University of Tartu

Faculty of Science and Technology

Institute of Technology

Oleksandr Kurylenko

Development of CNN-Based Models for Short-Term Load Forecasting in Energy Systems

Master's thesis (30 ECTS)
Robotics and Computer Engineering

Supervisors:

Alan Henry Tkaczyk, PhD
Florentin Dam, MSc
Ulf Roar Aakenes, PhD

Tartu 2020

Resümee

Konvolutsioonilistel närvivõrkudel põhinevate mudelite arendus lühiajalise koormuse prognoosimiseks energiasüsteemides

Selles magistritöös on arendatud kaks konvolutsioonilistel närvivõrkudel (CNN) põhinevat sügavõppe mudelit, mis prognoosivad tunnipõhiselt elektri tarbimist järgmiseks kalendripäevaks. Nende eesmärk on prognoosida tarbimiskoormuse algtaset, mis aitab hinnata elektri nõudlusele reageerimise tulemuslikkust ja ennustada tarbimismuudatustest sõltuvate teenuste kättesaadavust.

Elektri tarbimise aegrea andmed pärinevad kolmest Norra piirkonnast. CNN struktuuril põhinevaid mudeleid võrreldakse nelja tööstusstandardi mudeliga (Asymmetric High Five of Ten, Similar Profile Five of Ten, Average, Daily Profile), lisaks tehakse võrdlusi naiivse (Naive) ning autoregressiivse integreeritud libiseva keskmise (ARIMA) mudeliga, mis sisaldab Fourieri rea komponente. Mudelite tulemuslikkust kontrollitakse kolme hindamisparameetri põhjal. Magstritöös antakse detailne ülevaade metoodikast, töö käigust ning tulemustest.

CNN struktuuril põhinevate mudelitega tehtud katsed olid edukad järgmise päeva elektri tarbimise tunnipõhiseks prognoosimiseks. Mitme sisendi ja mitme väljundiga (MIMO) 24 tundi ette prognoosivatest mudelitest näitasid parimaid tulemusi CNN struktuuril põhinev ning kombinatsioon CNN struktuuril põhineva ja pika lühiajalise mälu (LSTM) mudelist. Daily Profile mudel andis parimaid tulemusi nende uuritud mudelite hulgast, mis ei võimalda 24 tundi ette prognoosida või ei rakenda MIMO lähenemist. Lisaks võib välja tuua Average mudeli, mille tulemuslikkus algtaseme koormuse hindamisel oli uuritud mudelite hulgas kõrgeim. Samas saab Average mudelit rakendada vaid võrdluseks, sest see kasutab sündmusejärgseid andmeid.

Töö CNN struktuuril põhinevate mudelitega energia tarbimise prognoosimiseks oli edukas, eriti arvestades seda, et CNN ja CNN+LSTM mudelid edestasid teisi sarnaseid prognoosimudeleid. Edasisi uuringuid on võimalik jätkata kahes suunas: juba arendatud CNN struktuuril põhinevate mudelite täiustamisega ning uute CNN struktuuril põhinevate arhitektuuride väljatöötamise ja rakendamisega.

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria); T140 Energeetika

Märksõnad: konvolutsioonilised närvivõrgud (CNN), pikk lühiajaline mälu (LSTM), ühe muutujaga aegrea prognoosimine, mitme muutujaga aegrea prognoosimine, elektri tarbimine, elektri nõudlus

Abstract

Development of CNN-Based Models for Short-Term Load Forecasting in Energy Systems

In this work, two deep learning models based on convolutional neural networks (CNNs) are developed for one-day-ahead hourly electricity consumption forecasting for the next calendar day. One-day-ahead electricity consumption forecasting is done to perform baseline load evaluation for the assessment of Demand Response (DR) performance and to predict the availability of flexibility products.

Using electricity consumption time series data for three regions in Norway, the developed CNN-based models are compared to four industry-standard baseline models (Asymmetric High Five of Ten, Similar Profile Five of Ten, Average, and Daily Profile). Additionally, comparisons are made to a Naive model and an Autoregressive Integrated Moving Average (ARIMA) model, which includes Fourier terms. Three evaluation metrics are used to estimate the models' performance. A detailed description of the methodology, work pipeline, and results is provided.

The conducted experiments were successful in developing and applying the CNN-based models to the problem of one-day-ahead hourly electricity consumption forecasting for the next calendar day. The developed CNN and combination "CNN + Long Short-Term Memory (LSTM)" models showed the best performance results among the predictive models which employ the Multiple-Input Multiple-Output (MIMO) strategy to forecast 24 hours ahead. The Daily Profile model showed the best performance among models which cannot forecast 24 hours ahead or do not necessarily employ the MIMO strategy. It is worth noting that the Average model showed the best performance in the baseline load evaluation among all the considered models. However, the Average model is only a reference comparison which does not actually perform forecasting, but rather uses post-event data.

It can be concluded that the work was successful in developing and applying the CNN-based models to short-term load forecasting in energy systems, especially since the developed CNN and CNN+LSTM models outperformed other similar forecasting models. Two different paths could be chosen for future work: one that intends to explore more and improve the CNN-based models developed in this work, and the one which aims to explore new CNN-based architectures.

CERCS: P170 Computer science, numerical analysis, systems, control; T140 Energy research

Keywords: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), univariate time series forecasting, multivariate time series forecasting, electricity consumption, electricity demand

Contents

List of Figures

List of Tables

Acronyms

AICc - Corrected Akaike Information Criterion

ARIMA - Autoregressive Integrated Moving Average

CNN - Convolutional Neural Network

CV-RMSE - Coefficient of Variation of the Root Mean Square Error

DR - Demand Response

DRSP - Demand Response Service Provider

HFoT - High Five of Ten

LSTM - Long Short-Term Memory

MAE - Mean Absolute Error

MAPE - Mean Absolute Percentage Error

MIMO - Multiple-Input Multiple-Output

NMSE - Normalized Mean Square Error

NRMSE - Normalized Root Mean Square Error

PReLU - Parametric Rectified Linear Unit

ReLU - Rectified Linear Unit

RMSE - Root Mean Square Error

RRMSE - Relative Root Mean Square Error

SPFoT - Similar Profile Five of Ten

STLF - Short-Term Load Forecasting

TPE - Tree Parzen Estimator

1 Introduction

In the electric grid, it is crucial to maintain a balance between demand and supply. Any significant imbalance in the demand-supply ratio may cause electric grid instability and failures within the grid. One of the traditional ways to avoid such situations is to adjust the supply according to the demand. However, this approach is quite limited from the electricity utility providers' side since some generating units may take a long time to ramp up to full power and be too expensive to operate, as well as at times demand may be higher than the overall capacity of all the available power plants.

Demand response (DR) mechanisms were designed to overcome these limitations. DR mechanisms aim to reduce electricity consumption or shift it from on-peak to off-peak periods to reduce the risk of potential electric grid disturbances and to ensure higher efficiency of usage of power plants for electric utility providers. It is done by reflecting supply expectations through consumer price signals when power is cheaper at specific times of the day or through smart metering when explicit requests or changes in pricing may be communicated to consumers. [AES07, BPKS11] According to the United States Federal Energy Commission, DR is defined as "changes in electric usage by end-use customers from their normal consumption patterns in response to changes in the price of electricity over time, or to incentive payments designed to induce lower electricity use at times of high wholesale market prices or when system reliability is jeopardized". [FER16] Therefore, DR mechanisms facilitate greater flexibility within the electric grid.

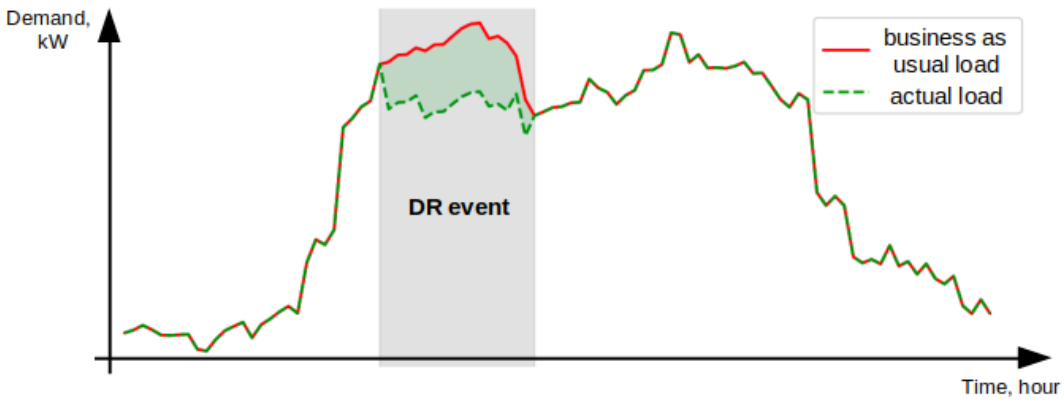


Figure 1.1: Demand Response (DR) event

When the demand needs to be decreased, a DR event is activated. As can be seen from Figure 1.1, during the DR event, demand is decreasing compared to business as usual load, or load under the assumption of no DR event. The green area in Figure 1.1 represents the curtailment in electricity consumption achieved during the DR event.

1.1 Problem Statement

To make DR mechanisms work on the market, the necessity to measure the real curtailment during the DR events arises. It is crucial for defining the extent to which flexibility service providers deliver the response. The actual load is compared to the business as usual load, known as baseline load, to get the real curtailment during the DR event. The difference between the baseline load and actual load during the DR event accounts for the DR performance.

As an example of the DR system,an incentive-based demand response system, given in Figure 1.2, is reviewed.

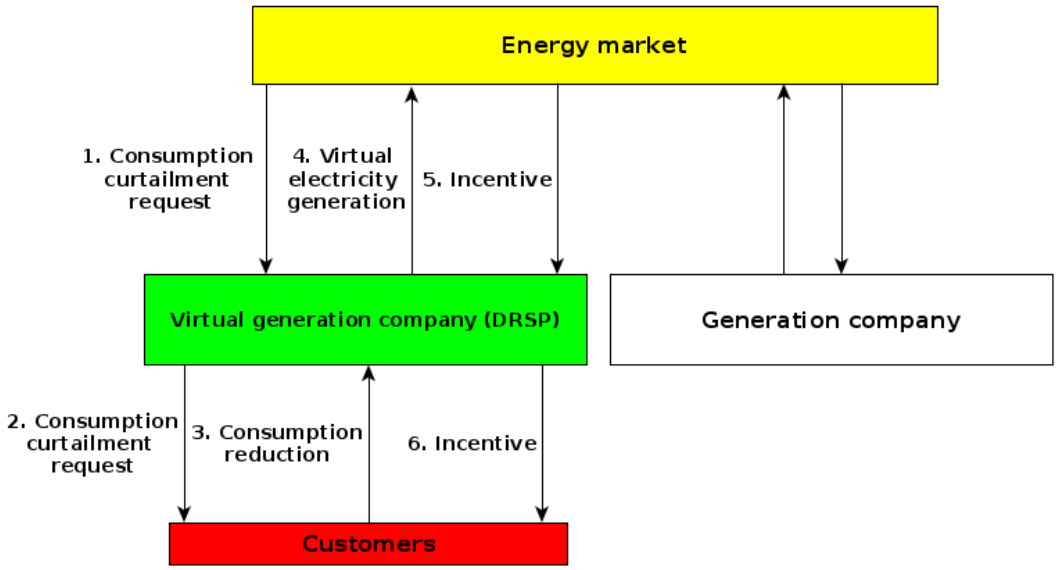


Figure 1.2: Incentive-based Demand Response (DR) system

In the incentive-based DR system, Demand Response Service Providers (DRSPs), also known as flexibility providers, gather electricity customers who have the potential to decrease electricity consumption upon a request. Each DRSP aggregates such customers and offers an aggregated DR resource, which is considered as a virtual generator. If consumption has to be reduced, the DR event is activated, and DRSP distributes the request to the participating customers who, in turn, decrease electricity consumption. In the next step, the real curtailment is measured as a difference between the baseline load and load during the DR event. After the electricity curtailment amount is measured, DRSP receives an incentive which is distributed among the customers, excluding the commission. [KL18, WUO+14] Therefore, DRSPs sell the deviation from the baseline load of their assets.

Different industry-standard baseline models such as Asymmetric HFoT, SPFoT, Average, or Daily Profile are widely adopted for the baseline load evaluation in the modern electric grids. They are developed to account for four main qualities of baseline models: accuracy, simplicity, integrity, and alignment. [KEM13a, KEM13b, Ene09]

Nevertheless, alternative methods can be considered to both improve the baseline load evaluation and produce the electricity consumption forecasts to allow assessment of the availability of the flexibility products on the market.

1.2 Objectives and Roadmap

The main objective of this work is to develop CNN-based deep learning models for the one-day-ahead hourly electricity consumption forecasting for the next calendar day, which belongs to short-term load forecasting (STLF). One-day-ahead forecasting of electricity consumption would allow not only to assess the real curtailment in case of DR events but also to predict the availability of flexibility products for that day.

The goal of this work is to demonstrate that the developed CNN-based deep learning architectures are relevant and flexible to allow "plugging-in" deep learning algorithms into the process of electricity consumption forecasting. As a powerful family of machine learning methods based on artificial neural networks, deep learning has shown great benefits and promising applications in load forecasting tasks. Unlike statistical methods, deep learning methods can better capture inherent non-linearity, volatility, and long-term dependencies present in the electricity consumption data. Although deep learning methods are not characterized as being relatively simple, they tend to provide high-accuracy forecasts, and ensure greater scalability, which is an essential property for growing electric grids. The convolutional neural networks (CNNs), in particular, have shown remarkable results in the scope of short-term load forecasting.

In this work, ARIMA and Naive model are used as baseline models for comparison to the developed deep learning CNN-based models and industry-standard baseline models.

It is worth mentioning that different types of customers in different regions have different electricity consumption behavior. Consequently, different models may be applied to each of the customer segments in the electric grid to achieve higher accuracy. However, in this work, the aim is to use the baseline load forecasting models for high-level aggregations of customers in Norway without performing preliminary customer segmentation.

1.3 Structure of the Thesis

This thesis consists of 6 chapters:

- Chapter 2 highlights the related works in the field of load forecasting.
- Chapter 3 describes the methodology applied in this work.
- Chapter 4 gives an overview of the experimental part of the study.
- Chapter 5 provides the results of the experimental part alongside with analysis and discussion over the received results.
- Chapter 6 provides a summary of the key findings.

2 Related Work

The availability of vast amounts of energy data, along with the various challenges in the energy sector, lead to an increased interest in short-term load forecasting. Different CNN-based models are developed and studied for solving the problem of STLF in numerous works. The demonstrated effectiveness, and high forecasting accuracy of hybrid CNN-based models in STLF tasks serves as an impetus for applying CNN-based models to STLF. In this chapter, some of the recent research, which has boosted the motivation to develop and employ CNN-based models for STLF in this work is reviewed.

In [KH18], Kuo et al. propose a CNN-based load forecasting model. The presented model is developed to make 72-hour-ahead forecasts based on the last week of hourly load data. The experiments on the real-world data show that the proposed model outperforms the compared models with MAPE and CV-RMSE of the proposed model being 9.77% and 11.66%. The experiments prove the effectiveness of the CNN structure in forecasting.

In [TMZZ18], Tian et al. propose a hybrid CNN-LSTM model for short-term load forecasting. The proposed model consists of LSTM, CNN, and feature-fusion modules, and is developed to make 24-hour-ahead forecasts based on the last three weeks of hourly load data. The experiments on the real-world data show that the proposed model outperforms the compared models by MAE, RMSE, and MAPE evaluation metrics averaged on eight partitions of the test set. The results also show that the proposed CNN-LSTM model demonstrates stability by being superior to the other models on all partitions of the test set.

In [LSSL19], Lang et al. perform a study on the performance of different modifications of one-dimensional CNN models for 36-hour-ahead load forecasting. The experiments are done on the real-life smart meter data sets of 15 minutes granularity, corresponding to a big apartment building and a whole city district. The authors conclude that good forecasts for volatile load profiles can be achieved already with rather simple CNN architectures.

In [HR19], Hong et al. propose a hybrid model based on CNN, cascaded with the Radial Basis Function (RBF) neural network with a double Gaussian function as its activation function. The proposed model takes into account the uncertainty of the wind power and is developed to make 24-hour-ahead forecasts based on the last 50 hours of hourly load data. The experiments on the real-world data show that the proposed model produces the best results considering the structure of the CNN with two convolution and two pooling layers, and outperforms the compared models based on R^2, RMSE, NMSE, and MAPE evaluation metrics.

In [KCJL19], Kim et al. propose a hybrid power demand forecasting model (c, l)-LSTM-CNN and introduce a bivariate-based context learning approach. In the proposed model, c bivariate sequences for each type of contextual information in the format $< Key, Context_{[1,c]} >$ are given as an input to c LSTM networks with l layers, which are cascaded with a CNN module. The proposed model is developed to make n-day forecasts. The authors perform numerous experiments on the different types of daily real-world power demand data. In particular, the proposed model shows the best results in terms of MAPE and RRMSE compared to the other considered models on the data set by day of the week without holidays.

In [HWG+19], Huang et al. propose a low training cost CNN-based model. The proposed model is developed to make day-ahead forecasts based on the last week of half-hourly residential load data. The experiments on the real-world data show that the proposed model outperforms the compared models in terms of training time and accuracy of forecasts based on RMSE, NRMSE, and MAE evaluation metrics. The authors emphasize that the forecasting accuracy of the model can be improved by obtaining and employing external information in addition to the electricity load data.

In [DWX+19], Deng et al. propose a multi-scale CNN with time cognition. The CNN module of the proposed model is based on the one-dimensional multi-scale convolutions. Additionally, the authors present periodic coding applied to the input time variables. The performance of the proposed and compared models are estimated on the real-world data both for point and probabilistic forecasts. Based on the experiments, the proposed model showed the best forecasting accuracy based on RMSE, MAE, and MAPE evaluation metrics for point forecasting, and average pinball score for probabilistic forecasting.

3 Methodology

In this chapter, the methodology for short-term load forecasting applied to baseline load evaluation is described. In section 3.1, the employed in this work approach to forecasting is discussed. In section 3.2, baseline models used for comparison to the given in section 3.4 developed deep learning CNN-based models are described. In sections 3.5 and 3.6, the discussion on the performance estimation of the implemented models is presented.

3.1 Approach to Forecasting

In this work, for baseline load evaluation, 24-hour-ahead hourly electricity consumption forecasting, also referred to as multi-step STLF, is performed. STLF covers real-time planning that aims to predict the availability of flexibility products to balance the requirements for the next calendar day. In this section, a general overview of forecasting approaches is given.

3.1.1 Types of Forecasting Models

The main objective of forecasting problems is to predict the values for a forecast horizon H based on the values of a time window W, where H is the number of steps for which the forecast is done, and W is the period on which the forecast is based. Each step in the forecast horizon is referred to as lead time.

For the cases $H > 1$, the problem is called a multi-step forecasting problem. In this work, the 24-hour-ahead hourly forecast is done, i.e., $H = 24$. Therefore, multi-step forecasting models for solving a multi-step forecasting problem are developed.

Multi-step forecasting models can be classified into two types: multi-step univariate forecasting models or multi-step multivariate forecasting models. In this work, both multi-step multivariate and univariate forecasting models are employed.

