Master's Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Benchmarking Transformer and xLSTM on Time-series Forecasting for Heat Data

Marja Wahl
Born 08.10.1999 in Jena
Matriculation number 2357517

Prof. Dr.-Ing. Marco Pruckner
Daniel Bayer M. Sc.
Sven Rausch M. Sc.
Supervisors

Submission
21.07.2025

Acknowledgement

I would like to express my sincere gratitude to my first supervisor, **Prof. Dr.-Ing. Marco Pruckner**, for his unwavering support throughout this research journey. His expert knowledge, timely guidance, and thought-provoking questions have significantly enhanced the quality of this work. His ability to always find time to address my queries, despite his busy schedule, has been invaluable to the completion of this thesis.

I am equally grateful to my second supervisor, **Daniel Bayer**, for his comprehensive assistance with writing, programming, and for consistently providing helpful ideas. His all-around tips and tricks have been instrumental in shaping this research.

Special thanks go to **Sven Rausch**, whose supervision and initiative made this master thesis possible. By bringing together the right people and guiding the progress, he has played a pivotal role in the success of this project. I am particularly indebted to him and **RAUSCH Technology** for providing access to the essential data that formed the backbone of this research on benchmarking Transformer and xLSTM architectures for time-series forecasting.

I would also like to extend my appreciation to my colleague at RAUSCH Technology, **Timo Krey**, who generously shared his expertise and patiently answered my questions, often taking time out of his busy day to assist me.

This thesis would not have been possible without the collective support, knowledge, and encouragement from these individuals. Their contributions have not only enhanced the quality of this research but have also enriched my academic journey.

Deutsche Zusammenfassung

Heizungssysteme tragen zu etwa 50% des globalen Energieverbrauchs und zum Klimawandel bei, was die Entwicklung softwaregestützter Lösungen vorantreibt, die maschinelle Lernverfahren zur Prognose von Wärmelasten und zur Reduzierung des Energieverbrauchs einsetzen. Neuerdings haben Transformer-Modelle in der Zeitreihenprognose in verschiedenen Bereichen an Bedeutung gewonnen, während erst kürzlich das xLSTM-Modell als Weiterentwicklung herkömmlicher LSTMs vorgestellt wurde.

Diese Masterarbeit führt einen Benchmark von Transformer-Modellen und xLSTM für Zeitreihenprognose von Wärmelastdaten durch. Dieser ist aufgebaut wie folgt: Erstens wird die Vorhersagegenauigkeit von Transformer-Architekturen und dem xLSTM evaluiert und mit konventionellen Methoden des maschinellen Lernens verglichen. Zweitens wird die Fähigkeit der Modelle aus heterogenen Zeitreihendaten zu lernen und externes Wissen einzubeziehen untersucht. Drittens wird die Robustheit der Modelle gegenüber der Reduktion von Trainingsdaten und Rauschen in zukünftigen Wetterdaten bewertet. Schließlich wird eine theoretische und empirische Zeit- und Speicherkomplexitätsanalyse durchgeführt.

Mehrere Modelle werden anhand von täglichen und stündlichen Wärmelastdaten mit verschiedenen Prognosehorizonten verglichen. Das xLSTM erzielt den niedrigsten Fehler in drei von vier Testszenarien und lässt sich effizienter trainieren als Transformer-Modelle, mit einer Dauer von mehreren Stunden auf der CPU, un 20 Minuten auf der GPU. Ein leichtgewichtiges neuronales Netz übertrifft alle anderen Modelle bei Sieben-Tage-Prognosen, obwohl es nur eine Minute auf der CPU trainiert.

Zukünftige Forschungen könnten hybride Modelle untersuchen, die xLSTM mit Ansätzen zur Feature-Integration kombinieren, um seine Leistung für Zeitreihenvorhersage weiter zu verbessern.

Abstract

Heating systems contribute to approximately 50% of global energy consumption and climate change, driving the development of software-driven solutions that use machine learning techniques to forecast heat load and reduce energy consumption. Recently, Transformer models have gained traction in time-series forecasting across multiple domains, while the xLSTM model has emerged as an enhancement of traditional LSTM networks.

This thesis benchmarks Transformer models and xLSTM for time-series forecasting of heat load data. The benchmark consists of following parts: Firstly, it evaluates the prediction accuracy of Transformer architectures and the xLSTM, comparing them to conventional machine learning methods. Secondly, it examines the models' capacities to learn from heterogeneous time-series data and incorporate external knowledge. Thirdly, it assesses the models' robustness against training data removal and noisy future weather data. Finally, it performs a theoretical and empirical time and memory complexity analysis.

The evaluation compares multiple models on daily and hourly heat load data with various forecasting horizons. The xLSTM achieves the lowest error in three out of four test scenarios and trains more efficiently than Transformer models, taking multiple hours on a CPU, and 20 minutes on a GPU. A lightweight fully-connected network outperforms all other models for seven day predictions, despite requiring only one minute of CPU training time.

Future research could explore hybrid architectures combining xLSTM with feature incorporation approaches to further enhance its performance for time-series forecasting.

Contents

1 Introduction **1**

2 Fundamentals and Related Work **5**

2.1 Background and Fundamentals 5

2.1.1 Environmental Impact of Heating and the Role of Forecasts . . 5

2.1.2 Time-series Forecasting 6

2.1.3 Methods for Time-series Forecasting 8

2.1.4 Advanced Models for Time-series Forecasting 13

2.1.5 Metrics for Time-series Forecasting 16

2.2 Related Work 17

2.2.1 Time-series Forecasting with Energy Data 17

2.2.2 Transformers for Time-series Forecasting 21

2.2.3 xLSTM for Time-series Forecasting 26

2.3 Open Research Questions 27

3 Methodology **28**

3.1 Overview 28

3.2 Model Implementations 29

3.2.1 FCN, RNN, and LSTM 29

3.2.2 Transformer Encoder 30

3.2.3 Temporal Fusion Transformer 31

3.2.4 xLSTM 33

4 Dataset **35**

4.1 Overview 35

4.2 Data Preprocessing 38

4.3 Features 42

4.4 Data Analysis 45

4.4.1 Correlation Analysis 45

4.4.2 Cluster Analysis . . . 47
4.5 Discussion . . . 48

5 Evaluation **50**
5.1 Training Data Generation . . . 50
5.2 Experiments . . . 53
5.2.1 Daily Heat Data . . . 54
5.2.2 Hourly Heat Data . . . 59
5.2.3 Influence of Amount of Training Data . . . 62
5.2.4 Influence of Accuracy of Future Weather Data . . . 68
5.3 Time and Memory Complexity Comparison . . . 70
5.3.1 Theoretical Complexity . . . 70
5.3.2 Empirical Performance Analysis . . . 72
5.4 Discussion . . . 75

6 Conclusion and Outlook **77**

A Appendix **80**
A.1 Dataset . . . 80
A.2 Experiments . . . 88

Acronyms **91**

Figures **94**

Tables **96**

Bibliography **98**

1 Introduction

Climate change is an undeniable problem we are facing today. The energy sector accounts for about three quarters of all harmful greenhouse gas emissions [1]. Especially the heating compartment is responsible for 50% of the world's total energy consumption [2]. Green energies are getting more popular to combat rising environmental pollution. On the software end, data-driven approaches introduce strategies on how to use available thermal energy, electrical energy, and natural gas more efficiently.

One approach is to forecast consumption of heating energy, which is valuable for reliable and precise planning of load distribution. In the Intra- and Interday market, heat energy is distributed by grid operators. If too much energy is bought and supplied, it evaporates and generates unnecessary costs for operators and environment. Also, not being able to provide enough energy might force providers to perform expensive short-term trades with external distributors. An accurate forecast enables heating systems to operate with minimal excess capacity, while still meeting demand requirements. Research has demonstrated that Artificial Intelligence (AI)-driven optimization of heating systems can achieve remarkable energy savings. Studies show average reductions of approximately 24% in energy consumption across buildings, corresponding to significant CO_2 emission reductions [3]. Furthermore, precise heat load forecasting enables more efficient integration of renewable energy sources into heating systems, supporting the transition away from fossil fuels. [4]. Therefore, the goal is to predict energy consumption as precisely as possible. Advanced forecasting models using machine learning (ML) approaches have demonstrated exceptional accuracy, with some models achieving correlation coefficients as high as 0.996 [5]. This optimization plays a crucial role in long-term sustainability goals and carbon neutrality strategies [6], as more efficient heating systems directly translate to reduced greenhouse gas emissions.

In more detail, the task of time-series forecasting on energy data is a method of estimating the future energy consumption. Traditional demand forecasting practices are statistical methods like Autoregressive Integrated Moving Average (ARIMA) or

Synthesized Load Profiles (SLPs). Statistical methods often fail to capture the complex relationships found in multivariate datasets, while SLPs generalize buildings, without regarding their individual differences ([7], [8]).

With increasing data availability and computing power, ML has become more popular for the task of time-series forecasting [9]. Long short-term memory (LSTM) [10] models are a prominent example of deep learning models used for sequence modeling [11]. A downside of the LSTM is that it suffers from vanishing gradients for long input sequences, creating a difficulty for time-series training [11]. In addition, its architecture does not allow parallel computation, resulting in long computing times for large time-series data input [11].

Models developed for Natural Language Processing (NLP) tasks have seen an uprise in the use for time-series forecasting. The sequential structure shared between text and time-series data, and the rise of Transformers [12], sparked interest in adapting the Transformer architecture for time-series modeling. Transformer models are attractive, as they show the potential to increase the prediction capacity [13]. They perform tasks based on the attention mechanism, which can be performed in parallel [12]. Running in parallel allows Transformer models to outpace LSTMs at scale for training on large amounts of data [14]. An additional benefit of the attention mechanism is that it is able to focus on important dates in the time-series data, which helps to increase prediction accuracy.

Other approaches to handle increasing data amounts were proposed. To leverage the techniques of Large Language Models (LLM), and to mitigate known downsides of LSTMs, Hochreiter et al. have introduced the extended Long short-term memory (xLSTM) [14]. The xLSTM tackles problems of the LSTM, like inability to revise storage decisions, limited storage capabilities, and lack of parallelization [14]. Also, xLSTMs are scaled to match the size of current LLMs [14]. xLSTM achieves good results on all tasks of the Long Range Arena [15] benchmark, indicating its capability to capture long-term dependencies.

This work aims at benchmarking the forecasting ability of Transformer models and the newly introduced xLSTM model on heat data. The results provide insights on the efficiency of attention-based and Artificial Neural Network (ANN) models on the task of heat load forecasting. Additionally, the time and memory complexity for each time-series forecasting method will be discussed.

Time-series forecasting encompasses multiple aspects. The data used for training can be univariate or multivariate, where the latter incorporates additional feature

information, like temperature, size, or location of the building. Further, the task can be split into short and long-term forecasting. Short-term predictions consider the next 24-48 hours, and long-term predictions are made for up to 365 days [16]. Another distinction is the method of predicting multiple forecasts. One option is to build a model that predicts a single forecast, and uses this to recursively create multiple forecasts. On the other hand, models can one-shot the prediction, by formatting the data into an appropriate format.

Other works have tested forecasting methods on heat data, but they always restrict themselves to data of one building ([17], [18]) or data from a major power plant ([19], [20], [21]). The data used for the benchmark is comprised of heat consumption recordings gathered from 149 buildings in Germany, with 131 out of them being residential structures. The diversity of the dataset presents challenges, like varying feature availability across buildings. A novel methodological problem addressed in this work is the integration of static building characteristics, such as heated area, building type, and consumed energy type, into the model training procedure. The dataset consists of monitorings from 96 gas meters and 53 heat energy meters, combined from three different research projects. Also, we merge recordings from different time periods from the years of 2017 to 2025. A major task of this work was to prepare the data from different data sources for unified and comparable model training. Another difference to most works is that we investigate using different granularities of the data, like hourly or daily. The size of the raw daily dataset amounts to 110,923 data points, while the raw hourly dataset has a size of 870,230 data points. The benchmark will evaluate how well the newly introduced xLSTM performs for time-series forecasting of heat load data, compared to attention-based Transformer architectures. Given this diverse dataset, models need to generalize well to achieve a high forecasting accuracy. We also investigate runtime and memory complexity, resulting in a complete evaluation of the ML models.

The following Sections are structured as follows. First, this work gives a background on environmental impact of heating and defines time-series forecasting in Chapter 2. Sec. 2.1.3 and Sec. 2.1.4 give an explanation of fundamental and advanced concepts regarding ML techniques for time-series prediction, like the ANN, LSTM, Transformer architecture, and xLSTM. Sec. 2.2 provides an overview of related works. In Chapter 3, the specific architectures used for benchmarking are explained. A detailed overview of the used dataset is given in Chapter 4. Then, Chapter 5 outlines the experiments, presents their results, and provides an evaluation of time and memory complexity of

used methods. After a discussion of the results, a final conclusion is drawn in Chapter 6.

2 Fundamentals and Related Work

2.1 Background and Fundamentals

This section covers background information about the topic of time-series forecasting of energy data. It explains the importance of heat consumption prediction for the environment and other stakeholders, as well as methods used for the task of time-series forecasting.

2.1.1 Environmental Impact of Heating and the Role of Forecasts

The building sector is a massive contributor to pollution due to energy consumption. The type of energy that is consumed plays an important role, too. Renewable energies and district heating are therefore considered more environmental friendly than fossil fuels, like gas. In general, buildings alone account for 30% of average global energy consumption and a third of the associated CO_2 emissions in 2017 [22]. At the same time, urbanization is rapidly growing [23]. The current global building energy consumption remains too high, hindering our progress toward meeting critical climate goals like the 2°C scenario outlined in the Paris Climate Agreement [22]. Therefore, every sector needs to find solutions for reducing pollution. In regards of the housing sector, this means minimizing harmful building energy consumption and increasing the use of renewable energies.

Heating is a big part of building energy consumption. Smart sustainable infrastructures are a way of modeling buildings and their heating behavior, with the goal of minimizing cost and environmental load ([24], [25]). Forecasting plays a big role in modeling buildings, as it helps to estimate future states. The more precise the energy consumption forecast, the more thorough the energy management can be planned. Creating smart buildings by monitoring consumption and applying forecast mechanisms does not only result in cost and energy savings, but also makes it easier to incorporate renewable energies in the long run ([26], [27]).

A forerunner of using ML for reducing heat load is RAUSCH Technology[1] (former Ener-IQ), a company with the vision to create a sustainable future for humanity's benefit. Their goal is to decarbonize the energy sector by using ML-tools, like time-series forecasting. They have successfully implemented AI-based optimization of heating systems in their KINERGY-project [3], which was able to reduce heating energy consumption by 24%. The AI-driven system proposed uses forecasting to simulate changes of control parameters and validate those. Also, energy consumption forecasts are used to determine the heat limit temperature, which is the outside temperature below which heating is needed. In another project they implement similar methods for reducing energy consumption in district heating systems [28].

RAUSCH Technology creates two success stories of reducing energy consumption with ML methods. This work uses the data collected from those projects to evaluate more complex forecasting models, the Transformer by Vaswani et al. [12], the Temporal Fusion Transformer (TFT) by Lim et al. [31], and the xLSTM by Beck et al. [14], on time-series forecasting of heat load data.

