiter and Schmidhuber introduced LSTM in 1997 [53].

2.5 Long Short-Term Memory

LSTM networks replace the hidden units in RNN with memory cells constructed of three gates [53]. In addition to these three gates, LSTMs introduce a Cell state vector, $\mathbf{C}^t$. The cell state vector is accountable for keeping track of the critical information in the system. Figure 2.8 shows the structure of an LSTM cell, with the cell state running across the top and connecting all cells. For each time step, information is removed and added, this is done by the forget gate and the input gate. The last gate, the output gate, decides what information should be sent to the next hidden state and output.

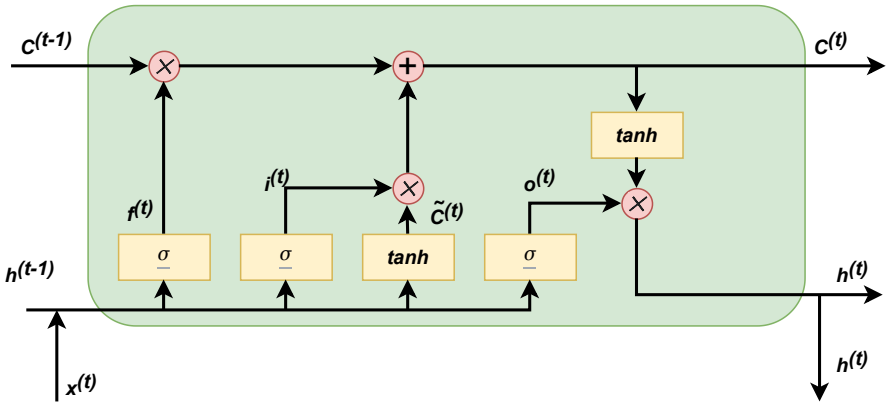


Figure 2.8: LSTM cell with gates, activation functions and states, inspired by [54].

The initial step of an LSTM cell is to decide what information from the previous hidden state, $\mathbf{h}^{(t-1)}$, and the input, $\mathbf{x}^t$, is surplus and should be forgotten. This is done mathematically by multiplying the concatenate of $\mathbf{h}^{(t-1)}$ and $\mathbf{x}^t$ with a weight matrix, $\mathbf{U}$. As with MLPs and RNNs, a bias, $\mathbf{b_f}$, is added before squashing the information using a sigmoid function, as shown in Equation (2.23a). A sigmoid is chosen because it squashes the value between 0 and 1, whereas 0 means forget everything and 1 keep everything [54]. Equation (2.23b) is nearly identical to the forget equation. However, it determines what information should be kept in the updated cell state. The only difference is the weights and biases, which are given by $\mathbf{V}$ and $\mathbf{b_i}$ for the input equation, respectively. A candidate vector, $\tilde{\mathbf{C}}$, is introduced to hold information to add to the cell state and is described mathematically in Equation (2.23c). The candidate activation function, tanh, is used to keep the values between -1 and 1. To calculate the updated cell state, unimportant parts are forgotten by multiplying the previous cell state with the forget vector. Then the new information is added by multiplying the input vector with the new candidate vector. This is shown by the plus sign in Figure 2.8 and Equation (2.23d). Finally, to create a new hidden state, an output vector is calculated using Equation (2.23e), where $\mathbf{W}$ is the output weights, and $\mathbf{b_o}$ is the output bias. This output vector is multiplied with $tanh(\mathbf{C}^t)$ to add old sequence data [54].

$$\mathbf{f}^{(t)} = \sigma(\mathbf{U} \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b_f}), \tag{2.23a}$$

$$\mathbf{i}^{(t)} = \sigma(\mathbf{V} \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b_i}), \tag{2.23b}$$

$$\tilde{\mathbf{C}}^{(t)} = tanh(\mathbf{W_c} \cdot [\mathbf{h}^{\lfloor}(t-1), \mathbf{x}^{(t)}] + \mathbf{b_C}, \tag{2.23c}$$

$$\mathbf{C}^{(t)} = \mathbf{f}^{(t)} * \mathbf{C}^{(t-1)} + \mathbf{i}^{(t)} * \tilde{\mathbf{C}}^{(t)}, \tag{2.23d}$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W} \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b_o}), \tag{2.23e}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} * tanh(C^{(t)}). \tag{2.23f}$$

2.6 Convolutional Neural Networks

Convolutional Neural Networks excel at pattern recognition. Therefore, CNN has been highly useful in image recognition as it decomposes the image, making it easier to identify distinct components in an image. This is done by different convolutional layers "looking" for different patterns. For photos, a 2D grid is used to process the picture. As mentioned earlier, time series data is often composed of different patterns transpiring at different frequencies. Instead of a 2D grid, time series data CNN takes advantage of 1D grids(or 2D if there are multiple time steps) [43]. As the name implies, CNNs take advantage of convolutions. The mathematical formulation of a convolution is seen in Equation (2.24). CNNs work similarly to MLPs described in Section 2.3.2. However, CNNs are not necessarily fully connected. This is called sparse interactions and can be viewed as a simplification of the input. Another advantage of CNN is parameter sharing for feature maps. These two traits of CNNs save computational time and memory.

$$(f * g)(t) = \int_{-\inf}^{\inf} f(\tau)g(t-\tau)d\tau \tag{2.24}$$

A convolutional layer can usually be divided into three stages: convolution, nonlinearity, and pooling. The first step consists of performing multiple convolutions, transforming the input into a set of outputs, often called a feature map. For the general convolution equation (Equation (2.24)) one of the arguments (for example, f) will be the input tensor, while the other (in this case, g) is the kernel. The kernel is another tensor extracting a feature, and is often called a filter. In the second step, a nonlinearity is added due to the first step being linear. As with MLPs, ReLu is a popular activation function for CNNs [43]. In the final stage, the input is divided into small rectangles (with the same size as the kernel size) and simplified. One popular simplification is the max pooling, which takes the max value of the given area. Forward propagation and back-propagation are similar to those previously described. Due to pooling, the index of the max value is stored to be used during back-propagation.

2.7 Explainable Artificial Intelligence

The sections about XAI is inspired by the specialization project [1] as much of the theory is relevant for this thesis as well.

A standard black box machine learning algorithm will leave the end user with the output and nothing more. To improve or change the results of such a model, one can try to change the hyperparameters and input data to find a better solution. On the other side of the spectrum, there are white box models with fully understandable features as well as the process being interpretable by humans. White box models are often limited in complexity because humans can not comprehend how machines think. The field of XAI is found between these, often categorized as grey box or glass box models. XAI maintain a lot of the complexity of black box models simultaneously as they give reason to why they arrived at their answer. The goal of XAI is to provide domain experts the possibility to have more interactive models, which can be applied more generally [12]. There are also hopes that XAI can help machine learning models to become more trustworthy and fair [55]. black box models used for sensitive information and human risk-associated situations often lack credibility and end up not being used. XAI has so far been especially useful for image recognition

as it provides insight into classifications that are done. However, it has become increasingly used for forecasting, for example, by Dikshit et al. [56], who used it for drought predictions in Australia, and by Zdravkovic et al. [57]. They gained insight into direct heating system models as well as a better understanding of the predictive maintenance of components.

The application of XAI can be divided into model agnostic and model specific. Model agnostic methods are universal for different types of machine learning algorithms. This is beneficial as it allows for comparing the output of multiple models [58]. This is an advantage over model specific, which only can be used on particular algorithms, limiting when they can be used. Explanation and representation flexibility are an important traits of model agnostic methods beyond being able to be used for different models. This entails methods that can give different types of explanations based on the problem at hand and for the features in the respective model [59]. XAI methods are also divided into local and global, describing individual predictions and average behavior, respectively [58]. Methods for increased transparency can be introduced before or after the machine learning model, namely ante-hoc and post-hoc methods. This study will focus on post-hoc methods as they allow for more general use of explainers and ML models. Among the most used post-hoc methods are **SHAP**, **LIME**, **DeepLIFT**, and **ELI5**, some will be explored more in the following sections [55].

2.7.1 Local Interpretable Model-agnostic Explanations

As the name implies, LIME is an explainable method designed for specific predictions [55]. As it is model agnostic, it works for all regression and classification models and is useful for image and text explanations [58]. LIME creates an interpretable model g from a set of potential models G, which try to fit the global model f. The chosen model is viewed as a surrogate model for the area around the specific prediction, x. The goal of the LIME is to find which model g approximates f the best. Riberio et al. [59] presented this mathematically as,

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g). \tag{2.25}$$

$\Omega(g)$ is a measure of complexity for each model g, it is preferable to have an interpretable model meaning that Ω needs to be small. $\mathcal{L}$ is the measure of locality-aware loss between g and f based on π_x, defined as the proximity weight to x. This can be formulated as,

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{2.26}$$

which can be seen as the square loss between the actual and proposed model timed with the weights determining the vicinity of x [59]. $\mathcal{Z}$ is a set of modified values based on x, which are used to understand the local area fpr which LIME is trying to define an approximation. LIME is a great way to simplify a complex black box model. On the other hand, LIME requires a domain expert to validate the explanations as LIME uses features to explain the model. Another disadvantage of LIME is that rerunning the algorithm can lead to different results. This is because machine learning models are stochastic.

2.7.2 SHapley Additive exPlanations

The SHAP methodology is based upon game theory ranking different participants' influence in a game and what share of the return they deserve. This is done by combining each possible subset of participants and recalculating the game's outcome. This gives an indication of how each group member contributed to the task and how different groups would fair. Lloyd S. Shapley proposed the following mathematical explanation for Shapley values,

$$\varphi_j(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!}[f_x(z') - f_x(z' \backslash j)] \tag{2.27}$$

Formulating this as a machine learning problem, the goal is to see how the different features impact the model's outcome. From Equation (2.27), the Shapley value of a feature j is calculated using the model f and a feature vector x [58]. As many models are complex, there can be a need to simplify the features into x'. The first part of Equation (2.27) can be viewed as a weight that incorporates how adding or subtracting different features impact the model based upon all features M. $[f_x(z') - f_x(z' \backslash j)]$ shows the contribution of a subset of features, i in z', where $f_x(z') = E[f(z)|z_S]$. A set S is given by non-zero indexes in z' [60]. Since Shapley values are calculated on all subsets of features, the numbers of calculations become exponentially high.

SHAP takes advantage of LIME and Shapley values to create model agnostic explanations. SHAP is primarily a local explainer. However, by averaging all local explanations, SHAP can give global explanations [55]. SHAP breaks features down to $z' \subseteq \{0,1\}^M$ indicating that a feature is either viewed as active or asleep. One formal way of expressing SHAP defined by Lundberg et al. [60] is,

$$g(z') = \varphi_0 + \sum_{i}^{M} \varphi_i z'_i. \tag{2.28}$$

It can be observed that the formulation of Equation (2.28) is a linear regression. This is similar to the Linear LIME model, but includes other weights. Where LIME focuses on the original instance's proximity, SHAP deviates and is weighted based on the coalition of features. Both LIME and SHAP (and other additive feature attribution methods) desire three properties to obtain a unique solution [60]. First of all it is important that $g(x') = f(x)$ in the local area when $x = h_x(x')$, where h_x is a function which maps the binary values x' to x. In other words, it is important to have local accuracy for the approximation. Secondly, features not included ($x'_i = 0$) shall not have an attributed impact. Equation (2.28) and Equation (2.26) show that both methods obey this property. Thirdly, it is important that the simplified input does not have less attribution than the original value. Lundberg et al. [60] found that Equation (2.27) follows Equation (2.28) and possesses the three properties mentioned. Taking advantage of Equation (2.25) and modifying the loss function and weight to the following,

$$\mathcal{L}(f,g,\pi_{x'}) = \sum_{z' \in Z} [f(h_x^{-1}(z')) - g(z')]^2 \pi_{x'}(z')$$

$$\pi_{x'}(z') = \frac{M-1}{\binom{M}{|z'|}|z'|(M-|z'|)}$$

as well as setting $\Omega(g) = 0$, the authors proved that one could use LIME to calculate the Shapley values. This explainer is called KernelSHAP [60].

Another explainer is DeepSHAP, a combination of DeepLift and Shapley values [60]. DeepLift was originally proposed by Shrikumar et al. [61] and is a deep learning tool explaining the difference in input and output compared to a reference input and output. For an output, o, Deeplift calculates a contribution score, $C_{\delta x \delta o}$, representing the contribution from each input compared to the reference. Mathematically this can be formulated as,

$$\sum_{i}^{n} C_{\Delta x_i \Delta o} = \Delta O, \tag{2.29}$$

where $\Delta o = f(x) - f(r)$ and $\Delta x_i = x_i - r_i$. The reference is defined as r. A multiplier, $m_{\Delta x \Delta o} = \frac{C_{\Delta x \Delta o}}{\Delta x}$ is defined as a multiplier, which is seen to find the contribution of the change in x to the change in o. DeepSHAP uses this to create a framework (thoroughly covered by Lundberg et al. [60]), which makes it possible to calculate smaller parts for a network into SHAP values for the entire network. This is done recursively by passing multipliers backward in the network, similar to back-propagation. The main advantage of DeepSHAP compared to KernelSHAP is that as DeepSHAP uses relationships established in neural networks computational time is saved.

2.8 Energy Forecasting

Energy forecasting is a crucial factor for the power sector. It allows the system operators to make educated estimations on everything from component wear to grid stability. Forecasting is also essential in market analysis, considering both expected production and load. Acquisition of data is often the initial step when creating a forecasting algorithm. After retrieving relevant data, one should inspect it for characteristics such as seasonal components, trends, sharp changes in behavior, and outlying data points [32]. To understand seasonal components and trends, it is important to have at least one year of training data to understand how these will affect the forecast. It is also essential to be aware of peculiar data points since they produce non-coherent predictions compared to the usual pattern. Removing these will leave the stationary residuals, which are subsequently used to create a model. It is important to define the forecast horizon and forecast interval. The horizon is defined as how far into the future the model should forecast, and the interval is how often the model shall deliver outputs. The problem at hand usually influences both horizon and interval. It is normal for energy forecasting to split the horizon into very short-, short-, medium-, and long-term. Very short-term forecasting are often categorized as less than an hour, whilst long-term can be categorized as more than 3 years [55]. This section will cover different types of power forecasting and their methods. The following sections are taken from the specialization project [1]

2.8.1 Wind Power Forecasting

Wind power is becoming increasingly popular because of its relatively cheap LCOE and production capacity [24]. However, the instability issues introduced with it need to be accounted for. Most wind power forecasting happens within the very short and short-term time frame. Forecasting performed with a horizon of less than 30 minutes (down to seconds ahead) are important to real-time tracking production and turbine control. This ensures optimal operation with minimal wear on the turbine. For these predictions, observations and sensor data are important because there is little to no time to perform advanced algorithms using tons of metrics [62].

For longer horizons, other input data become more relevant. NWP such as temperature, pressure, wind speed, and direction are essential attributes for wind power forecasting. In addition to weather prediction, historical data is valuable information for many forecasting methods. Finally, having good knowledge of the wind turbine's technical specifications and performance helps give accurate power forecasts [55]. Wind power forecasting is separated between single turbine forecasting and wind farm forecasting. With the advancements in turbine and platform technology, offshore wind farms have become increasingly popular. This has introduced the need for input data relating to the behavior of the ocean.

Statistical models such as the ARIMA are popular for wind power forecasting. By collecting historical data and running it through a statistical algorithm, one can obtain reasonably accurate results for both short- and medium-term forecasts. Statistical models are relatively cheap and easy to incorporate, making them an excellent baseline for comparing other models, as done in studies by Gonzalez-Sopena et al. [63] and Malhan et al. [64]. A disadvantage to statistical methods is the lack of complexity introduced with AI. The nature of machine learning makes it possible to make accurate forecasts without physical conditions, even though it is advantageous to include [55]. However, machine learning introduces issues such as lacking interpretability and being computationally harder compared to statistical models. Machine learning methods additionally proved effective in learning capacity with limited data. In an article by Tao et al. [65], deep belief networks were applied to 3 months of wind data and achieved accurate results compared to other benchmarks.

However, research has shown that hybrid models combining statistical components and ML are superior to exclusively using one method. In an extensive review performed by Wang et al. [66], multiple deep learning networks were investigated and compared. The study concluded with most hybrid models outperforming single models. Additionally, the review identified challenges for wind power forecasting. It was found that the complexity of wind data and efficiently identifying important features of said data still needs improvements [66].

2.8.2 Photo-Voltaic Power Forecasting

Forecasting of PV production inherits many similar traits to wind power forecasting. Solar production is vulnerable to weather changes, and the presence of shadows can limit power production for singular panels and lead to irregularities in grid frequency and stability. An advantage of PV panels over wind turbines is the lack of moving parts, leading to less need for forecasting related to component wear. In addition, PV panels have the benefit of being more scalable. This has led to a need for forecasting related to roof-mounted systems and large solar farms.

The most relevant NWP data for solar power forecasting is solar irradiance or solar power per square meter, in addition to weather data such as pressure and temperature. As the primary concern of PV panels is a clear sky, satellite data and cloud detection is often used in addition to weather data [55]. In a review by Ahmed et al. [67], multiple different forecasting techniques were explored for different time horizons. It was concluded that the hardest forecast horizons related to solar production were long- and very short-term. Long-term forecasting has the challenge of being a very dynamic problem requiring accurate NWP and historical data. On the other hand, very short-term forecasting is vulnerable to the volatility of cloud dynamics. The review also found variations of CNNs in different hybrid forms are the most prominent and promising forecasting

methods. LSTM hybrid models were also among the methods found to be used in a substantial amount of the papers reviewed. Li et al. [68] and Agga et al. [69] applied hybrid models of CNN-LSTM to day-ahead problems. Both reports concluded with positive results compared to other benchmark models. Li et al. [68] established that their model performed better for sunny conditions compared to cloudy and rainy days.

2.8.3 Electrical Load Forecasting

Formerly information regarding the energy flow through the grid and to residences was limited. Grid operators essentially applied voltage and hoped for the best. If someone complained, one would try and find where and the reason for the issue manually. With the emerging sensor and data technology, load flow analysis and forecasting have given opportunities for transmission and distribution planning, maintenance, and financial planning, among other applications [55]. As with other forecasting areas, load forecasting is divided into different time horizons. Furthermore, load forecasting can be split into categories based on the topology, ranging from household-level to regional and national forecasts.