Multi-step Univariate Forecasting Models

In the multi-step univariate forecasting models, the succeeding sequence members $\{x_n, \ldots, x_{n+H-1}\}$ depend only on the preceding members of the same time series $\{x_{n-W}, \ldots, x_{n-1}\}$ for the forecast horizon H and time window W, as given in Eq. (3.1).

$$x_n, \ldots, x_{n+H-1} = f(x_{n-W}, \ldots, x_{n-1}) \tag{3.1}$$

In Eq. (3.1), f is some function which defines the dependency between the succeeding and preceding values. [ASVVMM16]

Multi-step Multivariate Forecasting Models

Multivariate forecasting models are an extension to the univariate forecasting models. They are used for the cases when the succeeding sequence members $\{x_n, \ldots, x_{n+H-1}\}$ depend not only on the preceding members of the same time series $\{x_{n-W_1}, \ldots, x_{n-1}\}$ but also on the members of another time series $\{y_{m-W_2}, \ldots, y_{m-1}\}$, $\{z_{k-W_3}, \ldots, z_{k-1}\}$, ... for the forecast horizon H and time windows $W_1, W_2, W_3, \ldots$, as given in Eq. (3.2).

$$x_n, \ldots, x_{n+H-1} = f(x_{n-W_1}, \ldots, x_{n-1}, y_{m-W_2}, \ldots, y_{m-1}, z_{k-W_3}, \ldots, z_{k-1}, \ldots) \tag{3.2}$$

In Eq. (3.2), f is some function which defines the dependency between the succeeding and preceding values. It is assumed that the time series $\{y_t\}_{t\in T}$, $\{z_t\}_{t\in T}$, ... are changing and measured synchronously with the time series $\{x_t\}_{t\in T}$. [ASVVMM16]

3.1.2 Multi-Step Forecasting Schemes

In this section, an overview of five different strategies for multi-step forecasting is given. In this work, the MIMO strategy for performing a multi-step STLF is adopted for the developed models.

Recursive Strategy

While employing a recursive strategy, a single predictor f is designed to make a one-step-ahead forecast x_i based on $x_{i-W}, \ldots, x_{i-1}$ for time window W.

For forecasting the succeeding sequence members $x_n, \ldots, x_{n+H-1}$ in the forecast horizon H, the value for each time step $t_n, \ldots, t_{n+H-1}$ is recursively forecasted and added to the time series sequence, being treated as a correct one. Therefore, for each time step $t_i, i \in \overline{n+1, n+H-1}$, the previously forecasted preceding value x_{i-1} is being used in forecasting. [GLA19]

Direct Strategy

While employing a direct strategy, a set of independent predictors $f_k, k \in \overline{1, H}$ is being designed to make forecasts $x_i, \ldots, x_{i+H-1}$ independently for each of the time steps $t_{i+k-1}, k \in \overline{1, H}$ in the forecast horizon H. The forecasts $x_i, \ldots, x_{i+H-1}$ are made by $f_k, k \in \overline{1, H}$ based on the same input $x_{i-W}, \ldots, x_{i-1}$ for time window W. [GLA19]

DiRec Strategy

DiRec strategy is a combination of the recursive and direct strategies.

While employing a direct strategy, a set of independent predictors $f_k, k \in \overline{1, H}$ is being designed to make forecasts $x_i, \ldots, x_{i+H-1}$ independently for each of the time steps $t_{i+k-1}, k \in \overline{1, H}$ in the forecast horizon H.

For forecasting the succeeding sequence members $x_n, \ldots, x_{n+H-1}$, each of the predictors $f_k, k \in \overline{1, H}$ uses an enlarged input set, obtained by adding the values x_i, predicted at the previous time steps $t_i, i \in \overline{n, n+k-2}, k \geq 2$. [GLA19]

MIMO Strategy

While employing a MIMO strategy, a single predictor f is designed to forecast the whole output sequence $x_n, \ldots, x_{n+H-1}$ in the forecast horizon H. [GLA19]

DIRMO Strategy

DIRMO strategy represents a trade-off between the Direct and MIMO strategies.

While employing a DIRMO strategy, the forecast horizon H is divided into smaller forecast horizons of length l, $H_1, \ldots, H_m, m = \lceil \frac{H}{l} \rceil$. This way, m forecasting problems for each of the forecast horizons $H_1, \ldots, H_m$ are created, each of which is solved employing the MIMO strategy. [GLA19]

3.2 Baseline Models

Baseline models provide a reference to make sure the developed CNN-based models are not performing worse. In this section, three types of baseline models are introduced.

3.2.1 Industry-Standard Baseline Models

In this work, along with the developed models, the aim is to explore the performance of the industry-standard baseline models. Industry-standard baseline models are widely used to evaluate whether flexibility has been offered or not. They account for four main qualities of baseline models: accuracy, simplicity, integrity, and alignment. Accuracy is necessary to provide adequate DR performance assessment. Simplicity is essential to make it easier for the stakeholders to estimate the real incentives depending on the load curtailment. Integrity is required to prevent the attempts of stakeholders to "game the system." Alignment is necessary to avoid the situations of under- or overestimation of the real load curtailment efforts.

Baseline load adjustments may be applied to models to capture customers' recent behavior on the day of the DR event. These adjustments are based on the day-of-event conditions and mitigate the possible bias in baseline load evaluation, which may occur since customer demand is often the heaviest on the event days. Instances of baseline load adjustments are symmetric or asymmetric. Symmetric adjustments shift the estimated initial baseline up if the estimated baseline load is lower than the event-day profile, and down if the estimated baseline load is higher than the event-day profile. Asymmetric adjustments only shift the estimated initial baseline load up if the estimated baseline load is lower than the event-day profile. [Ene09, WUO+14]

Asymmetric HFoT

Asymmetric HFoT ("high five of ten") baseline load evaluation model was developed by EnerNOC and is described in [Ene09].

For a given time interval t, baseline load b_t is estimated as shown in Eq. (3.3) (the calculation is done for each time interval during the DR event).

$$b_t = \frac{c_{td1} + c_{td2} + c_{td3} + c_{td4} + c_{td5}}{5} + \max[\frac{c_{t-1} - b_{t-1} + c_{t-2} - b_{t-2}}{2}; 0] \tag{3.3}$$

The first summand represents the initial baseline load estimation, while the second one represents the additive upward asymmetric adjustment. The initial baseline load is equal to the average electricity consumption of corresponding hours from five days with the highest consumption c_{td1}-c_{td5} within the last ten non-event days excluding holidays. Baseline load adjustment is equal to the difference in observed electricity consumption c_{t-2} and c_{t-1} and evaluated baseline load b_{t-2} and b_{t-1} beginning two hours before the DR event activation with the minimum adjustment of zero.

This baseline model satisfies all four crucial qualities of the baseline. Using a 10-day baseline time window, it provides a proper balance to take into account the recent trends and to limit possibilities for manipulations. The formula utilizes the corresponding hours with the highest consumption, specifically to avoid understating the customers' baseline load.

SPFoT

SPFoT ("similar profile five of ten") baseline load evaluation model was developed for Low Carbon London and is described in [WUO+14].

For a given time interval t, baseline load b_t is estimated as shown in Eq. (3.4) (the calculation is done for each time interval during the DR event).

$$b_t = \frac{c_{tds1} + c_{tds2} + c_{tds3} + c_{tds4} + c_{tds5}}{5} + \frac{c_{t-1} - b_{t-1} + c_{t-2} - b_{t-2}}{2} \tag{3.4}$$

The first summand represents the initial baseline load estimation, while the second one represents the day-of-event symmetric adjustment. The initial baseline load is equal to the average load consumption of corresponding hours from five days with a similar electricity consumption profile c_{tds1}-c_{tds5} within the last ten non-event days excluding holidays. The electricity consumption profile similarity between the event day and other days is defined by (Pearson) correlation coefficient. Baseline load adjustment is equal to the difference in observed electricity consumption c_{t-2} and c_{t-1} and evaluated baseline load b_{t-2} and b_{t-1} beginning two hours before the DR event activation.

Since SPFoT estimates the baseline load based on the daily electricity consumption profile of a similar shape, not just based on the days with the highest consumption, more variable profiles can benefit from this approach.

Average

Average baseline load evaluation model was proposed in [EAL17].

For a given time interval t, baseline load b_t is estimated as shown in Eq. (3.5) (the calculation is done for each time interval during the DR event).

$$b_t = \frac{c_{t-1} + c_{t+1}}{2} \tag{3.5}$$

Baseline load is equal to the average electricity consumption one hour before c_{t-1} and one hour after c_{t+1} the DR event activation.

Daily Profile

Daily Profile baseline load evaluation model was proposed in [EAL17].

For a given time interval t, baseline load b_t is estimated as shown in Eq. (3.6) (the calculation is done for each time interval during the DR event).

$$b_t = \frac{c_{d,t-1} \cdot c_{d-1,t}}{c_{d-1,t-1}} \tag{3.6}$$

Baseline load is equal to the electricity consumption within the preceding hour $c_{d,t-1}$ multiplied by the increase/decrease of electricity consumption in the corresponding hours $c_{d-1,t-1}$ and $c_{d-1,t}$ a day before the DR event activation.

3.2.2 Naive Model

Naive model, also known as persistence model, is a conventional benchmark in forecasting. For the forecast horizon H, a formula for forecasting the values $\hat{x}_{T+h}, h \in \overline{1, H}$, succeeding after x_T, with the Naive model is given in Eq. (3.7). [HA18, Chapter 3.1]

$$\hat{x}_{T+h|T} = x_T \tag{3.7}$$

Naive model belongs to a class of multi-step univariate forecasting models.

3.2.3 ARIMA

Autoregressive Integrated Moving Average $ARIMA(p, d, q)$, where p, d and q are parameters, represents a class of models which model time series (or stochastic processes—collections of random variables indexed by time) based on its own lagged values and lagged forecast errors. $ARIMA(p, d, q)$ is the compound of autoregressive $AR(p)$ (Eq. (3.8)), integrating $I(d)$, and moving average $MA(q)$ (Eq. (3.9)) parts.

$$AR(p) : x_t = \sum_{i=1}^{p} a_i \cdot x_{t-i} + \epsilon_t \tag{3.8}$$

$$MA(q) : x_t = \sum_{j=1}^{q} b_j \cdot \epsilon_{t-j} + \epsilon_t \tag{3.9}$$

In Eq. (3.8) and (3.9), ϵ_t is white noise, $\epsilon_t \sim N(0, \sigma_\epsilon^2)$. $AR(p)$ establishes the relationship between the value at time t, x_t, and its lagged values x_{t-p}, ..., x_{t-1}. $MA(q)$ establishes the relationship between the value at time t, x_t, and lagged forecast errors ϵ_{t-q}, ..., ϵ_{t-1}.

Adding $AR(p)$ and $MA(q)$, an Autoregressive Moving Average $ARMA(p, q)$ model given in Eq. (3.10) is obtained; $\epsilon_t \sim N(0, \sigma_\epsilon^2)$.

$$ARMA(p, q) : x_t = \sum_{i=1}^{p} a_i \cdot x_{t-i} + \sum_{j=1}^{q} b_j \cdot \epsilon_{t-j} + \epsilon_t \tag{3.10}$$

Applying a backshift operator **B**, given in Eq. (3.11), $ARMA(p, q)$ can be rewritten in the way given in Eq. (3.12).

$$\mathbf{B}x_t = x_{t-1} \tag{3.11}$$

$$ARMA(p, q) : \sum_{j=1}^{q} b_j \cdot \mathbf{B}^j \epsilon_t + \epsilon_t = (1 - \sum_{i=1}^{p} a_i \cdot \mathbf{B}^i) x_t = \theta(\mathbf{B}) x_t \tag{3.12}$$

$ARMA(p, q)$ is restricted to modelling stationary (statistical characteristics do not change over time) stochastic processes only. Still, time series are not usually realizations of stationary stochastic processes. For example, a time series may contain a trend and is, therefore, a realization of a non-stationary stochastic process.

Stationarity of a stochastic process is satisfied in case all of the absolute values of the inverse roots $r_k, k \in \overline{1, p}$ of the characteristic equation (3.13) (solved for **B**) are lower than unity, i.e., $|r_k| < 1, k \in \overline{1, p}$. If one or more of the inverse roots r_k satisfy the condition $|r_k| = 1$ (unit root is the solution of the characteristic equation (3.13)), the stochastic process is considered to be

non-stationary. In case a trend is present in time series, the multiplicity of a unit root defines its order.

$$\theta(\mathbf{B}) = 0 \tag{3.13}$$

To transform the non-stationary process with a unit root of the characteristic equation (3.13) of multiplicity d into a stationary stochastic process, a difference operator ∇, given in Eq. (3.14), is applied to the values of time series for d times.

$$\nabla x_t = x_t - x_{t-1} = (1 - \mathbf{B})x_t \tag{3.14}$$

$ARIMA(p, d, q)$ is the modification of $ARMA(p, q)$ which involves the integrating part $I(d) = \nabla^d$ and can be used for both stationary and non-stationary stochastic processes. The formula for $ARIMA(p, d, q)$ is given in Eq. (3.15); $\epsilon_t \sim N(0, \sigma_\epsilon^2)$.

$$ARIMA(p, d, q) : x_t = \sum_{i=1}^{p} a_i \cdot \nabla^d x_{t-i} + \sum_{j=1}^{q} b_j \cdot \epsilon_{t-j} + \epsilon_t \tag{3.15}$$

Any of the parameters p, d or q may be set to zero and in that case the corresponding part ($AR(p)$, $I(d)$ or $MA(q)$) of the $ARIMA(p, d, q)$ model is not used. [LHZS16, Zha18, Rao18]

ARIMA belongs to a class of multi-step univariate forecasting models.

3.3 Framing the Problem as Supervised Learning

Supervised learning is solving a problem of predicting target values (labels) y given input values (features) x. The goal of supervised learning is to construct a model f_Θ that maps x to y, $f_\Theta(x) = y$. The parameters Θ of the f_Θ are chosen based on some collection of n labeled instances $\{x_i, y_i\}_{i=1}^n$ to establish the relationship between the input and target values and be able to correctly predict target values for previously unobserved input values. [ZLLS20]

In this work, supervised learning is applied to fit deep learning models for electric load forecasting based on historical data. The considered forecasting problem is a regression problem where H values are forecasted based on W historical observations; H is a forecast horizon, $H > 1$; W is a time window, $W > 1$.

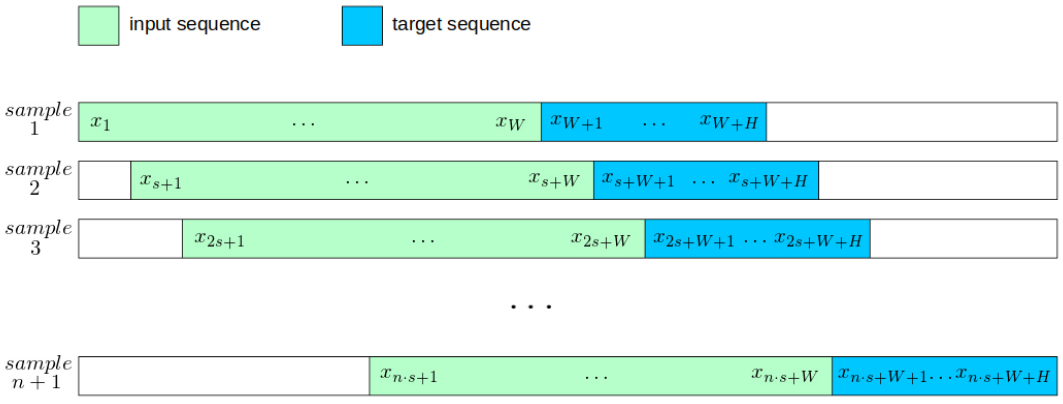


Figure 3.1: Sliding window method

The aforementioned regression problem can be framed as a supervised learning problem by applying the sliding window method to the time series: a window of size W is placed on the input sequence, and a window of size H is placed on the corresponding target sequence. The

target sequence is usually located right after the input one. To cover all the time series data, the widows are simultaneously shifted to the end of the time series by a fixed step s, s is usually taken equal to one. The process of applying the sliding window method to time series data is visualized in Figure 3.1.

3.4 Deep Learning Models

In this work, two CNN-based models are applied to the multi-step univariate and multivariate forecasting problems. In this section, a description of CNNs and LSTM networks as well as the two developed CNN-based models is provided.

3.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were first introduced by LeCun et al. in [LBD+89]. Usually, the CNNs consist of convolution layers, pooling layers, and fully connected layers. The CNNs are capable of capturing the local trend and scale-invariant features when the nearby data points have a strong relationship with each other. The CNNs typically combine three important properties: sparse connectivity, parameter sharing, and equivariant representations. [GBC16, Chapter 9]

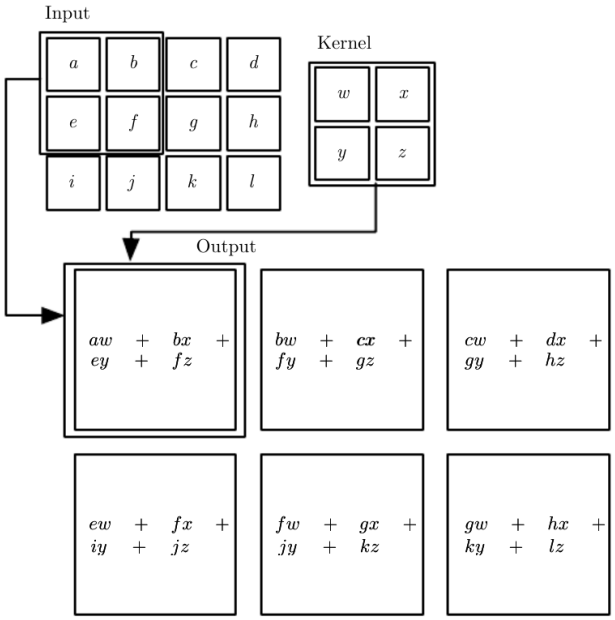


Figure 3.2: An example of two-dimensional convolution [GBC16, Chapter 9]

A convolution layer is a key component of the CNN architecture where feature extraction is performed. In a convolution layer, a convolution operation is applied to its inputs, which are multidimensional arrays, or tensors. A convolution operation is "a dot product operation between a grid-structured set of weights and similar grid-structured inputs drawn from different spatial localities in the input volume." [Agg18, Chapter 8] For instance, for two-dimensional input tensor, such as an image, a discrete convolution is defined as given in Eq. (3.16), where I is a two-dimensional input tensor, K is a kernel, and $S(i, j)$ is an output of the convolution operation. In practice, an infinite summation is implemented as a summation over a finite number of elements of the input and kernel arrays. An example of a two-dimensional convolution op-

eration is given in Figure 3.2. [GBC16, Chapter 9] The outputs of convolution layers are called feature maps, or activation maps.

$$S(i,j) = (I * K)(i,j) = \sum_{m=-\infty}^{+\infty} \sum_{n-\infty}^{+\infty} I(m,n)K(i-m, j-n) \tag{3.16}$$

A pooling layer typically follows convolution layers. In a pooling layer, a downsampling operation is applied to its input, i.e., rectangular neighborhoods of input feature maps are summarized into single output neurons. For example, max pooling outputs the maximum value within a rectangular neighborhood, as shown in Figure 3.3. Downsampling results in dimensionality reduction, which minimizes computational load, and reduces overfitting. [GBC16, Chapter 9]

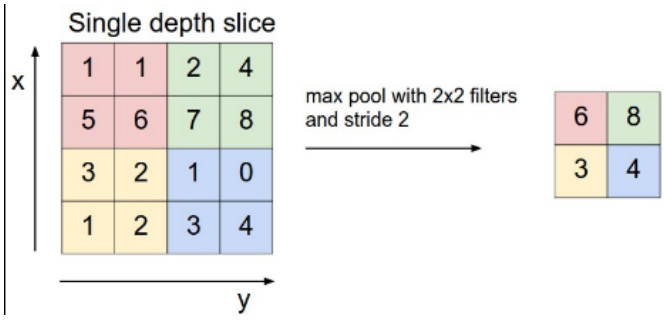


Figure 3.3: An example of max pooling [CS220]

Fully connected layers are typically added at the end of the CNN architecture. Neurons in fully connected layers have full connections to all neurons in the previous layer as in the regular neural networks.