2.1.2 Time-series Forecasting

Time-series forecasting finds application in many fields, for example climate modeling, biological sciences, medicine, commercial decision making, and finance [9]. It can also be used for heat load prediction, to estimate the energy consumption due to heating for the next hours or days.

A time-series X of length N can be defined as a mapping between time-steps t and scalar values [29]:

$$X = \{X_1, X_2, X_3, \ldots, X_N\}, X_t \in \mathbb{R} \tag{2.1}$$

In general, time-series forecasting is a supervised regression task, where a model is computed, which can predict such a time-series given some input. The regression task consists of finding the function f mapping the input of length n to k output values, with $n, k \in \mathbb{N}$ and $n, k \geq 1$ [30]:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^k \tag{2.2}$$

[1] https://www.rausch.se/

Specifically, time-series forecasting is the task of predicting k future values of target $y_{t:t+k} = (y_t, y_{t+1}, \ldots, y_{t+k})$ for time step t, given a sequence of n past values $y_{t-n:t-1} = (y_{t-n}, \ldots, y_{t-2}, y_{t-1})$. The one-step prediction with future steps $k = 1$ can be defined as:

$$y_t = f(y_{t-n:t-1}, X_{i,t-n:t}, s_i), \tag{2.3}$$

where $X_{i,t-n:t}$ are exogenous variable time-series of entity type i for look-back window n, and s_i defines static metadata. The function f is the model that performs the predictions. Eq. 2.3 is based on the definition for multi-horizon forecasting presented by Lim et al. [31].

Generally, precise point estimates, like shown by Eq. 2.3, are considered forecasting. This means the expected value of the future target is computed. Other possible methods are probabilistic outputs, where the model predicts parameters of a distribution, for example mean and variance of a Gaussian distribution [9]. Another method is quantile regression, where the model outputs prediction intervals via quantiles [31].

Multi-horizon forecasts are predictions with $k > 1$ [9]. The input for a model performing multi-horizon forecasts can be enriched by future covariate series, expanding Eq. 2.3:

$$y_{t:t+k} = f(y_{t-n:t-1}, X_{i,t-n:t+l}, s_i), \tag{2.4}$$

with prediction horizon length k and future covariate length l.

Multi-horizon forecasts can be performed by iteratively using single-step forecasts as input for the model and generating k forecasts, or by increasing output size of the model to k. The iterative approach's advantage is that the underlying model can be simpler, but error can propagate for each new forecast made. The one-shot approach does not have that problem, but might require a more complex model.

The load forecasting problem can be categorized according to horizon length k. Hong et al. [32] argue that the definitions used for short and long-term forecasting are often ambiguous. Zhao et al. propose a definition which divides the domain into very short-term load forecasting, short-term load forecasting, medium-term load forecasting, and long-term load forecasting [16]. For a prediction period ranging from a few minutes to hours, the task is considered very-short term load forecasting. This might be implemented for aiding with decisions for real-time operations, like on the gas and district heat market [16]. It is considered a short-term prediction, when the forecast length ranges from one day to one week ahead. Those forecasts can be

helpful for analyzing energy schedules for improving efficiency [16]. Medium-term forecasting starts at a forecast length of one month to a year, with the goal of making an effective operational plan [33]. Long-term forecasting is considered when more than a year is predicted. Due to the large forecast horizon, it has special difficulties like computation time plunge and keeping substantial prediction accuracy [13]. Longer input and output sequences result in longer range dependencies [13]. Most heat load prediction research focuses on very-short to short-term forecasting ([16], [18], [34]). When performing those types of forecasting, the granularity of the data is usually higher, meaning the data is monitored every 15 minutes or every hour. Medium- and long-term forecasting often rely on aggregated data, which might be daily or weekly.

Traditionally, statistical methods are used for time-series forecasting, like autoregressive (AR) models ([35], [36]), exponential smoothing [37], or structural time-series models [38]. More recently, data-driven approaches are implemented, as they are found to be more flexible and reliable for modeling forecasting [25]. Another reason is the increase in data availability. Buildings are deploying more and more sensors to ensure integration in smart environments, which makes datasets for training models more accessible ([28], [39], [40]).

One more advantage of data-driven models is the possibility to integrate knowledge from external factors, like weather and building characteristics. Covariate time-series, like weather data, can be used as additional input to data-driven models. Some models allow using future covariate time-series, which provide information about the timesteps to be predicted. This could, for example, be a weather forecast. Static covariates are features that do not change over time, like building information. They might be handled differently than other covariates, for example by the TFT [31].

2.1.3 Methods for Time-series Forecasting

Multiple methods have been proposed for time-series forecasting. They can be divided into statistical and data-driven approaches. AR models are widely implemented for the use of time-series forecasting, as they provide a balanced implementation simplicity and forecasting accuracy [25]. Prominent examples of AR models are ARIMA, seasonal ARIMA (SARIMA), and ARIMA with inclusion of eXogenous variables ([35], [36]). While AR models are quick to fit, they are not able to model multi-linear patterns effectively [25]. Therefore, the following section provides a short overview of more recent and data-driven prediction techniques.

2.1.3.1 Artificial Neural Network

The base idea for ANNs was presented by McCulloch and Pitts [41]. They first proposed a mathematical model representing biological neurons as a network of simple binary units. The idea was then further developed by different researchers.

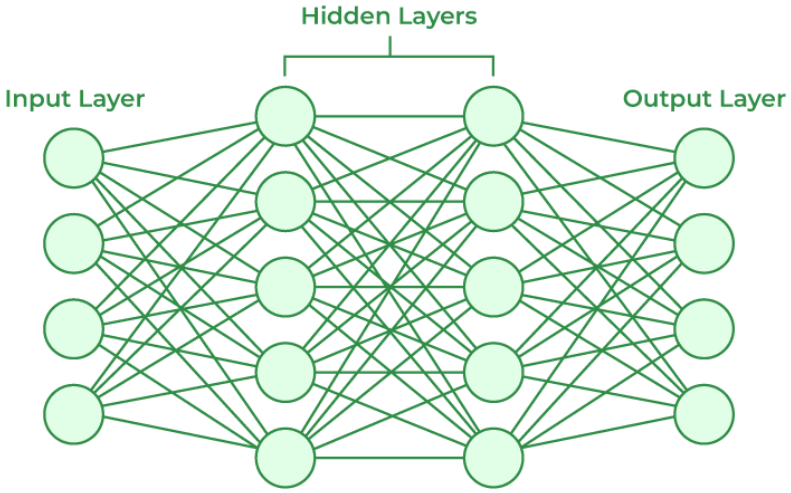

Figure 2.1: An example ANN [42]

The ANN is a network of units, each unit belonging to a layer. The first layer is considered the input layer, the last the output layer. Intermediate layers are called hidden layers. A Fully-connected Network (FCN) has connections between every pair of units of two consecutive layers. The number of layers and units per layer are hyperparameters. An example ANN is depicted in Fig. 2.1.

The ANN learns using supervised learning. That means, it sees examples of data and the corresponding target value during training. First, the data examples are passed through the neural net during the forward pass. During this step, the input data is passed along on multiple paths through the network. Each unit multiplies a weight w_i with incoming data, adds a bias b, and applies an activation function f. Prominent activation functions are sigmoid, tanh, and Rectified Linear Unit (RelU). Eq. 2.5 formulates the output of a single node in the network as iterations over every input value x_i. Mathematically, the forward pass is realized by matrix multiplication.

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.5}$$

After the forward pass, the results are compared to the desired target values by computing a loss L. For this, different loss functions can be chosen, depending on the task of the neural network. The loss is then backpropagated through the network. First the partial derivative of the loss L for each weight and bias in the network is computed. Then, a process called gradient descent uses the gradients calculated during backpropagation and updates weights and biases by moving them in the opposite direction of the gradient. The degree of change is decided by a parameter called learning rate. [30]

A more sophisticated strategy is implemented by the Adaptive Moment Estimation (Adam) optimizer [43], which combines momentum and Root Mean Square Propagation for updating weights and biases. It estimates mean and variance of past gradients and uses that information to update model parameters more smoothly and with less oscillations.

2.1.3.2 Recurrent Neural Network

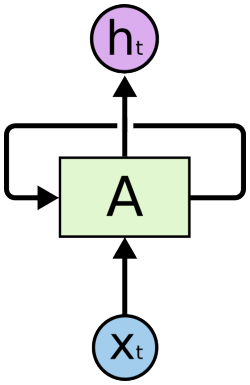

Figure 2.2: Recurrent Neural Network cell with feedback loop, input x_t and hidden state h_t [44]

Recurrent Neural Networks (RNNs) revolutionized sequential data processing by introducing a simple yet powerful concept: memory. Unlike feedforward networks, RNNs maintain an internal state that captures information about previous inputs. This state is updated as new data arrives, allowing the network to build a cumulative understanding of the sequence.

The key innovation of RNNs lies in their recurrent connections. While traditional networks flow strictly forward, RNNs include connections that loop back, allowing information to persist from one step to the next. The feedback loop of one cell in the RNN is visualized in Fig. 2.2.

Training RNNs involves a specialized form of backpropagation called Backpropagation Through Time. Since RNNs process sequences, the error must be propagated not just through layers, as in traditional networks, but also backward through time steps. This involves unrolling the network through time and treating each time step as if it were a layer in a very deep network.

While powerful, this approach introduces challenges like the vanishing and exploding gradient problems, where gradients either become too small or too large as they are propagated back through many time steps.

2.1.3.3 Long Short-Term Memory Network

The LSTM was proposed by Hochreiter et al. [10] as a solution to the vanishing gradient problem of RNNs. Later it was refined by many more researchers to achieve the architecture commonly used now.

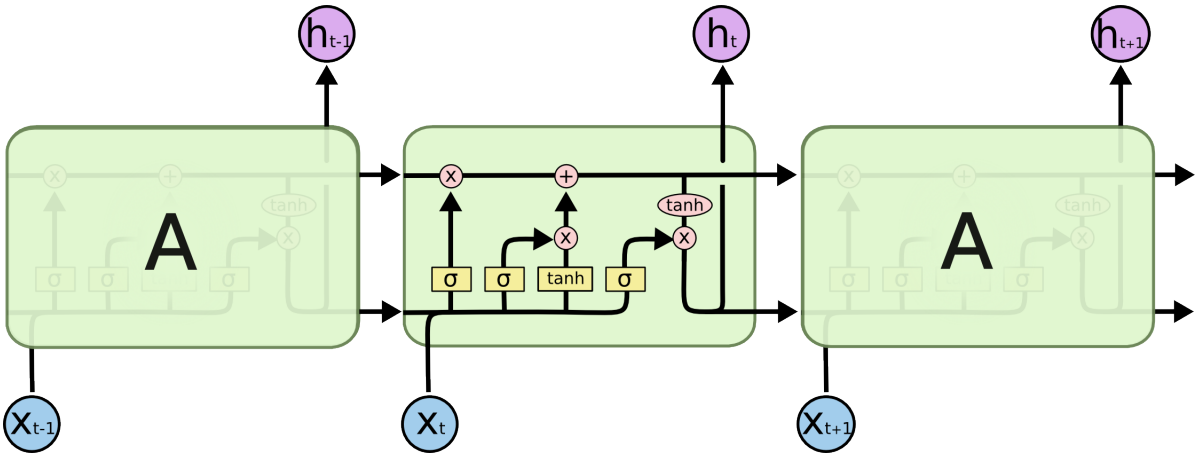

Figure 2.3: The LSTM and LSTM cell architecture [44]

The LSTM consists of a memory cell, designed to keep from forgetting information. The cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates determine how information is kept. The LSTM architecture is shown in Fig. 2.3. The forget gate was added later by Gers et al. [45].

Furthermore, the states and gates of the LSTM cell can be formulated as follows [14]:

$$c_t = \mathbf{f}_t \odot c_{t-1} + \mathbf{i}_t \odot z_t \qquad \text{cell state} \qquad (2.6)$$

$$h_t = \mathbf{o}_t \odot \tilde{h}_t, \quad \tilde{h}_t = \psi(c_t) \qquad \text{hidden state} \tag{2.7}$$

$$z_t = \varphi(\tilde{z}_t), \quad \tilde{z}_t = \mathbf{w}_z^\top x_t + r_z h_{t-1} + b_z \qquad \text{cell input} \tag{2.8}$$

$$\mathbf{i}_t = \sigma(\tilde{\mathbf{i}}_t), \quad \tilde{\mathbf{i}}_t = \mathbf{w}_i^\top x_t + r_i h_{t-1} + b_i \qquad \text{input gate} \tag{2.9}$$

$$\mathbf{f}_t = \sigma(\tilde{\mathbf{f}}_t), \quad \tilde{\mathbf{f}}_t = \mathbf{w}_f^\top x_t + r_f h_{t-1} + b_f \qquad \text{forget gate} \tag{2.10}$$

$$\mathbf{o}_t = \sigma(\tilde{\mathbf{o}}_t), \quad \tilde{\mathbf{o}}_t = \mathbf{w}_o^\top x_t + r_o h_{t-1} + b_o \qquad \text{output gate} \tag{2.11}$$

Cell input and the gates are defined by their individual weight vectors w, recurrent weights r, hidden states h, and bias terms b. ψ and ϕ denote activation functions, usually set to `tanh`, as depicted in Fig. 2.3.

The cell state passes through the three gates of the LSTM cell, before being forwarded to the next cell. This process is resembled by the top arrow in Fig. 2.3. The first gate that is applied is the forget gate, deciding how much information to retain, based on Eq. 2.10. Its inputs are the hidden state of the previous cell h_{t-1} and the input for the current cell x_t. By passing through a sigmoid layer, their information determines how much knowledge from the previous cell state c_{t-1} is kept. Then, a proportion of new information is added to c_t by passing the cell input z_t (Eq. 2.8) through the input gate (Eq. 2.9), and adding its result to c_t. The updated hidden state h_t (Eq. 2.7) is the result of the activated cell state c_t passed through the output gate (Eq. 2.11).

Another version of the LSTM is the Gated Recurrent Unit (GRU) [46]. It combines forget and input gates into a single update gate. Also, it merges cell state and hidden state, which makes it a simpler version of the LSTM.

Despite its success for time-series forecasting, the LSTM also has known downsides when handling long sequences of data. For example, it has difficulties updating previously stored information, when more relevant data is encountered. Also, constantly updating the states of the LSTM can cause it to forget important information. This is also a side-effect of having limited memory capacity, because of storing the history information as a scalar value. Another problem is that the training of the LSTM, like the one of the RNN, can not be parallelized due to its architecture.

2.1.4 Advanced Models for Time-series Forecasting

Transformers and their attention mechanism are the base structure for a great portion of LLMs [47]. Therefore, the current rise in LLMs can also be tracked back to the success of the Transformer architecture. Bidirectional Encoder Representations from Transformers (BERT) [48] and Generative Pre-trained Transformer (GPT) [49] are both prominent Transformer-based models. BERT uses the Transformer Encoder and trained it on the masking task, which makes it suitable for Google's search query completion. GPT successfully uses a Decoder for text generation.

More recently, the extended LSTM (xLSTM) was proposed by Beck et al. [14]. This model is said to overcome problems of the original LSTM and achieve competitive results compared to Transformer-based models on language tasks.

The following sections discuss how these advanced architectures can be valuable for time-series forecasting.