Input Variables

Deciding which input variables to use for load forecasting forms the perspective and focus of the model. NWP and historical data have proved to be important features in multiple load forecasting models, both local and regional models [70], [71]. A review by Kuster et al. [71] found that meteorological data was particularly vital for short-term forecasts and on smaller scales. For pure statistical models such as regression analysis and ARIMA models, historical load data has been the most important feature. ML models, on the other hand, are more flexible when it comes to input data because of adaptability and nonlinearity. Other important features found in numerous methods are socioeconomic variables. A study by Kipping et al. [26] used a series of explanatory socioeconomic variables in addition to meter data to create forecasts for Norwegian electricity consumption. Among these variables were household size and information about the house, such as whether it was attached or detached. The same study also incorporated seasonality and information about Norwegian holidays. It can be challenging to obtain accurate load forecasts for local forecasts where the residence has local energy production. One way to increase the accuracy is by taking inspiration from PV power forecasting, like Chu et al. [72], where cloud images were used for net load forecasts in San Diego.

Forecasting Models

Lee et al. [73] compared a variety of seasonal ARIMA called SARIMAX with Support Vector Regression (SVR), LSTM, and ANN. Additionally, the hybrid version of the statistical model with the three ML models was tested. The models were used to forecast peak loads on nationally in South Korea. The authors concluded that the machine learning models outperformed SARIMAX. Furthermore, the singular version of LSTM and the hybrid version were found to be the best performers in total. In their literature review Kuster et al. [71], found that ANNs were used in most cases for short-term forecasting using a short resolution. Regression models, on the other

hand, were only used for long-term forecasts with annual data. For household-level load forecasting, machine learning algorithms are found to outperform traditional methods [74].

Challenges

With the increasing share of prosumers connected to the grid, it becomes increasingly hard to create accurate load forecasting models. The distribution system operators will, in most cases, not observe the contribution of PV panels, as the measurements are NILM based on the electric meter and can not directly distinguish production from loads. This increased volatility and flexibility can be hard to predict. Load forecasting on an individual household is harder to forecast compared to aggregated systems. Kong et al. [75] showed how demand for a single residence had less correlation compared to regional day-ahead loads. The study found a less consistent daily pattern for single households, but by accumulating multiple households, a more consistent pattern appeared. Increased difficulty of local forecasting leads to sub-optimal scheduling [76]. With many prosumers having limited production and storage capacity, sub-optimal scheduling creates economic expenses and power losses.

2.9 Forecast Performance

The following section (Section 2.9) and the ensuing subsections (Subsection 2.31, Subsection 2.30, and Subsection 2.9.3) are retrieved from the specialization project [1] mentioned in the preface.

It is important to get a notion of how well the finished model performs. As discussed in previous sections, the data of a forecast is split into a training and a test set. Even though the forecasting model performs well for the training set, it does not necessarily translate to a model working well for the test set and future analysis. Error analysis for forecast performance is similar to the error estimation for regression and ANNs in Section 2.3.4. There are multiple types of error predictions. Among the popular methods are scale-dependent and percentage errors. Scale-dependent errors keep the same scale as the output data. These types of accuracy measures cannot be used to compare forecasts with different scales or units. Percentage errors, on the other hand, can compare the performance of different data sets [30]. Percentage error-based accuracy measures require the data to be valid ratio scale measures. This entails having a meaningful zero. For example, for temperatures are Celsius and Fahrenheit invalid actions, but Kelvin is permissible. Percentage errors can also return extreme values if the observed value is close to zero [30].

2.9.1 Root Mean Square Error

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((y_i - \hat{y})^2} \tag{2.30}$$

Root Mean Square Error (RMSE) is a continuation of MSE seen in Equation (2.16). Even though RMSE has historically been frequently used, it has some concerns. First of all, as it is a scale-dependent accuracy measure, it can not be used to compare different scales. Furthermore, both MSE and RMSE are sensitive to outliers. Outliers are data points that are substantially far away

from the other data points. Since RMSE is susceptible to deviations, models judged with this method will be based on the mean value.

2.9.2 Mean Absolute Error

Mean Absolute Error (MAE) is another scale-dependent performance reviewer. It has many similar traits to RMSE, but as it is simply the absolute value of the error instead of the root mean square, it is easier to interpret. MAE is median-based, making it less vulnerable to outliers compared to RMSE [77].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.31}$$

2.9.3 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) is similar to MAE, however by dividing the observed value, one obtains the scaled version. MAPE inherits the robustness to outliers from MAE [77]. As a percentage-based form performance indicator, MAPE is a great way to compare the performance of different data sets. However, MAPE has some downfalls.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{2.32}$$

From Equation (2.32), it is observed that cases where $y_i > \hat{y}_i$ will be punished less than $y_i < \hat{y}_i$. Another issue with the mathematical formulation of MAPE is cases where the observed value is zero or close to zero, creating an undefined or high response. There are variations of MAPE which counteract these issues. The symmetric MAPE (sMAPE) modification of MAPE removes extra penalties given to exaggerated predictions and is less affected by small observed values. However, Hyndman et al. [77] argued that MAPE is the preferred option if all observed values are significantly distant from zero because of its simplicity. They also argued that MAE might be preferable for the same reason if all data is of the same scale.

2.9.4 R-squared

R-squared, also known as the **coefficient of determination** or simply R^2, is a measure of the relationship between the independent and dependent variables. By calculating the residual sum of squares, $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and dividing it by the total sum of squares. The outcome of this is the Fraction of Variance Unexplained (FVU). As with any regression model, the target is for the residual sum of squares to be as close to zero as possible. On the other hand, if FVU is 1, the prediction, $\hat{y}$, is predicted to be the mean, $\overline{y}$, for all predictions. R^2 is mathematically described in Equation (2.33). It is seen that another way of writing the equation is $R^2 = 1 - FVU$. The optimal R-squared value is consequently 1, while 0 represents a model describing the mean. In some cases, a negative R^2 is obtained, indicating the predictions perform worse than the average

value of the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})} \tag{2.33}$$

Chapter 3

Methodology

The following chapter will describe the methodology for this thesis. The approach can be divided into five steps and is visualized in Figure 3.1. **(1)** Collect raw data, consisting of electrical load data and weather prediction data. **(2)** preprocessing of said data. Here, outliers and irregularities are removed, and categorical data is added. In addition to the aforementioned the data is visualized to get insight into patterns. **(3)** Training data is used for model development. During model development, different input variables and hyperparameters are tuned and experimented with to find the best model. **(4)** The different models are investigated using XAI methods, such as Shapley Additive exPlanations. **(5)** The different models from step three are compared using the traditional performance reviews: RMSE, R^2, MAPE, and MAE.

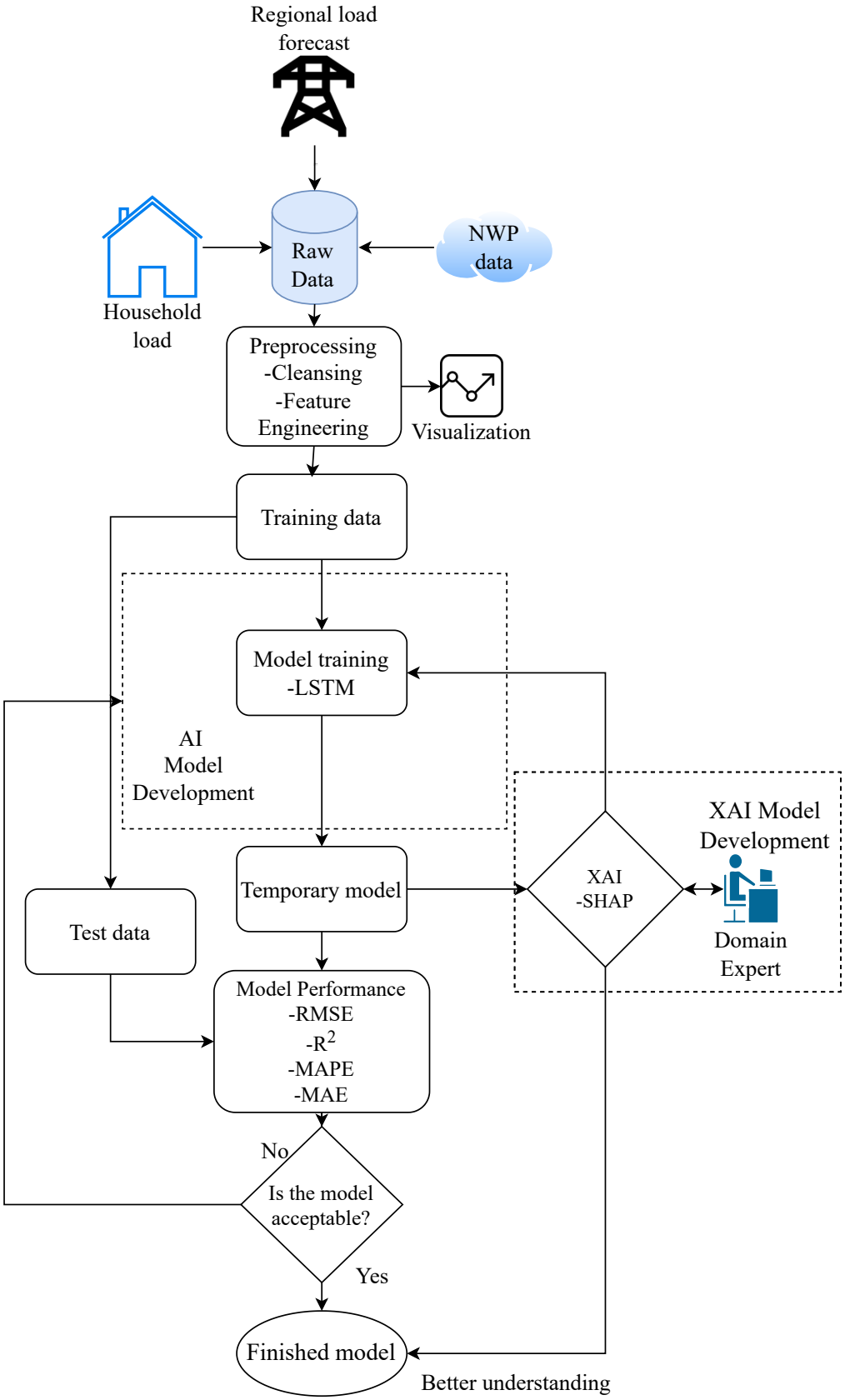


Figure 3.1: The proposed methodology for the thesis.

3.1 Experimental Setup

Throughout model development, a ZenBook Pro 15 with an Intel(R) Core(TM) i7-8750H CPU with a core speed of 2.20GHz was used. Due to complexity in the model and model tuning, it later in the thesis proved to be insufficient to use only this device. It was therefore acquired use of another server more suitable for heavy ML computations, running on an Intel Core i9-9920X @ 3.5 GHz, with four NVIDIA GV102 graphics cards.

3.2 Data Collection

To create reliable neural networks that perform to high standards the first step is to collect, analyze and understand the data. Inaccurate analysis of data can lead to errors as a result of both humans and defective sensors. For example, a faulty wattmeter could result in outliers which in return could create inaccurate connections in the neural network. This section will cover how data was recovered as well as visualizations of data ahead of preprocessing.

3.2.1 Residential Load Data

Consumption data for Norwegian households are managed and stored by Elhub [78]. Elhub are obliged to collect measurements from smart metering devices which are required in all households in Norway. The data is used by TSOs and production companies to keep track of consumption and corresponding invoices. The data is openly available to end users with an hourly resolution. Third parties can access private data if granted by the respective owner. In this thesis, a detached house is investigated. The house is located in Oslo, in the southeast of Norway.

In Norway the standard for building efficiency is set by NS 3031:2007 [25]. From this, *Energimerkingen* was created, a simple way to identify an energy grade and a heating grade. Each grade is independent of the other but forms a comprehensive and straightforward understanding of the house's energy use and demand. The Energy grade stretches between A and G, where A is the best, signifying an energy-efficient home regarding its size. Heating grade, on the other hand, is dependent on what kind of heating is used. A high share of pure electric (electric heater etc.) or fossil heating is bad and denoted with a red color. The grading system is five steps, red, orange, yellow, light green and dark green. Examples of measures to increase heating grade are the installation of thermal solar energy and heat pumps. Figure 3.2 depicts the calculated Energimerke for the residence in question, as most heating is done by electric heating and a fireplace. Secondly, the Energy grade is the second to last worst due to the large size and the year it is built.

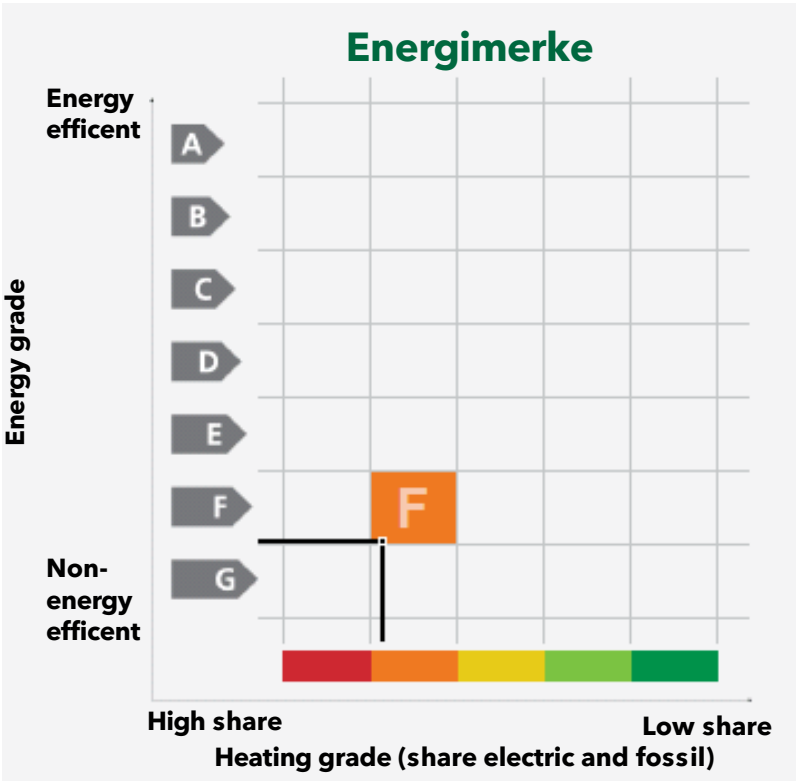


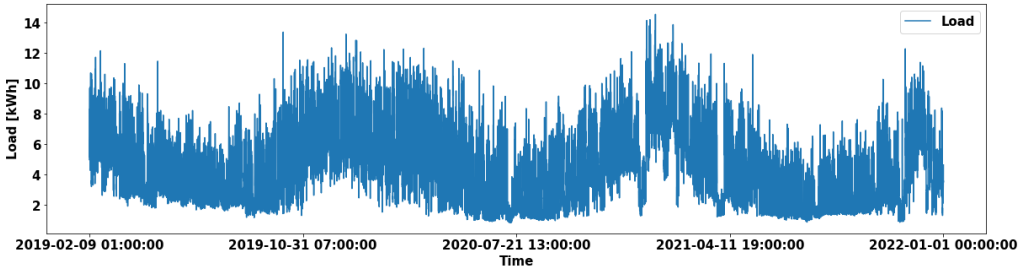
Figure 3.2: Energimerking of the detached house. Y-axis is energy grade and X-axis is heating grade. Calculated from [79].

The available load data spans from February 2019 until 2021 with an hourly resolution, creating 25368 data points. Figure 3.3 illustrate all data points, as well as, short time spans to get a sense of the distribution. The format of the data is given as accumulated active power each hour, E^t, each hour and can be simplified into the following equation,

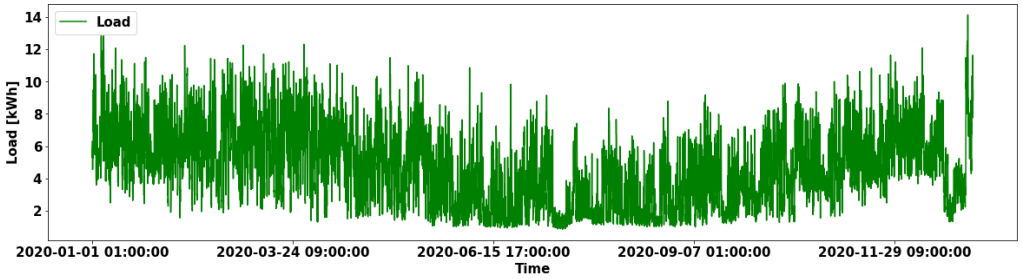
$$E^t = \int_t^{t+\tau} P(t)dt. \tag{3.1}$$

$P(t)$ is the instantaneous active power load at time t and τ denotes a time shift of one hour with an unknown temporal resolution. Statistics Norway reported an average power consumption per household of about 16000 kWh in 2018 [80]. With the study object being a detached house, it is expected to consume more. In 2020 the total load accumulated to 40527 kWh, and in 2021 it was slightly lower at 36269 kWh. Table 3.1 shows the descriptive data of the load data, average load, Standard Deviation (STD) as well as lowest, highest values, and each quarterly percentile (25 %, 50 %, and 75 %). Average consumed power slightly decreased in the time period, with a significant reduction of 10.3% in 2021 compared to the previous year. It is observed that the highest peak has increased at the same time as the minimal hourly consumption has decreased. The changes during last years could come down to a couple of reasons. With the emergence of COVID-19 many changed habits, which in return changed power usage.

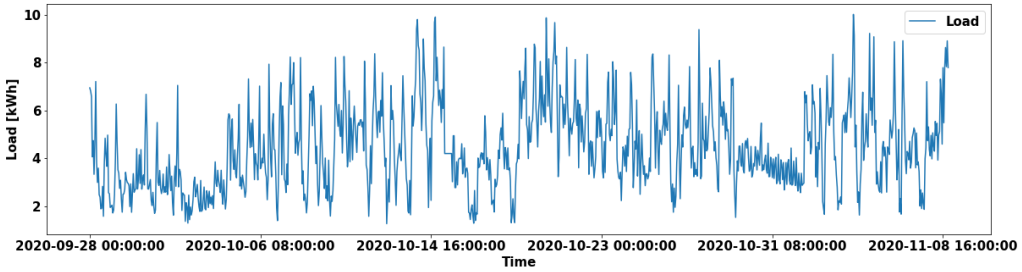
Additionally, increasing electricity prices can inflict saving measurements by house owners. A study performed by Sæle et al. [81] found that 53.9% of participants (1011 participants) would change their habits if the yearly expenditure increased by 3000 NOK or more. The actual reason for the behavior change has not been investigated and can not be attributed to anything specific.