3.4.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks, introduced in [HS97], are a variation of Recurrent Neural Networks (RNNs), designed to overcome the problems of vanishing and exploding gradient issues in RNNs, and solving the problem of learning long-term dependencies. LSTM networks, same as RNNs, are intended for processing sequences. The architecture of LSTM is given in Figure 3.4. In Figure 3.4, the yellow rectangles are learned neural network layers, the pink circles are pointwise operations, the merging lines denote concatenation.

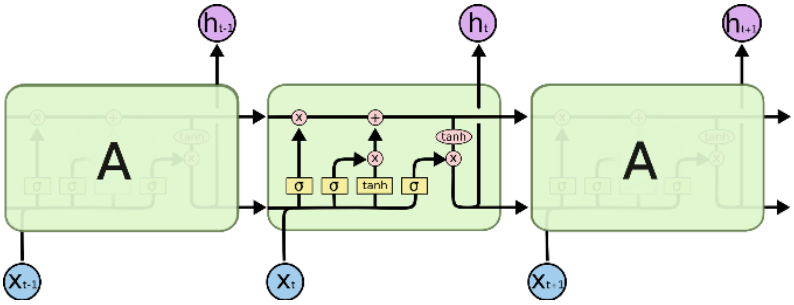


Figure 3.4: Architecture of LSTM [Ola15]

A more detailed view of an LSTM memory cell, or unit, is given in Figure 3.5.

An LSTM unit gets the hidden state h_{t-1} and the cell state C_{t-1} as inputs from the previous step as well as the input x_t. Each unit has gates that control the flow of the information in and out of the cell. An LSTM unit has three types of gates: forget gate f_t, input gate i_t, and output gate

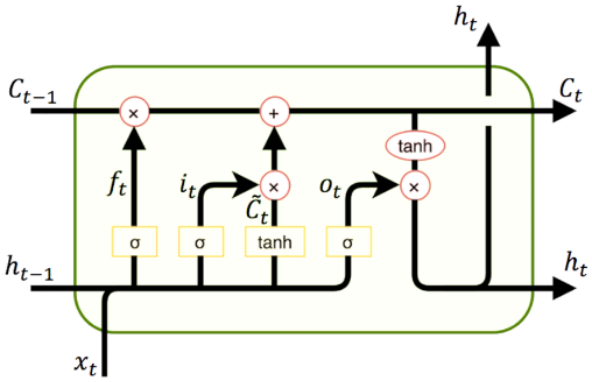


Figure 3.5: LSTM unit [BFOS19]

o_t. [Agg18, Chapter 7.5] The information flow in an LSTM unit is represented by Eqs. (3.17) to (3.22); σ is a sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$; $tanh$ is a hyperbolic tangent, $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$; W and b are weights and biases of the neural network layers respectively. [BFOS19]

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.17}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.18}$$

$$\widetilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3.19}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.20}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{3.21}$$

$$h_t = o_t \odot tanh(C_t) \tag{3.22}$$

3.4.3 Developed CNN Model

A major part of the architecture of the developed CNN model is adopted from [KH18] and is given in Figure 3.6. As an input, the model takes hourly electricity consumption data for the last week in the shape of $(168, 1)$. The explanation of the choice of the input sequence length as one week is given in section 4.2.

The developed CNN model consists of one-dimensional convolution layers (Conv1D), one-dimensional max pooling layers (MaxPooling1D), Dropout layer (Dropout), flattening layer (Flatten), fully connected layer (Dense), and two activation layers—leaky rectified linear unit (LeakyReLU) and parametric rectified linear unit (PReLU). The kernel sizes of the convolution and pooling layers, and the depths of feature maps of the convolution layers (in order from input to output of the model) are given in Table 3.1. Number of units in the Dense layer is equal to the forecast horizon $H = 24$. The developed CNN model is implemented using `Keras` library with `Tensorflow` backend; Figure 3.6 is made using `Netron` version 4.1.4 [Net].

In the proposed architecture, after each Conv1D layer, a leaky ReLU activation function is employed, given in Eq. (3.23), instead of the widely used ReLU activation function, $ReLU(z) = \max(0, z)$. It is done to overcome the "dying ReLU" problem, which refers to the neurons

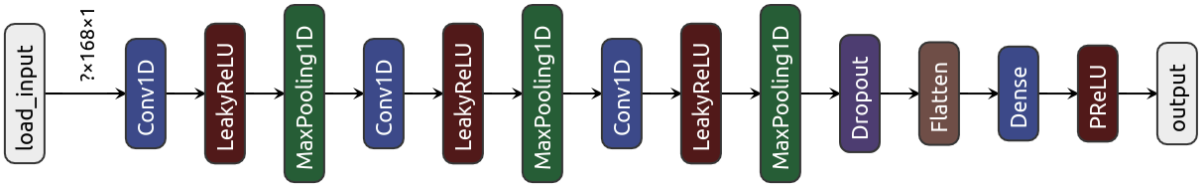


Figure 3.6: Developed CNN model architecture

Table 3.1: Parameters of convolution and max pooling layers of CNN model

Layer	Kernel size	Depth of the feature maps
Conv1D	9	16
MaxPooling1D	2	-
Conv1D	5	32
MaxPooling1D	2	-
Conv1D	5	64
MaxPooling1D	2	-

becoming inactive during training phase (once the sum of the weighted inputs of the neurons becomes negative) and only outputting zeros from that point on for any input. In Eq. (3.23), $\alpha \leq 1$ controls the leak for $z < 0$ and is set to a constant value for the whole training phase. [RGH+20, Chapter 3.4]

$$LeakyReLU(z) = \begin{cases} z, & z > 0 \\ \alpha z, & z \leq 0 \end{cases} \tag{3.23}$$

Additionally, parametric ReLU (PReLU) activation function, given in Eq. (3.24), is applied to the output of the Dense layer. It is stated in [HZRS15] that "PReLU improves model fitting with nearly zero extra computational cost and little overfitting risk." As opposed to leaky ReLU, in the parametric ReLU parameter $a \leq 1$ is trainable and is learned using the backpropagation method during the training phase. [HZRS15]

$$PReLU(z) = \begin{cases} z, & z > 0 \\ az, & z \leq 0 \end{cases} \tag{3.24}$$

The necessity for using leaky ReLU and parametric ReLU as activation functions in the developed CNN model instead of the ReLU activation function also stems from the data. After the pre-processing, described in section 4.2, the vast amount of transformed electricity consumption data values lies below zero. Hence, using ReLU activation both after the Conv1D layers and the Dense layer would result in a large number of zeros being predicted by the model. Overall, using leaky and parametric ReLU activation functions resulted in the best performance of the developed CNN model during the experiments performed in chapter 4 as compared to using ReLU, sigmoid, and hyperbolic tangent activation functions.

The Dropout layer is added before the Flatten layer to reduce overfitting. A dropout is an approach to regularization which drops out a random fraction of neurons and corresponding activations with some probability $p_{Dropout}$ during the forward and backward propagation at each step of the training phase. [RGH+20, Chapter 3.4]

The developed CNN model belongs to a class of multi-step univariate forecasting models.

3.4.4 Developed CNN+LSTM Model

The architecture of the developed CNN+LSTM model is given in Figure 3.7. The major part of CNN channels' architecture is adopted from [HWG+19]. The horizontal CNN channel is aimed to extract the electricity consumption regularity among a day, and the vertical CNN channel is aimed to extract the electricity consumption regularity between days. The LSTM branch is added to process the weather and day information data. In the developed model, the features extracted by the CNN channels and the dependencies learned by the LSTM are fused together and forwarded to the fully connected layer. As an input, the model takes hourly electricity consumption data for the last week in the shape of $(7, 24, 1)$ and weather and day information data for the next calendar day, for which the forecast is done, in the shape of $(24, 12)$ (for 12 input features). The explanation for the choice of the input sequence length as one week is given in section 4.2.

The developed CNN+LSTM model consists of two-dimensional convolution layers (Conv2D), two-dimensional max pooling layers (MaxPooling2D), LSTM layer (LSTM), Dropout layers (Dropout), batch normalization layer (BatchNormalization), concatenation layers (Concatenate), flattening layers (Flatten), fully connected layer (Dense), and two activation layers—leaky rectified linear unit (LeakyReLU) and parametric rectified linear unit (PReLU).

The kernel sizes of the convolution and pooling layers, and the depths of feature maps of the convolution layers (in order from input to output of the model) are given in Table 3.2. Number of units in the Dense layer is equal to the forecast horizon $H = 24$. In the LSTM layer, the hidden state output for each input time step is returned; the number of LSTM units is a model hyperparameter. The developed CNN+LSTM model is implemented using `Keras` library with `Tensorflow` backend; Figure 3.6 is made using `Netron` version 4.1.4 [Net].

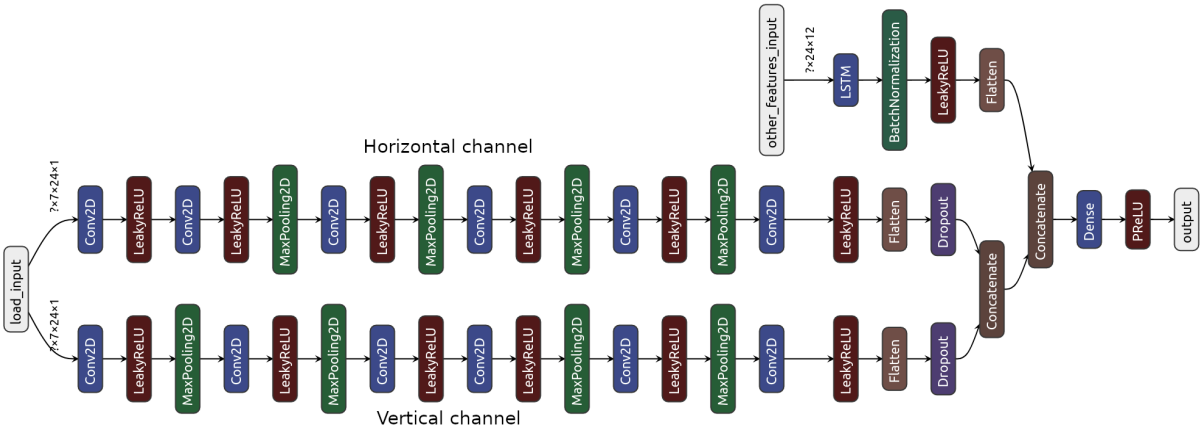


Figure 3.7: Developed CNN+LSTM model architecture

The necessity for using leaky ReLU, given in Eq. (3.23), and parametric ReLU, given in Eq. (3.24), as activation functions in the developed CNN+LSTM model instead of the widely used ReLU activation function stems mostly from the data. After the pre-processing, described in section 4.2, the vast amount of transformed electricity consumption data values lies below zero. Hence, using ReLU activation after the Conv2D layers in the CNN channels, the BatchNormalization layer in the LSTM branch, and the Dense layer would result in a large number of zeros being predicted by the model. Overall, using leaky and parametric ReLU activation functions resulted in the best performance of the CNN model during the experiments performed in chapter 4 as compared to using ReLU, sigmoid, and hyperbolic tangent activation functions.

Table 3.2: Parameters of convolution and max pooling layers of CNN+LSTM model

Horizontal channel			Vertical channel		
Layer	Kernel size	Depth of the feature maps	Layer	Kernel size	Depth of the feature maps
Conv2D	(1,7)	16	Conv2D	(4,1)	16
Conv2D	(1,5)	24	MaxPooling2D	(1,2)	-
MaxPooling2D	(1,2)	-	Conv2D	(4,1)	24
Conv2D	(1,5)	24	MaxPooling2D	(1,2)	-
MaxPooling2D	(1,2)	-	Conv2D	(3,1)	24
Conv2D	(1,4)	64	Conv2D	(3,1)	64
MaxPooling2D	(2,1)	-	MaxPooling2D	(1,2)	-
Conv2D	(1,3)	64	Conv2D	(2,1)	64
MaxPooling2D	(2,1)	-	MaxPooling2D	(2,1)	-
Conv2D	(1,3)	64	Conv2D	(2,1)	64

The Dropout layers are added before the concatenation of electricity consumption patterns extracted by both CNN channels to reduce overfitting. The recurrent dropout, which drops the linear transformation of the recurrent state with some probability $p_{RecDropout}$, is employed in the LSTM.

The batch normalization layer is added after the LSTM layer to reduce the internal covariate shift by normalizing the inputs. The internal covariate shift is "the change in the distribution of network activations due to the change in network parameters during training" [IS15]. Batch normalization facilitates faster convergence during training and additionally has a regularization effect. [IS15]

The developed CNN+LSTM model belongs to a class of multi-step multivariate forecasting models.

3.5 Evaluation Metrics

Well-adopted in papers, RMSE, MAE, and MAPE evaluation metrics are used for performance estimation of the models considered in this work.

3.5.1 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) belongs to the class of scale-dependent metrics, and is given in Eq. (3.25), where y_i is the actual value, $\hat{y}_i$ is the forecasted value, and n is the number of observations.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{3.25}$$

RMSE has the same order as the input data and is equivalent to the square root of the variance of the model output if it is unbiased. Larger error terms are penalized more, and RMSE tends to become significantly larger than MAE for outliers. [HK06]

3.5.2 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) belongs to the class of scale-dependent metrics, and is given in Eq. (3.26), where y_i is the actual value, $\hat{y}_i$ is the forecasted value, and n is the number of observations.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3.26}$$

MAE measures the average magnitude of the forecast errors without taking into account their direction. [HK06]

3.5.3 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) belongs to the class of metrics based on percentage errors, and is given in Eq. (3.27), where y_i is the actual value, $\hat{y}_i$ is the forecasted value, and n is the number of observations.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}|\frac{\hat{y}_i - y_i}{y_i}| \tag{3.27}$$

One of the main drawbacks of MAPE is being infinite or undefined for $y_i = 0$ and having a considerably skewed distribution for y_i being close to zero. [HK06] However, since in this work the aim is to use MAPE for performance evaluation of electricity consumption forecasting for high-level aggregations of customers, y_i is not expected to be close to zero.

3.6 Performance Estimation

In this section, the methods used for the performance estimation of the ARIMA and the developed CNN-based deep learning models are described. ARIMA and deep learning models require special attention in terms of performance estimation since multiple approaches can be utilized for their assessment. However, the aim is to choose the ones that produce the least biased estimates for the sake of a fair comparison of the models.

3.6.1 ARIMA

One of the most popular approaches to the performance estimation of regression models is out-of-sample evaluation, i.e., holding out the last W_{last} records of the time series as a test data. Nevertheless, the choice of W_{last} plays a significant part in the final performance estimate.

In [HA18, Chapter 3.4], Hyndman et al. suggest utilizing walk-forward cross-validation for time series forecasting problems. Walk-forward cross-validation works by forming a series of test sets, each containing H observations, where H is the forecast horizon; $H > 1$. The corresponding training sets are formed as the observations prior to the test set. The initial training set consists of m observations, which is a sufficient amount of data to obtain a reliable forecast; $m > 1$.

Therefore, at the first iteration of walk-forward cross-validation, $\{x_i\}_{i=1}^{m}$ is taken as a training set, $\{x_i\}_{i=m+1}^{m+H}$ as a test set. At the next iteration, training set is formed as $\{x_i\}_{i=1}^{m+s}$, test set is formed as $\{x_i\}_{i=m+s+1}^{m+s+H}$, where $s \geq 1$ is an algorithm step, and so on. The final performance estimate is taken as the average of performance estimates at each iteration. The process of walk-forward cross-validation is visualized in Figure 3.8.

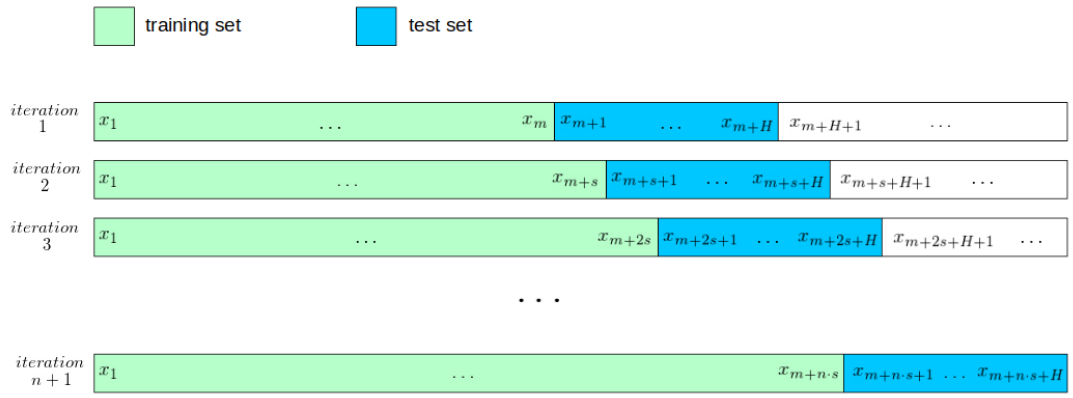


Figure 3.8: Walk-forward cross-validation algorithm

In this work, walk-forward cross-validation is applied to the performance estimation of the ARIMA.

3.6.2 Deep Learning Models

Originally, the intention was to use k-fold cross-validation as a good and robust solution for the performance estimation of deep learning models. K-fold cross-validation algorithms work by splitting the data into k separate sets and at each iteration use each of the k sets as a test set, while the rest $k-1$ sets are used as a training set. K-fold cross-validation guarantees an effective usage of data for the model training and evaluation. K-fold cross-validation is known for giving a more robust performance estimation than out-of-sample evaluation.

K-fold cross-validation is considered to be the most appropriate approach in case both independence and identical distribution among observations can be assumed. However, these assumptions are usually not valid for the time series data. In most sources, k-fold cross-validation is not recommended for the time series forecasting tasks. Despite that, research has been done in this direction.

In [BB12] Bergmeir et al., based on the empirical study, suggest the usage of blocked k-fold cross-validation, together with the adequate control for stationarity of the time series. Blocked k-fold cross-validation works by splitting the data into k separate adjacent continuous sets $J = \{J_1, \ldots, J_k\}$, where $J_i \cap_{i \neq j} J_j = \varnothing$. Let $J_{i^-} = \cup_{i \neq j} J_j$. Then, for the i-th iteration of blocked k-fold cross-validation, J_{i^-} is utilized as a training set, J_i is utilized as a test set; $i \in \overline{1,k}$. The final performance estimate is taken as the average of estimates at each iteration.

In [BHK18], Bergmeir et al. elaborate on the previous results and both theoretically and empirically prove that in the case of purely (non-linear and non-parametric) autoregressive models, it is possible to utilize standard k-fold cross-validation as long as the considered models have uncorrelated errors. In the case of serial correlation of errors, k-fold cross-validation is considered to be biased. Although, based on the simulation study, as long as the data are fitted well by the model, k-fold cross-validation showed to be a better choice than out-of-sample evaluation. The authors recommend avoiding cross-validation if the models underfit the data as using cross-validation in case of underfitting may lead to underestimation of the real error.