2.1.4.1 Transformer and Time-series Forecasting

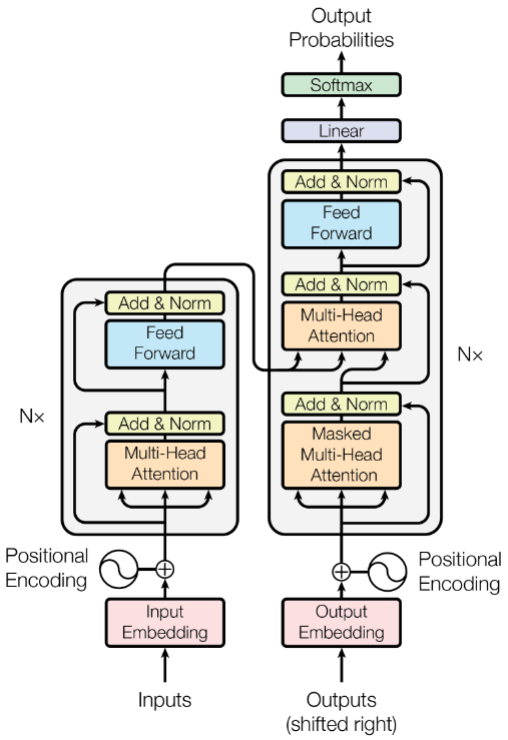

Figure 2.4: The Transformer architecture [12]

The Transformer by Vaswani et. al is a state-of-the-art model completely relying on attention mechanism [12]. Its architecture is based on the common encoder-decoder structure, depicted in Fig. 2.4. The encoder, as well as the decoder, consist of N identical layers, these again are composed of multi-head attention and feed-forward layers. [12]

Multi-head attention means that the input queries Q, keys K, and values V are projected on h attention layers running in parallel. The h attention heads use different learned linear projections of the input, maximizing the information that can be attended. Each head h performs scaled dot-product attention on the learned input representations of Q, K, and V. This means that Q and K are used to compute weights, which are then applied on V, in order to retrieve the most salient part of the sequence. [12]

Originally designed for working with text data, Transformers have recently been used for tasks in time-series modeling [50]. The Transformer is built to predict a value given a sequence of tokens, which is also the basic idea of time-series forecasting.

A major advantage of the Transformer structure is that it can be run in parallel, because individual attention heads share no dependencies. This stands in contrast to RNN and LSTM modules, which inhibit a recurrent structure. Therefore, Transformers are also appealing to time series modeling [50].

Another theoretical advantage over a recurrent structure is the attention mechanism, and how it is able to focus on the important tokens in a sequence [50]. This makes it possible to capture even long-range dependencies in sequential data.

One difference to language modeling tasks is the embedding of tokens. For time-series forecasting, the tokens already represent scalar values, therefore no embedding mechanism is needed. Yet, the order of tokens needs to be given to the model by using positional encodings. The Transformer, in contrast to the LSTM, has no knowledge of input token order otherwise. The vanilla Transformer uses static embeddings, added to embeddings of the input tokens. Also, learnable embeddings can be used [50].

2.1.4.2 xLSTM and Time-series Forecasting

Recently, an advanced LSTM architecture called extended LSTM (xLSTM) has been presented by the team of Hochreiter et al., the author of the original LSTM paper [14]. The xLSTM mitigates known disadvantages of the LSTM, like their inability to revise storage decisions, their limited storage capacities, and their lack of parallelization [14].

To solve those problems, the xLSTM introduces two modifications to the LSTM: exponential gating and novel memory structures. To do so, two new LSTM structures are created, scalar LSTM (sLSTM) and matrix LSTM (mLSTM).

To tackle the problem that LSTMs are not able to easily reset their weights, the sLSTM adds exponential activation functions to input and forget gate. They also use normalization and stabilization techniques, to avoid overflows. The updated gates look like this

$$i'_t = \mathbf{exp}(\log(i_t) - m_t) = \mathbf{exp}(i_t - m_t) \tag{2.12}$$

$$f'_t = \mathbf{exp}(\log(f_t) + m_{t-1} - m_t) \tag{2.13}$$

where Eq. 2.12 references the stabilized input gate and Eq. 2.13 the stabilized forget gate [14]. The stabilization function is described as follows [14]:

$$m_t = max(\log(f_t)) + m_{t-1}, \log(i_t) \tag{2.14}$$

In addition to adding exponential gates, the sLSTM introduces memory mixing. Similar to the original LSTMs multiple memory cells, the sLSTM can have multiple heads. Each head can have individual memory mixing, but it can not be applied across multiple heads [14].

The mLSTM uses a matrix memory cell instead of a scalar one, which increases storage capacities. The retrieval of the key-value pairs is implemented using a covariance update rule, inspired by Bidirectional Associative Memories [51]. The updated cell, normalizer and hidden state can be formulated as follows:

$$\boldsymbol{C}_t = \boldsymbol{f}_t \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \boldsymbol{v}_t \boldsymbol{k}_t^\top \qquad \text{cell state} \tag{2.15}$$

$$\boldsymbol{n}_t = \boldsymbol{f}_t \boldsymbol{n}_{t-1} + \boldsymbol{i}_t \boldsymbol{k}_t \qquad \text{normalizer state} \tag{2.16}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \hat{\boldsymbol{h}}_t, \quad \hat{\boldsymbol{h}}_t = \boldsymbol{C}_t \boldsymbol{q}_t \,/ \max\left\{\boldsymbol{n}_t^\top \boldsymbol{q}_t, 1\right\} \qquad \text{hidden state} \tag{2.17}$$

Without memory mixing, the mLSTM is able to be fully parallelized on modern hardware accelerators. [14]

The final xLSTM architecture is constructed by residually stacking building blocks, either mLSTM or sLSTM. The proposed model is made up from 7 blocks, 2 sLSTM blocks at position 2 and 5, with the rest being mLSTM blocks.

2.1.5 Metrics for Time-series Forecasting

To measure the prediction accuracy several metrics are commonly used. The Mean Squared Error (MSE) measures the average squares of errors between predicted values $\hat{y}_t$ and actual values y_t for timestep t, with forecast horizon of length k:

$$\text{MSE} = \frac{1}{k}\sum_{t=1}^{k}(y_t - \hat{y}_t)^2 \tag{2.18}$$

It penalizes larger errors more due to squaring, which makes it more sensitive to outliers. Also, the unit is not the same as the original data. To remedy that, the Root Mean Squared Error (RMSE) can be computed

$$\text{RMSE} = \sqrt{\frac{1}{k}\sum_{t=1}^{k}(y_t - \hat{y}_t)^2} \tag{2.19}$$

which takes the square root of the MSE. This makes the RMSE more interpretable, which is why it is widely used in forecasting literature [25].

To make the RMSE scale-invariant, the normalized RMSE (nRMSE) can be computed, which is the RMSE divided by the range of observed values:

$$\text{nRMSE} = \frac{\sqrt{\frac{1}{k}\sum_{t=1}^{k}(y_t - \hat{y}_t)^2}}{\max(y) - \min(y)} \tag{2.20}$$

where the range of observed values is the difference between maximum and minimum values.

Even more interpretable is the Mean Absolute Error (MAE), which measures the average of absolute differences between predicted and actual values:

$$\text{MAE} = \frac{1}{k}\sum_{t=1}^{k}|y_t - \hat{y}_t| \tag{2.21}$$

It is more robust to outliers than MSE/RMSE, but does not penalize larger errors as heavily either.

One of the most used metrics for time-series forecasting evaluation is the Mean Absolute Percentage Error (MAPE) [25]. It expresses accuracy as a percentage of the error:

$$\text{MAPE} = \frac{100\%}{k} \sum_{t=1}^{k} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{2.22}$$

, which makes it scale-independent and easy to interpret. On the other hand, it is problematic when the actual values are zero or close to zero, resulting in very large errors.

These metrics are often used together, to create a comprehensive overview of the model's prediction performance.

2.2 Related Work

In this section, multiple works on the topic of energy data prediction are presented. Additionally, research on Transformer models and xLSTM for time-series forecasting is highlighted.

2.2.1 Time-series Forecasting with Energy Data

Over the last decades, thousands of energy forecasting papers have been published, which demonstrates the topics importance [27]. Energy forecasting can be split according to the energy type: electricity, oil and gas forecasting, and district heating. Wind and solar power generation is a subdomain of electricity forecasting, but deals with generation rather than consumption. The dataset used in this work consists of recorded gas and district heat consumption values. Nevertheless, papers discussing all types of energy forecasting were investigated. A large proportion of reviewed papers focuses on electricity data ([16], [19], [34], [52], [53], [54], [55], [56], [57], [58]), several papers discuss district heating data ([18], [20], [21], [59], [60], [61], [62], [63], [64], [65]), one reference analyzes a heat, ventilation, air, conditioning (HVAC) system [17], and one work investigates wind and solar power [66]. All works concern themselves with the topic of short-term energy data prediction. The types of energies are to their own degree dependent on features like weather, building characteristics, and consumer behavior.

Support Vector Machine Eseye et al. experiment with a Support Vector Machine (SVM) on hourly district heat demand data, training the model to forecast 24 hours ahead [18]. The data is recorded over a time span of three years, where two years

are used for feature selection and forecast model training, and one year is used for testing. The features used in training are calendar data, holiday, weather data, and the heat demand 24 hours prior, the average of the day before, and 186 hours before. For feature selection they use a hybrid of a binary genetic algorithm and Gaussian Process Regression (GPR). As an additional preprocessing step, they implement Empirical Mode Decomposition (EMD) on the target variable. This process decomposes the heat demand time series into a set of subseries, with the goal of extracting important detail information of the raw time series. As optimization algorithm they choose the Imperialistic Competitive Algorithm (ICA) rather than backpropagation, to find global instead of local minima. Their data consists of recordings for four buildings, each building is fit a different model. Their proposed EMD-ICA-SVMs outperform EMD-ICA-ANNs, achieving a MAPE score of 4.65% and RMSE score of 14.04 (kW) on the residential buildings forecast. Their results also show that the model performs best on weekends in Spring according to the MAPE (2.91%), and worst on weekends in summer (7.78%).

Gaussian Process Regression Potočnik et al. fit a GPR for the task of short-term prediction of district heat data [21]. The data they use describes the hourly transmitted heat from the border of production facilities in Ljubljana. Therefore, the data models the behavior of the whole city, rather than those of single households. They gathered a training corpus of data of three years and perform a 48-hour multi-step heat demand forecast. They use features like linear time since beginning of recordings, day of week, binary encoded holiday, weather data, and historical load data. Additionally, they add a feature day of week with holidays encoded as sunday and time encoded with the cosine function. Potočnik et al. find that out of linear models, like the SVM, neural networks, and GPR, the latter had the best results with a Mean Absolute Normalized Error (MANE) score of 2.94%. The MANE describes the MAE on normalized data. Also, the authors investigated the influence of accurate future weather forecasts. They found that using actual future temperature, compared to using future temperature forecasts, resulted in a drop of the MANE to 2.11%. Unfortunately, this case is not realistic, but also reveals the existence of a basic forecasting error caused by an unknown district heating process, rather than by inaccurate weather forecasting [21].

MLP and Ensemble Learning Tang et al. analyze the forecasting of one hour of the energy consumption of a HVAC system in a single facility [17]. They train an

Multi-Layer Perceptron (MLP) on hourly data collected for a few days at each season of the year. For features, they use temperature, relative humidity, CO_2 concentration, solar radiation, and past load information. Feeature selection is performed using a boosting tree algorithm. Instead of training a single MLP, the authors first cluster the data using the k-means algorithm. Then, they train a model on each of the resulting clusters. The predictions of the ensemble are averaged to achieve a singular output. The ensemble method with clustering outperforms training a single model with a MAPE of 8.63%, compared to 9.83%.

Neural Networks Xue et al. perform a 24 hour forecast and use hourly data of about four months of heat load in megawatt from a Combined Heat and Power plant in China [20]. They implement exponential weighted moving average imputation for handling missing data and select features based on their partial autocorrelation score. Also, they perform hierarchical clustering on heat load patterns for dividing samples into train and test sets. Extra information that the model gets is the calendar month, hour of the day, temperature, and historical load data. In their evaluation of the three model types, Support Vector Regression (SVR), Deep Neural Network (DNN) and eXtreme Gradient Boosting (XGBoost), they also compared a direct and recursive strategy. The recursive strategy would take predicted values as input to the same model to fill the prediction horizon, while the direct strategy trains one model for each of the 24 hours. The recursive XGBoost results in the lowest MAPE score of 9.59%, with the recursive DNN achieving the second best results with a MAPE score of 10.09%.

Further works have evaluated ANNs on the task of short-term prediction for district heating demand ([65], [67]). Petrichenko et al. [65] compare an ANN, polynomial regression and the hybrid version, and find out that the hybrid method performs the best. Johansson et al. [67] perform a review on district heat load forecasting, where they compare an ensemble of decision tree regressors to an ensemble of FCNs. Their data was collected from a district heating system in Rottne, Sweden, for a period of three months. The model was trained to predict a day ahead, using past heat consumption and weather forecast data. The comparison results in the decision trees outperforming the FCNs with a MAPE of 11.7%, compared to 14.41%.

Recurrent Neural Network Kato et al. [59] compare an RNN to a three-layer FCN on the task of predicting the heat load of a District Heating and Cooling (DHC) system

in Giga-Joules (GJs). They gather hourly data from an actual DHC for the period of 37 days, using the first 30 days as training data. The features they use are past heat load data, highest open-air temperature, and lowest open-air temperature. The RNN is able to achieve a lower MSE of 11.82 GJ, compared to 21.05 GJ.

Long Short-Term Memory Muzaffar et al. [62] train and evaluate an LSTM on electricity load data. The dataset consists of hourly recordings of electricity load over a span of 13 months. The first twelve months are used as training data. The outside temperature is used as single exogenous feature. During preprocessing, the yearly seasonality found in the electricity data is removed, daily and weekly seasonality are removed as well. The LSTM is compared to baseline statistical methods Autoregressive Moving Average (ARMA), SARIMA, and ARMA with eXogenous inputs. The LSTM is trained with a look-back window of five hours. The results show that the LSTM is only able to outperform the statistical methods for a forecast horizon of 24 hours (1.522% MAPE compared to 5.42%) and 48 hours (2.16% MAPE compared to 4.48%). For seven and 30 days, the LSTM is outperformed by the ARMA model (3.7% MAPE compared to 5.97%), and the SARIMA and ARMA (5.24% MAPE compared to 9.75%), respectively.

A study by Li et al. [68] compared a pure LSTM model with five improved LSTM architectures, using a Convolutional Neural Network (CNN) (LSTM-CNN, CNN-LSTM, LSTM-Attention, CNN-Attention-LSTM, and LSTM-Attention-CNN) for building energy consumption forecasting. The modifications included basic approaches, such as adding CNN layers before or after LSTM layers, or incorporating attention layers between two LSTM layers. They also tested hybrid modifications that combined CNN layers and the attention mechanism with an LSTM.

The analysis utilized data from the Building Data Genome Project 2 [40], encompassing 60 randomly selected buildings across six different building types: office, university, library, dormitory, hospital, but none of them residential structures. The dataset contained hourly building energy consumption records spanning either one or two years. The energy data represented electricity alone or electricity combined with hot water, gas, or steam, depending on the specific building. Individual models were constructed for each building, incorporating features such as calendar data (time stamp, date, month, week) and weather parameters (outdoor air temperature, dew point temperature, sea level pressure, wind direction, wind speed).