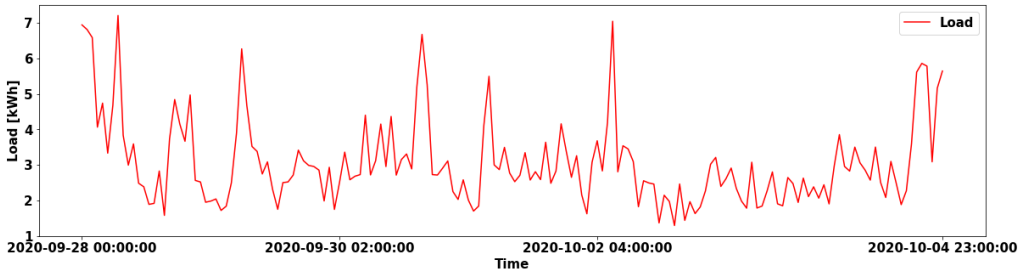
(a) Electrical load from 2019 to the end of 2021.



(b) Electrical load of 2020.



(c) Electrical load between week 40 and 45 in 2020.



(d) Electrical load during week 40 in 2020.

Figure 3.3: Electrical consumption with different time spans.

Table 3.1: Statistics of the residential load data.

Year	Mean [kWh]	STD [kWh]	Min [kWh]	25%	50%	75%	Max [kWh]
2019	4.6355	2.0847	1.1719	2.85	4.374	5.962	13.384
2020	4.6142	2.3131	0.833	2.712	4.469	6.191	14.149
2021	4.140	2.449	0.873	2.09	3.54	5.664	14.555

3.2.2 Regional Load Data

Norway is divided into five price regions, NO1, NO2, NO3, NO4, and NO5. Each zone operates independently from the others, and power consumption and production disparities can lead to

different prices in each region. This is further supported by grid limitations as seen in Figure 3.4, which depict the Norwegian transmission lines. To calculate the accumulated hourly load, one can take the aggregate of power production in the price region and the sum of power trade between the areas. This can be done due to the electricity having the characteristic of being used instantly. Formulating power production and consumption as a balanced equation where $\sum G(t) = \sum P(t)$, with $G(t)$ signifying generation and $P(t)$ loads. Power trade between each price zone can be formulated as $\sum I_{i,j}(t)dt$, where i and j signify price zones [82]. Redefining Equation (3.1) to accommodate power production and power trade, the following equation is obtained,

$$E^t = \int_t^{t+\tau} (G(t) + \sum I(t)_{i,j})dt. \tag{3.2}$$

Figure 3.4: Overview of transmission lines in Norway. Adopted from [83].

Oslo is a part of price zone NO1, and it is therefore decided to collect load from this zone. Hourly accumulated load is collected with permission from Nord pool, the primary power market in Europe [84]. Load data dating back to 2014 is collected and cleaned as described in [82]. Power consumption in each price zone from 2014 to 2021 is depicted in Figure 3.5. Contrary to residential load data, regional data is given by MWh/h or $10^3 \times kWh/h$. During winter, NO1 can be seen to have the highest load consistently. However, during summer, this shifts with NO2 and NO3 sees less of a dip in electrical load compared to NO1. There can be multiple reasons to this. An aspect can be addressed to population, as NO1 is the area of Norway with the densest population and most inhabitants. The smaller dip in loads in NO2 and NO3 might be attributed to more energy-intensive industry located in these areas. The regional load data will be used on an LSTM model with a

24 − *hour* horizon. Model development will be described in a later section.

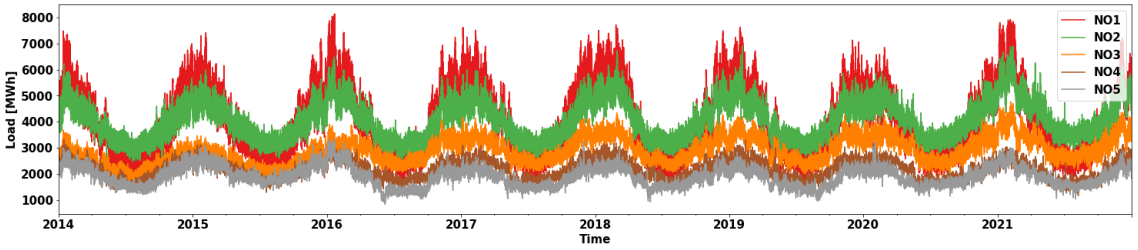


Figure 3.5: Regional loads by price zone.

3.2.3 Numerical Weather Prediction

The Meteorological Cooperation on Operational (MetCoOp) Numeric Weather Prediction is a collaboration between the Norwegian Meteorological Institute (MET), Finnish Meteorological Institute (FMI), and Swedish Meteorological and Hydrological Institute (SMHI). Together they have created the MetCoOp Ensemble Prediction System (MEPS). The forecasts are made by 30 ensemble members, meaning 30 different forecasts creating a new one based on probabilities [85]. A new forecast is created every sixth hour, 00:00, 06:00, 12:00, 18:00 and predicts up to 54 hours ahead [85]. The model is run with 2.5 km grid spacing and spans 900×960 grid points in zonal and meridional directions. In the vertical direction, the model has 65 levels.

The data is retrieved from MET Norway's THREDDS data server, which is available to the public [86]. Each day is logged in separate NetCDF files, retrieved through Python using OPenDAP protocol. The MEPS dataset offers a variety of different parameters. To provide proof of concept for XAI, some weather factors such as **upwards wind** and **cloud type** are seen as unimportant and therefore not included in the model development. The parameters deemed relevant for further investigation is shown below and a short excerpt of the downloaded format is shown in Table 3.2.

1. Ambient temperature $(T)[K]$
2. Relative humidity (H) [%]
3. Air pressure (p) $[Pa]$
4. Wind speed (w) $[m/s]$
5. Cloud cover (c) [%]

MEPS delivers wind data in the form of two vectors, one for wind along the zonal axis (x-axis) and one along the meridional (y-axis). Wind direction is deemed superfluous and instead the magnitude of the wind vector (see Equation (3.3)) is decided to be used.

$$w = ||(w_x, w_y)|| = \sqrt{w_x + w_y} \tag{3.3}$$

Table 3.2: Excerpt of NWP data format an afternoon in January.

Time	**T** $[K]$	**H** [%]	$\mathbf{w_y}$ $[m/s]$	$\mathbf{w_x}$ $[m/s]$	**P** $[Pa]$	**c** [%]
21-01-07 16:00	268.669	0.7912	-1.934	-1.397	100595.3	0.9932
21-01-07 17:00	268.8244	0.7818	-1.856	-1.102	100602.836	0.9990
21-01-07 18:00	268.787	0.7848	-1.284	-0.5635	100628.98	0.9980
21-01-07 19:00	268.814	0.7844	-1.211	-0.5379	100631.38	0.9985

3.2.4 Data Visualization

To get a better understand how the different input features impact each other and the context in regards to the load visualization is used. Here, the time-dependents variable are visualized to get a grasp of patterns and insights which can be further used during model development. The typical daily load profile for households consists of two peaks, one in the morning as people prepare to leave for work and one in the afternoon when people get home. Accumulating this for multiple households, a distinct pattern is created, as observed by the orange line in Figure 3.6. Each day has one prominent peak in the morning, followed by a smaller rise in the afternoon.On the other hand, a single household will be more susceptible to deviations in routines. Additionally, energy-intensive activities such as showering and washing machines will have a more significant impact on the load profile as it makes up a larger percentage of the total load. This is clear by the example shown in Figure 3.6, where the blue line indicates the residential load. The two loads are normalized as described in Section 3.3.1.

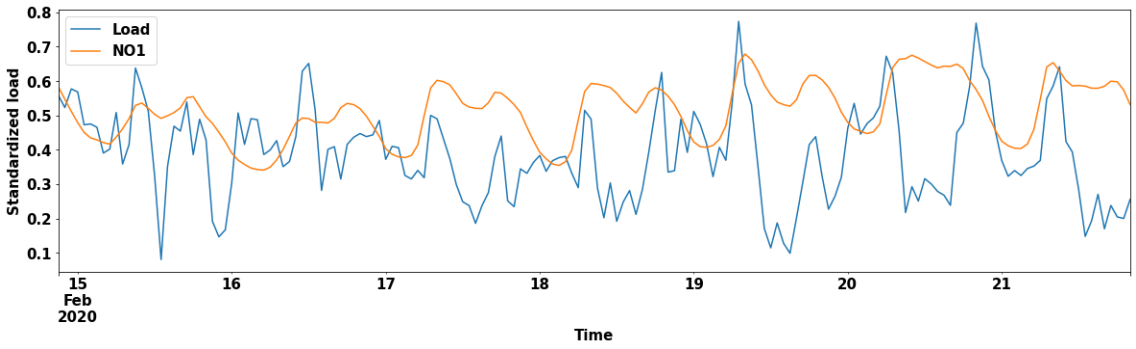


Figure 3.6: Standardized regional and residential load for week in February. Load imply residential load, whilst NO1 the load of price zone NO1.

Even though the pattern of the residential load is not as clear compared to the regional one, there are still usually two peaks as with the regional one. This is especially well observed on the 19^{th} of February, where the peaks are extra dominant. Furthermore, by accumulating every hour over a year, one can see a daily pattern form for each weekday. Monday till Friday are very similar with a peak at 06:00 and a second peak at 18:00. During the weekends, the load profile becomes more evenly distributed throughout the day, as shown by the cyan and green profiles in Figure 3.7. Another observation is that weekend peaks are later than compared to weekdays. This is most likely down to residents sleeping longer during the weekends.

Investigating the NWP, data multiple interesting observations are found. First, cloud cover and wind speed are quite irregular factors. Cloud cover is often predicted as either 0% or 100%, with significant deviations between hourly predictions. The inclusion of cloud cover is done to

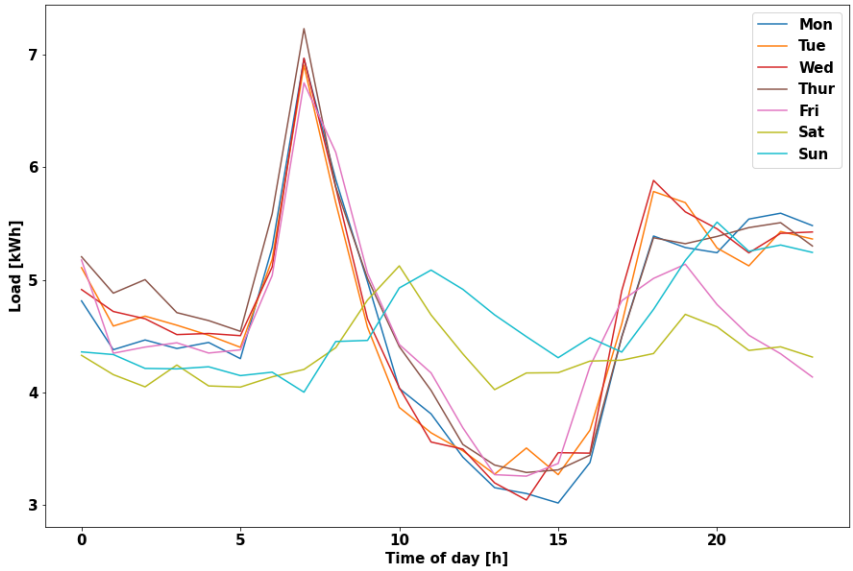


Figure 3.7: Average consumption each hour in 2020.

investigate whether or not the machine learning model can learn patterns not seen by the human eye. Furthermore, wind speed is not seen to have any seasonal patterns, nor does it seem to impact load directly from preliminary visualization. Humidity and pressure both have a small seasonal component. Humidity is furthermore seen to have a more daily component, as seen in Figure 3.8.

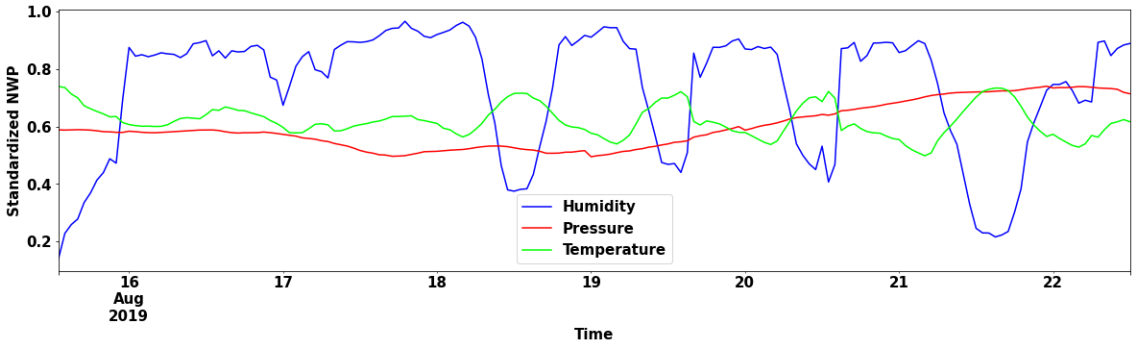


Figure 3.8: An excerpt of temperature, humidity and pressure from a week in August.

Finally, the temperature is seen as the most relevant NWP variable. With heating being such a large part of Norwegian power consumption, it is clear that this is important. This is backed by the literature in the Energy forecasting section (Section 2.8). Figure 3.8 shows how the nightly humidity high is opposite to the temperature, which increases during the day. Looking at each month's average temperature and comparing it to residential load and the load of NO1, the impact of heating is further proved, as illustrated in Figure 3.9.The same figure further demonstrates the relationship between local load and regional. It is worth noting that the reason the temperature dip in July is due to it being the coldest and rainiest July in 50 years in south-east Norway [87].

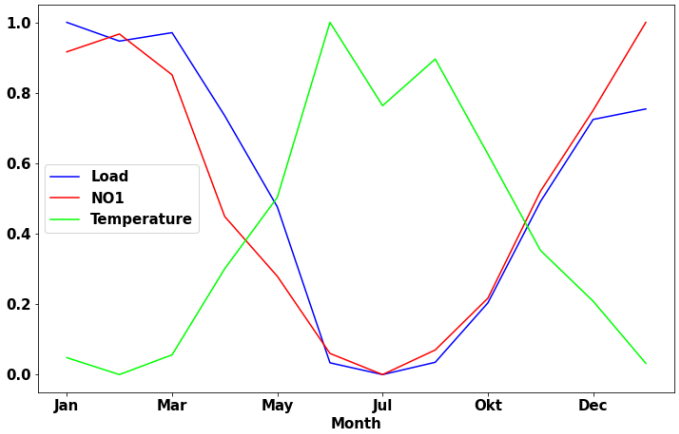


Figure 3.9: Average monthly residential load, regional load in NO1 and temperature, All have been standardized between 0 and 1.

3.2.5 Data Changes

Daylight Saving

Norway operates like many other countries with DayLight Time (DLT) savings. During the summer, Norway follows the UTC+2, and in the winter, the clock is turned backward, effectively turning the clock backward to UTC+1. This happens on the final Sunday in March and October each year. However, this creates an obstacle in the supplied data because an hour is lost in March, and an additional hour is added in October. The time shift happens between 02:00 and 03:00. Consequently, there is no data for the extra hour in March and an hour too much in October. This is dealt with by interpolating the previous value and the next value. This causes some inaccuracies, but is seen as the most convenient solution.

Missing Values and Irregularities

The residential household data was investigated for outliers and other irregularities, but none were found. For the residential load data and NWP data see [82].

3.3 Data Preparation

The suggested input parameters can be divided into two different categories. First, there are the time-dependent variables, which from now on called quantitative variables. The retrieval and data cleansing of these has been described in Section 3.2. Secondly, qualitative input variables are used for formulating fixed events, for example, separate which season it currently is.

3.3.1 Quantitative Variables

The collection of quantitative data is described in Section 3.2. The following is the list of quantitative variables which will be explored in this thesis,

- Residential load, P(t)
- Regional load predictions, $\hat{P}_{NO1}(t)$
- Ambient temperature, $\hat{T}(t)$
- Humidity, $\hat{H}(t)$
- Pressure, $\hat{p}(t)$
- Wind speed, $\hat{w}(t)$
- Cloud cover, $\hat{c}(t)$

Residential load is the only variable that is not another prediction, hence the other variables being denoted with a hat ($\hat{}$). NWP are predicted by MEPS as described earlier, whilst regional load predictions are performed as described in Section 3.4.3. The quantitative data is at wildly different scales and have other units. One measure to simplify the input for the model is normalization. Normalization confines all numeric inputs to be between a fixed interval. This can aid learning and increase performance [88]. For this thesis, Sklearn's **MinMaxScaler** [89] is used. Each feature is transformed using Equation (3.4), where X denotes every single value and X_{min} and X_{max} are the smallest and largest values in the set, respectively.

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.4}$$

In a real-life scenario one would only know previous data. Therefore when choosing the minima and maxima for the normalization, it is done using the training set. This is done using the *fit function*, which computes the two values. Then, the training and test input data are transformed using the built in *fit_transform*.

3.3.2 Qualitative Variables

There are a set number of possible outcomes for the qualitative variables based on some other variable. Usually, ML algorithms struggle to understand qualitative data. One way to work around this is by one-hot encoding the data to create *dummy variables*. One-hot encoding consists of creating a binary matrix, where each row is filled with a 0 or a 1, with 1 meaning true. For nominal variables, one-hot encoding works great. However, some problems are introduced with it. Mainly memory issues by introducing large matrices, slowing or stopping the model's progress. The following categorical variables were investigated during model development:

- Hour of day: h_0, ...,h_{12},..., h_{23}
- Day of week: d_0, d_1 d_2, d_3, d_4, d_5, d_6
- Months: m_0,m_1,..., m_{11}
- Season: s_0, s_1, s_2, s_3

Table 3.3 shows an excerpt of the one-hot encoded seasonal variable. Each season is three months with the winter months being defined as December, January and February.