In [CTM19], Cerqueira et al. elaborate on the topic by comparing different evaluation techniques on two case studies. Empirical results suggest the usage of blocked k-fold cross-validation for the stationary time series, while for non-stationary ones out-of-sample evaluation shows the most accurate performance estimation of the models.

As an alternative to using blocked k-fold cross-validation in this work, a prequential block

method is considered. It shows comparatively good empirical results in the study [CTM19] performed by Cerqueira et al. and is similar in its approach to the walk-forward cross-validation chosen for the evaluation of ARIMA. Prequential block method works by splitting the data into k separate adjacent continuous sets $J = \{J_1, \ldots, J_k\}$, where $J_i \cap_{i \neq j} J_j = \varnothing$. Then, for the i-th iteration of prequential block algorithm, sets $\{J_j\}_{j \leq i}$ are utilized as a training set, J_{i+1} is utilized as a test set; $i \in \overline{1, k-1}$. The final performance estimate is taken as the average of performance estimates at each iteration.

Based on the empirical studies in [CTM19], k-fold cross-validation tends to underestimate, while the prequential method tends to overestimate the real error. Employing both performance estimation methods in this work can help to better assess the real performance of the CNN-based deep learning models.

More specifically, in this work, an adjusted prequential block method is used: the initial training set is formed as the first m sets $\{J_j\}_{j \leq m}$. This adjustment helps to diminish the bias of the prequential block performance estimation method since deep learning models require large amounts of data to properly capture the present in the data patterns, and a non-sufficient amount of training data would result in a drop in the models' performance.

4 Experiments

In this chapter, the models discussed in section 3 are applied to the problem of short-term load forecasting. A description of the data, the specific models' and data configurations, and the implementation details of the experiments are provided.

4.1 Data Description

4.1.1 Electricity consumption data

This work is done in collaboration with a national operator of a Norwegian Energy Monitoring System (EMS), Enoco AS, that provided total electricity consumption hourly data for three regions in Norway from 1 a.m. on the 1st of January, 2016, to 12 a.m. on the 1st of January, 2019.

Electricity is nationally distributed through a cable grid and the power is downscaled in a regular tree structure by transformers (nodes) down to a fine mesh of consumers. At a particular level in the tree, a region can be represented by 2–3 transformers. The sum of a load of these 2–3 transformers reflects the total consumption for the specific region and is defined as transformer consumption data.

However, a regular tree structure would not be sufficiently robust for a modern society. Therefore, the grids are designed with redundancy. Electricity can be routed to the end consumer through different paths and transformers (nodes). Just as in a mesh network, a failure in one node may cause a sudden change of load traffic in the neighboring transformers. Such grid construction and operational behavior can lead to unusual load curves in a single or a set of transformers. Therefore, the information around grid operational issues is needed while using the transformer data.

A different approach to get the total consumption of the specific region without operational grid disturbance is to summarise metering data from all consumers in the region: households, industry, office buildings, etc. The complete set of metering data is present, but not accessible, in the Norwegian national data hub EL-Hub. The closest approach to get the full metering data set is through a regionally dominating utility company. But since this is not a monopoly market, there are many suppliers in every region and their number of subscribers varies continuously. Enoco AS has a significant share of the regional meters in its database. Nevertheless, the number of subscribers in the Norwegian EMS database strongly varies on a daily basis. Therefore, it is not possible to be sure that the metering data set represents the actual total consumption of the region.

Hence, to use a more precise total electricity consumption in each of the three regions, in this work, the hourly transformer data is used in the experiments.

4.1.2 Weather data

In this work, hourly weather data, collected from the Norwegian Meteorological Institute's web page `eKlima`[1] is used. The description of the collected data is given in Table 4.1.

Table 4.1: Weather data

Variable	Explanation	Units
FF	Average wind speed at 10 meters above the ground in the last 10 minutes before the time of observation	m/s
DD	The general wind direction (ref. FF) in the last 10 minutes before the time of observation	degrees
FG1	The highest wind speed at 10 meters above the ground in the last hour before the time of observation	m/s
TA	Air temperature at 2 meters above the ground at the time of observation	°C
TAX	The highest noted temperature during the hour of observation	°C
TAN	The lowest noted temperature during the hour of observation	°C
RR1	Amount of precipitation in the last hour before the time of observation	mm
UU	Relative air humidity at the time of observation	%

4.2 Data Pre-Processing & Feature Engineering

As a first step of the data pre-processing, the originally collected data is expanded with the additional variables: DOW (day of the week), values are from 0 to 6 (corresponding to Monday to Sunday); Week (ISO week number), values are from 1 to 53; Holiday (holiday flag), 1 for weekends and national holidays in Norway, 0 otherwise. The total electricity consumption is saved under the variable Load.

At the further data exploration step, the data was checked for the missing values as well as for the correlations between the variables. As a result of the data exploration step, the variables TAX and TAN were removed due to the high correlation with the variable TA (>99%). The variables with missing values and the corresponding percentages of missing values for each region data are given in Table 4.2.

Based on Table 4.2, the variable RR1 was removed due to a high number of missing values. The rest of the missing values are imputed using the `R` implementation of multivariate imputation by chained equations (MICE) [vBGO11]. MICE creates multiple imputations for the same missing data, which are later pooled together. The data is imputed variable by variable according to the imputation method defined. In this work, a predictive mean matching imputation method is applied for the imputation of $m = 10$ multiple imputations for each region data. In the predictive mean matching, each missing entry is imputed with a value, randomly chosen from a small set of observed donor values with the regression-predicted values being the closest to the regression-predicted value for the missing entry. [vB18, Chapter 3.4]

The final variables present in the data set are: Date (date of the measurement), Week (ISO week number), DOW (day of the week), Time (hour of the measurement), Holiday (holiday

[1] `http://sharki.oslo.dnmi.no`

Table 4.2: Variables with missing values

Variable	Region 1	Region 2	Region 3
Load	<0.1 %	<0.1 %	<0.1 %
FF	0.1 %	0 %	<0.1 %
DD	0.1 %	0.2 %	1.5 %
FG1	0.1 %	0.2 %	1.3 %
TA	0.3 %	0 %	<0.1 %
RR1	22.4 %	100 %	100 %
UU	0.3 %	0 %	<0.1 %

flag), Load (total electricity consumption at the given date and hour), FF (average wind speed at 10 meters above the ground in the last 10 minutes), DD (the general wind direction (ref. FF) in the last 10 minutes), FG1 (the highest wind speed at 10 meters above the ground in the last hour), TA (air temperature at 2 meters above the ground at the given date and hour), UU (relative air humidity at the given date and hour). Total electricity consumption for 3 regions obtained after imputing missing values is visualized in Figure 4.1.

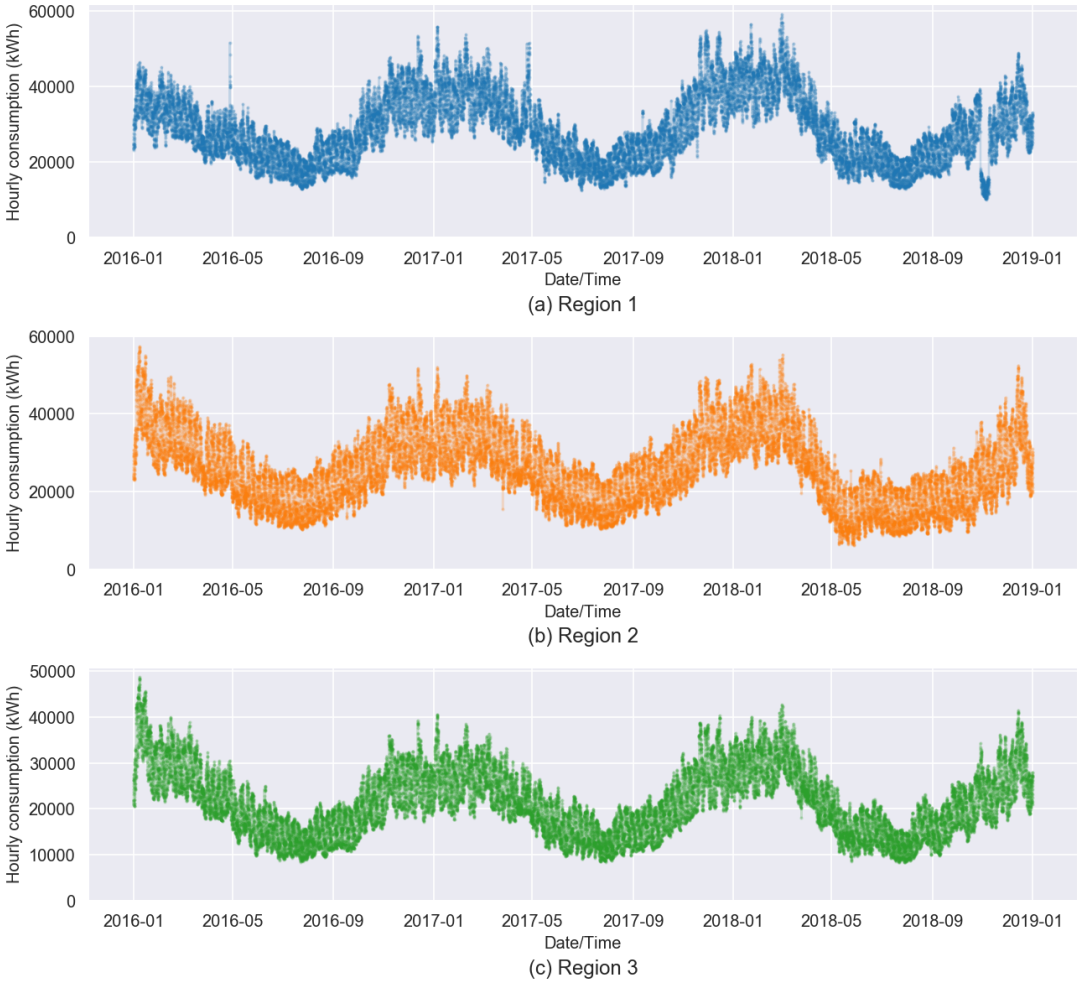


Figure 4.1: Hourly transformer data for 3 regions in Norway

Based on 4.1, one segment initially present in the data with significant outliers can be ob-

served for the region 1 data in October–November, 2018, where a significant reduction for approximately 20000 kWh occurred for around 10 days. The closer inspection of this segment is given in Figure 4.2. Based on the information provided by Enoco AS, one of the reasons for this behavior is one of the transformers used for calculating the total electricity consumption for region 1 being out of function, and electricity is being routed via a different transformer in the grid. Another reason can also be missing measurements for one of the transformers for this specific period.

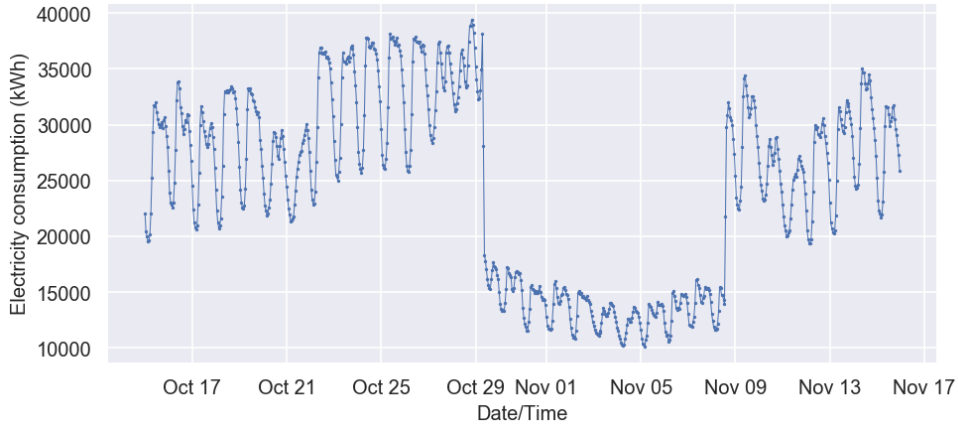


Figure 4.2: Segment of data for region 1 from 15 October 2018 to 15 November 2018

Other unusual electricity consumption curves initially present in the data include the reduction of the electricity consumption for approximately 8000 kWh for a couple of hours on the 8th of January, 2018, for region 3 data which can be caused by a large industry player stopping their production for a short period; and an unexpected reduction for approximately 6 000 kWh during night hours between the 18th and 19th of November, 2017, for region 1 data which may also be caused by an industry player stopping their production.

All the unusual data curves mentioned above were preserved unmodified in the data since they represent the real-life transformer data patterns. A small number of other irregularities with data being missing for an hour, which are supposedly errors that occurred in the production of the data sets, were fixed in the process of data pre-processing by the imputation procedure.

Continuing to explore the total electricity consumption data, based on Figure 4.1, the presence of yearly patterns in the data can be observed. To have a better overview of these patterns, box plot visualizations of the electricity consumption per month were created for each region, given in Figure 4.3. Box plot visualizations allow examination of the data distribution by displaying the minimum, first quartile, median, third quartile, and maximum values. Figure 4.3 shows that total electricity consumption data has a yearly pattern for all three regions: electricity consumption is the lowest during summer months and the highest during winter months. The extensive usage of heating devices can explain high electricity consumption during the winter months.

The data was also explored for other patterns, such as daily and hourly regularities. It was found that electricity consumption is lower during holidays/weekends as well as that there is an hourly pattern of electricity consumption as given in Figure 4.4. From Figure 4.4, it can be observed that electricity consumption is generally lower during the night hours than during the rest of the day.

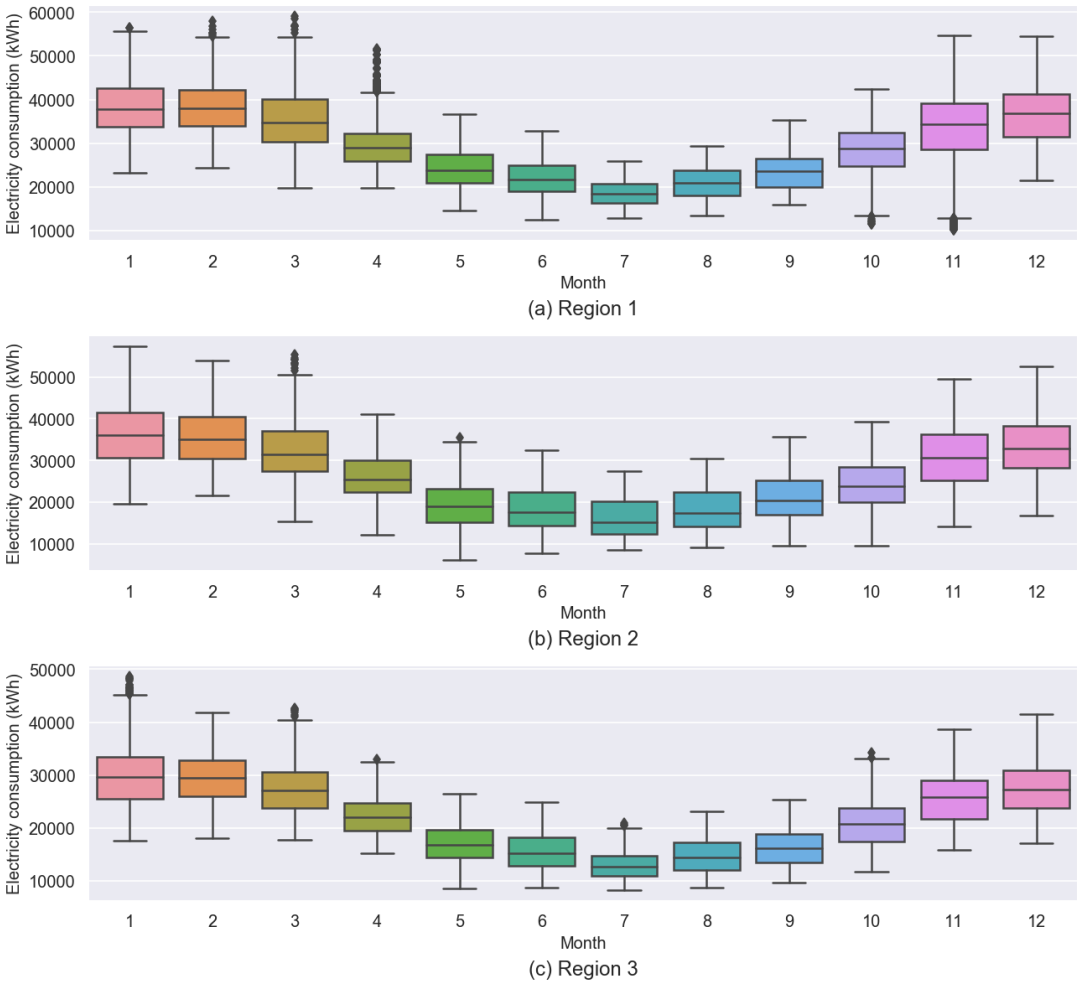


Figure 4.3: Hourly transformer data by month

4.2.1 Baseline Models

Here, an overview of the data preparation for use in the baseline models (the industry-standard baseline models, Naive model, and ARIMA) is provided.

Industry-Standard Baseline Models & Naive Model

For the industry-standard baseline models and Naive model, the data from 00:00 on the 2nd of January, 2016, to 23:00 on the 31st of December, 2018, is used. A subset of the initial data is taken to conduct a performance estimation on the same data for all models in this work.

No modifications are applied to the data.

ARIMA

For the ARIMA, the data from 00:00 on the 2nd of January, 2016, to 23:59 on the 31st of December, 2018, is used. A subset of the initial data is taken to conduct a performance estimation on the same data for all models in this work.