Results revealed that basic LSTM model modifications outperformed hybrid modifications, suggesting no additional benefit from combining attention mechanisms with LSTM and CNN architectures. For buildings with only one year of available data, the CNN-LSTM configuration demonstrated superior performance, reducing the average RMSE by 2.9%. Conversely, when two years of data were available, the LSTM with attention architecture exhibited more stable predictions and performance, decreasing the average RMSE of the standard LSTM by 5.6%.

All related works train the models they propose building-wise. In this work, one model is going to be trained on data from multiple buildings. This will test the models ability to generalize well for different types of buildings.

As can be seen in the review, not a lot of works analyze data from residential buildings. The consumer behavior might have a different influence in buildings like schools, hospitals, or offices. In contrast, our dataset consists of mainly residential buildings, inherently considering consumer behavior.

Out of the papers found for the review, no work performs long-term forecasting, meaning predicting a week, month, or year of energy consumption.

2.2.2 Transformers for Time-series Forecasting

2.2.2.1 Adapting Transformers for Time-series Forecasting

When using Transformers for time-series forecasting, special characteristics of time-series data need be considered. The specific challenges are how to include the time component into the input and how to handle long series of data.

Since the Transformer has no recurrent design like the LSTM model, hand-crafted positional encodings need to be added to the input to inject a sense of order into the model. One option is to use the absolute positional encodings proposed by Vaswani et al. [12], where a fixed positional vector is added to the input time-series embeddings, before feeding them to the Transformer. This is for example done by Li et al. [53].

A more expressive option is to use learnable positional encodings. Zerveas et al. [69] add an embedding layer for learning vectors for each position index, while Lim et al. [31] use an LSTM network.

Incorporating timestamp information commonly accessible for time-series data is another challenge for the vanilla Transformer architecture. Informer [13], Autoformer [70] and FEDformer [71] overcome this problem by encoding timestamp information as an additional positional encoding using learnable embedding layers.

Transformers have the ability to model extraordinary long-range dependencies, but for long sequences like time-series data, self-attention is not efficient due to its quadratic computation time [13]. Some adapted Transformer models focus on improving efficiency of the self-attention mechanism on the module and architecture level ([13], [53], [71], [72], [73], [74], [75], [76], [77]).

The original Transformer architecture can be adjusted on the module level in two ways: designing new attention modules or utilizing bias for token inputs [50]. The LogSparse Transformer [53] and the Pyraformer [77] adjust the self-attention module by introducing a sparsity bias into the attention mechanism. Additionally, the LogSparse Transformer uses convolutional self-attention, by introducing causal convolutions, to generate queries and keys.

Another method, used by the Informer [13] and FEDformer [71], is speeding up computation by exploring the low-rank property of the self-attention matrix. Specifically, the Informer pre-selects dominant queries based on key similarities, to make the self-attention mechanism more efficient. The FEDformer transforms the input into the frequency domain by applying a Fourier and wavelet transform. Then, it randomly selects a fixed-size subset of the frequency.

The AST [78] implements a generative adversarial encoder-decoder architecture for training a sparse Transformer model. The Quatformer [79] modifies the attention-mechanism by using learnable quaternions to depict periodical patterns.

A few works create models with higher interpretability. The TFT [31] incorporates static and continuous covariates for time-series forecasting. Like the LSTM, it is built for multi-horizon forecasting, meaning it can predict multiple time-series at a time. The TFT can handle multiple input covariates, which makes it possible to incorporate global, as well as temporal dependencies, and events. It also proposes a gating feature selection and a temporal self-attention decoder. By creating the ability to analyze the trainable feature selection network, the TFT provides insights in global feature importance for making an accurate forecast.

ProTran [80] and SSDNet [81] combine the Transformer with State Space Models (SSMs) for the task of time-series forecasting. ProTran uses a generative modeling approach and inference procedure based on variational inference. SSDNet uses the Transformer to learn temporal patterns and estimate parameters of an SSM. Then, the SSM computes a seasonal-trend decomposition, which can be used to interpret the time-series further.

The Autoformer [70] and Crossformer [82] utilize the bias for token input on attention-module level. In detail, the Autoformer uses segment-based representation and employs a straightforward seasonal-trend breakdown structure featuring an auto-correlation component that functions as attention. This component evaluates time-lag similarities between input signals and combines the most similar subseries to create a less complex output. The Crossformer introduces a Transformer model that leverages relationships across dimensions for multivariate forecasting. It transforms inputs into 2D vectors using a new embedding technique that maintains both temporal and dimensional data. A dual-attention mechanism then effectively captures dependencies across both time and dimensions.

Following works integrate adjustements on the architecture level. Between attention blocks, the Informer [13] samples down the data size by inserting max-pooling layers with stride 2. By doing so, it introduces a hierarchical architecture into the Transformer, which takes into account the multi-resolution aspect of a time series [50]. The Pyraformer [77] introduces hierarchy by using a C-ary tree-based attention mechanism.

2.2.2.2 Transformers in Energy Data Forecasting

Recently, many authors have tried the Transformer model and different variations of it for the task of energy data forecasting ([7], [16], [19], [34], [57]).

Zhao et al. [16] evaluate the vanilla Transformer for next-day electricity load prediction. Their study analyzed half-hourly electrical consumption data in megawatts from New South Wales, spanning January 1, 2006, to December 31, 2010, obtained from the Australian Energy Market Operator. The training dataset encompassed the period from January 1, 2006, to July 4, 2010, with additional features including temperature, humidity, and electricity price. A novel similar day selection approach was implemented, utilizing comparable historical load patterns rather than previous day exogenous factors as input. To determine similarity between days, the researchers first assigned weights to each exogenous feature using the Light Gradient Boosting Machine [83] algorithm, followed by k-means clustering [84] based on these weighted features. The proposed Transformer model incorporated learnable position embeddings derived from calendar data and was configured for day-ahead load prediction, forecasting 48 time steps into the future. Comparative analysis against several baseline models—including GRU, LSTM, RNN, CNN+LSTM, and seq2seq+LSTM—demonstrated the Transformer models' superior performance across both MAPE and RMSE metrics.

Specifically, the Transformer achieved an average MAPE value of 1.13%, outperforming the second-best model (seq2seq+LSTM), which recorded an average MAPE of 1.29%.

Some works use an adapted version of the Transformer, to better fit the characteristics of time-series data.

The SL-Transformer is proposed by Zhu et al. [7], for performing time-series power forecasting on wind and solar energy. This task is different to predicting energy load, since no consumer is involved. The study was conducted on renewable energy forecasting using data from one wind farm over a year and five photovoltaic farms over four months. The wind farm data was collected at 5-minute intervals, capturing wind speed, wind power, and power generation metrics, while the photovoltaik data was sampled hourly, recording solar power generation (kWh) and collection time. During preprocessing, a Savitzky-Golay filter was applied for data smoothing, complemented by a Local Outlier Factor filter to effectively reduce noise and outliers in the dataset. The researchers proposed a novel encoder-decoder framework, designated as SL-Transformer, which integrated a Transformer architecture for the encoder component and an LSTM structure with an attention mechanism for the decoder component. Performance evaluation for wind power prediction demonstrated that the SL-Transformer significantly outperformed alternative methods including SVM, standard Transformer, NBEATS, LSTM, and ARIMA, achieving a superior Symmetric MAPE (SMAPE) of 5.85% compared to the second-best model (LSTM) which recorded a SMAPE of 16.42%. Similarly, for solar power prediction, the SL-Transformer maintained its performance advantage with a SMAPE of 4.22%, substantially surpassing the runner-up LSTM model which achieved a SMAPE of 11.14%.

Zhang et al. [34] introduce the Time Augmented Transformer, which incorporates a temporal augmented module to learn temporal relationship representations. The authors argue that input calendar data introduces too much noise into the model. Instead, the Transformer is prepended with a time augmentation layer. This layer gets temporal information about a time step, like year, month, day, time-stamp, day of week, holiday, as a one-hot encoded input vector. Then, the model learns on the encodings using two fully connected network layers, before concatenating the embeddings with the actual time-series data. This creates the final input for the Transformer model.

The study tests their model on the same dataset that Zhao et al. [16] use, half-hourly data from 2006 to 2010 of electrical load data of New South Wales, obtained from the Australian National Electricity Market. Apart from the electrical load, they use hours,

dry bulb temperature, wet bulb temperature, dew point temperature, humidity, and electrical price as features. They evaluate the architecture on different forecasting horizons, 30 minutes, one hour, 12 hours, and one day. Their lookback window has the size of 48 hours. The approach is compared to ARIMA, SVR, LSTM, bidirectional LSTM [85], CNN-LSTM, Seq2Seq [52], and the vanilla Transformer [12].

As a result, the Time Augmented Transformer outperforms all models for the prediction horizons of 30 minutes, one hour, 12 hours, and one day, with a MAPE of 0.465%, 0.618%, 1.69%, and 2.62%, respectively. The second best scores are achieved by the vanilla Transformer, with a MAPE of 0.744%, 0.618%, 3.27%, and 3.89%, respectively.

Two works evaluated the TFT [31] on forecasting electricity consumption ([19], [57]). Huy et al. [19] perform forecasting on electricity consumption data of Hanoi city of seven years. Six years are used for training, the last year for testing. As a preprocessing step, they remove trend value from load value using the differencing transform method. The detrended values are used as training input of the TFT, while the trend values are fed to a linear regression model. The final output is computed by summing the prediction of TFT and linear regression. Additional features used are temperature, relative humidity, calendar data, and holiday. The authors evaluate the following models on a 24 hours forecast: ARIMA, Prophet [86], Seq2Seq [52], linear regresssion + LSTM, and linear regression + TFT. The proposed method combining linear regression and TFT outperforms all models, achieving a MAPE score of 3.38%. The method combining linear regression and an LSTM achieves the second-best results, with a MAPE of 4.64%.

Giacomazzi et al. [57] apply the TFT to short-term electricity load forecasting, utilizing hourly data for both day-ahead and week-ahead prediction horizons. The model incorporated additional input features like historical load, calendar data, temperature, epidemic data, and grid node information. The TFT implementation featured a separate variable selection network to connect the various input features. When benchmarked against baseline LSTM and ARIMA models, TFT demonstrated superiority for larger forecasting horizons. For the task of week-ahead prediction on the grid level, the TFT achieves a MAPE of 3.88%, compared to 4.24% achieved by the LSTM. Although, the results produced for day-ahead forecasts (4.22%) were worse than those of the LSTM (3.52%). On the hierarchical and substation level, the TFT outperformed the LSTM on the day-ahead forecast as well. The study highlighted the

interpretable multi-head self-attention mechanism of TFT as a significant advantage, which must be weighed against its higher training computational requirements.

2.2.3 xLSTM for Time-series Forecasting

The xLSTM is presented as an alternative for the Transformer-based language models. Therefore, its authors heavily test it on language tasks and multi-query associative recall tasks [14]. Although the xLSTM is evaluated on tasks from the long range arena, verifying its abilities to capture long range dependencies, it is not directly build for the task of time-series forecasting. Yet, the xLSTM architecture provides several tweaks that make it suitable for that task. The sLSTM has multiple memory cells, and can therefore store more complex patterns, which is useful for time-series prediction [87]. In general, the xLSTM can handle long-term context better in terms of finding and extracting information from important past patterns than the LSTM.

Alharthi et al. present xLSTMTime, which is an adaptation of the xLSTM for time-series forecasting [87]. The model incorporates additional steps like series decomposition, extraction of trend and seasonal components, and a post-reversible instance normalization. The latter is done by reversible instance normalization [88], which is a normalization-and-denormalization method with learnable transformations. It removes and restores statistical information of time-series, even when mean and variance change over time. The xLSTMTime outperforms several Transformer-based architectures (iTransformer, PatchTST, Crossformer, FEDformer, Autoformer, Informer, Pyraformer) on the task of long-term time-series forecasting on the Electricity Transformer Temperature (ETT) dataset [13]. When forecasting electric power consumption, the xLSTMTime achieves comparable results to PatchTST.

Kraus et al. present the xLSTM-Mixer, a model using linear layers and sLSTM blocks for time-series forecasting [89]. First, they apply channel-independent linear layers on the different input time-series. Then, the forecast is up-projected into a hidden dimension and refined by an xLSTM stack. The model uses stacks of only sLSTM blocks, for time and variant mixing. The blocks create two forecasts, one from the original and one from the reversed up-projection. As a last step, the two forecasts are mixed by a final linear layer.

They use linear layers, because of recent findings that simple linear models with additional normalization schemes perform well enough on forecasting tasks ([90], [91]). Yet, they clarify that linear models alone are not able to capture the complex de-

pendencies present in time-series datasets. As a normaliztation scheme they choose NLinear over trend-seasonality decomposition, stating the latter to be redundant. The xLSTM-Mixer outperforms xLSTMTime on mentioned datasets ETT and Electricity, as well as other Transformer-based models (PatchTST, iTransformer, FEDformer, Autoformer).

2.3 Open Research Questions

To advance current research in heat load forecasting, this work aims at benchmarking ML models on the task. The experiments cover forecasting of heat load in hourly and daily resolutions, as well as for shorter and longer horizons. The main goal is to determine which model is best suited for heat load prediction on the data at hand, combining assessments on accuracy and training effort. Past research indicates that Transformer-based models or the xLSTM could improve precision on forecasting tasks over simpler models like FCN, RNN, and LSTM. Another aspect, not found in heat load forecasting yet, is that the model is trained and tested on several time-series of different buildings and energy types. This brings new challenges, such as incorporating static covariates like building type, consumed energy type, and building size. A critical research objective is to evaluate the capacity of ML models to effectively process the heterogeneous dataset and encompass knowledge from external features.

3 Methodology

3.1 Overview

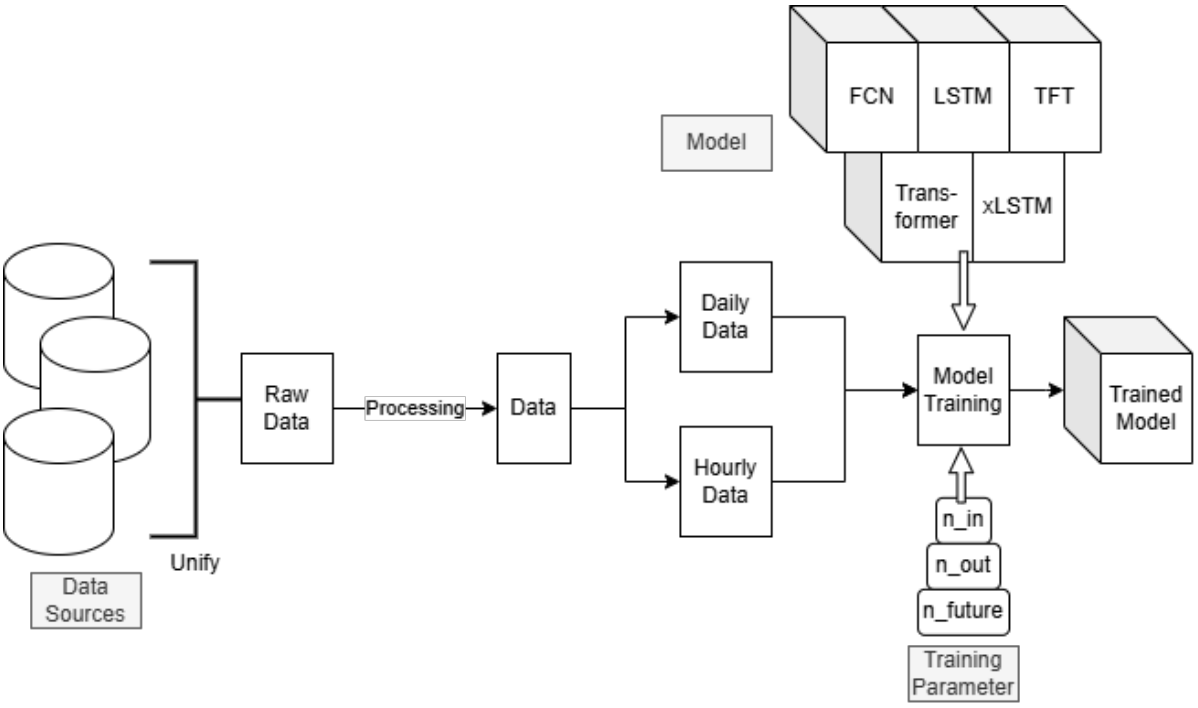

Figure 3.1: High-Level overview of model training process.