Table 3.3: Example of how seasons are one-hot encoded.

Date	Winter (s_0)	Spring(s_1)	Summer(s_2)	Autumn(s_3)
2021-01-01	1	0	0	0
⋮				
2021-06-06	0	0	1	0
⋮				
2021-12-12	1	0	0	0

3.4 Model Development

All models are preprocessed and developed using Python [90], with the machine learning models are developed using the TensorFlow [91] library. Within Tensorflow, one finds Keras[92], Tensorflow's API. The initial step is to split the data into training and test set. The residential data set is limited to only three years worth of data. To give the model most the data to train on, it is decided to split it 80/20 in favor of training. It is recommended with a split of around 70/30 to 80/20, depending on the amount of available data [37]. With 80% of the data being used for training, it leaves the final 5074 hours of load data to test the model. This equates to June 3^{rd} until December 31^{st}. It is unfortunate not to be able to test the model on an entire year worth of data since it inhibits the possibility of seeing performance for all seasons.

Each model consists of independent variables, X, and the target variable, y, which in this case is the residential load, $P(t)$. The goal of the forecast is **hour ahead** forecasting, also known as having a *horizon* of one hour, using the previous 24 hours as an input. The independent and target variables are then reshaped to fit the horizon and model input. Each input is formed into tensors with the shape (**number of samples**, **number of lags**, **number of features**) and the y tensors having the shape (**number of samples** , **number of target variables**). In all cases, the number of target variables remains as one, whilst the number of features will vary depending on the model. With 24 lags per prediction, it is not only possible to predict until the 25^{th} value in the test set, the length of the test set is therefore reduced to 5050 hours.

Model Compositions

During development, two different types of models are investigated. The first model is a pure LSTM model composed of two layers. The first *LSTM layer* has return sequences set to *True* so that the layer outputs a vector for each time step instead of a vector at the final input of a sequence. This is needed to be able to stack multiple LSTM layers. Recurrent dropout is added to the layer in addition to a standard dropout layer between the two layers. *Dropout* and recurrent dropout are added to prevent overfitting [92]. This is done by dropping some connections, with the difference between the two being whether connections are dropped between input and output (vanilla dropout) or in the cell state (recurrent dropout). For a full explanation of this topic, the reader is recommended to explore [93], [94]. The dropout layer is followed by another LSTM layer with similar specifications. However, as this is the final LSTM layer, return sequences are turned off. To obtain the desired output shape a *Dense* layer calculates the dot products of the input and squishes it to the desired output shape.

During testing, it was found that the optimizer *Adam* generally produced the best results and is therefore used on all models. Adam is also found by the literature to work well on many different problems [49]–[51]. Furthermore, during early development many models were prone to *NaN* errors when optimizing. As this can often result from exploding gradients, gradient clipping was tested with great success. It is therefore kept for all upcoming models. To further prevent overfitting, *early stopping* is added with a patience of 5 epochs thus the training is stopped if the losses from gradient descent do not improve over five epochs. The second model explored is a CNN-LSTM model. The general model is built as the model described in the previous paragraphs. However, a *Conv1D* layer is added before the first LSTM layer.

Hyperparameter Optimization Using *Keras Tuner*

Hyperparameters can be divided into model and algorithm hyperparameters [37]. Model-dependent hyperparameters define the structure of the model, whilst activation functions and learning rate amongst others, are algorithm-based. Tuning these parameters is seen as crucial to obtaining a well-performing model. Adapting models with respect to the issue and available features is widely accepted as a superior solution compared to preset models [95]. There is a wast number of optimization algorithms, such as *Grid-based*, *Random search* and *Bayesian optimization*. Grid-based is a thorough tuning mechanism, testing out all possible combinations within a search space. Bayesian on the other hand uses probability and an acquisition function for optimization. One of the available optimization softwares is the Keras tuner[96]. The Keras tuner offers three different tuning algorithms, Random Search, Bayesian and Hyperband tuning, with Random search being used in this thesis. Table 3.4 shows the different parameters tuned and their respective search space. Dropout is not confined to a search space. Instead, the function *Boolean* is used, implying that models are tested with and without a dropout of 0.2. The different activation functions are described in Section 2.3.3.

Table 3.4: Hyperparameter space for all proposed models. the red color signify the CNN layer which is only used for the hybrid models.

Layer	Units/ Filters	Activation	Recurrent dropout	Dropout	Kernel size
LSTM 1	[8,128]	*ReLu, Tanh, Sigmoid*	[0, 0.2]	0.2	**-**
LSTM 2	[8,128]	ReLu, Tanh, Sigmoid	[0, 0.2]	0.2	**-**
Conv1D	[8,1000]	ReLu, Tanh, Sigmoid	**-**	**-**	[1,10]

In addition to hyperparameters in Table 3.4, learning rate and clip value are tuned. The intervals for these parameters are $[1 \times 10^{-4}, 1 \times 10^{-2}]$ and [0.001, 0.5] for learning rate and clip value, respectively. Each test is ran with **75 different trials and each trial is ran twice with ten epochs per trial**. The best performing model is then ran again with 75 epochs with the same early stopping as before.

3.4.1 Models with Load and NWP

The first set of trials were created using only residential load and different variants of NWP. The initial model explored included all available NWP variables, giving the machine learning function,

$$\hat{\mathbf{P}}_{\mathbf{A1}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{p}}, \hat{\mathbf{w}}, \hat{\mathbf{c}}),\ with:\ f : \mathbb{R}^{6\times 24} \to \mathbb{R}^{1\times 1}.$$

Here, $\hat{\mathbf{P}}$ denotes the predicted function given by the variables, $\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{P}}, \hat{\mathbf{w}}, \hat{\mathbf{c}}$. The input domain consists of six input features over 24 hours, with an output of 1×1. This mathematical formulation will be continued to be used for describing the rest of the models. Both a CNN-LSTM and an LSTM model were created. From now on, they are named CNN-LSTM-A1 and LSTM-A1. The A will henceforward be used to denote models with only load, and NWP, whilst later additions will be given subsequent letters. The number following the letter will be used as an indication of model iteration in the given development phase.

Recall in Section 3.2.4, cloud cover, and wind speed were found to be erratic in change, and relatively small pattern was found. Two models were again created, CNN-LSTM-A2 and LSTM-A2, given by,

$$\hat{\mathbf{P}}_{\mathbf{A2}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{p}})\ with:\ f : \mathbb{R}^{4\times 24} \to \mathbb{R}^{1\times 1}.$$

This iteration of models were tested to observe whether the model would improve with less noise or if cloud cover and wind were useful features. More models were explored without significant results and are therefore left out.

3.4.2 Models with Qualitative Variables

The second generation of models took advantage of the results found in the previous section. Be aware due to uncertainty given by deep learning models, one can not be sure whether a different set of features than the ones which are taken from Section 3.4.1 would outperform the one used here with additional features. The first set of features used for this generation were the following,

$$\hat{\mathbf{P}}_{\mathbf{B1}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{p}}, \hat{\mathbf{w}}, \hat{\mathbf{c}}, \mathbf{h}, \mathbf{d})\ with:\ f : \mathbb{R}^{37\times 24} \to \mathbb{R}^{1\times 1}.$$

The changes from models A1 is the addition of hourly (h), and daily (d) one-hot encoded variables. The shape of the one-hot encoded variables will be (# **samples**, # **different categories**), where categories, for example, represent each day, *Monday*, *Tuesday*, ..., *Friday*. This increases computational complexity and time. With the Norwegian weather being heavily influenced by seasonal changes, with high loads in the winter and low in the summer, the second version of the model with qualitative variables added monthly and seasonal one-hot encoded variables. The shape of each input tensor is thus (53×24), as seen below.

$$\hat{\mathbf{P}}_{\mathbf{B2}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{p}}, \hat{\mathbf{w}}, \hat{\mathbf{c}}, \mathbf{h}, \mathbf{d}, \mathbf{m}, \mathbf{s})\ with:\ f : \mathbb{R}^{53\times 24} \to \mathbb{R}^{1\times 1}$$

.

3.4.3 Models with Regional Prediction Data

Regional Load Forecasting Model

In a master's thesis performed by Bolstad [82], day-ahead forecasts were performed for NO1 with 168 lags. Using the findings from that thesis, a CNN-LSTM hybrid model was created. Trying to replicate similar results as Bolstad, the goal of this set of models was to investigate if regional load forecast would improve upon the earlier cases. Due to time limitations and objective focus, only one version of the regional load model was tuned. Bolstad performed the forecast using NWP data from a variety of ten cities in the region to achieve a more accurate representation of the region.

$$\hat{\mathbf{P}}_{\mathbf{NO1}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{p}}, \hat{\mathbf{w}}, \hat{\mathbf{c}}, \mathbf{h}, \mathbf{d}, \mathbf{m}, \mathbf{s}),$$

forms the function trying the replicate the regional load with $f : \mathbb{R}^{98\times168} \rightarrow \mathbb{R}^{1\times24}$. Each NWP variable is understood to be a 10×168 tensor to fit all ten cities used in the study. Tuning is performed as explained in Section 3.4 with the same search space as Table 3.4. Since the local data set spans from 2019 to the end of 2021, the test set for regional load forecasting had to be of the same size. This equates to about 38% of the entire available data set for NO1. With both a longer time span and a larger domain, the run time of this model was accordingly longer.

Regional Load Forecasting Applied to Local Load Forecasting

The first version of *generation C* was created to further examine whether regional loads could improve the forecast. C1 uses all available features available in this study.

$$\hat{\mathbf{P}}_{\mathbf{C1}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{H}}, \hat{\mathbf{p}}, \hat{\mathbf{w}}, \hat{\mathbf{c}}, \mathbf{h}, \mathbf{d}, \mathbf{m}, \mathbf{s}, \hat{\mathbf{P}}_{\mathbf{NO1}}) \; with: \; f : \mathbb{R}^{53\times24} \rightarrow \mathbb{R}^{54\times1.}$$

From the results from the first and second generations (see Sections 4.1 and 4.2), it was decided to focus on the LSTM model as it provided better results. A hybrid model will additionally introduce a higher degree of complexity, increasing the general run time. Therefore, a second model was tested as a "bare bones" test where the features found to be less prominent were removed. In this test, only historic load with temperature and regional forecasts were taken advantage of, giving the function,

$$\hat{\mathbf{P}}_{\mathbf{C2}} = f(\mathbf{P}, \hat{\mathbf{T}}, \hat{\mathbf{P}}_{\mathbf{NO1}}) \; with: \; f : \mathbb{R}^{3\times24} \rightarrow \mathbb{R}^{1\times1}.$$

Even though run time optimization is not a objective, having a fluid and quick program will always be beneficial. A model with fewer features can also potentially be less vulnerable to faulty input data. However, if one it can also introduce larger error as the model has less parameters to consider.

3.5 SHAP Development

SHAP's DeepSHAP algorithm mentioned in Section 2.7.2 is initialized using the DeepExplainer object found in the SHAP library [60]. SHAP is chosen over LIME and other XAI tools because of its versatile abilities. First of all, SHAP offers both local and global explanations. Secondly,

the mathematical foundation of SHAP gives confidence in explanations due to the extensive research on Shapley values. On a final note, the SHAP library offers a wide variety of visualization tools, making explanations understandable for domain experts and more inexperienced users. One downfall to SHAP is that it is a work in progress, hence there can arise problems which are not implemented solutions for yet.

The explanations are based on 1000 random samples from the training set, giving a reasonable estimate of the expected values [97]. Calculated Shapley values are given in the same form as the model input, namely, (#**samples**, #**lags**, #**features**). This creates a conundrum where SHAP has multiple inputs to explain per output. An example using only historic load would be written as,

$$\hat{\mathbf{P}}_{\mathbf{h}} = f(P_{h-1}, P_{h-2}, P_{h-3}, ..., P_{h-\#lags}),$$

with h being the current hour. With multiple features and 24 lags, the number of input variables SHAP explains becomes $24^{\#features}$. To see the impact of every singular time variant of a feature, the data can be reshaped into the form $(\#lags \times \#samples, \#features)$. To see the impact of each feature on an output regardless of time lag, one can use the following equation [82],

$$\phi_{\hat{x}} = \sum_{h=1}^{24} \phi_{\hat{x}h}. \tag{3.5}$$

One issue with doing this is that the importance of Shapley values is given in the $\mathbb{R}$ domain, meaning they can be both negative and positive. Equation (3.5) changes this and can lead to explanations becoming leveled. Bolstad [82] solved this by taking the absolute value of each Shapley value in Equation (3.5), giving Equation (3.6).

$$\phi_{\hat{x}} = \sum_{h=1}^{24} |\phi_{\hat{x}h}| \tag{3.6}$$

However, using Equation (3.6) violates one of the local properties of SHAP, namely that the sum of all explanations equals the base value, $\sum \phi = E[f(z)]$. For global explanations using SHAP, the aggregated importance of each feature is calculated.

Local explanations are applied in hopes of interpreting the model's evaluation of short instances or singular outputs. This can be especially useful for cases where the model deviates from the actual load.

3.6 Choice of Performance Measures

Section 2.9 explained four different ways of reviewing the performance of the various models. Each one of them has its advantages and disadvantages. R^2 and MAPE are percentage based measures, whilst RMSE and MAE return the same unit as the given variable it is trying to predict. Since this thesis uses the same dataset for all models, making it possible to compare the models using RMSE and MAE. With RMSE being vulnerable to outliers and MAE susceptible to disregarding the outliers, one should be careful trusting either one too much [98]. One study [99] argues that R^2 is the most informative performance measure due to the other metrics looking at the singular values, hence not informing whether the model performs well or not. At the same time, R^2 is

criticized for not detecting goodness of fit. For example, if the variance is high, the R^2 can become artificially low, even though the model fits the data well. MAPE is not perfect as it struggles if any value is 0, as discussed Section 2.9.3.

For this thesis, it is decided to use MAPE as the main indicator. As a precaution, each data set is investigated for 0-values. The other metrics will nevertheless be used to gain insight into the performance. In addition to these measurements, visual confirmation will be used to validate each model.

Chapter 4

Results and Discussion

In this chapter, the results of the different models are presented together with local and global SHAP explanations. Results are discussed simultaneously for simplicity, and a final discussion is found in Section 4.5. The chosen hyperparameters for the models presented in this section is found in Appendix A.

4.1 Load and NWP

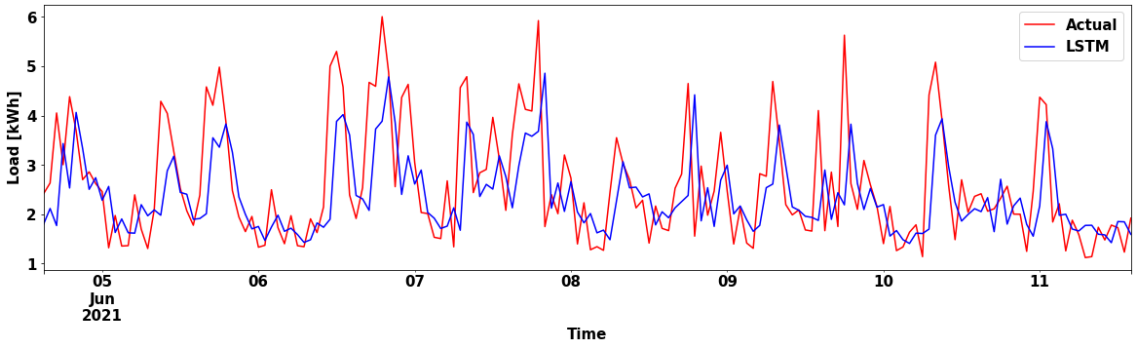
The first set of models was created using only different combinations of NWP data (see Section 3.3.1). The four different models investigated and their total performance measures are shown in Table 4.1. Comparing the different models, it can be seen that the A1 models, with all available NWP data outperform the models with only temperature, humidity and pressure. This can suggest that the DL model can extract information from cloud cover and wind speed, even though it is hard for humans to find. Secondly, the difference in performance between the hybrid model and the LSTM model is relatively small. With a slightly lower MAE, but a higher RMSE, one could argue that LSTM-A1 has more significant absolute errors, but follows the median better than the CNN-LSTM-A1. Looking at the relative metrics, R^2 and MAPE, the two perform similarly. CNN-LSTM-A1 is slightly better according to R^2, whilst MAPE indicates that LSTM-A1 is better.

Table 4.1: Performance measures for first generation models. The best performing measures are highlighted.

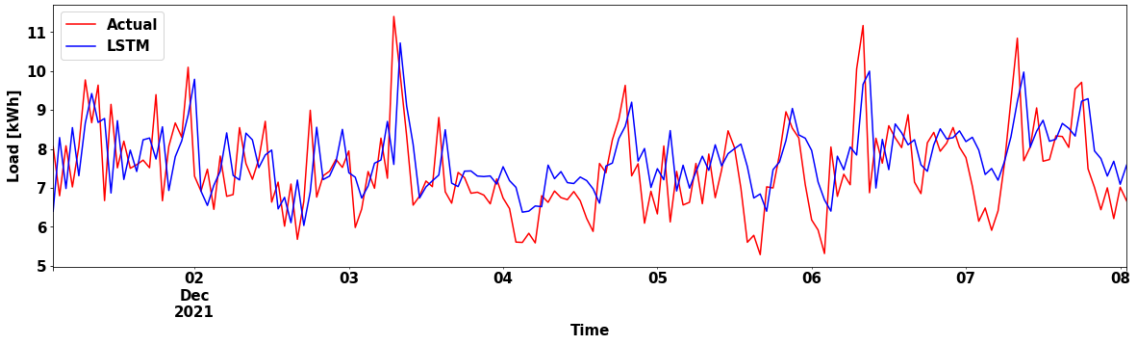
Model Generation A	$\mathbf{R^2}$[%]	**MAPE**[%]	**RMSE [kWh]**	**MAE [kWh]**
LSTM-A1	70.17	**24.31**	1.028	**0.728**
CNN-LSTM-A1	**71.49**	27.77	**1.005**	0.7515
LSTM-A2	70.77	26.67	1.017	0.7501
CNN-LSTM-A2	69.79	27.13	1.034	0.765

Calculating the best and worst week for LSTM-A1 according to MAPE, it is found that this is the 1^{st} to 8^{th} of December and the 4^{th} to 11^{th} of June, respectively. Visually, the relative peaks in June are more extensive compared to December. One could argue that MAPE returns the given week as the poorest due to punishing negative errors more than positive errors. In the summer, the electric load is generally low, and the model underestimates peaks. During the winter months,

the load is generally much higher. It is never lower than 5 kWh/h, in the best performing month. However, the model does seem to have the opposite problem over this week as to the summer week displayed in Figure 4.1a. Hours with a lower load compared to the norm have higher deviations. If this has a large impact on the calculated MAPE is challenging to prove.