Box-Cox power transformation is applied to the electricity consumption variable Load, as given in Eq. (4.1), where x_t is the original value, λ is the transformation parameter, and $\psi(\lambda, x_t)$ is the transformed value. Box-Cox transformation helps to stabilize the variance, eliminate skewness, and makes the data look approximately normally distributed. It is applied to strictly

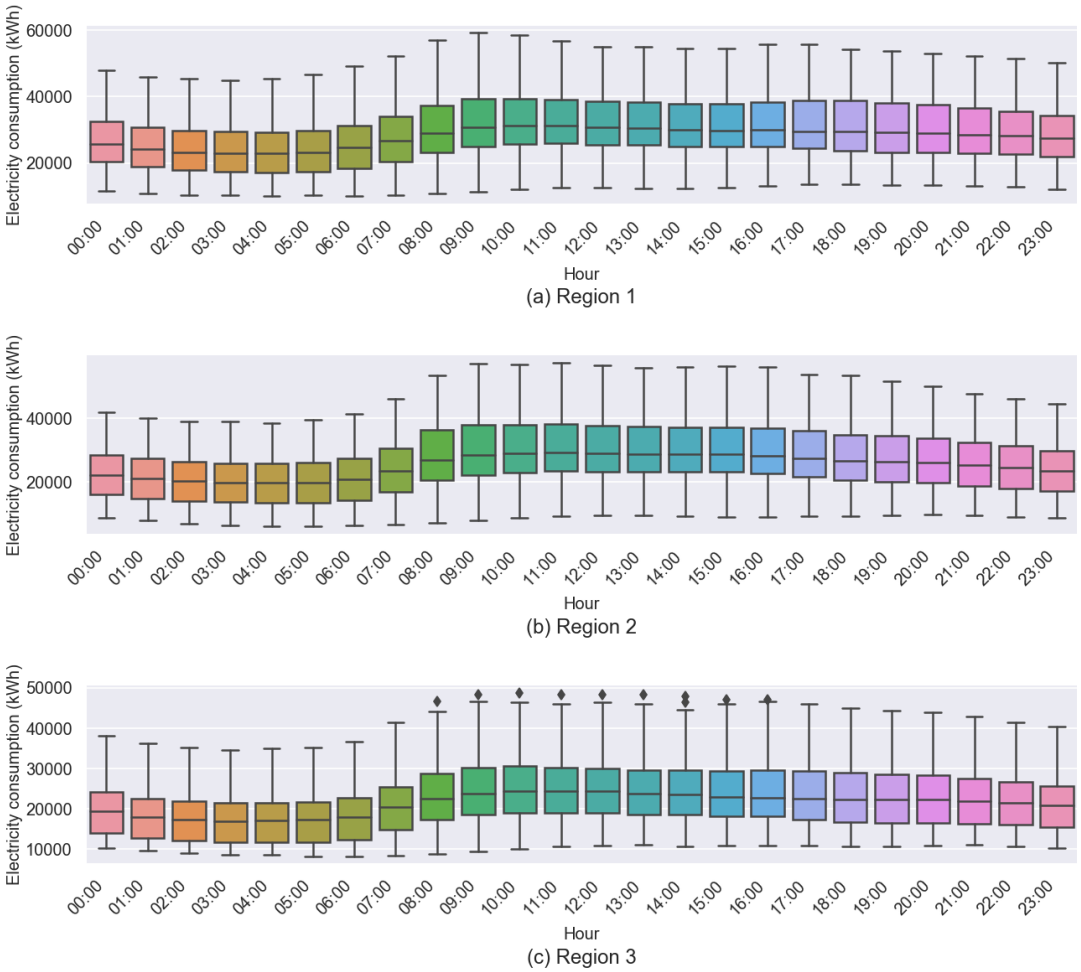


Figure 4.4: Hourly transformer data by hour

positive values and helps the model to better capture the present in the data patterns, as simpler patterns are easier to model. [HA18, Chapter 3.2]

$$\psi(\lambda, x_t) = \begin{cases} log(x_t), & \lambda = 0 \\ (x_t^\lambda - 1)/\lambda, & \lambda \neq 0 \end{cases} \tag{4.1}$$

Parameter λ is defined to maximize the profile log-likelihood of a linear model fit to the transformed data. [BC64] In this work, `sklearn` implementation of Box-Cox transformation is used.[2]

4.2.2 Deep Learning Models

For the deep learning CNN and CNN+LSTM models, the data from 00:00 on the 2nd of January, 2016, to 23:59 on the 31st of December, 2018, is used. A subset of the initial data is taken since the deep learning models are applied to the problem of one-day-ahead hourly electricity consumption forecasting for the next calendar day, starting from 00:00. Accordingly, in this work, only the hourly data for the full days, from 00:00 to 23:59 included, is used.

Week, DOW and Time variables expose periodic behavior, e.g., they repeat with some periodicity T each. Therefore, keeping them sequential, i. e., in a natural encoding, causes the

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html`

loss of information since natural encoding has no explicit representation of periodic behavior. To explicitly represent the periodic behavior of the variables, the periodic coding [DWX+19], given in Eq. (4.2), is applied to the variables DOW, Week, and Time, where n is the value in natural encoding, and T is the length of the period. $T = 52.5$ for Week, $T = 7$ for DOW, and $T = 24$ for Time. As a result, each of the periodically encoded variables is replaced by the respective sin and cos representations.

$$\begin{cases} p^{sin} = \sin \frac{2\pi n}{T} \\ p^{cos} = \cos \frac{2\pi n}{T} \end{cases} \tag{4.2}$$

Yeo-Johnson power transformation is applied to the variables Load, FG1, FF, DD, TA, and UU, as given in Eq. (4.3), where x_t is the original value, λ is the transformation parameter, $\psi(\lambda, x_t)$ is the transformed value. Similarly to Box-Cox transformation, Yeo-Johnson transformation helps to stabilize the variance, eliminate skewness, and makes the data look approximately normally distributed. Yeo-Johnson transformation can be applied to both positive and negative values.

$$\psi(\lambda, x_t) = \begin{cases} ((x_t + 1)^\lambda - 1)/\lambda, & \lambda \neq 0, x_t \geq 0 \\ log(x_t + 1), & \lambda = 0, x_t \geq 0 \\ -((-x_t + 1)^{2-\lambda} - 1)/(2 - \lambda), & \lambda \neq 2, x_t < 0 \\ -log(-x_t + 1), & \lambda = 2, x_t < 0 \end{cases} \tag{4.3}$$

Parameter λ is defined to maximize the profile log-likelihood of a linear model fit to the transformed data. [YJ00] In this work, the `sklearn` implementation of Yeo-Johnson transformation is used.[3]

After applying the Yeo-Johnson transformation, all the continuous variables are put in approximately the same range employing Z-score normalization, given in Eq. (4.4), to the variables Load, FG1, FF, DD, TA, and UU; x_t is the original value, μ is the mean of the values of each separate variable, σ is the standard deviation of the values of each separate variable. Putting variables in approximately the same range eliminates the prevalence of one variable's values over the other variables' values in the deep learning model and facilitates faster convergence during training.

$$z_t = \frac{x_t - \mu}{\sigma} \tag{4.4}$$

In the experiments, the best results were achieved by employing Z-score normalization, as compared to Min-Max normalization, given in Eq. (4.5), and Decimal scaling normalization, given in Eq. (4.6), where j is the smallest integer such that $\max |y_t| < 1$.

$$y_t = \frac{x_t - min}{max - min} \tag{4.5}$$

$$y_t = \frac{x_t}{10^j}, \quad j \in \mathbb{Z} \tag{4.6}$$

Eventually, the data is transformed to supervised learning, as described in section 3.3.

The serial correlations between the daily electricity consumption profiles, i.e., hourly sequences from 00:00 to 23:59 for each day, for 28 lags (=4 weeks) are examined to define how "informative" for the one-day-ahead hourly forecast is the data from the lag l day.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html`

The results have shown a strong correlation (> 0.7) for all 28 lags for all three regions' data sets. The highest correlations (≈ 0.9 or > 0.9) are observed for the lags $7 \cdot n$ for $n = \overline{1,4}$. It shows the presence of a weekly pattern in the data for all three regions. This observation is also supported by the previous exploration of the data sets.

It was decided to use the previous week's observations as an input sequence for the CNN and CNN+LSTM models to have the right balance between the models' training time and performance. However, in future works, it would be useful to choose the input sequence length as each model's hyperparameter.

To transform the total electricity consumption variable Load to supervised learning samples, the time window is specified as $W = 168$ (one week of hourly observations), the forecast horizon as $H = 24$, the step for shifting the input/target windows as $s = 24$. As for the rest of the variables, only the input (training) set is created utilizing the values which correspond to the time steps for the Load variable's target (test) set.

4.3 Implementation Details

4.3.1 Industry-Standard Baseline Models

Industry-standard baseline models, described in section 3.2.1, are used for the baseline load evaluation and are not meant for the 24-hour-ahead hourly electricity consumption forecasting. Hence, the performance estimation of the industry-standard baseline models differs from an approach used for the rest of the models in this work.

To evaluate the performance of the industry-standard baseline models, each model is applied to every date/time present in the data (where applicable according to the model definition), and the specified in section 3.5 metrics are calculated on the whole data sets at once. For Asymmetric HFoT and SPFoT, for baseline load estimation for holidays, holidays are not excluded. In this work, the performance estimation of the industry-standard baseline models is done under the assumptions of DR events not lasting over an hour and being activated at the beginning of the hour.

4.3.2 Naive Model

In this section, estimation of the performance of the Naive model, described in section 3.2.2, is done. The Naive model is solving the problem of the one-day-ahead hourly electricity consumption forecasting for the next calendar day, starting from 00:00. For each day present in the data, total electricity consumption is forecasted according to the Eq. (3.7) where applicable. It means that hourly electricity consumption for the day, for which the one-day-ahead forecast is done, is forecasted as electricity consumption at 23:00 of the preceding day. The specified in section 3.5 metrics are calculated on the whole data sets at once.

4.3.3 ARIMA

In this section, estimation of the performance of the ARIMA, described in section 3.2.3, is done. The ARIMA is solving the problem of 24-hour-ahead forecasting. Based on the previous data exploratory analysis, the data for all regions contain periodic (seasonal) patterns. However, the previously described ARIMA model does not capture periodic patters and consequently cannot possibly model the periodic (seasonal) data well enough.

To check for the data periodicity, the R implementation of periodogram [4] is used. Periodogram is a mathematical tool used to compute the significance of different frequencies in time-series. It is calculated by averaging the squared absolute value of discrete Fourier transform (DFT) of the signal and returns spectral density estimates for various frequencies of the signal. [SS17, Chapter 4.3] After computing the periodogram, the two most significant frequencies for each region data are taken. The chosen frequencies and the corresponding spectral density estimates are given in Table 4.3. In Table 4.3, values of Frequency are rounded to 6 decimal places, values of Period and Spectral density are rounded to the nearest integers.

Table 4.3: The most significant periodogram frequencies and spectral densities

Data	#	Frequency $(cycles/hour)$	Period=1/Frequency $(hours)$	Spectral density $(kWh^2/(cycles/hour))$
Region 1	1	0.000111	9000	444529
Region 1	2	0.041667	24	72424
Region 2	1	0.000111	9000	31698710
Region 2	2	0.041667	24	7940782
Region 3	1	0.000111	9000	1307294
Region 3	2	0.041667	24	226819

As can be seen from the table, all regions data has the same periodicity: $p_1 = 9000$ and $p_2 = 24$. To capture the multiple periodicity in the model, a modification of the ARIMA by adding Fourier terms for the corresponding periods p_1 and p_2 is implemented. The modification of the ARIMA is given in Eq. (4.7).

$$x_t = a + \sum_{i=1}^{M} \sum_{k=1}^{K_i} [\alpha sin(\frac{2\pi kt}{p_i}) + \beta cos(\frac{2\pi kt}{p_i})] + N_t \tag{4.7}$$

In Eq. (4.7), N_t is a pure ARIMA model, as described in section 3.2.3, M is a number of periods p_i, and K_i is a number of Fourier terms per each period. Based on the blog of Rob J Hyndman [Hyn10], this modification especially helps to handle the data with long seasonal periods like $p_1 = 9000$.

The best ARIMA + Fourier terms model coefficients are determined using the corrected Akaike Information Criterion (AICc): the smaller value of AICc corresponds to the better model. Based on [HA18, Chapter 8.6], the formula of AICc for model presented in Eq. (4.7) is given in Eq. (4.8).

$$AICc = -2 \cdot log(L) + \frac{2 \cdot T \cdot (p + q + k + \sum_{i=1}^{M} 2 \cdot K_i + 1)}{T - p - q - k - \sum_{i=1}^{M} 2 \cdot K_i - 2)} \tag{4.8}$$

In Eq. (4.8), L is the likelihood of the data, i.e., the probability of the data coming from the estimated model, T is the number of observations in the data, and $k = 1$ if $a \neq 0$, $k = 0$ otherwise.

T R function auto.arima() [5] is used to determine the parameters p, d, q of the ARIMA model N_t for each region data set. The order of differencing d is determined by utilizing

[4] https://www.rdocumentation.org/packages/TSA/versions/1.2/topics/periodogram

[5] https://www.rdocumentation.org/packages/forecast/versions/8.12/topics/auto.arima

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. KPSS test is testing the null hypothesis of time-series stationarity around a fixed level against an alternative of a unit root. [KPSS92] p and q are determined as such which minimize AICc of the model.

As the first step of performance estimation of ARIMA + Fourier terms model, the values K_i, $i = \overline{1,2}$, and parameters p, d, q are chosen by minimizing AICc on the whole data sets.

First of all, the pure ARIMA N_t is fitted on the whole data sets. The optimal parameters and corresponding AICc are given in Table 4.4. Values of AICc in Table 4.4 are rounded to the nearest integers.

Table 4.4: ARIMA optimal parameters and corresponding AICc

Data	p	d	q	AICc
Region 1	2	1	2	33913
Region 2	2	1	4	160354
Region 3	4	1	5	59729

In the next step, the goal is find the optimal K_i and p, d, q to minimize AICc compared to the pure ARIMA. The optimal K_1 and K_2 are determined by applying brute-force search. For K_1, the search is performed among values $\overline{1,5}$, for K_2, among values $\overline{1,12}$. By definition, $K_i \leq \frac{p_i}{2}$. p, d, and q are determined separately for each combination of K_1 and K_2. The optimal parameters obtained are given in Table 4.5. Values of AICc in Table 4.5 are rounded to the nearest integers.

Table 4.5: ARIMA + Fourier terms optimal parameters and corresponding AICc

Data	K_1	K_2	p	d	q	AICc
Region 1	1	12	3	1	1	22098
Region 2	1	12	2	1	1	148020
Region 3	2	12	5	1	4	43291

Comparing AICc in Table 4.4 and Table 4.5, a significant improvement in models AICc can be seen after complementing pure ARIMA with Fourier terms.

In order to evaluate the performance of the determined in Table 4.5 ARIMA + Fourier terms model configurations using specified in section 3.5 metrics, walk-forward cross-validation, described in section 3.6.1, is applied. As an initial data set, the first 9000 observations, corresponding to the period p_1 for each region data, are taken. The walk-forward cross-validation is implemented with the step $s = 24$ and the forecast horizon $H = 24$.

4.3.4 Deep Learning Models

In this section, estimation of the performance of the CNN and CNN+LSTM models, described in section 3.4, is done. Both the CNN and CNN+LSTM are solving the problem of the one-day-ahead hourly forecasting for the next calendar day.

Training (fitting) of the models is done by applying a gradient descent optimization algorithm Adam (Adaptive Moment Estimation) [KB14] to the minimization of the RMSE loss

function given in Eq. (4.9). RMSE increases as increases the variance linked with the frequency distribution of error magnitudes. It is beneficial when large errors are particularly undesirable. In Eq. (4.9), $\hat{X}_f$ is the vector of forecasted values, X_f is the vector of true values.

$$loss(\hat{X}_f, X_f) = \sqrt{\frac{1}{24} \cdot \sum_{i=T}^{T+24} (x_i - \hat{x}_i)^2} \tag{4.9}$$

Each model architecture and model training process is described by a set of hyperparameters such as dropout rate, leaky ReLU α, learning rate, etc. Optimally chosen hyperparameters guarantee faster and better convergence of the models during training process. In this work, Tree Parzen Estimator (TPE) method is used for hyperparameter optimization proposed in [BBBK11] by Bergstra et al. TPE method is suitable for high-dimensional optimization tasks and small model fitness evaluation budgets.

Tree Parzen Estimator (TPE) is a hyperparameter optimization method which belongs to the Sequential Model Bayesian Optimization (SMBO) family of methods. SMBO methods, which are a succinct formalism of Bayesian optimization, work by sequentially narrowing down a search space specified for each hyperparameter based on the previous results.

TPE works in the way of trials (iterations). At first, hyperparameters are chosen randomly from the search space. At each of the following iterations, given a search history in the format of $(hyperparameter, loss)$ and stochastic expression of each separate hyperparameter, the algorithm chooses the hyperparameter value for the next trial. Correlation between the hyperparameters is not taken into account, e.g., the algorithm is applied separately to each axis of the search space.

The ultimate goal of the TPE algorithm is to maximize the criterion of Expected Improvement (EI) given in Eq. (4.10).

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) \cdot p(y \backslash x) dy \tag{4.10}$$

In Eq. (4.10), x is the hyperparameter, y^* is the target performance, and y is the loss. y^* is defined as some quantile γ of the observed y values so that the condition given in Eq. (4.11) is fulfilled.

$$\gamma = p(y < y^*) \tag{4.11}$$

TPE algorithm models $p(y)$ and $p(x \backslash y)$ instead of modelling directly $p(y \backslash x)$. TPE defines $p(y \backslash x)$ as given in Eq. (4.12), where $l(x)$ is the density formed by using the observations $\{x_i\}$ for which the corresponding loss $y(x_i) < y^*$, and $g(x)$ is the density formed by using the rest of the observations.

$$p(y \backslash x) = \begin{cases} l(x), & y < y^* \\ g(x), & y \geq y^* \end{cases} \tag{4.12}$$

Eventually, the main task of the algorithm comes down to Eq. (4.13). In each trial, the algorithm returns a candidate x^* with the largest EI.

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) \cdot \frac{p(x \backslash y) \cdot p(y)}{p(x)} dy \propto (\gamma + \frac{g(x)}{l(x)} \cdot (1 - \gamma))^{-1} \rightarrow max \tag{4.13}$$

In the modification of TPE algorithm proposed in [BYC13] Bergstra et al., $\sqrt{T}/4$ top-performing trials are used to estimate density $l(x)$ and the rest to estimate density $g(x)$. Another

modification proposed in the same work is assigning weights to the results of each trial: the most recent 25 trials results are assigned full weight, and older trials are assigned weights as values of a linear ramp from 0 to 1.0.

In this work, `Hyperas` [6] wrapper for `Hyperopt`[7] implementation of the TPE algorithm with the modifications mentioned above is used. For optimizing hyperparameters for each model for each region data, 15% of the data are taken as a validation set, and 85% are taken as a training set. Samples for the validation set are picked at equal intervals from all the supervised learning samples for each region data, resulting in the well representative of each region data validation sets. The validation loss minimization is used as an objective for the TPE algorithm.

CNN

Using the described in this section TPE algorithm, a hyperparameter optimization is performed for the developed CNN model. The search spaces were chosen based on the initial manual hyperparameters tweaking.

After 70 trials of TPE algorithm, the optimal hyperparameters, given in Table 4.6, are obtained for each region data set. In Table 4.6, $U(a, b)$ represents a uniform distribution at the interval (a, b). All values in Table 4.6 are rounded to at least 8 decimal places.

Table 4.6: CNN optimal hyperparameters

		Data		
Hyperparameter	Search space	Region 1	Region 2	Region 3
Batch size	64, 72, 80, 88	64	88	80
Number of epochs	100, 150, 200, 250	100	100	250
Learning rate	$U(0.00001, 0.01)$	0.00213016	0.00338289	0.00072592
Dropout rate	$U(0.2, 0.5)$	0.49	0.37	0.35
Leaky ReLU α_1	$U(0.2, 1.0)$	0.96220293	0.63699618	0.62338190
Leaky ReLU α_2	$U(0.2, 1.0)$	0.54657803	0.98549111	0.688064096
Leaky ReLU α_3	$U(0.2, 1.0)$	0.88307523	0.57319752	0.73069689

In order to evaluate the performance of the determined in Table 4.6 CNN model configurations using specified in section 3.5 metrics, blocked k-fold cross-validation and prequential block method are employed. The necessity of utilizing both performance estimation methods and their description are provided in section 3.6.1.

The number of folds used in both methods is taken as $k = 20$. In the prequential block method, the first 10 folds are used as the initial training set. The visualizations of the performance estimation methods for the deep learning models are provided in Appendices.

CNN+LSTM

Using the described in this section TPE algorithm, a hyperparameter optimization is performed for the developed CNN+LSTM model. The search spaces were chosen based on the initial manual hyperparameters tweaking.