The process of model training can be viewed in Fig. 3.1. After unifying the data from all three data sources ([28], [92]), the raw data is processed. More details on data preprocessing can be found in Sec. 4.2. Experiments are done for daily and hourly samples of the data, while not all data is available in hourly resolution. The experiments include the training of FCN, LSTM, Transformer-Encoder (TE), TFT, and xLSTM. All models are trained on both temporal resolutions, with different input lengths $n_{\mathbf{in}}$, forecast horizons $n_{\mathbf{out}}$, and length of future covariates $n_{\mathbf{future}}$.

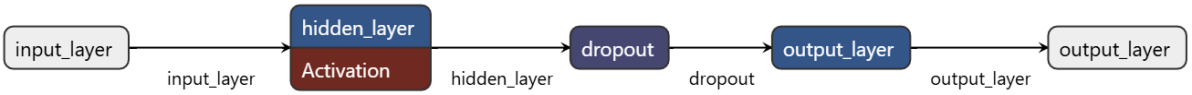

Figure 3.2: Architecture of FCN with one hidden layer

3.2 Model Implementations

This section covers the methods used in this work for performing heat data forecasting on the collected data. It describes the details of the implementations of evaluated ML models.

For implementing models and model training, the programming language Python [93] version 3.10 is used. Especially, the Python libraries TensorFlow (v2.18) [94] and PyTorch (v2.6) [95] are employed. For logging and experiment tracking, Weights&Biases [96] is utilized. The code for the TFT model and training is published by Google [97], and only runs for TensorFlow 1.15 and Python 3.7. It was slightly adapted for TFT experiments. Also, the xLSTM code is published on GitHub [98] and written in PyTorch, which was slightly adapted for the xLSTM training and evaluation. All other models are implemented using TensorFlow 2.18. The code with all implementations is available on GitLab[1].

3.2.1 FCN, RNN, and LSTM

The FCN architecture has one layer with dropout, like seen in previous works ([17], [21]). Fig. 3.2 shows the basic architecture with one input layer, hidden layer, dropout layer, and output layer. Hyperparameters are number of neurons in the hidden layer and dropout rate.

The activation function used in the hidden layer is RelU [99] and the output layer uses linear activation, as recommended for regression tasks [100]. The bias values are initialized with zeros, while the weights are initialized using the Glorot initializer [101]. The size of the output layer depends on the n_{out} parameter, which decides how many points the model will predict at once.

The RNN architecture chosen for time-series prediction experiments resembles the FCN architecture, but the hidden layer is replaced by a RNN-layer. The number

[1] https://gitlab2.informatik.uni-wuerzburg.de/modsimu/abschlussarbeiten/2025-heat-energy-forecast-marja-wahl

of cells in the RNN-layer and the dropout rate are hyperparameters. Again, bias is initialized with zeros and weights are initialized using Glorot initializer. For the RNN, the activation of the RNN-layer is tanh, while the output's layers activation function is still linear.

The LSTM for heat load prediction is structured as simple as FCN and RNN, but the hidden layer is an LSTM layer of variable size. The activation function for this layer is tanh, and the rest of the architecture is equal to that of the RNN.

3.2.2 Transformer Encoder

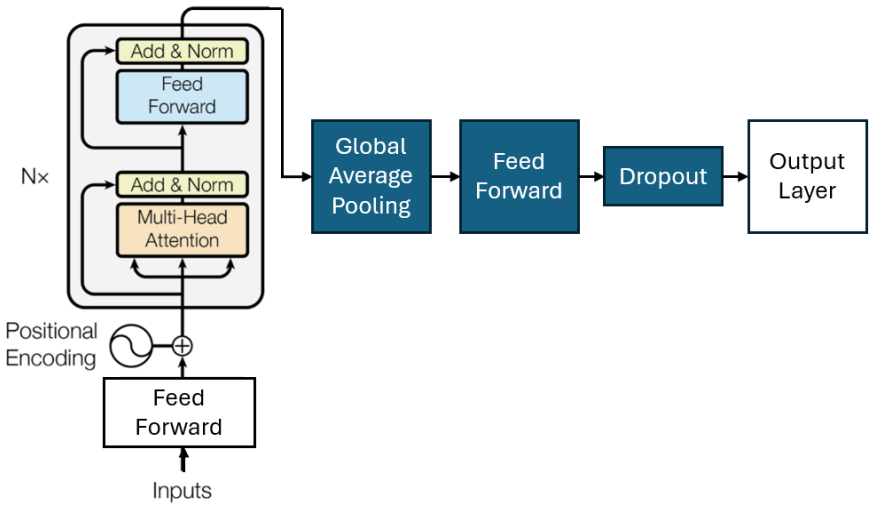

Figure 3.3: Architecture of Transformer encoder with N encoder blocks and preceding up-projection layer, as well as successive down-projection layers.

The vanilla Transformer architecture is adapted for time-series forecasting and used in the evaluation. For the experiments, only the Transformer Encoder is used, like done by Zerveas et al. [69].

First, a masking layer is implemented, to mask any future targets. Then the values are passed through an up-projection layer, which increases the size of input vectors using a fully-connected layer to a hidden dimension. Afterwards, a learnable positional embedding layer is applied. Now, the modified data is used as inputs to a stack of Transformer encoder blocks, where each has a specified number of heads, head size, and hidden dimension of the fully-connected layer. The outputs of the Transformer stack are down-projected to one dimension using global average pooling. Its outputs

are fed to a fully-connected layer with dropout. As a final layer, a dense layers is used for down-projecting to the desired output length. The architecture is shown in Fig. 3.3.

3.2.3 Temporal Fusion Transformer

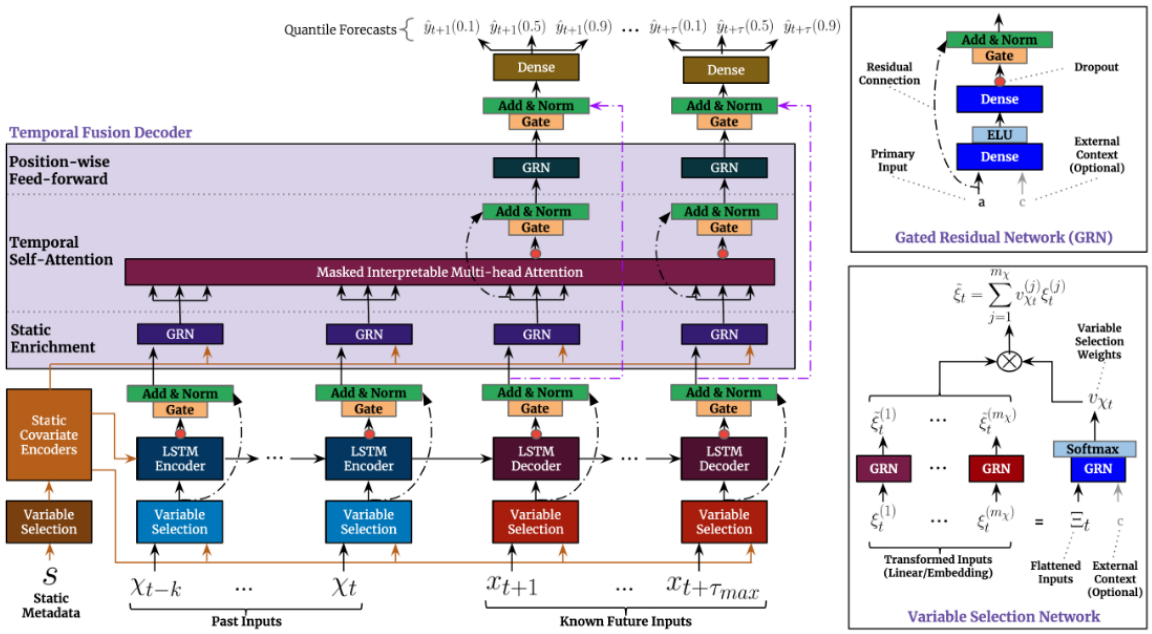

Figure 3.4: Architecture of TFT as presented by Lim et al. [31].

The original Temporal Fusion Transformer (TFT) architecture, like presented by Lim et al. [31] is used in the experiments. It can be seen in Fig. 3.4.

The TFT is able to handle past inputs, known future inputs, and static covariates individually. The latter are fed into a separate static covariate encoder. All inputs are passed through a Variable Selection Network (VSN), which uses Gated Residual Networks (GRNs) for selecting the most salient features.

Gating Mechanisms Throughout the architecture, multiple GRNs are installed. Those building blocks give the TFT the ability to apply non-linear processing only when needed. That process is realized by implementing skip connections and Gated Linear Units (GLUs) [102]. The GRN is defined as

$$\text{GRN}_\omega(\boldsymbol{a}, \boldsymbol{c}) = \text{LayerNorm}(\boldsymbol{a} + \text{GLU}_\omega(\boldsymbol{\eta}_1)) \tag{3.1}$$

$$\boldsymbol{\eta}_1 = \boldsymbol{W}_{1,\omega}\boldsymbol{\eta}_2 + \boldsymbol{b}_{1,\omega} \tag{3.2}$$

$$\boldsymbol{\eta}_2 = \text{ELU}(\boldsymbol{W}_{2,\omega}\boldsymbol{a} + \boldsymbol{W}_{3,\omega}\boldsymbol{c} + \boldsymbol{b}_{2,\omega}) \tag{3.3}$$

, where $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2$ are the hidden layers of the GRN, ELU is the Exponential Linear Unit activation function [103], LayerNorm is the standard layer normalization as proposed by Ba et al. [104], and ω is indexing weight sharing. Vector $\boldsymbol{c}$ is optitonal, representing external context. The GLU is defined by following equation:

$$\text{GLU}_\omega(\gamma) = \sigma(\boldsymbol{W}_{4,\omega}\gamma + \boldsymbol{b}_{4,\omega}) \odot (\boldsymbol{W}_{5,\omega}\gamma + \boldsymbol{b}_{5,\omega}) \tag{3.4}$$

, where $\sigma(.)$ is the sigmoid activation function, $\boldsymbol{W}_{.,\omega}$, $\boldsymbol{b}_{.,\omega}$ are weights and biases and $\odot$ is the Hadamard product. The GRN can adjust the degree of non-linearity by regulating how much information is added to input vector $\boldsymbol{a}$ using the GLU, as shown in Eq. 3.1. When the output of GLU is small, the non-linear transformations are skipped entirely.

Variable Selection Networks The VSNs of the TFT are applied to all variables, static and time-dependent. Yet, static, past, and future inputs are assigned to different VSNs. The VSNs serve two purposes. Firstly, they gather insights on importance of variables and secondly, they remove unnecessary information which might lead to clutter, improving model performance. First, input variables are embedded, where linear transformations are applied for continuous variables and entity embeddings [105] are used for categorical variables. Each input is transformed into a vector of length $d_{\textbf{model}}$, which is the hidden dimension used throughout the layers of the TFT. As a second step, every input is passed through its own GRN. Additionally, the flattened inputs, the vector of all inputs for timestep t and optional external context are passed through a GRN with following softmax layer. This creates a vector of variable selection weights, used to weigh the outputs of the individual GRNs. Finally, those are combined and used as output of the VSN.

Static Covariate Encoders The static covariate encoder is build to separately handle static covariate information and distribute it throughout the TFT. It produces four context vectors by applying four individual GRN to the static input data. Each vector

serves a different purpose, one is used as context for temporal variable selection, two are used as context for local processing of temporal features, and the last one enriches temporal features with static information.

Temporal Processing The main task of the TFT is temporal processing, which is realized by four layers in series. The first layer is a sequence-to-sequence LSTM encoder-decoder, which extracts information about surrounding values, leveraging local context. The created set of uniform temporal features are used as input to the main component, the temporal fusion decoder, consisting of three layers. The input gets enriched by static context and then processed by GRNs, which share their weights across the layer. The static-enriched temporal features are grouped per time-step t and are used as input for interpretable multi-head attention. This version aggregates the information spread across multiple heads by addition, so they can be interpreted more easily. The output is fed to a feed-forward layer consisting of a GRN. Each structure performing temporal processing is equipped with a skip connection, making it possible for the TFT to bypass those layers if they are not found necessary during training.

Prediction Outputs The original TFT implementation outputs quantile prediction intervals rather than point forecasts. They simultaneously predict values for the 10^{th}, 50^{th} and 90^{th} percentile. A prediction y_t for timestep t and for the 90^{th} percentile can be understood as the predicted outcome being smaller than y_t with a 90% probability. As a result, the model is trained using the quantile loss [106]. To make our experiments comparable, the TFT is set to point prediction by adapting the output layer to predict a single value. MSE is used as loss function during training.

3.2.4 xLSTM

The xLSTM used for time-series forecasting evaluations is almost the same architecture as the xLSTM presented by Beck et al. [14]. Yet, the original xLSTM is build for NLP tasks, why some changes are implemented to its structure, as depicted in Fig. 3.5.

First, the input is processed by an up-projection layer, to increase the size of the third dimension, which are the features, to the embedding size. The embedding size is set to $d = 256$. Then, the output of the up-projection is passed through mLSTM and sLSTM, which are described in Sec. 2.1.4.2. Here, the chain consists of seven entities, where all except the second one are mLSTM blocks. The second one is an sLSTM

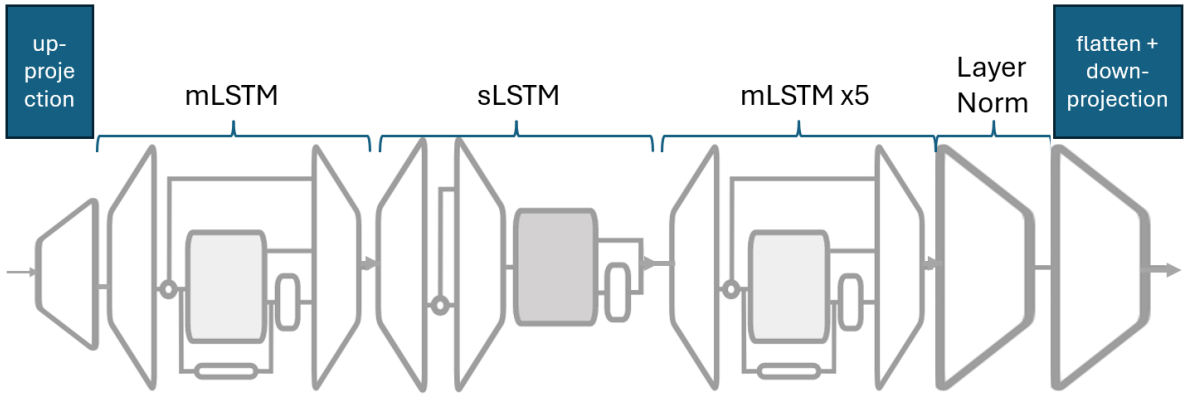

Figure 3.5: Architecture of the adapted xLSTM. mLSTM and sLSTM are like in the original xLSTM paper, added layers are described with blue text box.

block. For all layers, the number of heads is four and the size of the one-dimensional convolution is four. The context length is set to the input size n_{in}. For the mLSTM layers, the query-key-value size is set to four. A post-normalization layer of size d is added after, as done in the original xLSTM. To receive a forecast of a desired length, the output of the xLSTM is flattened to a two-dimensional vector. Then, the vector with input length equal to $d * n_{\text{in}}$ is down-projected to the output length n_{out}.