(a) The worst predicted week for LSTM-A1 according to MAPE (28.30%).



(b) The best predicted week for LSTM-A1 according to MAPE (12.26%).

Figure 4.1: Comparison of the best and worst predicted weeks according to MAPE.

From Figure 3.7, it was observed that the load during the weekends is lower compared to weekdays. To see if this has impacted the performance of each day is calculated in Table 4.2. Surprisingly, performance is relatively similar for all days. Maybe even more surprising is that Saturday and Sunday perform a little better than the others. One could argue that this is because the load is more stable the entire day compared to workdays with morning and afternoon peaks. However, from seasonal performance, it was found months with low loads performed worse than winter months. These finds further emphasize that the model mainly struggles with sharp load changes, especially morning and afternoon peaks.

Table 4.2: Performance of LSTM-A1 for each weekday.

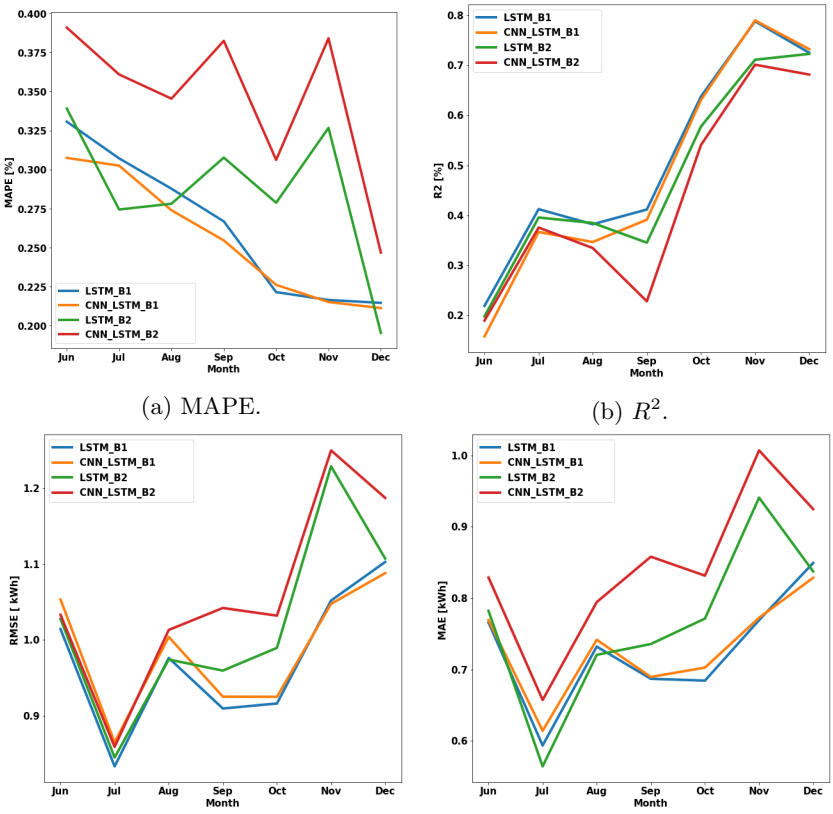
Metric	**Mon**	**Tue**	**Wed**	**Thur**	**Fri**	**Sat**	**Sun**
R^2	71.88	66.77	69.40	66.91	68.69	**0.7297**	71.15
MAPE	24.59	24.02	25.04	24.20	24.71	24.56	**23.00**
RMSE	1.111	1.074	1.016	1.045	1.002	**0.8970**	1.036
MAE	0.7927	0.7650	0.7373	0.7585	0.7109	**0.6329**	0.6966

4.2 Qualitative Variables

Table 4.3: Performance measures for second generation models.

Model Generation B	R^2[%]	MAPE [%]	RMSE [kWh]	MAE [kWh]
LSTM-B1	**73.17**	26.22	**0.975**	**0.7252**
CNN-LSTM-B1	72.41	**25.48**	0.9883	0.7305
LSTM-B2	70.38	28.42	1.024	0.7637
CNN-LSTM-B2	67.93	34.39	1.066	0.8427

Utilizing the best results and findings from the first generation of models, it was tried to improve the model by incorporating qualitative variables. First, by adding hourly and daily one-hot encoded values to determine whether sharp load changes (found in Section 4.1 can be detected by information about the hour. The day of the week is implemented to observe if it impacts the results found in Table 4.2. The second instance of models with qualitative variables added seasonal and monthly features the first gen proved to work best during winter. The performance of B1 and B2, both LSTM and CNN-LSTM, is shown in Table 4.3. Looking purely at the performance metrics, the pure LSTM models outperform the hybrid models, again. Secondly, the models, including season and month, performed worse compared to those without, with a 3.81% (for LSTM) and 6.18% (for CNN-LSTM) decrease in accordance with R^2. Interestingly, the models proposed with seasonal and monthly characteristics did not only perform worse than B1, but they performed worse than all models presented in the first generation. If this is a fluke during development or adding too much information to the model will be discussed in Section 4.5.



(a) MAPE.

(b) R^2.

(c) RMSE.

(d) MAE.

Figure 4.2: Performance per month for the second generation of models.

To see if qualitative measures impacted performance for each day and month, performance metrics are calculated and visualized in Table 4.4 and Figure 4.2. Remember that MAPE, RMSE, and MAE are to be minimized, whilst the performance of R^2 is best at 100%. As seen earlier, CNN-LSTM-B2 generally performed worse than the other models proposed during development. This is further highlighted in Table 4.4, where it performs worse than LSTM-B1 and LSTM-A1 (from the previous generation). However, comparing LSTM-B1 to LSTM-A1, it is seen that R^2 improves for all days, implying that adding weekdays and hours improved performance. It is observed that for all three models, Thursday is significantly worse than to the other days. This is most likely down to divergence in residence habits which the model has not entirely figured out. Monthly and seasonal variable were added to B2 to highlight month changes to the models. Figure 4.2 shows that this is not the case. The B1 drafted models outperform their B2 counterparts for nearly all months in all statistics. The only exception is that LSTM-B2 performed surprisingly well for the summer months. The Figures 4.2a-4.2d make it clear that even though the four models produce different results, they all follow a similar pattern. With an increasing load during the winter months, RMSE and MAE increase considerably. However, the percentage-based metrics decrease. These trends indicate that even though the error measurements increase during the winter, the total load increases more, reducing the percentage error.

Table 4.4: Performance of LSTM-B1 and CNN-LSTM-B2 for each weekday.

Model	**Metric**	**Mon**	**Tue**	**Wed**	**Thur**	**Fri**	**Sat**	**Sun**
LSTM-B1	R^2 [%]	**76.92**	70.28	74.15	67.90	72.44	74.30	72.56
	MAPE [%]	25.02	27.80	25.78	27.45	**25.29**	25.99	26.21
	RMSE [kWh]	1.007	1.016	0.9336	1.030	0.9432	**0.8747**	1.010
	MAE [kWh]	0.7567	0.7750	0.7046	0.7813	0.6900	**0.6410**	0.7287
CNN-LSTM-B2	R^2 [%]	72.77	70.27	70.90	65.53	69.88	70.083	69.52
	MAPE [%]	28.18	28.10	28.42	28.63	27.31	28.50	29.81
	RMSE [kWh]	1.093	1.0162	0.9905	1.067	0.9859	0.9438	1.065
	MAE [kWh]	0.8294	0.7686	0.7396	0.8003	0.7224	0.6993	0.7871

The two highest achieving models for the second generation (LSTM-B1 and CNN-LSTM-B1) are further inspected in Figure 4.3. The same week is predicted in Figure 4.1b, with the main difference between the two plots being that the second generation seems to follow the peaks and dips better than the previous generation. This is especially apparent in the dips on the 5^{th} and 6^{th} of December. However, the same figure shows how the CNN-LSTM-B1 and LSTM-B1 exaggerate more compared to LSTM-A1. On the 2^{nd} of December (which coincidentally is a Thursday), it is seen that B1 models predict values close to 11 kWh, whilst the actual load only amounted to 8.55 kWh. Even though these are only two examples, they are found throughout the data sets and are also represented in the performance metrics. This could be down to generation A models being more overfit than the second generation.

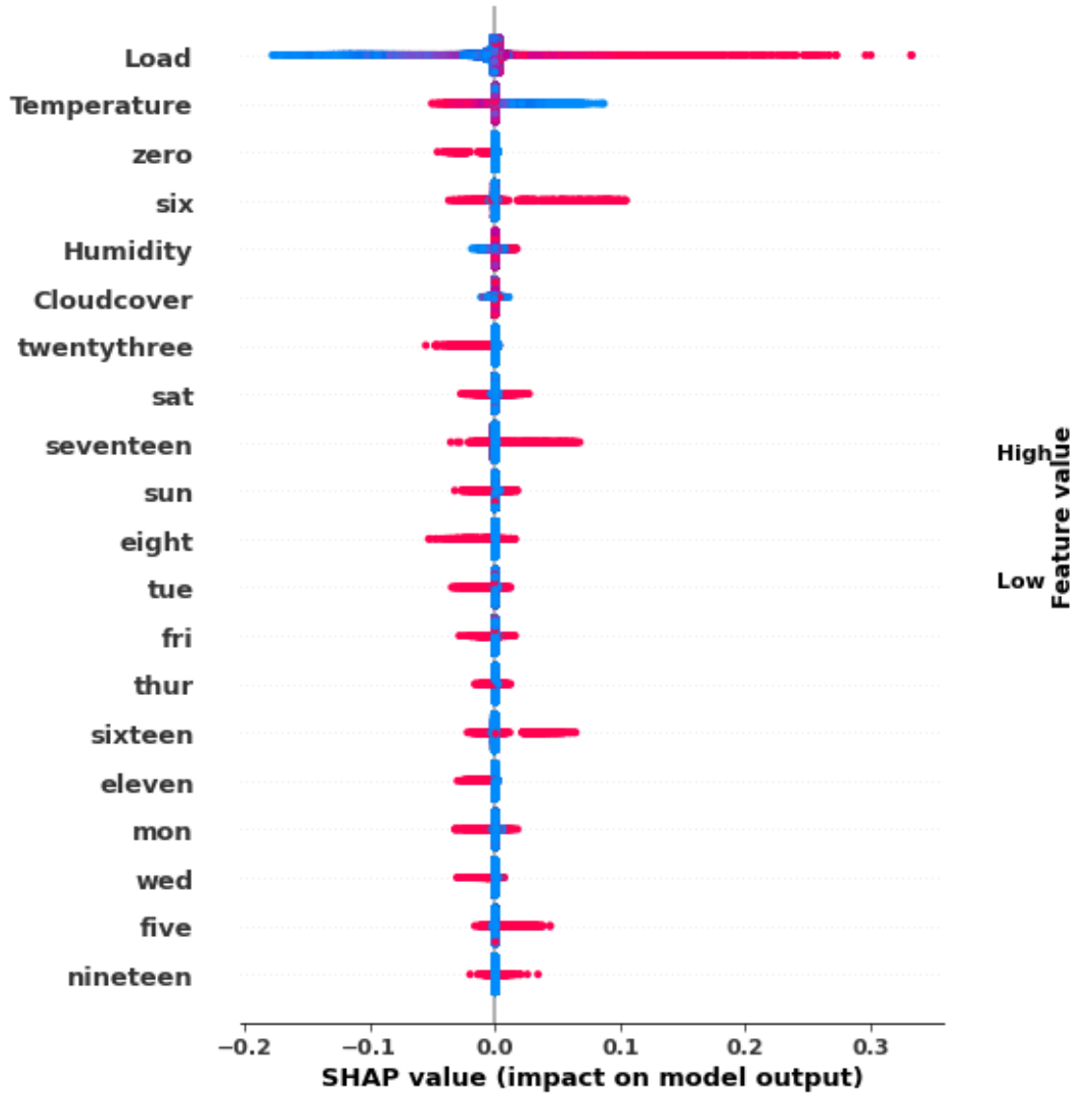


Figure 4.4: Summary plot of model LSTM-B1.

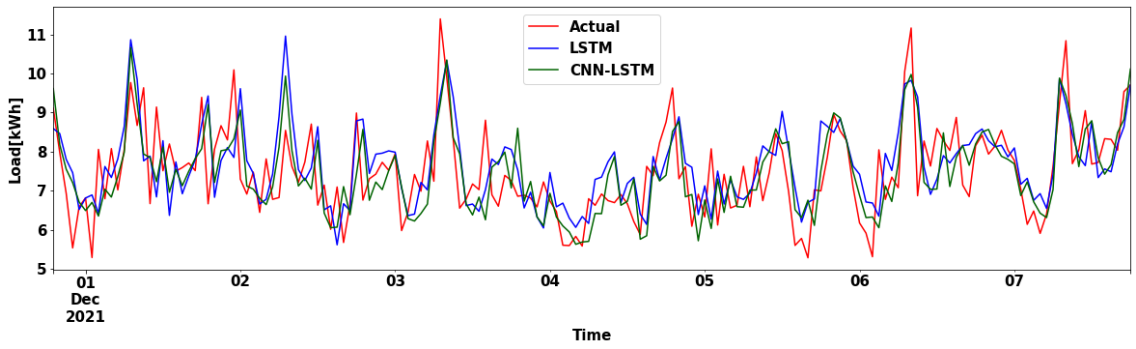


Figure 4.3: Comparison of LSTM-B1 model and CNN-LSTM-B1 model for second generation models.

4.2.1 SHAP Explanations

The two best-performing models from generation B are explained using DeepShap to find important features and patterns to understand the model's workings better. Figures 4.4 and 4.5 are summary plots depicting the importance of the 20 most important features for the LSTM-B1 model and CNN-LSTM-B1, respectively. A colored dot represents each explanation (feature and time variable). The color identifies the feature value. For example, a relatively high ambient temperature will be a shade of red, whilst colder temperatures are accordingly blue. Be aware that qualitative variables will only be clear red or blue due to being one-hot encoded. The x-axis describes the SHAP value for each point. A high absolute SHAP value means that that sample is of great importance. If it is negative, it is understood to decrease the expected output compared to the base value.

The first observation for both figures is that the most important features are historical load and temperature forecast. Unsurprisingly it is seen that low historic loads impact the model output negatively, and that high loads usually affect the model output positively. However, it is observed some points close to the origin (0.0) are the opposite color of the general pattern. This is most likely due to there being 24-hour lags, meaning that a high peak yesterday morning does not imply that there will be a high peak the following noon. Luckily it seems like the model has understood this, as most of these points have relatively little impact on the model. Temperatures are seen to have the opposite color pattern to load, where cold temperatures give higher output, which reasons with logic. Surprisingly, cloud cover is given much higher importance than initially thought during visualization. For some reason, the CNN-LSTM model is heavily reliant on load as every other feature has close to zero impact on the model output. The pure LSTM model, on the other hand, is more impacted by multiple features. Especially the one-hot encoded variables for peak hours have high SHAP values, indicating that the model has understood the connection between peaks and hours. There could be that the hybridization of the CNN and LSTM confuses SHAP or that the convolutions in the CNN emphasize the load more.

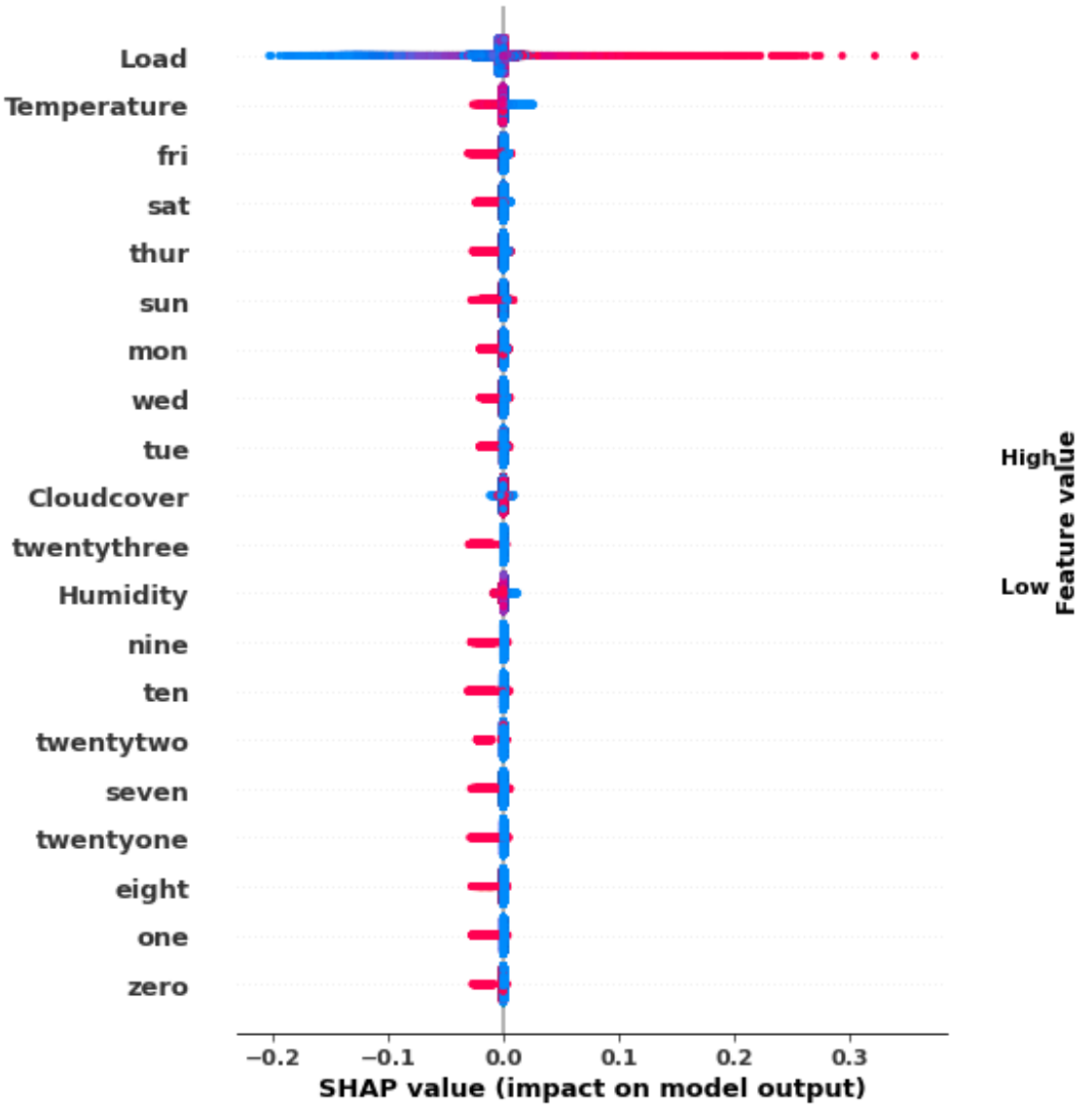


Figure 4.5: Summary plot of model CNN-LSTM-B1.