[6] `https://github.com/maxpumperla/hyperas`
[7] `https://github.com/hyperopt/hyperopt`

Table 4.7: CNN+LSTM optimal hyperparameters

		Data		
Hyperparameter	Search space	Region 1	Region 2	Region 3
Batch size	64, 72, 80, 88	80	80	80
Number of epochs	100, 150, 200, 250	200	250	200
Learning rate	$U(0.0001, 0.01)$	0.00322775	0.00295860	0.00590047
Number LSTM units	6, 8, 16, 24	6	8	8
Dropout rate (CNN vertical channel)	$U(0.2, 0.5)$	0.31	0.24	0.3
Dropout rate (CNN horizontal channel)	$U(0.2, 0.5)$	0.43	0.49	0.45
Recurrent LSTM dropout rate	$U(0.2, 0.5)$	0.32	0.49	0.32
Leaky ReLU α_v (CNN vertical channel)	$U(0.2, 1.0)$	0.84586596	0.89359327	0.91367263
Leaky ReLU α_h (CNN horizontal channel)	$U(0.2, 1.0)$	0.48200531	0.41049660	0.50027713
Leaky ReLU α_{lstm} (LSTM)	$U(0.2, 1.0)$	0.48754533	0.40702055	0.66107681

The same leaky ReLU α_v was assigned for all leaky ReLU layers in the CNN vertical channel, and the same leaky ReLU α_h was assigned for all leaky ReLU layers in the CNN horizontal channel. Therefore, one leaky ReLU α hyperparameter was optimized per each CNN channel of the CNN+LSTM model.

After 70 trials of TPE algorithm, the optimal hyperparameters, given in Table 4.7, are obtained for each region data set. In Table 4.7, $U(a, b)$ represents a uniform distribution at the interval (a, b). All values in Table 4.7 are rounded to at least 8 decimal places.

In order to evaluate the performance of the determined in Table 4.7 CNN+LSTM model configurations using specified in section 3.5 metrics, blocked k-fold cross-validation and prequential block method are employed. The necessity of utilizing both performance estimation methods and their description are provided in section 3.6.1.

The number of folds used in both methods is taken as $k = 20$. In the prequential block method, the first 10 folds are used as the initial training set. The visualizations of the performance estimation methods for the deep learning models are provided in Appendices.

4.3.5 Technical Details

This work has been done on a PC with 8 CPUs Intel(R) Core(TM) i7-7700HQ @ 2.80GHz and 1 GPU NVIDIA GeForce GTX 1050Ti. An overview of the packages used in the implementation of this work is given in Appendices.

5 Results and Discussion

In this chapter, the results of the performance estimation of the explored models obtained after running experiments with the configurations described in section 4.3 are presented. Visual representations of the results are presented as well as the discussion on the comparability of the performance estimates of the considered models is introduced.

5.1 Main Results

Mean evaluation metrics obtained after running the experiments with configurations described in section 4.3 are given in Tables 5.1 and 5.2. In Table 5.1, ”method A” denotes prequential block method (20 folds, 10 folds as the initial training set), ”method B” denotes blocked 20-fold cross-validation for performance estimation. All values in Tables 5.1 and 5.2 are rounded to 2 decimal places. The evaluation metrics are calculated based on the forecasted and actual electricity consumption values after applying transformations inverse to the ones applied to the data as described in section 4.2.

Here, mean evaluation metrics for the Asymmetric HFoT, SPFoT, Average, and Daily Profile models are presented separately from the rest of the methods. The differences in approaches to forecasting between the Asymmetric HFoT, SPFoT, Average, and Daily Profile, and the rest of the models explain the necessity for this separation. This point is discussed in more detail in subsection 5.2.1.

Visual representations of the results presented in Tables 5.1 and 5.2 together with the standard deviations of evaluation metrics for the ARIMA + Fourier terms, CNN, and CNN+LSTM (due to performance estimation on various folds of the data) are given in Figures 5.1, 5.2, and 5.3. In Figures 5.1, 5.2, and 5.3, the yellow, purple, and orange bars are the mean values of evaluation metrics, and the error bars are the standard deviation of the evaluation metrics. The evaluation metrics for the methods with no error bars were calculated on the whole data sets at once.

5.2 Analysis of Models' Performance

5.2.1 Comparability of Models

In this work, the Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM are implemented to facilitate 24-hour-ahead hourly electricity consumption forecasting. It is done to allow not only baseline load evaluation in case the DR event was activated but also an assessment of the availability of flexibility products for the specific day.

On the other hand, the industry-standard baseline models, described in section 3.2.1, are used for the baseline load evaluation and are not meant for 24-hour-ahead hourly electricity consumption forecasting (considering the assumption of DR events not lasting over an hour).

Table 5.1: Mean evaluation metrics for Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM

Model	Data	RMSE (kWh)	MAE (kWh)	MAPE (%)
Naive model	Region 1	3936.76	3273.45	11.64
ARIMA + Fourier terms	Region 1	2326.62	1945.49	6.80
CNN (method A)	Region 1	1971.47	1385.02	4.93
CNN (method B)	Region 1	1864.74	1311.84	4.50
CNN+LSTM (method A)	Region 1	1819.89	1230.75	4.61
CNN+LSTM (method B)	Region 1	1742.67	1217.40	4.37
Naive model	Region 2	4152.95	3469.20	14.46
ARIMA + Fourier terms	Region 2	2628.48	2147.61	9.12
CNN (method A)	Region 2	1820.58	1344.11	5.79
CNN (method B)	Region 2	1813.34	1344.06	5.51
CNN+LSTM (method A)	Region 2	1645.46	1309.29	6.31
CNN+LSTM (method B)	Region 2	1411.83	1095.82	4.79
Naive model	Region 3	3253.71	2701.60	13.08
ARIMA + Fourier terms	Region 3	1786.83	1517.23	7.40
CNN (method A)	Region 3	1173.38	888.67	4.20
CNN (method B)	Region 3	1125.18	852.76	4.02
CNN+LSTM (method A)	Region 3	981.21	739.37	3.55
CNN+LSTM (method B)	Region 3	912.90	702.30	3.34

Table 5.2: Mean evaluation metrics for Asymmetric HFoT, SPFoT, Average, and Daily Profile

Model	Data	RMSE (kWh)	MAE (kWh)	MAPE (%)
Asymmetric HFoT	Region 1	4427.94	3035.47	11.64
SPFoT	Region 1	1819.74	1294.30	4.79
Average	Region 1	474.45	328.83	1.17
Daily Profile	Region 1	818.92	504.12	1.79
Asymmetric HFoT	Region 2	5014.45	3472.48	15.98
SPFoT	Region 2	1810.66	1365.44	5.97
Average	Region 2	637.14	430.28	1.76
Daily Profile	Region 2	1067.56	720.70	3.00
Asymmetric HFoT	Region 3	3387.97	2382.75	12.23
SPFoT	Region 3	1274.84	972.09	4.80
Average	Region 3	394.58	274.42	1.33
Daily Profile	Region 3	618.80	406.01	1.98

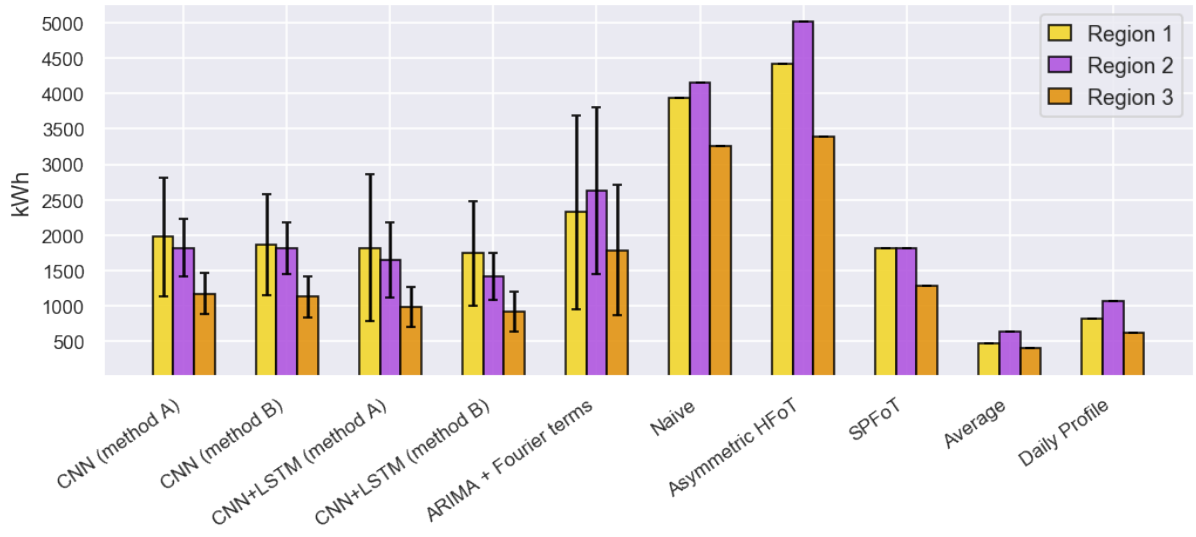


Figure 5.1: Values of RMSE on all data sets

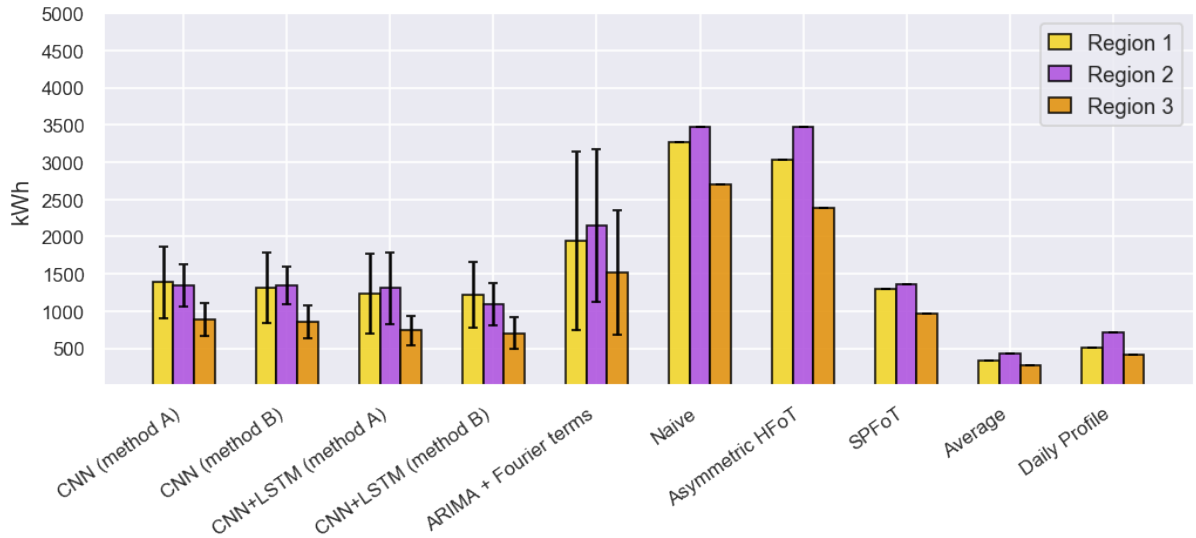


Figure 5.2: Values of MAE on all data sets

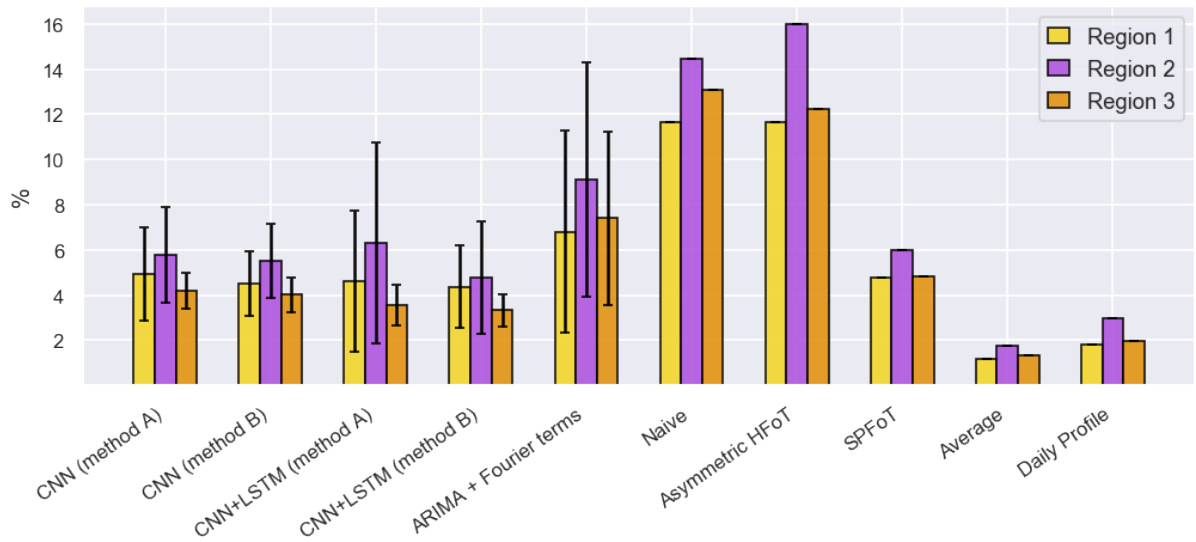


Figure 5.3: Values of MAPE on all data sets

Moreover, the approach they use for the baseline load evaluation significantly differs from the one employed in the Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM.

The industry-standard baseline models considered in this work can be divided into two classes: predictive models and analytic models. Predictive models use only the electricity consumption data prior to the DR event activation, while analytic models may also use the electricity consumption data following the DR event activation.

SPFoT and Average belong to the class of analytic models: Average uses the electricity consumption data one hour following the DR event activation for baseline load evaluation, and SPFoT uses data for the whole calendar day (independent of time of the DR event activation) to detect the days with the most similar electricity consumption patterns. Analytic models tend to be more accurate due to the usage of future electricity consumption data (following the DR event activation) and cannot be used in forecasting in the way the Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM are implemented. Throughout the chapter, by "forecast" for SPFoT and Average models, an evaluation of the baseline load is meant.

Asymmetric HFoT and Daily Profile belong to the class of predictive models. However, in the forecasts, they rely on the most recent electricity consumption data: Asymmetric HFoT uses the electricity consumption data from 1-2 hours before the DR event activation for the upward adjustment calculation, and Daily Profile uses the electricity consumption data from one hour before the DR event activation. Usage of the very recent electricity consumption data differs from the MIMO strategy with a 24-hour forecast horizon, employed in the Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM. Usage of more recent data gives a significant advantage to the Asymmetric HFoT and Daily Profile models over the Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM.

Based on the above arguments, the results for the Asymmetric HFoT, SPFoT, Average, and Daily Profile are presented separately from the Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM. Nevertheless, the comparison of the performance estimates of all used models is performed, keeping in mind the massive advantages of the Asymmetric HFoT, SPFoT, Average, and Daily Profile over the rest of the models.

5.2.2 Asymmetric HFoT

Summarizing by all evaluation metrics, the Asymmetric HFoT performs as one of the worst (at approximately the same level as Naive model) among the models presented in Tables 5.1 and 5.2.

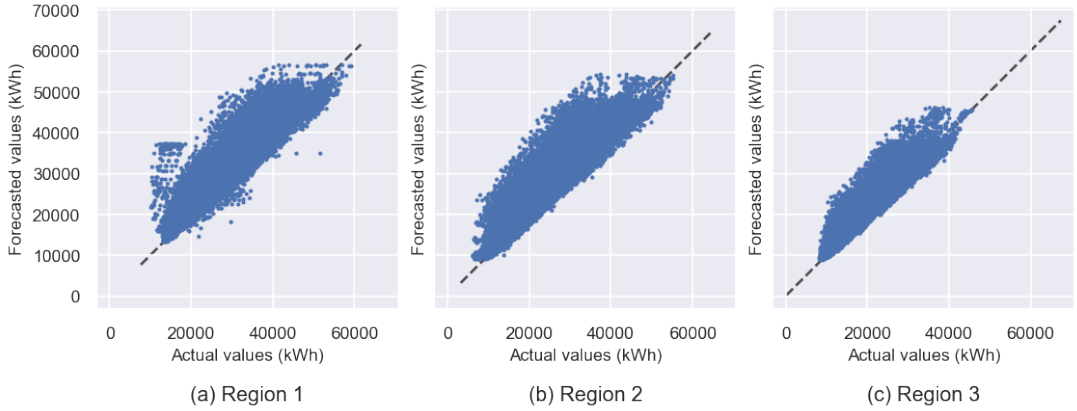


Figure 5.4: Forecasted vs. Actual electricity consumption (Asymmetric HFoT)

From Figure 5.4, it can be seen that the Asymmetric HFoT tends to overestimate the actual

electricity consumption. Significant deviations from the actual consumption can be observed for all regions' data. The deviations occur since in the electricity consumption forecast the Asymmetric HFoT relies on the days with the highest electricity consumption, and only an upward asymmetric adjustment based on the recent electricity consumption and baseline load evaluations is applied.

5.2.3 SPFoT

Summarizing by all evaluation metrics, the SPFoT performs better than the Asymmetric HFoT among the models presented in Table 5.2, and better than the ARIMA + Fourier terms and Naive model presented in Table 5.1.

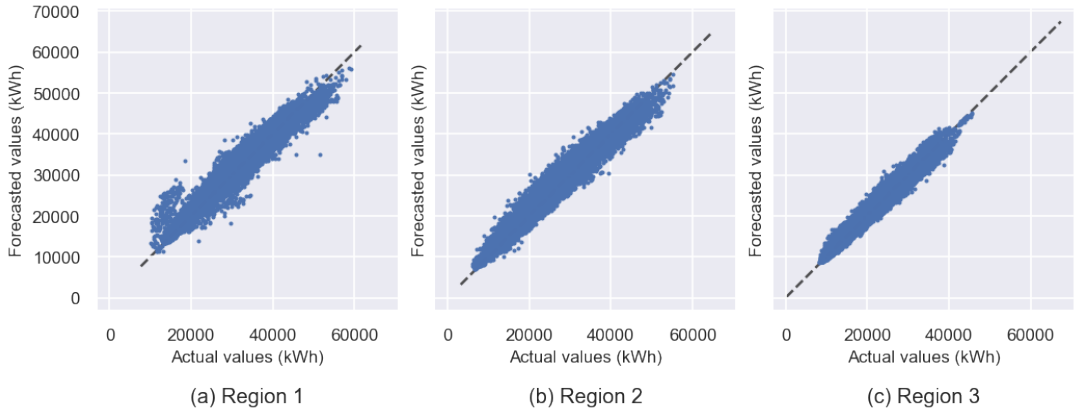


Figure 5.5: Forecasted vs. Actual electricity consumption (SPFoT)

From Figure 5.5, it can be seen that the SPFoT tends to rather equally under- and overestimate the actual electricity consumption. Considerably significant deviations from the actual consumption can be observed, especially for region 1 data, as shown in 5.5a. One of the reasons for deviations to occur may be sudden changes in electricity consumption patterns.

Comparatively good results by the SPFoT can be explained by the usage of the electricity consumption data from after the time for which the "forecast" is made to detect the days with the most similar electricity consumption patterns as well as by the usage of the most recent electricity consumption data for the symmetric adjustment calculation.

5.2.4 Average

Summarizing by all evaluation metrics, the Average performs the best among the models presented in Tables 5.1 and 5.2.

From Figure 5.6, it can be seen that the Average tends to equally under- and overestimate the actual electricity consumption and produces quite precise "forecasts" of electricity consumption. Deviations from the actual consumption are rather rare and do not seem to affect the "forecasting" ability of the model significantly. One of the reasons for deviations to occur may be sudden changes in electricity consumption patterns.

Good results by the Average can be explained by the usage of the most recent information as well as the future information (following the DR event activation) about the electricity consumption for making the "forecast", namely one hour before and one hour after the time for which the "forecast" is made.