4 Dataset

For the evaluation of the methods presented in Chapter 3, data provided by RAUSCH Technology is used. The following section give an overview of the composition of the dataset and the preprocessing steps.

4.1 Overview

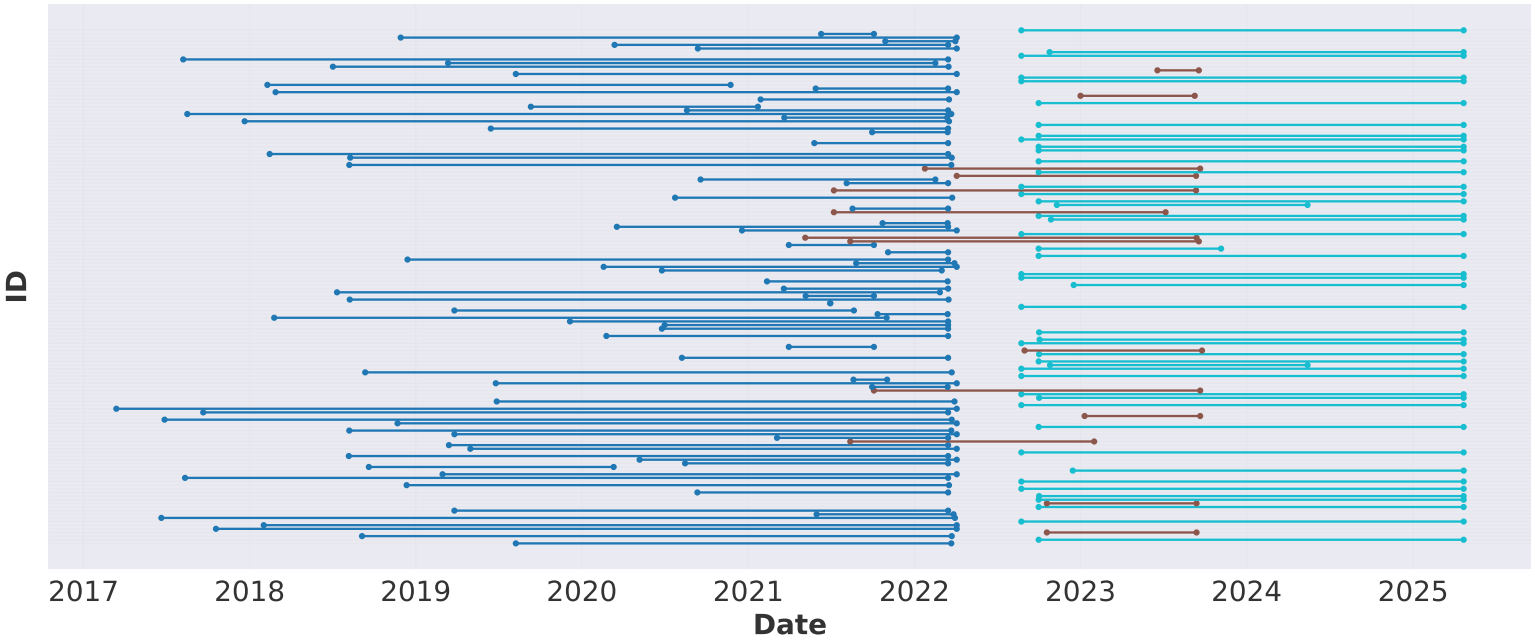

Figure 4.1: Time-series recording span overview for the raw daily dataset. Time-series are from following sources: Legacy (82, dark blue), KI-NERGY (14, brown), AI in district heating (47, cyan)

This section gives an overview of the datasets gathered to benchmark the different models on heat data forecasting, a numerical count can be seen in Tab. 4.5. The available time-series data is visualized in Fig. 4.1, and the different types of buildings

	KI-NERGY	Legacy	AI in District Heating	Sum
Sensors	20	82	47	149
gas meter	14	82	-	96
heat energy meter	6	-	47	53

Figure 4.2: Number of Sensors Overview

	KI-NERGY	Legacy	AI in District Heating	Sum
Data Points (daily)	7,128	65,752	38,043	110,923
gas meter	5,434	65,752	-	71,186
heat energy meter	1,694	-	38,043	39,737

Figure 4.3: Number of Daily Data Points Overview

	KI-NERGY	Legacy	AI in District Heating	Sum
Data Points (hourly)	168,834	-	701,396	870,230
gas meter	128,697	-	-	128,697
heat energy meter	40,137	-	701,396	741,533

Figure 4.4: Number of Hourly Data Points Overview

Figure 4.5: Overview of raw data counts and sources by category

are listed in Tab. 4.6. The location of sensors is visualized in A.4. The following paragraphs describe the three data sources and the specifics of each corresponding dataset.

KI-NERGY The KI-NERGY research project [3] investigated the application of AI for optimizing heating systems in the building sector. It aimed at developing an AI-based system that continuously monitors heating systems, identifies inefficient operating modes, and derives targeted recommendations for improvement of efficiency. For this purpose, multiple buildings in Germany were equipped with sensors that recorded heat consumption over time periods within the years of 2021 to 2023. Not all buildings' heat consumption were monitored over the whole time period, as errors with sensor setups and connection failures occurred. The project monitored gas and district heating consumption, along with other parameters consumption and useful energies. This work uses the heating energy consumption records only. These were observed in twenty buildings of type residential (7), student housing (4), school

Building Type	Count	Percentage
Residential Buildings	131	87.9%
School	11	7.4%
Student Housing	4	2.7%
Museum	2	1.3%
Social Housing	1	0.7%

Figure 4.6: Overview of types of buildings in the dataset.

(7), and museum (2). Gas meters measure heating and warm water energy consumption simultaneously. The heat load is recorded in kilowatt-hour (kWh) for buildings connected to district heating, and in m^3 for gas-fueled buildings. To unify the dataset, m^3 is converted into kWh by multiplying it with $11.3 kWh/m^3$, the higher heating value, like done by Djebko et al. [3]. In the data, information about the heated area is available for all buildings but one.

AI in District Heating RAUSCH Technology collaborated with the German Energy Agency (dena) on the "AI in District Heating" project [28], focusing on integrating AI into district heating networks to optimize processes and reduce CO_2 emissions. The main goal was to identify and evaluate AI-supported use cases for district heating systems and develop a practical guide for district heating providers to implement their own AI projects. The project was conducted in partnership with Stadtwerke Norderstedt (a municipal utility company) as a pilot partner. Throughout the project, heat energy meters in buildings in Norderstedt recorded district heat load in 47 residential buildings. Unfortunately, none of the buildings hold information about the heated area. The data was measured in the years of 2022 until 2025.

Legacy The legacy dataset consists of gas consumption readings done by Ener-IQ (now RAUSCH Technology). These recordings were taken as part of an effort to create an AI-based product for recommending actions regarding heating systems in residential buildings. It consists of 96 properties, whereas for 19 of them there is no information about the heated area. Two readings of energy meters had to be summed 5 times, as their sum was equal to the total energy consumption of a building. The gas consumption is measured in m^3, but was converted to kWh by multiplying with 11.3 kWh/m^3. The different time-series were recorded in the years of 2017 until 2022.

4.2 Data Preprocessing

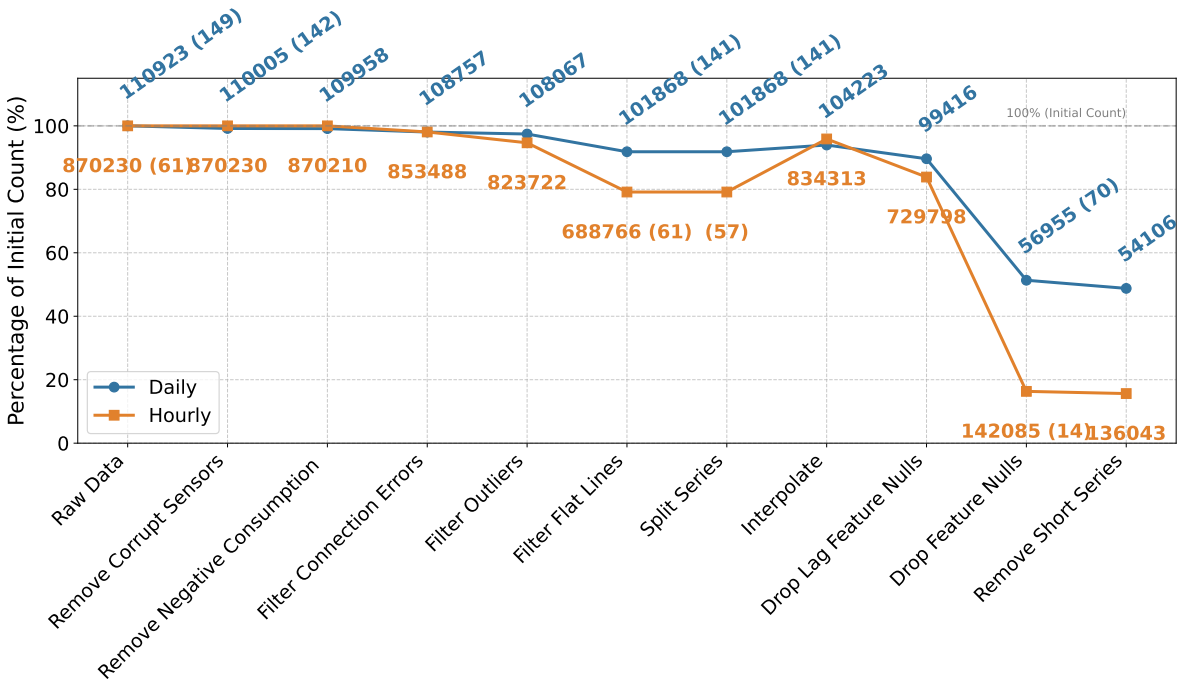

Figure 4.7: Count of data points after each preprocessing step in daily and hourly format. The numbers in brackets indicate the amount of energy meters .

After joining the data from all three sources, the raw data count sums up to 110, 923 daily values and 870, 230 hourly values. Unfortunately, when collecting data over long periods of time, there is always a risk of errors. Corruptions in the data can occur due to failed transmissions or wrong placements of the sensor. Therefore, the data needs to be put through an intensive preprocessing pipeline. The data is cleaned to provide the best possible foundation for successful model training. The following sections explain the single steps that were taken during preprocessing. Fig. 4.7 visualizes the amount of data left after each preprocessing steps for the daily and hourly datasets.

Removing Corrupt Sensors A corruption in the sensor can occur, when the sensor is not set up right, the transmission failed, and the data is simply too inconsistent to be used. As a first iteration, the corrupted sensors are identified by manually viewing the time-series and deciding whether or not to keep them. This step removed 7 out of the 149 daily data series, while all the hourly datasets are kept.

Transform Resolution The data has different temporal resolutions, since it is a collection of data from different projects and sensors. To unify the dataset, one resolution is chosen, daily or hourly. A part of the data was either recorded in a 1 hour or 30 minute interval. For the other part, only the averaged values of each day were accessible, see Tab. 4.5 for more detailed counts. To create two equally distributed datasets, the maximum energy meter reading of every hour or day are used to create an hourly and a daily dataset. This also means that time-series of an hourly format might be present in the daily dataset in an aggregated form. The data point count comes to 110,005 for the daily data and 870,230 for the hourly format.

Compute Energy Consumption The goal is to train models to predict the future energy consumption. Therefore, the data needs to be transformed from meter level to energy consumption. This is realized by computing the difference between each consecutive data point in a series. As a result, we lose one data point per series.

Remove Negative Values In case of data jumps due to system failures and counter resets, the data might contain negative consumption values after the previous step. These are simply removed and regarded like missing values.

Filter Connection Errors It is possible that sensors send large values due to connection errors. These values are stored and summed in the sensor, because it is not able to process them further. When the connection returns, an erroneous large value is sent. This creates larger jumps in the dataset. To make sure that these values are not part of the dataset, rows right after periods of missing values are removed.

Filter Outliers Apart from connectivity problems, other circumstances can cause outliers in the dataset. To get rid of any other outliers, the IQR method is used. It is a robust statistical technique for filtering sensor data contaminated by connection errors that produce abnormally large values. An example time-series, where IQR is applied for removing outliers, can be seen in Fig. 4.8.

First, the 25^{th} (Q25) and 75^{th} (Q75) quantile for every time-series is calculated. Then, the IQR is determined as the difference between these quantiles: $Q75 - Q25 = IQR$. An upper bound is defined as $Q75 + 1.5 * IQR$, and all values above are identified as outliers and are removed.

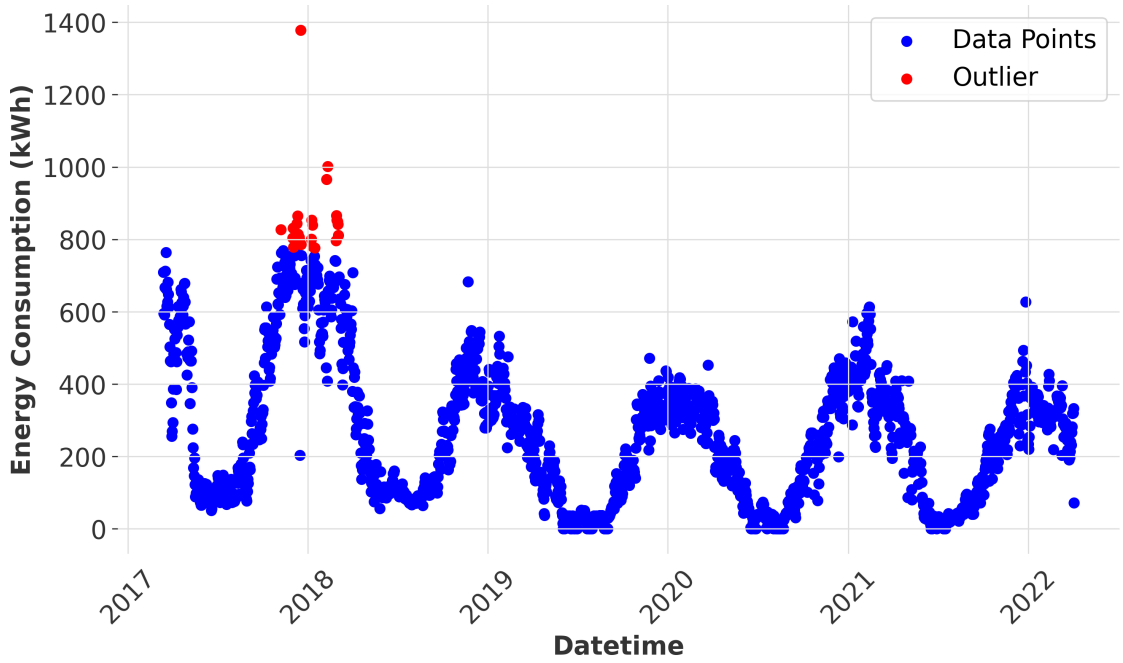

Figure 4.8: Example time-series, where outliers detected by the IQR method are colored in red. These points are removed for further processing.