4.3 Regional Prediction Data

After tuning the regional forecast ($\hat{\mathbf{P}}_{\mathbf{NO1}}$) once, it achieved these results, **MAPE: 5.17 %**, $\mathbf{R^2}$**: 95.44%**, **RMSE:** 275.27 kWh, and **MAE:** 204.95 kWh. Comparing these metrics to the literature (see [82]), they are inferior, but seen as sufficient to be used in the proposed models. Figure 4.6 depicts the entire period during which the regional load forecast is performed.

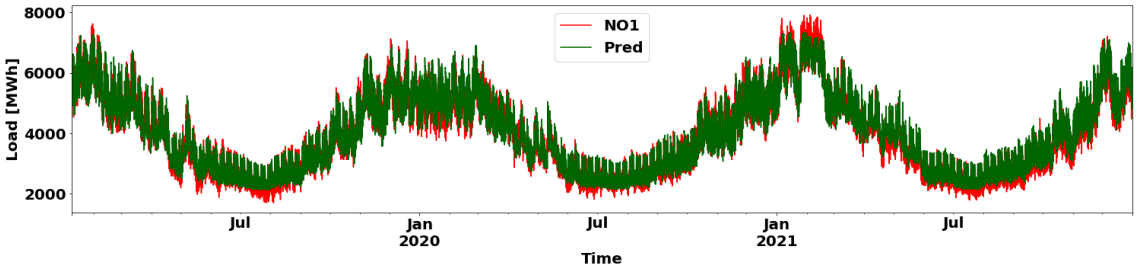


Figure 4.6: Regional forecast compared to actual load, from January 2019 - December 2021.

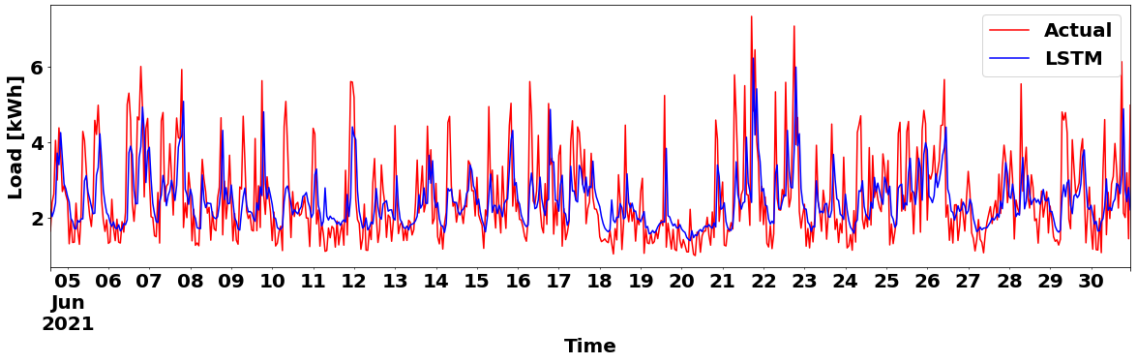
LSTM-C1 is the first model created during the development of the third generation. Recall that C1 comprises all qualitative (Section 3.3.2) and quantitative (Section 3.3.1) features explained in their respective sections. As with the B2, the models with abundant features perform subpar. From Table 4.5, it is seen that LSTM-C1 achieves lower scores than the LSTM-B1 for all metrics. As the pattern of the previous models favored fewer features, the bare bones model, LSTM-C2, as described in Section 3.4.3, was tested. LSTM-C2 outperforms all earlier models, as seen in Table 4.5. The only exception is the MAPE of LSTM-A1, which performed 0.35 percentage points better. Since LSTM-C2 has removed qualitative features altogether, it was interesting to examine how it would affect monthly performance compared to the second generation. Generally, LSTM-C2 did better in most months, as expected, since it did better overall. However, LSTM-C2 seems to excel during the winter compared to B1. This could be down to it putting higher importance on temperature. Using a regional load forecast can also have impacted the accuracy during winter, since the regional forecast is heavily impacted by weather and season, the model could indirectly get a sense of seasonal changes and other parameters that influence load. One example is holidays and weekends, which Bolstad [82] found to impact load profile and peaks. Figure 4.7 shows the July and December for LSTM-C2. In July, it is seen that the forecast generally struggles with peaks. In addition to peaks, the dips seem to be exaggerated by the forecast. These peaks are especially prevalent in the periods $12^{th} - 16^{th}$ and $26^{th} - 30^{th}$. For December, the model generally follows the actual load better and struggles less with sudden changes. For periods with sharp differences in load, most likely due to homeowners being away, have a slight divergence, but the forecast seems to understand this and thus correct itself quickly. All monthly results for C2 are shown in Table 4.6.

Table 4.5: Performance measures for third generation models.

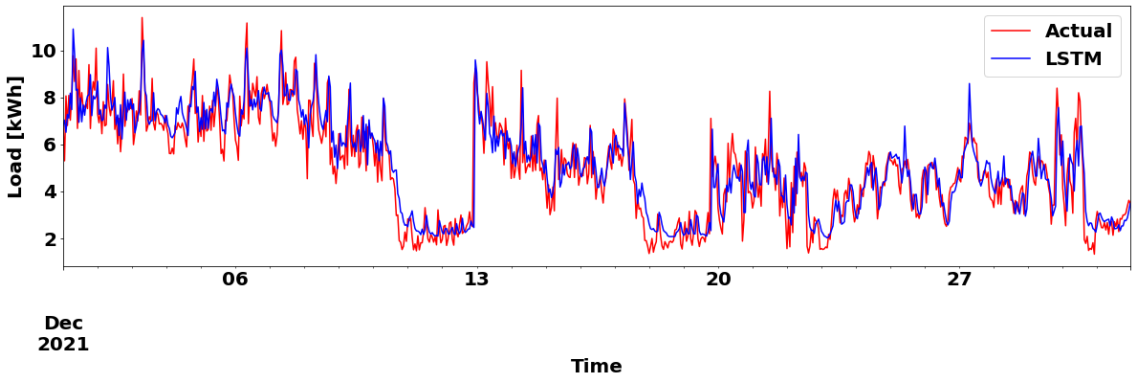
Model Generation C	**R^2**[%]	**MAPE** [%]	**RMSE** [kWh]	**MAE** [kWh]
LSTM-C1	70.78	29.12	1.017	0.7671
LSTM-C2	**74.56**	**24.66**	**0.949**	**0.6942**

Table 4.6: Monthly results for C2.

Metric	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**
R^2[%]	22.03	42.00	35.31	41.30	68.05	**79.04**	78.36
MAPE [%]	30.36	28.83	26.28	24.90	20.82	24.24	**17.97**
RMSE [kWh]	1.013	**0.8276**	0.9986	0.9083	0.8605	1.0471	0.9790
MAE [kWh]	0.7388	**0.5707**	0.7166	0.6736	0.6407	0.78979	0.7380



(a) The month of June, red is actual load and blue is LSTM-C2 forecast.



(b) The month of December, red is actual load and blue is LSTM-C2 forecast.

Figure 4.7: Excerpt from results using model C2.

4.3.1 SHAP Explanations

From the summary plot in Figure 4.8, the feature importance is plotted. As with previous results, historical load and temperature are the two most important features. Surprisingly, LSTM-C1 seems to have put a lot of confidence in the seasonal parameters, autumn, winter, and summer. For some reason, even the zero valued one-hot encoded values have been given some importance, which is prominent in the spring and autumn features. It is counter-intuitive for the forecast to put weight on zero valued features as another feature in the same category is 1. It could have an effect on predictions where lagged values are in the margins between two categories. One categorical value seen to increase the impact on the model output is the hourly value for 06:00. This is also the hour that usually has the highest peak, indicating that the model has found a connection between the two. In Figure 4.8, **LSTM** represents regional forecasts from NO1. The forecast is weighted as the fourth most important feature, confirming the suspicion that regional load significantly impacts development.

The importance of the load justifies the inclusion of the regional load forecast in LSTM-C2 during the SHAP review for LSTM-C1. A summary plot is created for LSTM-C2 as well and is shown in Figure 4.9. Here, it is seen that regional load forecast (LSTM) is actually given more significance than historical load and temperature. Furthermore, the color pattern given to the LSTM points is similar to temperature and not historical load. In Section 4.2.1, it was argued that their respective impact on future loads gave the color pattern of load and temperature. Why relatively low regional values (blue dots) increase expected model output and high regional forecast (red dots) is hard to pinpoint. One reason could be that old lags are given more weight compared to lags closer to

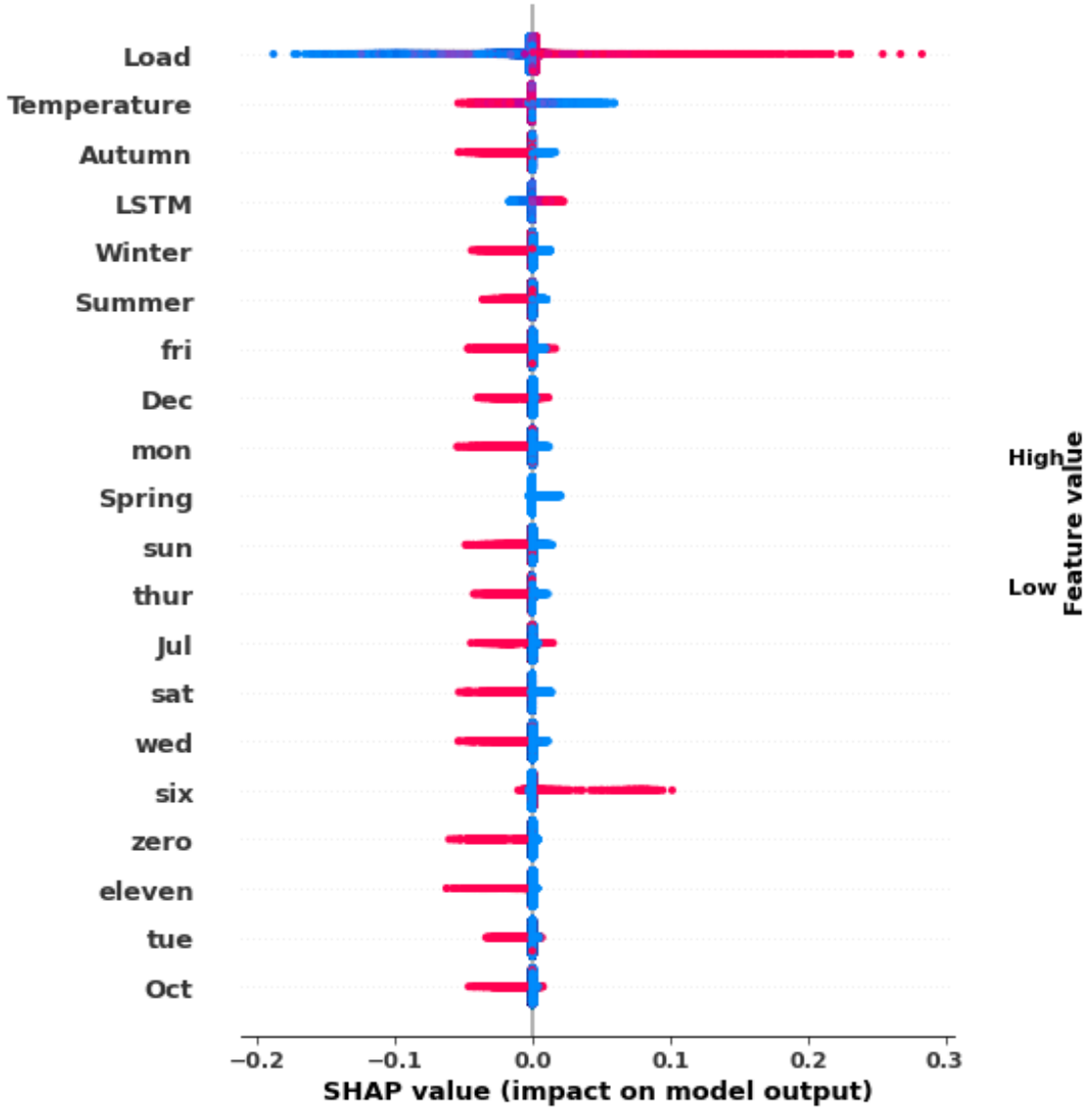


Figure 4.8: Summary plot of LSTM-C1.

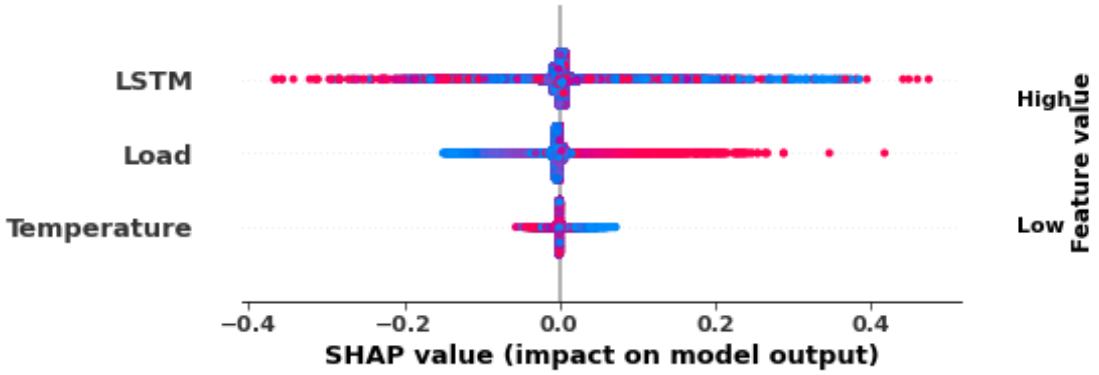


Figure 4.9: Summary plot of LSTM-C2.

the actual forecast horizon. Another reason contributing to the difference is that the residential load could have a slightly different pattern compared to the load. This was explored in Figure 3.6, where the two loads were seen to have similar structures, but the residential loads are inconsistent.

4.4 Local Explanations

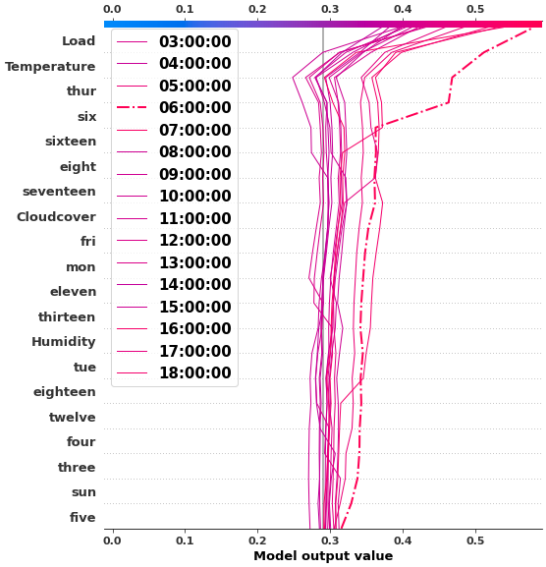
A couple of local explanations are highlighted in this section to achieve a more comprehensive understanding of the SHAP values and their respective explanations.

4.4.1 Single Day Explanation

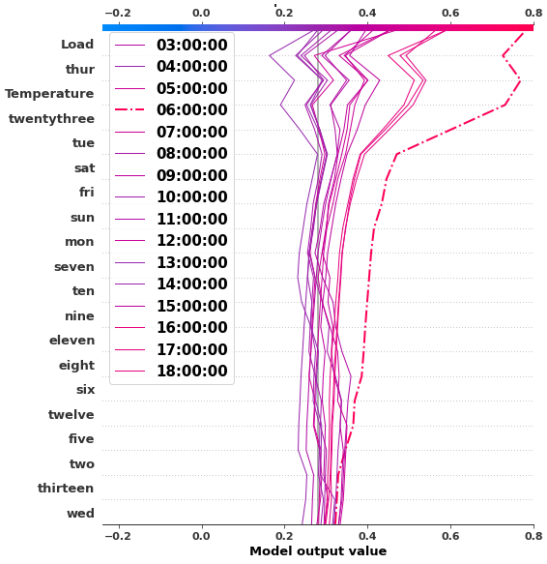
Using the aggregated SHAP equation (Equation (3.5)) and plotting a time span in a decision plot, one can obtain insight into how each feature impacts the given forecast over a short time period. The x-axis displays model output, with the y-axis representing each feature. The plot is read from the bottom to the top, as all lines begin at the base value and each feature either pushes it above or below the base. Figure 4.10 depicts two decision plots for generation B1, one for each model type. Below the decision plots, the load profile for the time period is plotted for the two models. The illustrated time frame is from 03:00 to 18:00 on December 9^{th}. Initially, one can observe a clear pattern of most features' impact for each hour. For the LSTM model, most predictions are lowered by it being Thursday and then heavily pushed above the base value by the historical load. The other features seem to have little to no impact on all outputs. For close to all outputs, there is a tiny push in either direction for the equivalent hour. This is most obvious for 06:00, where the one-hot encoded value for time pushes the output towards a higher output. 06:00 is highlighted with a dashed line.