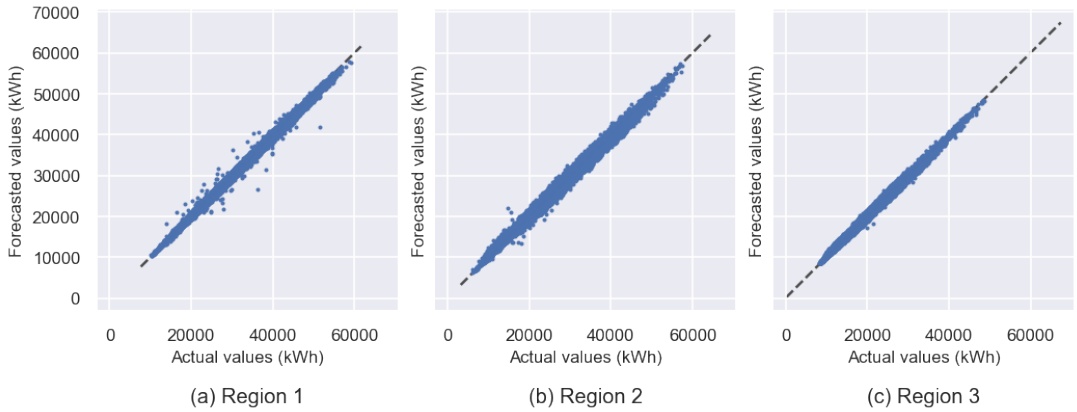


Figure 5.6: Forecasted vs. Actual electricity consumption (Average)

5.2.5 Daily Profile

Summarizing by all evaluation metrics, the Daily Profile performs worse than the Average but better than the rest of the models presented in Tables 5.1 and 5.2.

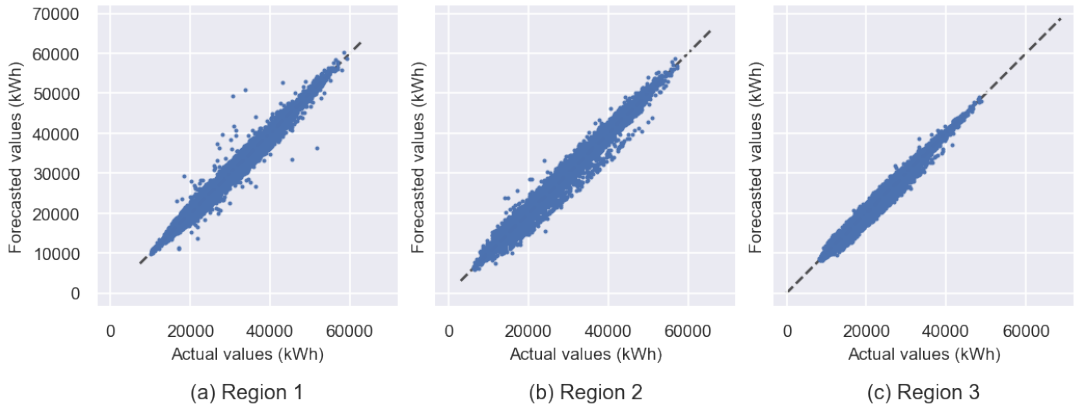


Figure 5.7: Forecasted vs. Actual electricity consumption (Daily Profile)

From Figure 5.7, it can be seen that the Daily Profile tends to rather equally under- and overestimate the actual electricity consumption. Considerably significant deviations from the actual consumption can be observed, especially for the data for regions 1 and 2, as shown in 5.7a and 5.7b. One of the reasons for deviations to occur may be sudden changes in electricity consumption patterns.

Good results by the Daily Profile can be explained by the usage of the most recent information about electricity consumption for making the forecast, namely one hour before the time for which the forecast is made.

5.2.6 Naive Model

Summarizing by all evaluation metrics, obtained by applying Eq. (3.7) to the one-day-ahead hourly electricity consumption forecasting for the next calendar day, the Naive model performs as one of the worst (at approximately the same level as the Asymmetric HFoT) among the models presented in Tables 5.1 and 5.2.

From Figure 5.8, it can be seen that the Naive model tends to underestimate the actual electricity consumption. Significant deviations from the actual consumption can be observed for all regions' data. The deviations occur since the Naive model does not learn the patterns

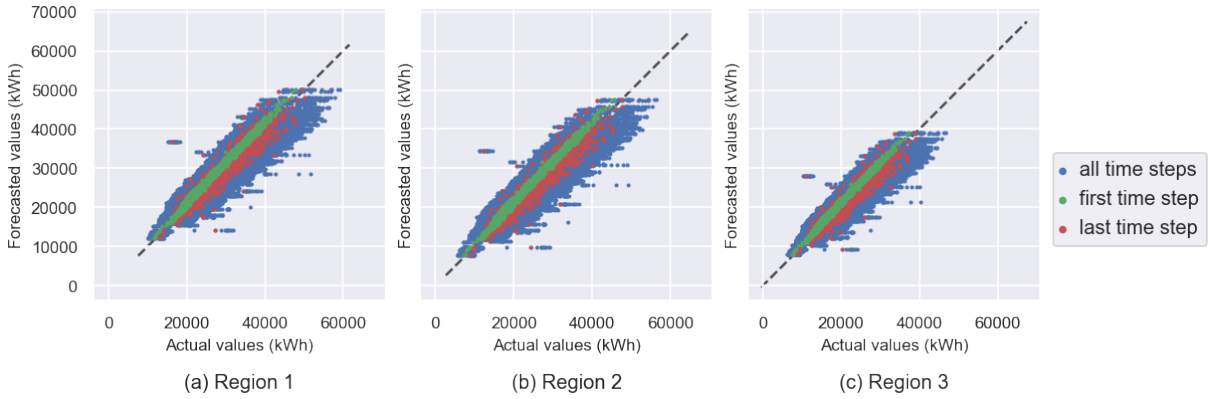


Figure 5.8: Forecasted vs. Actual electricity consumption (Naive model)

present in the data and propagates the electricity consumption at 23:00 as the constant hourly forecast for the next calendar day, for which the one-day-ahead forecast is done.

Figure 5.8 shows that the Naive model produces reliable forecasts for the first lead time, while for the later lead times, the model mostly produces values rather deviated from the actual electricity consumption. One of the reasons for such behavior is the recentness of the information used for each lead time in the forecast horizon. For the first lead time, which is 00:00 of the next calendar day, for which the forecast is done, the value from one hour before is used as a forecast, which is a relatively short period. There is a low probability of significant changes in electricity consumption in a short period. However, the probability of significant changes in electricity consumption increases with the increase of the time interval between 23:00 and the lead time. Supposedly, utilizing a smaller forecast horizon H would yield better performance estimates of the model.

5.2.7 ARIMA + Fourier Terms

Summarizing by all evaluation metrics, obtained by walk-forward cross-validation as described in section 3.6.1, the ARIMA + Fourier terms model performs better than the Naive model but worse than the CNN and CNN+LSTM among the models presented in Table 5.1. Comparing to the models presented in Table 5.2, the ARIMA + Fourier terms performs better than the Asymmetric HFoT but worse than the rest of the models.

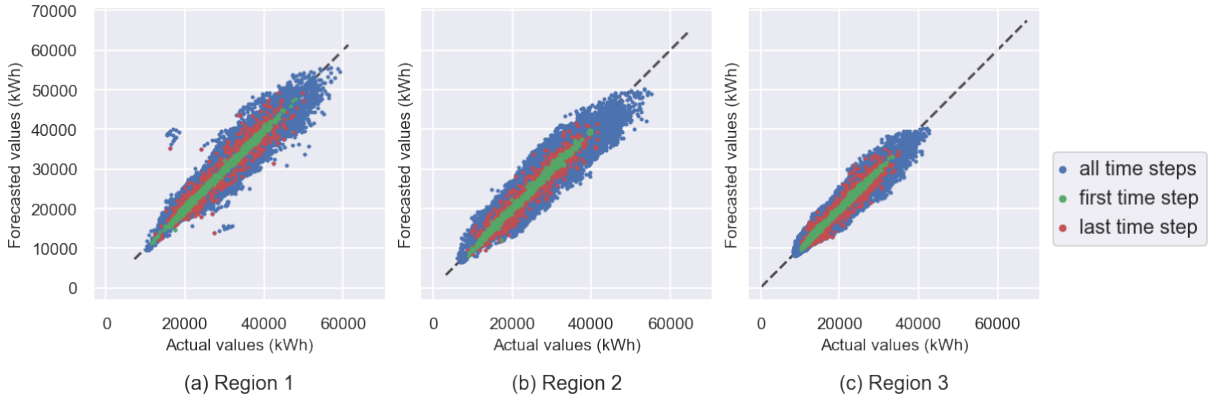


Figure 5.9: Forecasted vs. Actual electricity consumption (ARIMA + Fourier terms)

From Figure 5.9, it can be seen that the ARIMA + Fourier terms tends to quite equally under- and overestimate the actual electricity consumption. Considerably significant deviations

from the actual consumption can be observed for all regions' data. The deviations may occur due to the new (not experienced in the training set) data patterns in the test set, or the inability of the final model to capture more complex patterns.

Figure 5.9 shows that the ARIMA + Fourier terms produces reliable forecasts for the first lead time, while for the later lead times, the model mostly produces forecasts rather deviated from the actual values. One of the reasons for such behavior may be the recentness of the information used for each lead time in the forecast horizon. The forecast for the first lead time is made by employing the most recent information about the electricity consumption, while the forecasts for the later lead times are made by employing the information from at least 2 to 24 hours before. Supposedly, utilizing a smaller forecast horizon H for ARIMA + Fourier terms would yield better performance estimates of the model.

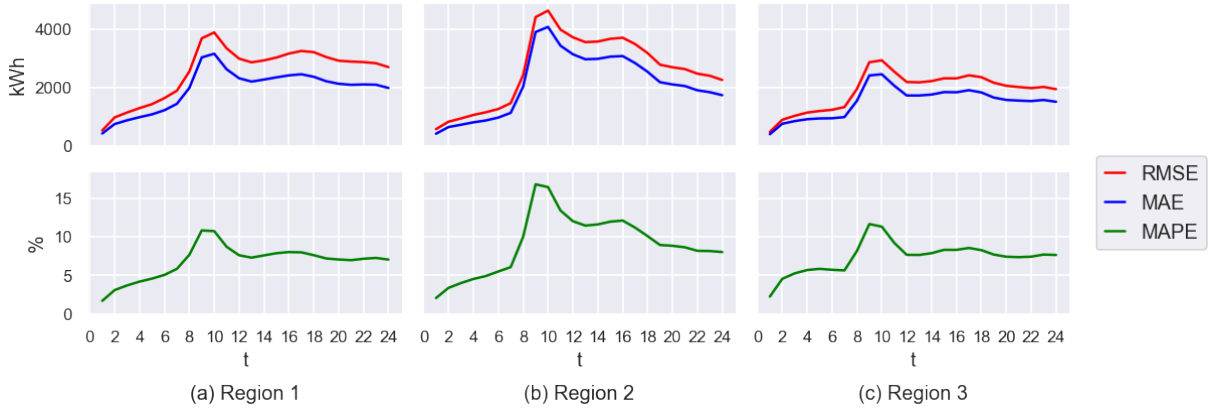


Figure 5.10: Evaluation metrics per forecast lead time (ARIMA + Fourier terms)

Based on the mean evaluation metrics per lead time for the ARIMA + Fourier terms presented in Figure 5.10, all evaluation metrics have the lowest values for the first lead time. The highest evaluation metrics for all three regions' data are observed at the lead times 8 to 10.

5.2.8 CNN and CNN+LSTM Models

In this work, the performance of the CNN and CNN+LSTM models was estimated by employing the prequential block method (20 folds, 10 folds as the initial training set) and blocked 20-fold cross-validation, described in section 3.6.2. Summarizing by all evaluation metrics, the CNN and CNN+LSTM models, employing both methods of performance estimation, have shown the best results among the models presented in Table 5.1. Comparing to the models presented in Table 5.2, the CNN and CNN+LSTM models outperform the SPFoT (except for the data for region 1, where the CNN model shows worse performance than the SPFoT) but show worse results than the Average and Daily Profile.

The CNN+LSTM model has shown better results than the CNN model for all regions' data and both methods of performance estimation, except for the data for region 2, where the CNN model has shown better mean MAPE for the prequential block method. It shows that the usage of more sophisticated CNN architectures, along with additional data like weather and day information the way it was done in the CNN+LSTM model, is beneficial for the forecasting ability of the models.

One of the crucial questions raised in section 3.6.2 was a choice of the performance estimation approach for the developed CNN-based models. Based on the results in Table 5.1, it can be seen that blocked 20-fold cross-validation produced significant results comparing to the prequential block method despite the concerns about the validity of cross-validation for time

series data. By significant results, it is meant that the ranks by evaluation metrics of the developed CNN-based models stay the same, independent of the choice of performance estimation approach as compared to the rest of the models and one another. Consequently, the aim is to study only visualizations for the results obtained by blocked 20-fold cross-validation (CV) as it can give a better overview of model performance by utilizing all available data for training and evaluation.

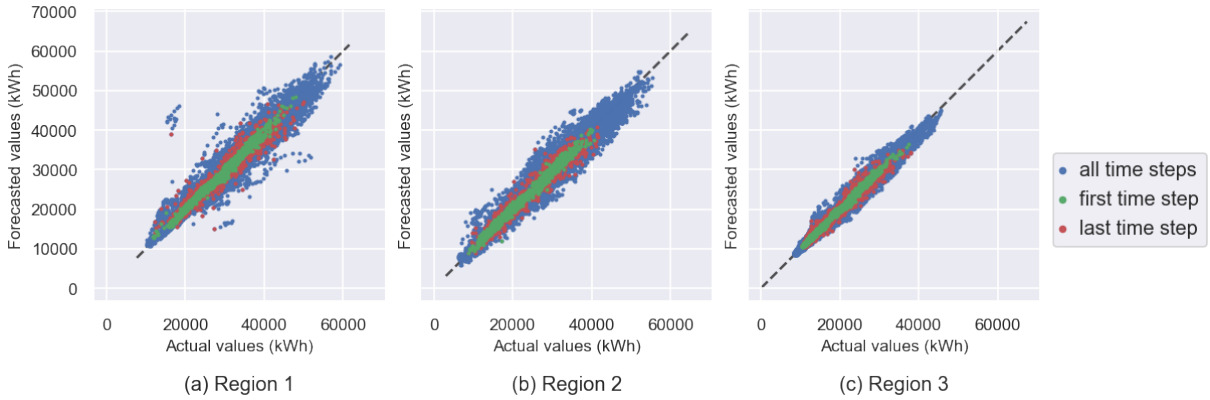


Figure 5.11: Forecasted vs. Actual electricity consumption (CNN, blocked 20-fold CV)

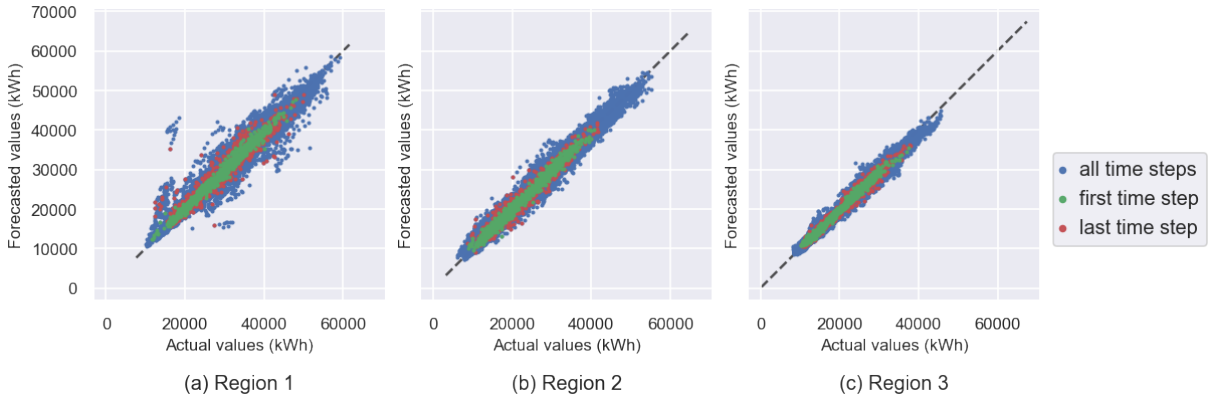


Figure 5.12: Forecasted vs. Actual electricity consumption (CNN+LSTM, blocked 20-fold CV)

Based on Figures 5.11 and 5.12, it can be seen that the developed CNN-based models produce relatively good forecasts comparing to the other methods presented in Table 5.1. Both the CNN and CNN+LSTM models tend to rather equally under- and overestimate the actual electricity consumption. However, there are considerably significant deviations from the actual values for both the CNN and CNN+LSTM models for the data for region 1, as shown in Figures 5.11a and 5.12a. The deviations may occur due to the new (not experienced in the training set) data patterns in the test set or the inability of the final model to capture more complex patterns.

From Figures 5.11 and 5.12, it can be observed that both the CNN and CNN+LSTM models produce reasonably good results for the first lead time, while for the later lead times the models mostly produce forecasts rather deviated from the actual values. One of the reasons for such behavior may be the recentness of the information used for each lead time in the forecast horizon. The forecast for the first lead time is made by employing the most recent information, while for the later lead times the forecasts are made by employing the information from at least 2 to 24 hours before. Supposedly, utilizing a smaller forecast horizon H would yield better performance estimates of the CNN and CNN+LSTM models.

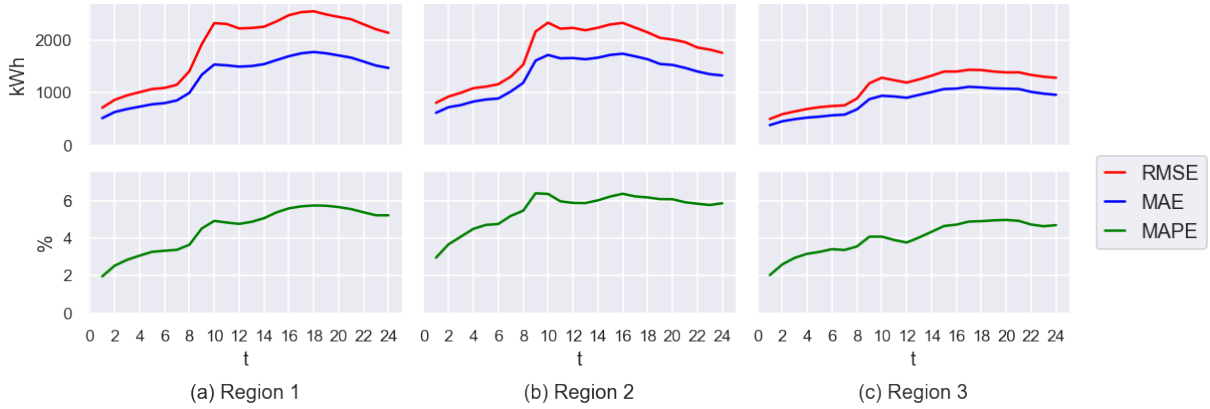


Figure 5.13: Evaluation metrics per forecast lead time (CNN, blocked 20-fold CV)

Based on the mean evaluation metrics per lead time for the CNN model, presented in Figure 5.13, all evaluation metrics have the lowest values for the first lead time and tend to increase for the later lead times. The same behavior is observed for the mean evaluation metrics per lead time for the CNN+LSTM model, presented in Figure 5.14. In some cases, mean evaluation metrics stay on nearly the same level starting from some lead time t (MAPE in Figures 5.13b, 5.14b, and 5.14c; RMSE and MAE in Figure 5.14c), or even decrease starting from some lead time t (RMSE and MAE in Figures 5.13a, 5.13b, and 5.14a).