The advantages of the IQR method lies in its resistance to extreme values; unlike mean-based filtering techniques, it is not skewed by the very outliers it aims to detect. This makes it particularly valuable for real-time sensor systems where connection errors can introduce spurious readings.

Filter Flat Lines Filtering flat lines in sensor data presents a nuanced challenge that requires careful threshold calibration. An example of a flat line is visualized in Fig. 4.9. When monitoring heating systems, consecutive zero or unchanging values may indicate either a legitimate system shutdown or a sensor malfunction. The filtering process typically involves setting a threshold for the maximum number of acceptable consecutive identical readings—particularly zeros—before flagging the sequence as problematic. The critical balancing act lies in determining this threshold: set it too low, and normal operational states (like a deliberately inactive heating system during warm periods) may be erroneously discarded as failures; set it too high, and actual sensor malfunctions might go undetected, compromising data integrity. To remove those flat lines, a threshold of 90 days is set. If the number of consecutive zeros is larger than the threshold, the data is discarded.

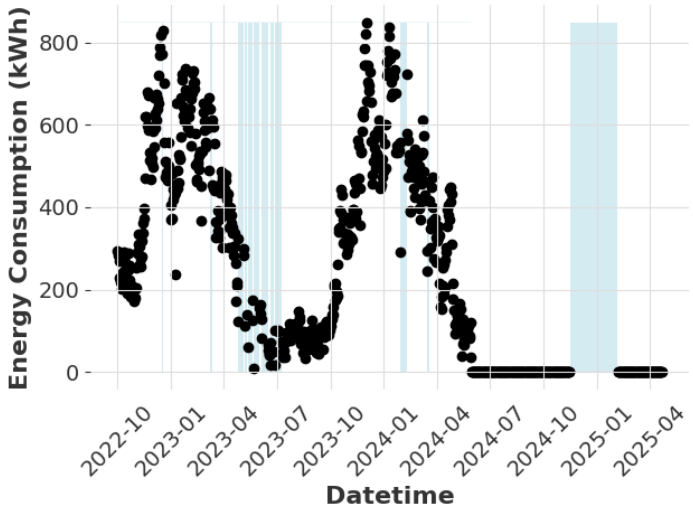

Figure 4.9: Plot of a time-series recording daily energy consumption. The series shows a flat line starting around June, 2024. Missing data is signaled by blue color. The consumption is shown in kWh.

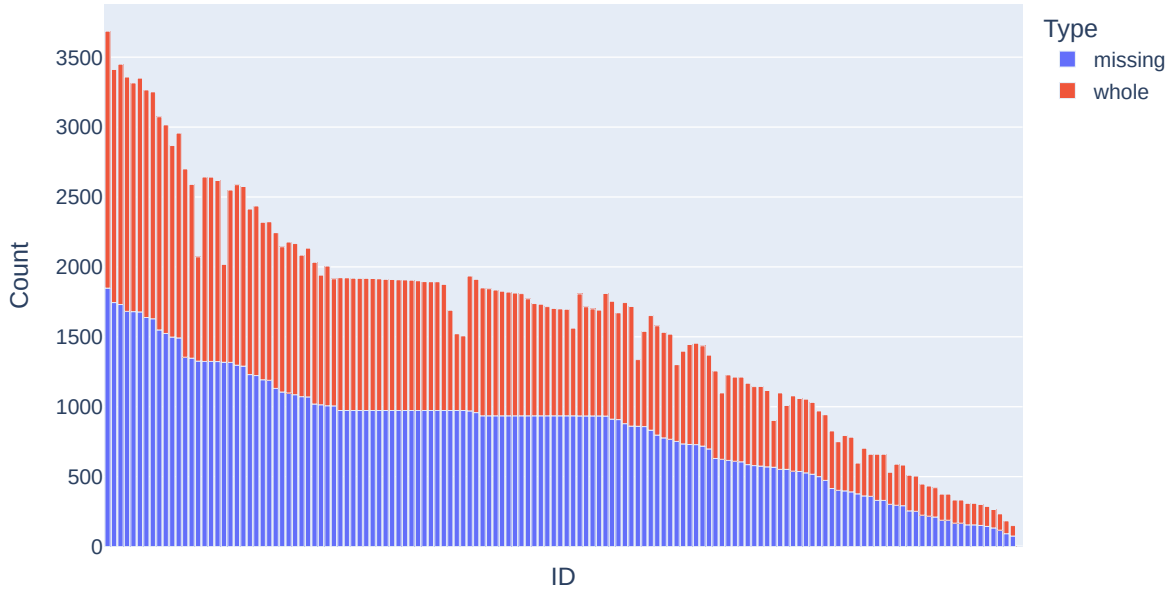

Figure 4.10: Number of missing data points compared to available data points per series for raw daily data.

Interpolation of Missing Data Points The dataset inherently existed with missing data points, as displayed in Fig. 4.10, and all previous preprocessing steps might have created data holes in the time-series. A model learns best on complete time-series, thus the missing data points need to be artificially replaced by using linear interpolation. To create more realistic values, the energy meter readings are interpolated, rather than the energy consumption values. Also, only data holes up to a certain threshold are filled by interpolation. Otherwise, long periods of missing values would be replaced with a constant value, resulting from a linearly interpolated meter reading. If the proportion of artificial values is too high, it might have negative consequences for the models prediction accuracy on real-world data.

The interpolation occurs three times in the data processing pipeline. First, before computing energy consumption values, to get rid of all initial missing values. During this step, only data holes up to a certain threshold are filled by interpolation. The thresholds are set to 7 days for daily values and 24 hours for hourly data. The second time interpolation takes place is after splitting of time-series. The data is interpolated once more after removing negative values, in case the removal created holes in the data.

Splitting of Time-series As mentioned in Sec. 4.2, only data holes up to a certain threshold are filled using interpolation. The rest of them are treated by splitting the time-series before and after the missing data. This is done to create time-series with the most amount of actual recorded data possible. While interpolating is a good practice to replace small holes in the time-series, for the data at hand, filling larger ones would create a too big proportion of artificial data. This might skew model accuracy, when presented with real-life recordings. A time-series that was split can be seen in Fig. 4.11.

4.3 Features

Apart from historical load data, exogenous features can hold valuable information for predicting future energy consumption. All works mentioned in Sec. 2.2.1 use at least one feature to improve model training.

Weather Data Many papers ([17], [18], [20], [21], [67]) use weather data as a feature. As a result, the dataset for this work is also enhanced with weather data. The data is

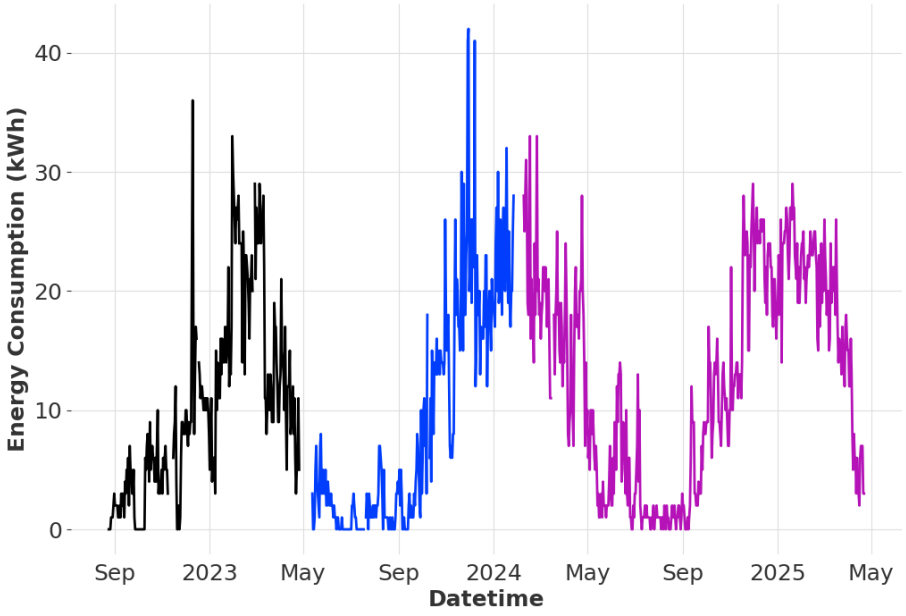

Figure 4.11: Example time-series that was split along missing data periods and interpolated. The different colored sections are new time-series.

collected from meteostat[1], an open weather and climate API. Among others, the data is provided by Germany's national meterological service (DWD).

Given a postal code, start, and end time, they provide daily and hourly data. They gather the data from a weather station closest to the postal codes dedicated location. The daily and hourly data consists of temperature, wind, air, and more information, as displayed in Tab. 4.1. The hourly data contains humidity information, for which the average, minimum, and maximum value for each day are added to the daily dataset as well.

Building Information Furthermore, static information about the buildings can be relevant for the model to make more accurate predictions. These type of features are not covered in related work, because other works train one model per building. Therefore, the building information is irrelevant, as it does not change. For this work, each time-series is added their buildings heated area in m^2 and the number of appartments, if available.

[1] https://dev.meteostat.net/python/

Column	Description	Daily	Hourly
tavg	The average air temperature in °C	X	
tmin	The minimum air temperature in °C	X	
tmax	The maximum air temperature in °C	X	
temp	The air temperature in °C		X
dwpt	The dew point in °C		X
rhum	The relative humidity in percent (%)		X
avghum*	The average humidity in percent (%)	X	
minhum*	The minimum humidity in percent (%)	X	
maxhum*	The maximum humidity in percent (%)	X	
prcp	The daily precipitation total in mm	X	X
snow	The snow depth in mm	X	X
wdir	The average wind direction in degrees (°)	X	X
wspd	The average wind speed in km/h	X	X
wpgt	The peak wind gust in km/h	X	X
pres	The average sea-level air pressure in hPa	X	X
tsun	The one hour sunshine total in minutes (m)	X	X

Table 4.1: Daily and hourly weather features provided by meteostat. Values with a (*) are calculated from hourly values.

To give the model even more sense of the order of magnitude of the current buildings consumption, the average daily consumption, computed on the training data, is added as a feature.

Since the model is trained on gas and district heat data simultaneously, a binary feature determining which energy type the time-series corresponds to is added as well. Using this information gives the model the possibility to identify any characteristics of different energy types and use it to make more accurate forecasts.

Calendar Data Features regarding the time of year are added to the consumption values as well. Binary features include `weekend` and `holiday`, which signal whether the current day is a weekend or holiday, respectively. Cyclic information, like `weekday` and `day_of_month`, are encoded using sine and cosine:

$$f_{\text{enc}} = \sin\left(\frac{2\pi \cdot f}{k}\right) \tag{4.1}$$

$$f_{\text{enc}} = \cos\left(\frac{2\pi \cdot f}{k}\right), \tag{4.2}$$

where $k = 7$ for `weekday` and $k = 31$ for `day_of_month`.

Adding Features Adding the features can be done by merging the different data points on the timestamp and ID column of each time-series. Unfortunately, not all data sources provide information about the buildings, like heated area or number of appartments. Therefore, the `building` dataset is created, which is the subset that has weather, calendar, and building features available, whereas building features are now defined as heated area, daily average, type of building, and primary energy. The number of appartments is left out, as it would reduce the data even further, and part of the information is contained in the heated area. For the daily dataset, data points reduce from 99,416 to 56,955, the hourly data points are reduced from 729,798 to 142,085.

4.4 Data Analysis

To get a better understanding of the relation of the features and our data, a statistical analysis is performed. This section shows results of a p-value and correlation analysis of daily and hourly datasets with features.

4.4.1 Correlation Analysis

As part of the correlation analysis, the R^2 value, the adjusted R^2, p-values, and a correlation matrix are computed.

The R^2 value [107] measures the proportion of variance in the dependent variable that is explained by the independent variables in a regression model. The dependent variable in our case is the energy consumption, the independent variables are the features. The R^2 value can be defined as $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$, where SS_{res} is the sum of squares of residuals and SS_{tot} the total sum of squares. The value ranges from 0 to 1, where 1 indicates that the model explains all the variability in the response data around its mean, and 0 indicates the model explains none of the variability.

The adjusted R^2 [107] modifies the standard R^2 by penalizing the addition of predictor variables that do not significantly improve the model. Unlike R^2, adjusted R^2 increases only when added variables improve the model more than would be expected by chance. It is particularly useful when comparing models with different numbers of predictors.

The p-value [107] indicates the probability of observing the data, or more extreme data, if the null hypothesis were true. In regression analysis, it helps determine if the

relationship between variables is statistically significant. A small p-value (typically ≤ 0.05) suggests strong evidence against the null hypothesis, indicating that the observed relationship between variables is unlikely to occur by random chance.

A correlation matrix displays the Pearson correlation coefficients [108] between multiple variables simultaneously. Each cell shows the correlation between two variables, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating no linear relationship. It is useful for identifying multicollinearity and understanding relationships between variables before building regression models.

Daily Dataset The analysis is performed on the preprocessed dataset of the daily format, which sums to 56,955 data points.

With 28 features, we achieve an R^2 value of 0.7783 for the dataset, and an adjusted R^2 of 0.7782. This means, about 77.8% of the variability of the target variable can be explained using the features at hand. With a significance level of 0.05, 19 out of the 28 features are deemed as significant. More details on the p-values can be found in A.2 and a detailed description of the features can be found in A.1.

The correlation of the target variable `diff`, which stands for the daily consumption, with other features is investigated. A correlation matrix can be seen in A.5. The highest positive correlation values can be identified for the daily average (0.84) and heated area (0.78). The largest negative correlation can be seen with the maximum temperature `tmax` (-0.3), the average temperature `tavg` (-0.29), and the minimum temperature `tmin` (-0.25). These features are within themselves highly correlated. All other features show a correlation to `diff` smaller than 0.2.

Hourly Dataset The preprocessed hourly dataset consists of 142,085 data points at the time of the analysis. With 25 features we compute an R^2 value of 0.55996 for the dataset, and an adjusted R^2 of 0.55989. This result shows that about 65% of the variability of the target variable can be explained by given features. This is less than for the daily dataset, indicating that achieving a higher forecasting accuracy on the hourly dataset might be more difficult. With the same significance level of 0.05, 24 out of the 25 features are calculated as significant. More details can be found in A.3.

The computed correlation matrix can be found in A.6. Again, a large positive correlation with the target variable `diff` can be found between the daily average (0.6) and the heated area (0.44). Also, the type of the building being a school is positively correlated (0.45), and the relative humidity (0.25). The largest negative correlation

can be found with the temperature (−0.43), the dew point temperature (−0.38), and the building being residential (−0.34).