Furthermore, the highlighted hour has a much higher output than the other hours, complying with former finds about peak hours. Be aware that the decision plot is plotted from inputs, meaning that 06:00 will forecast 07:00. CNN-LSTM-B1 has a similar feature pattern as LSTM-B1. However, CNN-LSTM-B1 puts a higher emphasis on it being a Thursday and that it should decrease the output. From the load profile in Figure 4.10c, it is seen that both models are accurate for the morning peak at 07:00, which is explained by the highlighted lines in the decision plots. The general load profile for the given day is, however, quite erratic, and both models are seen to

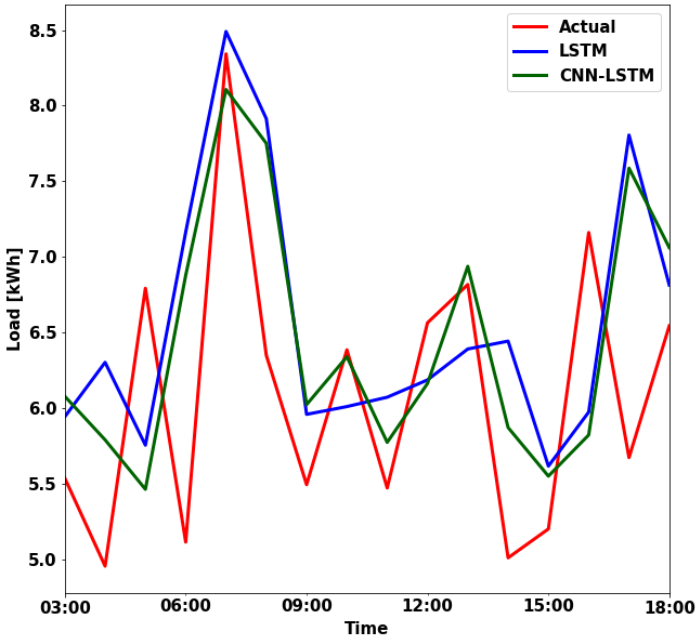
deviate for multiple hours, especially at 17:00.



(a) Decision plot for LSTM-B1.



(b) Decision plot for CNN-LSTM-B1.



(c) Load profile for the time period 03:00 to 18:00.
$\mathbf{MAPE}_{LSTM} = \mathbf{15.54}\%$,
$\mathbf{MAPE}_{CNN-LSTM} = \mathbf{13.36}\%$.

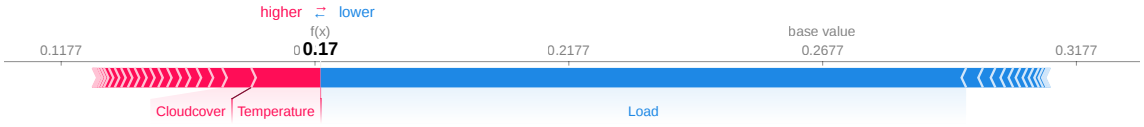
Figure 4.10: Decision plot for B1 models and load profile for December 9^{th}.

Single Hour Deviation

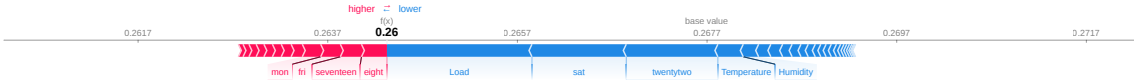
The largest deviation is chosen for further examination to understand why forecasts misbehave. This is found to be December 12^{th} at 22:00, where the prediction underestimated with 5.46 kWh. Figure 4.11 depicts four different figures, three force plots (Figure 4.11a, Figure 4.11b, and Figure 4.11c), and a load profile (Figure 4.11d) for the given evening. The load profile shows a sudden change in the load between nine and ten o'clock that evening. In Figure 3.7, it was shown that

the evening load generally happens at 18:00. A deviation of this magnitude could be because the residents have been away and then returned home and turned on heating and other appliances.The available features show no obvious warning signs if this is the case. To see what the model has emphasized, SHAP is used, specifically force plots [100]. Figure 4.11a is the aggregated SHAP values for each feature using Equation (3.5). A force plot shows the impact of each feature compared to the base value, $E[f(z)]$. Load is seen as the major impact, pushing the output below the base value. Temperature and cloud cover, on the other hand, increases the output. Because the aggregated SHAP values are vulnerable to equal out opposite impacts the SHAP values for lags $T-24$ and $T-1$, visualized in Figure 4.11b and 4.11c, respectively. For the inputs 24 hours ahead of time (Figure 4.11b), the three most important features are load, Saturday, and twenty-two. With December 12^{th} being a Sunday, it makes sense that the previous day has some impact and lowers the output, as weekend loads were found to be lower than weekdays. Additionally, the model has also identified that 22:00 is a time when the load is usually lower, since it also lower the output.

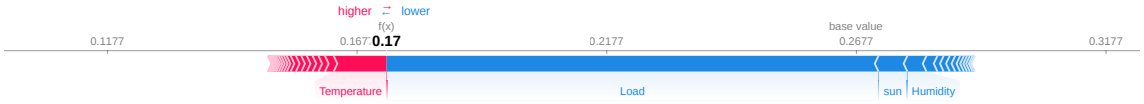
The explanations for inputs with an hour lag show that load and temperature are the most important, equivalently to Figure 4.11a. Understandably the load is an essential feature throughout the explanations, as seen in the global explanations. With the temperature being such an important feature to pushing the value higher, it is natural to suspect that the temperature is relatively high. This is confirmed via analysis in retrospect, with the average temperature of December being $3.49°C$ colder and the average temperature for 22:00 in December being $3.52°C$ colder.



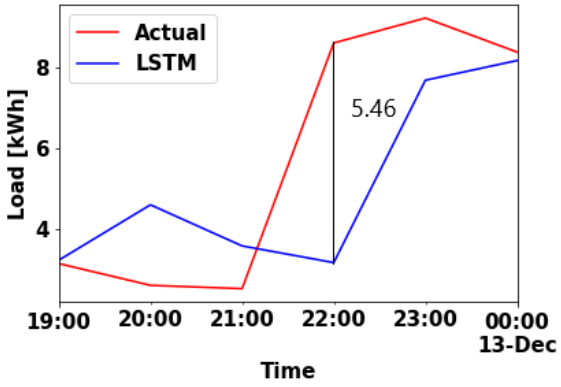
(a) Sum of all SHAP values for T=22, on 12^{th} December.



(b) Force plot for SHAP values from $T - 24$.



(c) Force plot for SHAP values from $T - 1$.



(d) Difference in actual load and predicted load.

Figure 4.11: Force plot for 12^{th} December 22:00.

4.5 Discussion of Results

With any machine learning problem, there will be issues and challenges. During development, it was found that it is not always the case that a model will improve by feeding more information. Contrary to the literature, LSTM outperform the hybrid CNN-LSTM model on a general basis. It must be emphasized that since developing a model using LSTM is the fundamental goal of this thesis, and more time has been used in this process. The structure of the CNN layer could have experimented with more to achieve its full potential. Others are left to explore whether an improved model could be created using more sophisticated LSTM and CNN methods. To the author's surprise, the best-performing model was one of the simplest (LSTM-C2).

With 11 different available features, countless combinations could be explored to find a better solution compared to LSTM-C2, which only used three. XAI and SHAP were applied to try to increase accuracy. There is a possibility that another combination would yield better results. However, from exploration, this is seen as less likely. Another way to improve the model is a broader hyperparameter search with a more extensive search space. The results show how the performance fluctuates with the months, with the best month being 40.9% better than the worst, according to MAPE. There are many factors contributing to this. First of all, during the summer, the loads are seen to be more irregular, becoming harder to forecast. As explained earlier, the forecasts

were usually prone to underestimate the load, which MAPE penalizes heavier. Even though the percentage-based metrics favor the winter months, it is seen that especially July performs better according to numerical metrics. Comparing the RMSE between June and December, it is seen that for this metric, December is only 3.36% better. This shows how singular metrics can deceive the results. From this, one could argue that percentage metrics will favor the winter as the general load is higher and the error is the same.

Comparing the results with similar studies in the literature is difficult due to discrepancies in objectives and data. For example Alhussein et al. [101], achieved a MAPE of 42.85% for a proposed CNN-LSTM model. However, this model used 12 lags instead of 24. In another article by Shaqour et al. [102], different subsets of dwellings were forecast using Deep Neural Networks (DNNs) and got a MAPE of 20.8% with three residences. Both from that study and this thesis, it has been explained that with multiple residences, the different peaks will be evened out, creating a more stable pattern. Finally, Abdel-Basset et al. [103] proposed a STLF-net on two different residential data sets([104], [105]), achieving MAPEs of 38.24% and 19.49%. The two data sets included a different number of features, with many indoor sensors available. Comparing the results from this thesis with the literature mentioned above, the proposed model performs equally well. However, it is clear that improvements can be made with more time and data at hand.

From inspection of local explanations, a deeper understanding of the model's inner workings is given. The force plots in Section 4.4.1 showed how the model prioritized different features depending on which lag it was. From this example, it was also seen that the SHAP value for the lag closest to the forecast hour resembled the aggregated SHAP value. If this is a coincidence or the norm was not explored enough to answer. It is also unsure whether the chosen time was the wisest for further investigation due to being an outlier where the features seemingly could not identify such a leap in load. On the other hand, it is important to be aware of these moments to be able to design a model which could handle these events. One possibility could have been to increase the number of lags in hopes of allowing the model to understand longer connections such as vacations. Another option could have been to implement calendar variables, like Easter and other holidays, where it assumed that most of these outliers stem from. On the other hand, decision plots were helpful for seeing connections in features over time. The decision plots in Figure 4.10 also revealed how the model identified Thursdays as a feature that generally lower model output. With Thursdays being the worst performing day (as seen in Table 4.3), the models' interpretation of Thursdays could contribute to the poor performance.

Chapter 5

Conclusion and Future Work

The contributions of this thesis consisted of creating an LSTM forecasting model using XAI on a Norwegian residence. A preliminary set of models were created using standard forecasting techniques. Implementation of SHAP proved to increase the performance as presented in Chapter 4. The complexity of DL models is one of the most significant downfalls of the current evolution. However, with SHAP, the mist surrounding the model was partly cleared. Using global explanations, the importance of different features was found. This allowed the removal of redundant features and seeing what different models emphasize. Local explanations gave a deeper understanding of outlying forecasts. The use of SHAP thoroughly improved the model and made it easier for domain experts and the general public to interpret. Furthermore, even though residential load patterns deviate from aggregated regional use, it was found to be an integral feature for the selected model.

It is evident that with the evolution in AI and increased data availability, residential load forecasting will be more accessible for system operators and house owners. With an ever-increasingly pressured grid and a rising price trend, the possibility of forecasting will relieve the most prominent peaks. In cooperation with automation, forecasting can be used as a supplementary tool to simplify house owners' life. Furthermore, aggregating multiple households and neighborhoods can give the DSOs meaningful insight. There is a significant focus in media and research on how to produce more renewable energy. However, it will be just as important to **reduce** excessive use of power in the future.

Implementation of XAI is proven as a vital tool to give domain experts insight during model development. In this thesis, using SHAP increased performance by identifying feature importance and outliers. It proved especially effective at separating important features from unimportant. For domain experts, such as DSOs, XAI will provide justification for predictions which will, in turn, provide confidence in the said forecast. For this, local explanations will be particularly useful for giving reassurance and inspecting outliers. Furthermore, XAI can potentially be helpful for end-users and shareholders if explanations are incorporated in an understandable fashion. It will therefore be essential to structure the output accordingly. The mathematical foundations of XAI and SHAP were not thoroughly investigated in this thesis, and the correctness of all outputs can therefore not be validated. However, as the model was improved in this process, the value of XAI in this context is clear. Given projections of XAI being fundamental for trustworthy AI, more research within energy forecasting is needed further to gain an understanding of the choice of background data.

As residential load profiles are unique and highly volatile to sharp changes, it would be interesting to further look into the following topics with the use of XAI:

(1) Increasing the amount of data sets, both with multiple households and over numerous years. Increasing the time frame would make it possible to train the model thoroughly and have a test set of at least one year, making it possible to observe the complete seasonal statistics. With data from multiple households, it would be possible to research if a model based on a singular household could be retrained to work on others. Another solution could be to train the model on multiple distinct households and then test it on singular residences.

(2) Using data with a higher time resolution. With an hourly resolution extracted from Elhub or similar services, it could be argued that some information is lost. This is because hourly aggregated peaks become steeper in contrast with data with a higher resolution. It can be argued that higher resolution would require more computational power, memory to store data, and the forecasts to be more accurate. This is a trade-off that could be a subject for further examination.

(3) Experiment with a more extensive variety of features. With the emergence of cheaper sensor technology, it would be natural to adopt more of these to examine whether they could significantly impact the forecasts. One example where this is available is Hebrail et al. [104], having temperature sensors in multiple rooms and with multiple submeters. This being a more expensive investment, the argument for if this is necessary has to be stated. As with this respective LSTM setup, having an abundance of features was found to create obscurities in the model. However, many of these features were qualitative, whilst quantitative factors could increase accuracy, being less susceptible to becoming too general. At the same time, sensors could lead to inaccuracies if the resolution is low. With the increasing share of prosumers on the market, it would be interesting seeing the impact and possibilities of performing such a study with loads and PV production.

Bibliography

[1] E. Henriksen, 'Electrical forecasting using xai norweigan residential buildings', Department of Electric Power Engineering Technology, NTNU – Norwegian University of Science and Technology, Project report in TET4525, Dec. 2021.

[2] Enerdata. (2021). 'Final electricity consumption', [Online]. Available: https://eneroutlook.enerdata.net/forecast-world-electricity-consumption.html (visited on 18/11/2021).

[3] D. Spilde, L. E. Hodge, I. H. Magnussen, J. Hole, M. Buvik and H. Horne, 'Strømforbruk mot 2040', *Rapport (Norges vassdrags-og energidirektorat)*, vol. 22, 2019.

[4] S. Statnett, *Nettutviklingsplan 2021*, 2021.

[5] E. facts Norway. (2021). 'Electricity production energifakta', [Online]. Available: https://energifaktanorge.no/norsk-energiforsyning/kraftforsyningen/ (visited on 09/11/2021).

[6] K. B. Debnath and M. Mourshed, 'Forecasting methods in energy planning models', *Renewable and Sustainable Energy Reviews*, vol. 88, pp. 297–325, 2018.

[7] H. Wang, N. Zhang, E. Du, J. Yan, S. Han and Y. Liu, 'A comprehensive review for wind, solar, and electrical load forecasting methods', *Global Energy Interconnection*, vol. 5, no. 1, pp. 9–30, 2022.

[8] G. Yang, Q. Ye and J. Xia, 'Unbox the black-box for the medical explainable ai via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond', *Information Fusion*, vol. 77, pp. 29–52, 2022.

[9] J. Kruse, B. Schäfer and D. Witthaut, 'Revealing drivers and risks for power grid frequency stability with explainable ai', *arXiv preprint arXiv:2106.04341*, 2021.

[10] K. Zhang, P. Xu and J. Zhang, 'Explainable ai in deep reinforcement learning models: A shap method applied in power system emergency control', in *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*, 2020, pp. 711–716. DOI: 10.1109/EI250167.2020.9347147.

[11] M. Kuzlu, U. Cali, V. Sharma and Ö. Güler, 'Gaining insight into solar photovoltaic power generation forecasting utilizing explainable artificial intelligence tools', *IEEE Access*, vol. 8, pp. 187 814–187 823, 2020. DOI: 10.1109/ACCESS.2020.3031477.

[12] A. B. Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins *et al.*, 'Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai', *Information Fusion*, vol. 58, pp. 82–115, 2020.

[13] P. Støa, N. A. Røkke, S. Ø. Størset, R. Spooren, G. A. J. Knutstad, N. Dahl, D. Akporiaye, R. Bredesen, G. Sand, E. G. Tveten *et al.*, 'Energi og industri-mulighetsrom verdikjeder-nho veikart for fremtidens næringsliv', *SINTEF Rapport*, 2019.

[14] I. E. Haukeli, A. Stavseng, D. Spilde, J. Hole, E. Skaansar, C. H. Skotland, I. Holm, M. H. Heien, K. R. Verlo and V. Røv, 'Elektrifiseringstiltak i norge hva er konsekvensene for kraftsystemet?', *Rapport (Norges vassdrags-og energidirektorat)*, vol. 36, 2020. [Online]. Available: https://publikasjoner.nve.no/rapport/2020/rapport2020_36.pdf.

[15] J. Aarnes, G. P. Haugom and B. Norheim, 'Produksjon og bruk av hydrogen i norge', *Rapport (DNV)*, vol. 0039, 2019. [Online]. Available: https://www.regjeringen.no/contentassets/0762c0682ad04e6abd66a9555e7468df/hydrogen-i-norge---synteserapport.pdf.

[16] Morrow-Batteries. (2021). 'Morrow batteries partner with the municipality of arendal to build battery gigafactory', [Online]. Available: https://www.morrowbatteries.com/post/morrow-batteries-partner-with-the-municipality-of-arendal-to-build-battery-gigafactory (visited on 12/11/2021).

[17] Freyr-Batteries. (2021). 'About freyr', [Online]. Available: https://www.freyrbattery.com/about (visited on 12/11/2021).

[18] S. D. Kurland, 'Energy use for gwh-scale lithium-ion battery production', *Environmental Research Communications*, vol. 2, no. 1, p. 012 001, 2019.

[19] Energi-Norge, 'Veikart for grønn vekst', *Rapport (Energi Norge)*, 2017.

[20] S. R. Sæther and E. Moe, 'A green maritime shift: Lessons from the electrification of ferries in norway', *Energy Research & Social Science*, vol. 81, p. 102 282, 2021.

[21] Avinor. (). 'Avinors elektriske fly', [Online]. Available: https://avinor.no/konsern/klima/elfly/ln-elb (visited on 12/11/2021).

[22] M. J. Mølnvik, J. O. G. Tande, A. Tomasgard, M. Torsæter, S. O. Gardarsdottir, M. Korpås, T. M. Skjølsvold, S. Heidenreich, M. Korsnes, G. Tangen *et al.*, 'Nordsjøen som plattform for grønn omstilling', 2021.