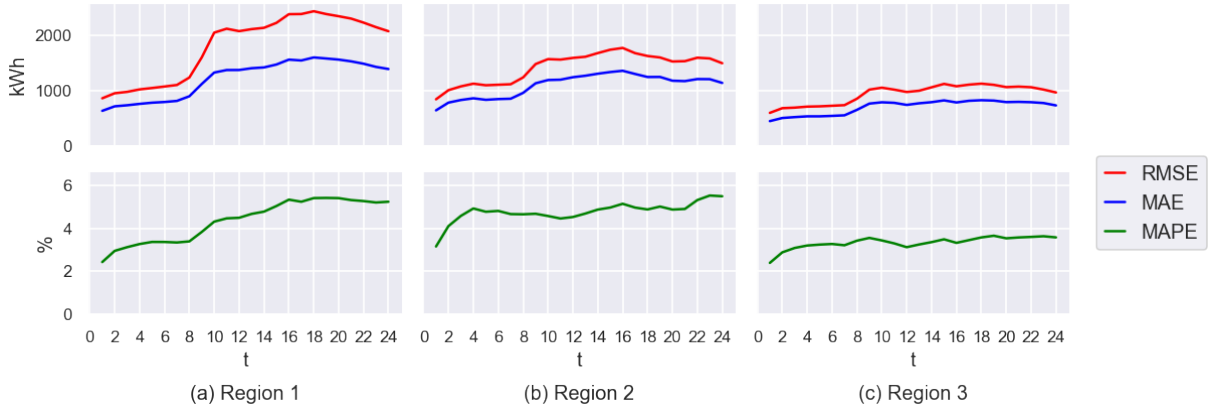


Figure 5.14: Evaluation metrics per forecast lead time (CNN+LSTM, blocked 20-fold CV)

5.3 Summary of Findings

In this chapter, the analysis and comparison of the performance of the Asymmetric HFoT, SPFoT, Average, Daily Profile, Naive model, ARIMA + Fourier terms, CNN, and CNN+LSTM models were performed.

The developed CNN+LSTM model showed the best performance results among the models presented in Table 5.1 which belong to the class of predictive models and perform the 24-hour-ahead forecasting by employing the MIMO strategy. However, as deep learning models require large amounts of data to properly capture the patterns present in the data, the performance for the same model could be improved if more data is available.

The Daily Profile showed the best performance among the predictive models which do not necessarily employ the MIMO strategy or are not capable of performing 24-hour-ahead forecasting. However, it should be taken into account that considering the assumption of DR events

not lasting over an hour, the Daily Profile in this work can only be applied to one-hour-ahead forecasting. It is assumed that given a sufficient amount of data, the CNN and CNN+LSTM models would outperform the Daily Profile if implementing one-step-ahead forecasts.

It is worth noting that the Average showed the best performance in the baseline load evaluation among all the models presented in Tables 5.1 and 5.2. However, the Average belongs to the class of analytic models which use the electricity consumption data following the DR event activation and consequently cannot be applied to electricity consumption forecasting.

6 Conclusion

In this work, two CNN-based deep learning models for the one-day-ahead hourly electricity consumption forecasting for the next day employing the MIMO strategy were developed. The industry-standard baseline models Asymmetric HFoT, SPFoT, Average, and Daily Profile, along with ARIMA + Fourier terms and Naive model, were used as baseline models for comparison.

The performance of the models was estimated on the electricity consumption data for three regions in Norway. Configurations of the developed CNN and CNN+LSTM models as well as configurations for the ARIMA + Fourier terms were determined separately for each region data. Performance estimation of the CNN-based models was done by employing both 20-fold cross-validation and prequential block method (20-folds, 10 folds as an initial training set). Performance estimation of the ARIMA + Fourier terms was done by employing walk-forward cross-validation.

The developed CNN+LSTM model showed the best performance results among the Naive model, ARIMA + Fourier terms, and CNN-based deep learning models that belong to the class of predictive models and perform the 24-hour-ahead hourly forecasting for the next calendar day by employing the MIMO strategy. It shows that using more sophisticated CNN architectures, along with additional data like weather and day information as it was done in the CNN+LSTM model, is beneficial for the models' forecasting ability.

The Daily Profile showed the best performance among the predictive models which do not necessarily employ the MIMO strategy or are not capable of performing 24-hour-ahead hourly forecasting. However, considering the assumption of DR events not lasting over an hour, the Daily Profile in this work can be applied only to one-hour-ahead forecasting and is not suitable in case a more extended forecast is needed.

The Average showed the best performance in the baseline load evaluation among all the models considered in this work. However, the Average belongs to analytic models which use the electricity consumption data following the DR event activation and consequently cannot be applied to electricity consumption forecasting.

The experiments have also shown that k-fold cross-validation produced significant results comparing to the prequential block method for performance estimation, despite the concerns about its validity in time-series forecasting. By significant results, it is meant that the ranks by evaluation metrics of the developed CNN-based models stayed the same, independent of the choice of performance estimation approach.

It can be concluded that the experiments with developing and applying the CNN-based models to the problem of the one-day-ahead hourly electricity consumption forecasting for the next calendar day were successful. Both the CNN and CNN+LSTM models have shown the best results among the models which employ the MIMO strategy for performing 24-hour-ahead hourly forecasting. Two different paths could be chosen for future work: one that intends to explore more and improve the CNN-based models developed in this work, and the one which aims to explore new CNN-based architectures.

Possible improvements and future work

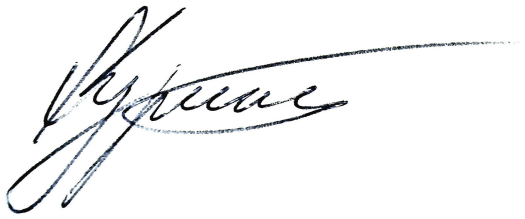
- Using another loss function for training of the CNN-based models.
- Using other activation functions in the CNN-based models, for instance, Exponential Linear Unit (ELU) [CUH15], which is smooth for all input values, helping to speed up the training.
- Modifying architectures of the CNN-based models.
- Defining the input sequence length for the CNN-based models as a hyperparameter.
- Using other hyperparameter optimization methods.
- Performing diagnostics of forecast residuals for better estimation of performance of the models. [HA18, Chapter 3.3]
- Exploring other hybrid CNN-based architectures.
- Applying the models considered in the current work to the problem of one-hour-ahead forecasting for comparison to the Daily Profile industry-standard baseline model.
- Applying the models considered in the current work to the problem of lower-level aggregations of consumers.

Acknowledgements

I would like to thank my thesis supervisors Alan Henry Tkaczyk of Institute of Technology at the University of Tartu, Florentin Dam at Akka Technologies, and Ulf Roar Aakenes at Enoco AS for helpful discussions and guidance.

I am grateful to my family, who were supporting and encouraging me throughout my studies, and during researching and writing this work. This accomplishment would not have been possible without them.

This work has been supported by the EU-SysFlex project, funded from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 773505. This publication reflects only the authors' view; responsibility for the information and views expressed herein lies entirely with the authors. Neither the European Commission nor the Innovation and Networks Executive Agency is responsible for any use that may be made of the information contained in this publication.



Bibliography

[AES07] M. H. Albadi and E. F. El-Saadany. Demand Response in Electricity Markets: An Overview. In *2007 IEEE Power Engineering Society General Meeting*, pages 1–5. IEEE, June 2007.

[Agg18] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

[ASVVMM16] Igor Aizenberg, Leonid Sheremetov, Luis Villa-Vargas, and Jorge Martinez-Muñoz. Multilayer Neural Network with Multi-Valued Neurons in time series forecasting of oil production. *Neurocomputing*, 175:980–989, January 2016.

[BB12] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, May 2012.

[BBBK11] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., December 2011.

[BC64] G. E. P. Box and D. R. Cox. An Analysis of Transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, July 1964.

[BFOS19] Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. Single and Multi-Sequence Deep Learning Models for Short and Medium Term Electric Load Forecasting. *Energies*, 12(1), January 2019.

[BHK18] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83, April 2018.

[BPKS11] V. S. K. Murthy Balijepalli, Vedanta Pradhan, S. A. Khaparde, and R. M. Shereef. Review of demand response under smart grid paradigm. In *ISGT2011-India*, pages 236–243. IEEE, December 2011.

[BYC13] James Bergstra, Dan Yamins, and David D. Cox. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *JMLR Workshop and Conference Proceedings*, volume 28 (1), pages 115–123, June 2013.

[CS220] CS231n Convolutional Neural Networks for Visual Recognition. `https://cs231n.github.io/convolutional-networks/`, Stanford University, 2020. [Online; accessed 15-May-2020].

[CTM19] Vítor Cerqueira, Luís Torgo, and Igor Mozetic. Evaluating time series forecasting models: An empirical study on performance estimation methods. *ArXiv*, abs/1905.11744, 2019.

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv*, abs/1511.07289, 2015.

[DWX+19] Zhuofu Deng, Binbin Wang, Yanlu Xu, Tengteng Xu, Chenxu Liu, and Zhiliang Zhu. Multi-Scale Convolutional Neural Network With Time-Cognition for Multi-Step Short-Term Load Forecasting. *IEEE Access*, 7:88058–88071, 2019.

[EAL17] Elering, AST, and Litgrid. Demand Response Through Aggregation — A Harmonized Approach In Baltic Region: Concept proposal, 2017.

[Ene09] EnerNOC. The Demand Response Baseline (White Paper), 2009.

[FER16] FERC. A National Assessment & Action Plan on Demand Response Potential. `https://www.ferc.gov/industries/electric/indus-act/demand-response/dr-potential.asp`, 2016. [Online; accessed 02-March-2020].

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GLA19] Alberto Gasparin, Slobodan Lukovic, and Cesare Alippi. Deep Learning for Time Series Forecasting: The Electric Load Case. *ArXiv*, abs/1907.09207, 2019.

[HA18] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice, 2nd edition*. OTexts: Melbourne, Australia, 2018. `https://OTexts.com/fpp2/` [Online; accessed 22-April-2020].

[HK06] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, October 2006.

[HR19] Ying-Yi Hong and Christian Lian Paulo P. Rioflorido. A hybrid deep learning-based neural network for 24-h ahead wind power forecasting. *Applied Energy*, 250:530–539, September 2019.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[HWG+19] Yunyou Huang, Nana Wang, Wanling Gao, Xiaoxu Guo, Chen Huang, Tianshu Hao, and Jianfeng Zhan. LoadCNN: A Low Training Cost Deep Learning Model for Day-Ahead Individual Residential Load Forecasting. *arXiv*, abs/1908.00298, 2019.

[Hyn10] Rob J. Hyndman. Forecasting with long seasonal periods. `https://robjhyndman.com/hyndsight/longseasonality/`, 2010. [Online; accessed 25-March-2020].

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, December 2015.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, July 2015.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv*, abs/1412.6980, 2014.

[KCJL19] Myoungsoo Kim, Wonik Choi, Youngjun Jeon, and Ling Liu. A Hybrid Neural Network Model for Power Demand Forecasting. *Energies*, 12(5), March 2019.

[KEM13a] AEMO DNV KEMA. Development of Demand Response Mechanism Baseline Consumption Methodology – Phase 1 Results Final Report. No. 20320008, 2013.

[KEM13b] AEMO DNV KEMA. Development of Demand Response Mechanism Baseline Consumption Methodology – Phase 2 Results Final Report. No. 20320008, 2013.

[KH18] Ping-Huan Kuo and Chiou-Jye Huang. A high precision artificial neural networks model for short-term energy load forecasting. *Energies*, 11(1), January 2018.

[KL18] Jimyung Kang and Soonwoo Lee. Data-Driven Prediction of Load Curtailment in Incentive-Based Demand Response System. *Energies*, 11(11), October 2018.

[KPSS92] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54(1-3):159–178, October–December 1992.

[LBD+89] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, December 1989.

[LHZS16] Chenghao Liu, Steven C. H. Hoi, Peilin Zhao, and Jianling Sun. Online ARIMA Algorithms for Time Series Prediction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1867–1873, 2016.

[LSSL19] Christian Lang, Florian Steinborn, Oliver Steffens, and Elmar Wolfgang Lang. Electricity Load Forecasting - An Evaluation of Simple 1D-CNN Network Structures. *arXiv*, abs/1911.11536, 2019.

[Net] Netron: Visualizer for neural network, deep learning and machine learning models. `https://www.lutzroeder.com/ai`. [Online; accessed 20-April-2020].

[Ola15] Christopher Olah. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Online; accessed 15-May-2020].

[Rao18] Suhasini Subba Rao. A course in Time Series Analysis. https://www.stat.tamu.edu/~suhasini/teaching673/time_series.pdf, Texas A&M University, December 2018. [Online; accessed 15-March-2020].

[RGH+20] João Paulo Pereira Rosa, Daniel J. D. Guerra, Nuno Horta, Ricardo Martins, and Nuno M T Lourenço. *Using Artificial Neural Networks for Analog Integrated Circuit Design Automation*. Springer Briefs in Applied Sciences and Technology. Springer, 2020.

[SS17] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R examples*. Springer Texts in Statistics. Springer, fourth edition, 2017.

[TMZZ18] Chujie Tian, Jian Ma, Chunhong Zhang, and Panpan Zhan. A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. *Energies*, 11(12), December 2018.

[vB18] Stef van Buuren. *Flexible Imputation of Missing Data*. Interdisciplinary Statistics. Chapman and Hall/CRC, second edition, July 2018.

[vBGO11] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3), 2011.

[WUO+14] Matt Woolf, Tatiana Ustinova, Enrique Ortega, Hariet O'Brien, Predrag Djapic, and Goran Strbac. Distributed generation & demand side response services for smart distribution networks, Report A7 for the "Low Carbon Learning Lab" LCNF project, September 2014.

[YJ00] In-Kwon Yeo and Richard A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, December 2000.

[Zha18] Mingda Zhang. Time Series: Autoregressive models AR, MA, ARMA, ARIMA. http://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class16.pdf, University of Pittsburgh, October 2018. [Online; accessed 14-March-2020].

[ZLLS20] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. https://d2l.ai [Online; accessed 10-May-2020].

Appendices

I. Visualizations of the performance estimation methods for the deep learning models

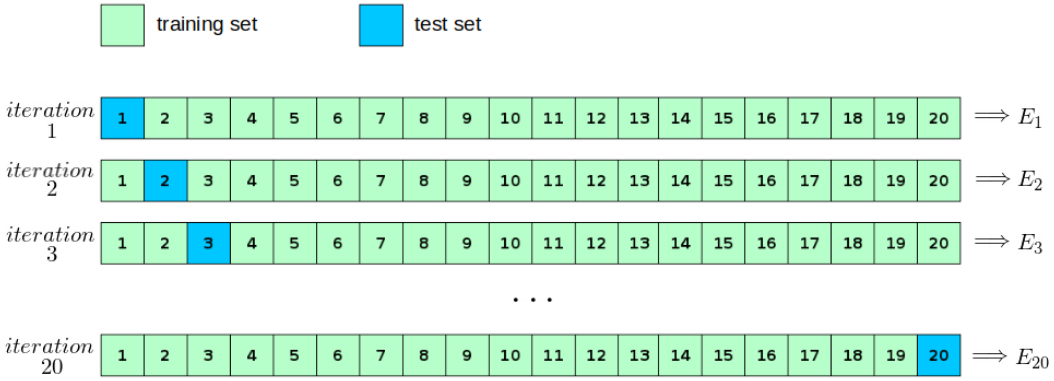


Figure 1: Blocked 20-fold cross-validation

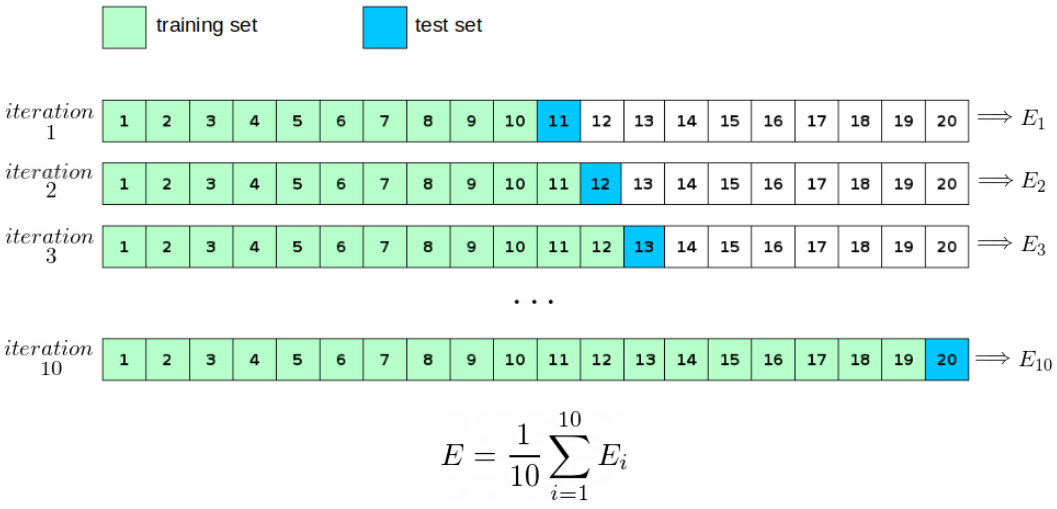


Figure 2: Prequential block method (20 folds, 10 folds as the initial training set)

II. An overview of the packages used in the implementation of this work

- `numpy` version `1.18.1` used for manipulation of data arrays in `Python`
- `pandas` version `1.0.1` used for manipulation of tables and time series data in `Python`
- `pandasql` version `0.7.3` used for manipulation of tables using SQL commands in `Python`
- `pickle` version `4.0` used for manipulation of `Python` objects
- `dill` version `0.3.1.1` used for manipulation `Python` objects
- `matplotlib` version `3.1.3` used for creating visualizations in `Python`
- `seaborn` version `0.10.0` used for creating visualizations in `Python`
- `sklearn` version `0.22.1` used for facilitating machine learning routines in `Python`
- `tensorflow` version `2.1.0` used for implementation and training of deep learning models in `Python`
- `keras` version `2.3.1` used for implementation and training of deep learning models in `Python`
- `hyperopt` version `0.2.3` used for hyperparameter optimization in `Python`
- `hyperas` version `0.4.1` used for hyperparameter optimization in `Python`
- `missingno` version `0.4.2` used for working with missing values in `Python`
- `rpy2` version `3.2.6` used for incorporating `R` libraries and commands into `Python` code
- `data.table` version `1.12.8` used for manipulation of tables in `R`
- `dplyr` version `0.8.5` used for manipulation of data in `R`
- `mice` version `3.8.0` used for data imputation in `R`
- `TSA` version `1.2` used for manipulation of time series data in `R`
- `forecast` version `8.12` used for ARIMA implementation in `R`

Non-exclusive licence to reproduce thesis and make thesis public

I, Oleksandr Kurylenko

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work from **08/12/2020** until the expiry of the term of copyright,

"Development of CNN-Based Models for Short-Term Load Forecasting in Energy Systems"

supervised by Alan Henry Tkaczyk, Florentin Dam, Ulf Roar Aakenes

2. I am aware of the fact that the author retains the rights specified in p. 1.

3. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Oleksandr Kurylenko
12.05.2020