[23] A. E. Winje, S. Hernes, L. Lind, G. Grimsby and E. Jakobsen, *Virkemidler for å realisere flytende havvind på norsk sokkel*, 2020.

[24] M. Buvik. (). 'Kostnader for kraftproduksjon', [Online]. Available: https://www.nve.no/energi/analyser-og-statistikk/kostnader-for-kraftproduksjon/ (visited on 15/11/2021).

[25] Standards Norway. (). 'Energy performance of buildings calculation of energy needs and supply', [Online]. Available: https://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=702386 (visited on 23/09/2021).

[26] A. Kipping and E. Trømborg, 'Modeling and disaggregating hourly electricity consumption in norwegian dwellings based on smart meter data', *Energy and Buildings*, vol. 118, pp. 350–369, 2016.

[27] A. Zoha, A. Gluhak, M. A. Imran and S. Rajasegarar, 'Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey', *Sensors*, vol. 12, no. 12, pp. 16 838–16 866, 2012, ISSN: 1424-8220. DOI: 10.3390/s121216838. [Online]. Available: https://www.mdpi.com/1424-8220/12/12/16838.

[28] A. Ridi, C. Gisler and J. Hennebert, 'A survey on intrusive load monitoring for appliance recognition', in *2014 22nd international conference on pattern recognition*, IEEE, 2014, pp. 3702–3707.

[29] *Encyclopedia of measurement and statistics au - salkind, neil*, pages 63-65, Thousand Oaks, Oct. 2007. DOI: 10.4135/9781412952644. [Online]. Available: https://doi.org/10.4135/9781412952644.

[30] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[31] J. D. Cryer and K.-S. Chan, *Time series analysis: with applications in R*. Springer, 2008, vol. 2.

[32] D. C. Montgomery, C. L. Jennings and M. Kulahci, *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[33] T. Hastie, R. Tibshirani and J. Friedman, 'The elements of statistical learning', *Cited on*, p. 33, 2009.

[34] E. Francesconi, 'The winter, the summer and the summer dream of artificial intelligence in law', *Artificial Intelligence and law*, pp. 1–15, 2022.

[35] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed. Pearson, 2009.

[36] Google-developers, *Training and test sets: Splitting data*, https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data, Last accessed: 12.08.2021, 2020.

[37] J. Moolayil, J. Moolayil and S. John, *Learn Keras for Deep Neural Networks*. Springer, 2019.

[38] SciKit. (). 'Underfitting vs. overfitting', [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html (visited on 30/11/2021).

[39] IBM-Cloud-Education, *Supervised learning*, https://www.ibm.com/cloud/learn/supervised-learning, Last accessed: 14.08.2021, 2020.

[40] Y.-S. Park and S. Lek, 'Artificial neural networks: Multilayer perceptron for ecological modeling', in *Developments in environmental modelling*, vol. 28, Elsevier, 2016, pp. 123–140.

[41] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, *Dive into Deep Learning*. 2020, https://d2l.ai.

[42] M. Awad and R. Khanna, *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer nature, 2015.

[43] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[44] D. Gupta. (2020). 'Fundamentals of deep learning – activation functions and when to use them?', [Online]. Available: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/ (visited on 16/11/2021).

[45] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, 'Activation functions: Comparison of trends in practice and research for deep learning', *arXiv preprint arXiv:1811.03378*, 2018.

[46] P. Baheti. (2021). '12 types of neural network activation functions: How to choose?', [Online]. Available: https://www.v7labs.com/blog/neural-networks-activation-functions (visited on 16/11/2021).

[47] V. Nair and G. E. Hinton, 'Rectified linear units improve restricted boltzmann machines', in *Icml*, 2010.

[48] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015, vol. 25.

[49] S. Ruder, 'An overview of gradient descent optimization algorithms', *arXiv preprint arXiv:1609.04747*, 2016.

[50] P. Li, 'Optimization algorithms for deep learning', *Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong*, 2017.

[51] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, 2014.

[52] S. Abirami and P. Chitra, 'Chapter fourteen - energy-efficient edge based real-time healthcare support system', in *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, ser. Advances in Computers 1, P. Raj and P. Evangeline, Eds., vol. 117, Elsevier, 2020, pp. 339–368. DOI: https://doi.org/10.1016/bs.adcom.2019.09.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245819300506.

[53] S. Hochreiter and J. Schmidhuber, 'Long short-term memory', *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[54] C. Olah, 'Understanding lstm networks', 2015.

[55] U. Cali, M. Kuzlu, M. Pipattanasomporn, J. Kempf and L. Bai, *Digitalization of power markets and systems using energy informatics*, 2021.

[56] A. Dikshit and B. Pradhan, 'Interpretable and explainable ai (xai) model for spatial drought prediction', *Science of The Total Environment*, vol. 801, p. 149 797, 2021.

[57] M. Zdravković, I. Ćirić and M. Ignjatović., 'Towards explainable ai-assisted operations in district heating systems', *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 390–395, 2021, 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.08.044. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896321007606.

[58] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[59] M. T. Ribeiro, S. Singh and C. Guestrin, '" why should i trust you?" explaining the predictions of any classifier', in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[60] 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*, 2017, pp. 4768–4777.

[61] A. Shrikumar, P. Greenside and A. Kundaje, 'Learning important features through propagating activation differences', in *International conference on machine learning*, PMLR, 2017, pp. 3145–3153.

[62] R. Tawn and J. Browell, 'A review of very short-term wind and solar power forecasting', *Renewable and Sustainable Energy Reviews*, vol. 153, p. 111 758, 2022.

[63] J. González-Sopeña, V. Pakrashi and B. Ghosh, 'An overview of performance evaluation metrics for short-term statistical wind power forecasting', *Renewable and Sustainable Energy Reviews*, vol. 138, p. 110 515, 2021.

[64] P. Malhan and M. Mittal, 'A novel ensemble model for long-term forecasting of wind and hydro power generation', *Energy Conversion and Management*, vol. 251, p. 114 983, 2022.

[65] Y. Tao, H. Chen and C. Qiu, 'Wind power prediction and pattern feature based on deep learning method', in *2014 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, IEEE, 2014, pp. 1–4.

[66] Y. Wang, R. Zou, F. Liu, L. Zhang and Q. Liu, 'A review of wind speed and wind power forecasting with deep neural networks', *Applied Energy*, vol. 304, p. 117 766, 2021.

[67] R. Ahmed, V. Sreeram, Y. Mishra and M. Arif, 'A review and evaluation of the state-of-the-art in pv solar power forecasting: Techniques and optimization', *Renewable and Sustainable Energy Reviews*, vol. 124, p. 109 792, 2020.

[68] P. Li, K. Zhou, X. Lu and S. Yang, 'A hybrid deep learning model for short-term pv power forecasting', *Applied Energy*, vol. 259, p. 114 216, 2020.

[69] A. Agga, A. Abbou, M. Labbadi and Y. El Houm, 'Short-term self consumption pv plant power production forecasts based on hybrid cnn-lstm, convlstm models', *Renewable Energy*, vol. 177, pp. 101–112, 2021.

[70] S. Haben, S. Arora, G. Giasemidis, M. Voss and D. V. Greetham, 'Review of low-voltage load forecasting: Methods, applications, and recommendations', *arXiv preprint arXiv:2106.00006*, 2021.

[71] C. Kuster, Y. Rezgui and M. Mourshed, 'Electrical load forecasting models: A critical systematic review', *Sustainable cities and society*, vol. 35, pp. 257–270, 2017.

[72] Y. Chu, H. T. Pedro, A. Kaur, J. Kleissl and C. F. Coimbra, 'Net load forecasts for solar-integrated operational grid feeders', *Solar Energy*, vol. 158, pp. 236–246, 2017.

[73] J. Lee and Y. Cho, 'National-scale electricity peak load forecasting: Traditional, machine learning, or hybrid model?', *Energy*, vol. 239, p. 122 366, 2022.

[74] H. Shi, M. Xu and R. Li, 'Deep learning for household load forecasting—a novel pooling deep rnn', *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2017.

[75] W. Kong, Z. Y. Dong, F. Luo, K. Meng, W. Zhang, F. Wang and X. Zhao, 'Effect of automatic hyperparameter tuning for residential load forecasting via deep learning', in *2017 Australasian Universities Power Engineering Conference (AUPEC)*, IEEE, 2017, pp. 1–6.

[76] A. Mashlakov, E. Pournaras, P. H. Nardelli and S. Honkapuro, 'Decentralized cooperative scheduling of prosumer flexibility under forecast uncertainties', *Applied Energy*, vol. 290, p. 116 706, 2021.

[77] R. J. Hyndman and A. B. Koehler, 'Another look at measures of forecast accuracy', *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207006000239.

[78] Elhub. (). 'Min side', [Online]. Available: https://elhub.no/om-elhub/elhub-for-sluttbruker/min-side/ (visited on 03/03/2022).

[79] Enova. (). 'Energimerking calculator', [Online]. Available: https://kalk.energimerking.no/Felles/InnledningKalk.aspx (visited on 03/05/2022).

[80] Statistics-Norway. (2018). 'Vi bruker mindre strøm hjemme', [Online]. Available: https://www.ssb.no/energi-og-industri/artikler-og-publikasjoner/vi-bruker-mindre-strom-hjemme (visited on 30/04/2022).

[81] H. Sæle and M. Aasen, 'Ny nettleiemodell for norske husholdninger', *SINTEF Rapport*, 2021.

[82] D. A. Bolstad, 'Interpretation of electrical load forecasts using explainable artificial intelligence', M.S. thesis, NTNU, 2021.

[83] NVE. (). 'Temakart', [Online]. Available: https://temakart.nve.no/link/?link=nettanlegg (visited on 04/05/2022).

[84] N. pool. (). 'Market data', [Online]. Available: https://www.nordpoolgroup.com/en/Market-data1/ (visited on 04/05/2022).

[85] U. Andrae, I.-L. Frogner, O. Vignes, A. Singleton, R. Azad, M. Partio and N. Sokka, 'A continuous eda based ensemble in metcoop', *ALADIN-HIRLAM Newsletter*, vol. 2020, pp. 189–199, 2020.

[86] N. M. Institute. (). 'Catalog https://thredds.met.no/thredds/catalog.html', [Online]. Available: https://thredds.met.no/thredds/catalog.html (visited on 30/11/2021).

[87] T. Spence. (2020). 'Rekordnedbør i juli. kjøligste julimåned på 50 år på østlandet.', [Online]. Available: https://www.aftenposten.no/norge/i/Ad6g53/rekordnedboer-i-juli-kjoeligste-julimaaned-paa-50-aar-paa-oestlandet (visited on 05/05/2022).

[88] I. H. Witten and E. Frank, 'Data mining: Practical machine learning tools and techniques with java implementations', *Acm Sigmod Record*, vol. 31, no. 1, pp. 76–77, 2002.

[89] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[90] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[91] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[92] F. Chollet *et al.* (2015). 'Keras', [Online]. Available: https://github.com/fchollet/keras.

[93] Y. Gal and Z. Ghahramani, 'A theoretically grounded application of dropout in recurrent neural networks', *Advances in neural information processing systems*, vol. 29, 2016.

[94] S. Semeniuta, A. Severyn and E. Barth, 'Recurrent dropout without memory loss', *arXiv preprint arXiv:1603.05118*, 2016.

[95] M. Feurer and F. Hutter, 'Hyperparameter optimization', in *Automated machine learning*, Springer, Cham, 2019, pp. 3–33.

[96] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, *Keras Tuner*, https://github.com/keras-team/keras-tuner, 2019.

[97] S. Lundberg. (2018). 'Deepexplainer', [Online]. Available: https://shap-lrjball.readthedocs.io/en/latest/generated/shap.DeepExplainer.html (visited on 10/05/2022).

[98] T. Chai and R. R. Draxler, 'Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature', *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[99] D. Chicco, M. J. Warrens and G. Jurman, 'The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation', *PeerJ Computer Science*, vol. 7, e623, 2021.

[100] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim *et al.*, 'Explainable machine-learning predictions for the prevention of hypoxaemia during surgery', *Nature Biomedical Engineering*, vol. 2, no. 10, p. 749, 2018.

[101] M. Alhussein, K. Aurangzeb and S. I. Haider, 'Hybrid cnn-lstm model for short-term individual household load forecasting', *IEEE Access*, vol. 8, pp. 180 544–180 557, 2020.

[102] A. Shaqour, T. Ono, A. Hagishima and H. Farzaneh, 'Electrical demand aggregation effects on the performance of deep learning-based short-term load forecasting of a residential building', *Energy and AI*, vol. 8, p. 100 141, 2022.

[103] M. Abdel-Basset, H. Hawash, K. Sallam, S. Askar and M. Abouhawwash, 'Stlf-net: Two-stream deep network for short-term load forecasting in residential buildings', *Journal of King Saud University-Computer and Information Sciences*, 2022.

[104] A. Hebrail Georges Berard, *Individual household electric power consumption*, UCI Machine Learning Repository, 2012.

[105] L. M. Candanedo, V. Feldheim and D. Deramaix, 'Data driven prediction models of energy use of appliances in a low-energy house', *Energy and buildings*, vol. 140, pp. 81–97, 2017.

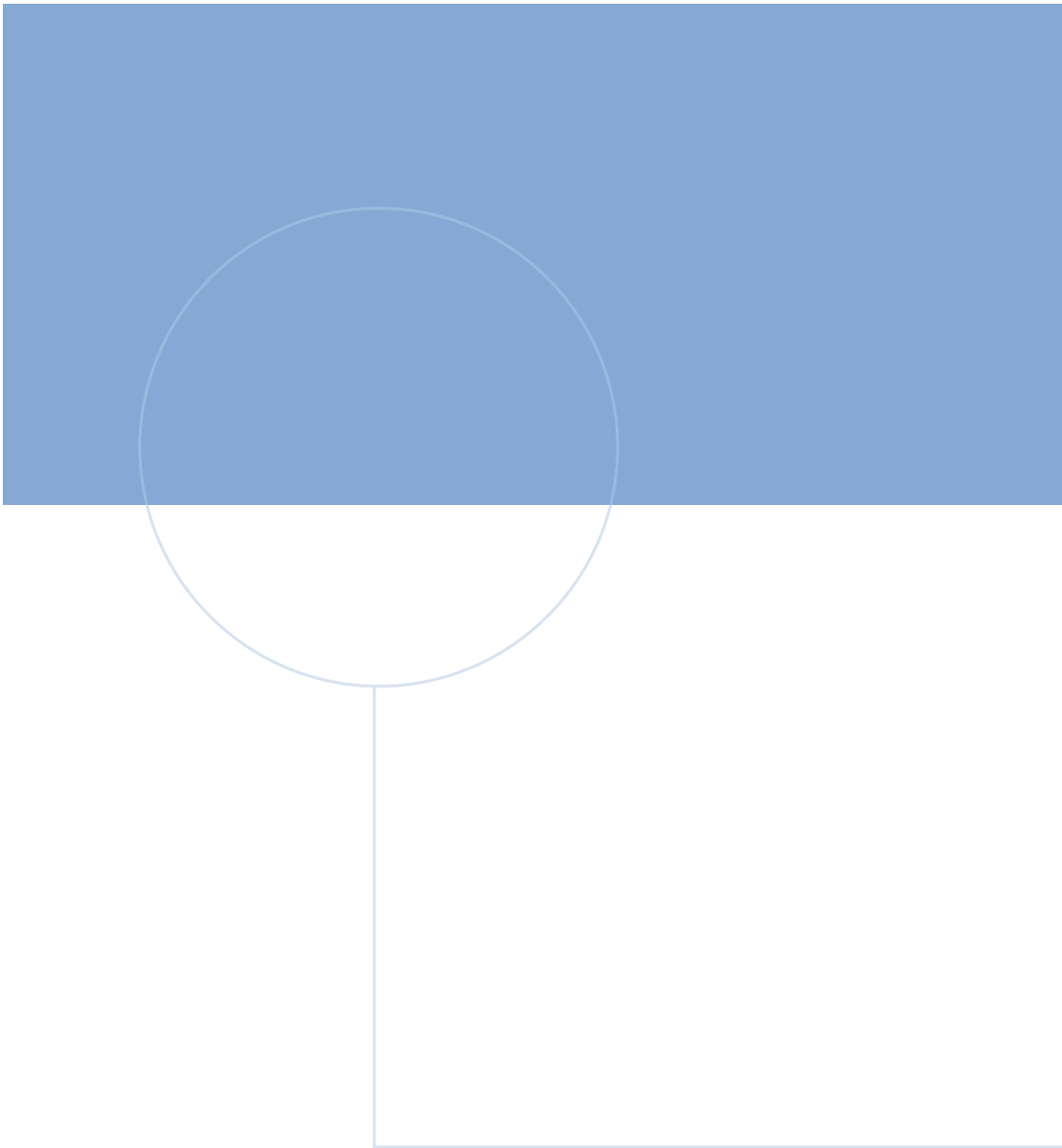
Appendix

Appendix A

Model Hyperparameters

Table A.1: Chosen hyperparameters for an assortment of models. Be aware due to a error some models were lost.

Layers Model	**LSTM-A2**	**CNN-LSTM-A2**	**LSTM-B1**	**CNN-LSTM-B1**	**LSTM-B2**	**CNN-LSTM-B2**	**LSTM-C2**
CNN layer							
Filters	-	64	-	826	-	930	-
Activation	-	ReLu	-	ReLu	-	ReLu	-
Kernel size	-	9	-	5	-	7	-
LSTM layer 1							
Units	24	118	116	110	106	50	76
Activation	Tanh	ReLu	ReLu	ReLu	Tanh	ReLu	Tanh
Recurrent dropout	0.0048	0.0901	0	0.0626	0.0714	0.0457	0.0998
Dropout	True	True	True	True	True	True	True
LSTM layer 2							
Units	112	110	118	102	88	12	120
Activation	ReLu	Tanh	ReLu	Sigmoid	ReLu	Sigmoid	ReLu
Recurrent dropout	0.0048	0.0905	0	0.0626	0.0714	0.0457	0.0998
Dropout	False		True	False	False	False	False
Optimizer values							
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Clipvalue	0.1384	0.3942	0.5	0.0038	0.0191	0.0995	0.0115
Learning rate	0.0006	0.0016	0.05	0.00013	0.0039	0.00013	0.0057



NTNU

Kunnskap for en bedre verden