4.4.2 Cluster Analysis

To get a better overview of the magnitude of energy consumption of the different buildings in our dataset, a cluster analysis is performed on the preprocessed data. For this, the mean, standard deviation, median, minimum, and maximum value for every building is computed and hierarchical clustering is performed with those metrics.

The hierarchical clustering algorithm used is called agglomerative clustering [109]. It builds nested clusters by merging smaller clusters into larger ones. It follows a bottom-up approach, where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. The distance calculation computes the similarity between all pairs of clusters, using distance metric and linkage criterion. The euclidean distance is used as distance metric and the ward's method is used as linkage criteria. The ward's method is defined as minimizing the increase in the sum of squared distances within clusters after merging. The number of clusters to be found is set to three.

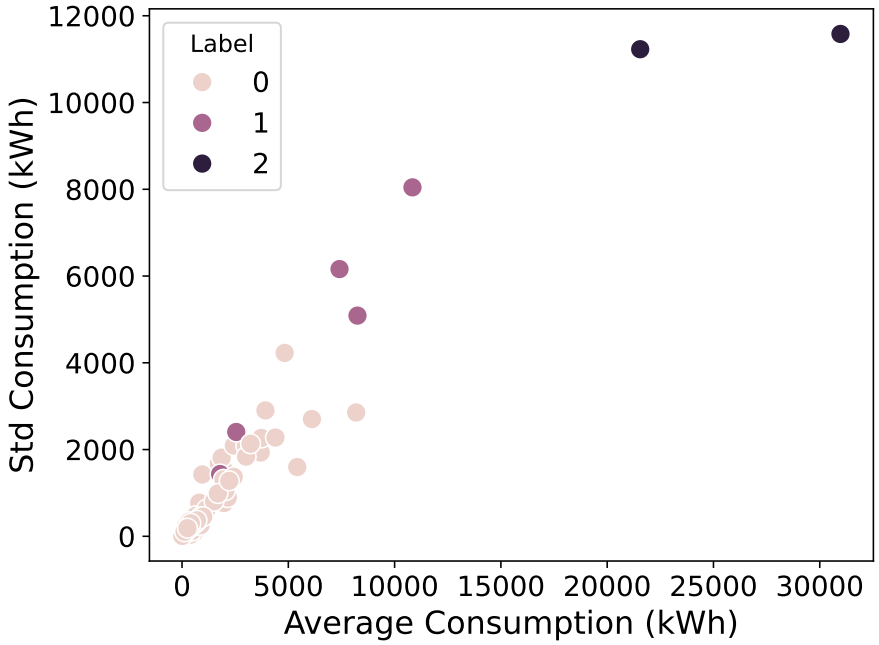

Figure 4.12: Result of clustering algorithm on mixed dataset. Each sensor belongs to one cluster and is visualized using the average of its daily consumption values and the standard deviation.

The data that is clustered consists of the combined daily and hourly preprocessed data, but the hourly data is transformed to the daily format. Series are remapped to one sensor, if they were split. Overlapping data, if the series was in daily and hourly dataset, is removed, so that only one occurrence remains. Therefore, the data consists of 57,168 data points from 74 sensors.

In Fig. 4.12, the different sensors and their respective clusters are visualized. Cluster 0 is the largest with 67 sensors, while cluster 1 only has 5, and cluster 2 only 2 corresponding sensors. The average daily consumption in cluster 0 is 1495.82 kWh, 6168.32 kWh in cluster 1, and a more than 4 times greater average of 26,264.13 kWh for cluster 2.

The cluster analysis shows that the dataset has two buildings with a very high energy consumption compared to the other buildings. This might be a direct result of their size, as one of the building has a heated area of 39,720 m^2 and the other of 49,339 m^2. By including the heated area as feature, the model should be able to identify the relation to higher energy consumption during training. Also, preprocessing techniques for training need to make sure to scale the data, to minimize effects caused by the difference in magnitude.

4.5 Discussion

The dataset is highly heterogenous, meaning the time-series have different lengths, come from different buildings, and cover different periods of the year. This can be seen in Fig. 4.1. The model needs to capture many more aspects, than if it would be trained on a time-series of only one building. Overall, the goal is to train a model that can generalize well, in order to be able to use this model for any building that might be added to the data space later on.

The data is split before and after missing data, to create continuous time-series. In practice, data might be missing due to connectivity or sensor issues as well. The model would have to either use filler values to create predictions, which might reduce accuracy, or wait for the amount of time that is set as the input period, before it can create forecasts again.

The feature giving information about the heated area is not available for almost half the dataset, which drastically reduces the data usable for training. It could be argued that in practice, the information is often also not available. Yet, given the correlation analysis results presented in Sec. 4.4.1, it the feature presumably helps a model to

create a more accurate forecast. In order to validate this hypotheses, the data needs to be the same, whether the feature is used or not. This made it necessary to only use the smaller dataset for experiments, even when the `heated_area` feature was not used during training.

Adding the heated area by computing it using Level of Detail (LoD) data was considered. LoD data is freely available data of german buildings, giving information about height, function, floor plan, and number of floors. For Schleswig-Holstein, the state where most of the buildings with missing area information are, a website[2] provides access to free to download LoD-data. With the information about height and floor plan, which can be used to compute ground surface, following function can be used to calculate the heated area: height $*$ ground surface $*$ 0.32. Yet, the provided data was found to not be reliable enough. Therefore, to prevent from feeding the model incorrect information, the idea was discarded.

[2] https://www.geodaten.schleswig-holstein.de/gaialight-sh/_apps/dladownload/dl-lod2.html

5 Evaluation

This section presents the experiments performed for benchmarking FCN, LSTM, xLSTM, TE, and TFT on the task of heat load prediction. First, the training data generation is explained in Sec. 5.1. Then, the experiment setting and results for one and seven day, as well as three and 24-hour prediction are presented in Sec. 5.2. Also, the influence of training data amount and future covariate accuracy on model performance is investigated. The Chapter is concluded by a theoretical and empirical time and memory complexity analysis of the benchmarked models in Sec. 5.3, and a discussion of the results (Sec. 5.4).

5.1 Training Data Generation

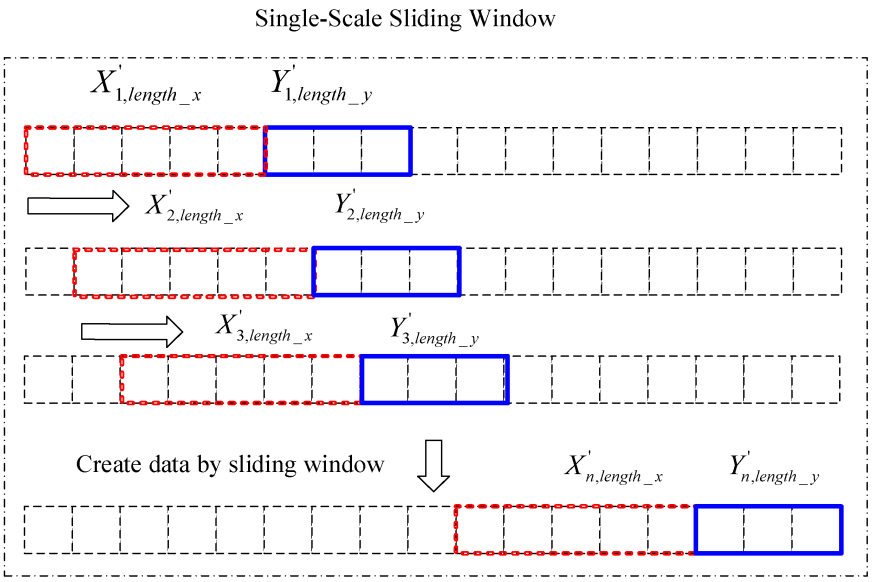

Figure 5.1: Sliding window technique with step size one [110]

Training Data Transformation As of now, each row in the dataset holds consumption and feature information about one time step t. For the model to be able to perform supervised training, the data needs to be transformed into input features $X'_{t-\text{length\_}x,\text{length\_}x}$ and target values $Y'_{t,t+\text{length\_}y}$, with look-back size length_x and look-forward length length_y. For this, the sliding window technique is used. It works by dividing a continuous time series into overlapping segments that serve as input-output pairs for model training. For each position of the window, the data within the window becomes the input features X, while the data points immediately following become the target values y to predict. This approach effectively transforms a time-series problem into a supervised learning problem, where each input window is associated with a specific output value or sequence. For our experiments, we use a step size of one, meaning the window is slid further one time step for each data point creation. The schema is depicted in fig. 5.1.

For training, we have one target value time-series and number of n_c continuous covariate time-series. When transforming the data, we treat all series the same, resulting in number of $c = 1 + n_c$ input feature series $X^c_{i,\text{length\_}x}$ and one target series $Y_{i,\text{length\_}y}$. Covariate target series can be used as future covariates by some models, which represent weather forecasts, for example. The input for the FCN groups all n input feature series in one row, each column being one feature at timestep t. Therefore, the input of the FCN is a matrix of size $X = (n, c{*}\text{length\_}x + n_s)$, with n number of data points and n_s number of static features. The target size is defined as $y = (n, \text{length\_}y)$. For all other models, the input needs to be of 3-dimensional form (number of data points, input length, number of features). Therefore, the input is defined as $X = (n, \text{length\_}x, 1 + n_c + n_s)$, where static covariates values are transformed to a series, by repeating their value for each time step t. The only model that considers static and future covariates separately is the TFT.

The future covariate handling can be explained in more detail. Models, like the FCN, LSTM, and TE, handle future covariates by implementing a masking layer, which makes the model ignore target series values during training. The length of future covariates is the same as length_y for LSTM, TE, and TFT. The input size changes to (n, length_x + length_y, 1 + number_of_features). Like mentioned, the TFT implements additional methods to handle future covariates. For the FCN, each future covariate for timestep t is simply a new input feature. Therefore, the number of future covariates can be bigger or smaller than length_y when training the FCN.

Training-test-validation Split To be able to perform supervised learning, each data point in the transformed dataset needs to be either in the training, test, or validation dataset. It is important that the model only sees data in the training set, so it does not get any information about test data beforehand. The validation data is used to create reference metrics during training, but is not used for training the model itself. As done by former research ([17], [18], [20], [21], [65], [68]), the data is split along the time axis, with 80% used for training, 10% for validation, and 10% as test data. Consequently, the model will see each time-series during training. Yet, since the time-series in the dataset cover different periods of time, the model might see a different time of year and amount of data for each time-series. After splitting, the daily data amounts to 42,147 data points for training, 4,165 for validation, and 4,206 for testing. The hourly training data has a length of 105,274 data points, 10,016 data points for validation, and 10,028 data points for testing.

Scaling When working with multiple time series that vary significantly in scale, standard scaling becomes essential to establish fair comparability and prevent larger value ranges from dominating the analysis. Therefore, standard scaling is applied to input and target values. It transforms the data to have a mean of 0 and a standard deviation of 1. The following equation is used: $z = \frac{x-\mu}{\sigma}$, with original value x, mean over the values μ, and standard deviation of the values σ, resulting in the standardized value z. It is important to apply the standardization individually on each numerical feature column, as well as on each time-series. Otherwise, extreme magnitudes in different time-series or features can skew the mean and variance parameters, making the scaling itself redundant. After training the model, the values are descaled, by using the associated μ and σ parameters.

One-hot Encoding One-hot encoding transforms categorical variables into a binary matrix format where each category becomes a separate feature column containing only 0s and 1s. This is essential for deep learning models that require numerical inputs and cannot directly process categorical data, preventing the algorithm from incorrectly interpreting ordinal relationships between categories that do not exist. One-hot encoding is applied to all categorical variables in the data, that are not binary, like type of building.

5.2 Experiments

This section provides explanations and results of the multiple experiments performed to benchmark the ML models on time-series forecasting. The experiments are grouped by the resolution of the data they are performed on, daily or hourly data. Apart from evaluating the forecasting accuracies of the different ML models, further experiments are executed. The influence of the amount of training data and the accuracy of future covariates on prediction accuracy are investigated as well.

Baseline Hong et al. [111] argue that a forecasting model needs to be compared against a simple benchmark. As a baseline, the assumption that the value for time step t is going to be the same as the value for time step $t - 1$ is implemented. This is a version of the similar day method, which is defined as using a consumption value from a similar day or hour in the historical data as prediction. This method is still used by many system operators [111]. Similar can be defined as the same day a week before or the same hour a day ago. We choose to take the consumption value of one day or one hour ago, as it is most similar in the context of time. This baseline model is used by Bayer et al. [58] as well.

Model	Batch Size	Dropout	Epochs	Hidden Size	Heads	Search Method
FCN	16–120	0.05–0.2	20–80	35–140	N/A	Bayesian
LSTM	16–120	0.05–0.2	20–80	35–140	N/A	Bayesian
TE	{32, 64}	0.1	{10, 20, 25, 30}	256	4	Manual
TFT	{64, 128, 256}	0.1–0.9	{20, 25, 30}	10–320	{1, 4}	Random
XLSTM	{32, 64}	0.1	{10, 20, 25, 30}	256	4	Manual

Table 5.1: Hyperparameter Search Space Configurations by Model

Hyperparameter Optimization All models have a considerable amount of hyperparameters. A well-chosen hyperparameter set can decide the performance quality of a model on the dataset. An overview of hyperparameter search spaces by model is given in Tab. 5.1.

For the smaller models FCN and LSTM, the Weights and Bias Sweep[1] functionality is used to find a suitable hyperparameter set. Additional to hyperparameters shown

[1] https://docs.wandb.ai/guides/sweeps/

in Tab. 5.1, length of future variables and length of input variables was tested, corresponding to maximum input feature lengths and outputs.

The TFT repository [97] provides a script for finding an optimal hyperparameter configuration. It randomly chooses hyperparameters from a pre-defined search space, which is shown in Tab. 5.1. Specific hyperparameter search spaces for the TFT are minibatch size out of {64, 128, 256}, learning rate from {0.0001, 0.001, 0.01}, and maximum gradient normalization value out of {0.01, 1.0, 100.0}. The script was run for twelve hours to find the best parameters.

For the xLSTM and TE the hyperparameters are manually chosen. The length of input features is set to the maximum, either seven days or 72 hours.

Overall, different parameter sets are tested for a certain amount of time. The model that performs best is chosen and the respective hyperparameters are used when further training that kind of model. The parameter set also includes on which features the model was trained. There are three options: on all of them (all), on all the features but the building information (no building), or on only the past consumption values (only diff). The presented evaluation results belong to the model that performed best out of all the tested ones.

Hardware The models are trained on Central Processing Units (CPUs) and Graphics Processing Units (GPUs) in the experiments, depending on their complexity. The CPU that is used is an Intel Core i7-1165G7. The GPU used is a NVIDIA L40S.

5.2.1 Daily Heat Data

The models are benchmarked on performing a one day and a seven day prediction of heat consumption.

Model	nRMSE	RMSE	MSE	MAE
Baseline	0.1609	808.16	653,120.8	248.19
FCN	0.1383	624.35	389,810.69	210.52
LSTM	0.1238	562.47	316,378.03	**168.68**
TE	0.1371	629.01	634,833.9539	217.94
TFT	0.124435	622.6761	387,725.6241	213.1
xLSTM	**0.1184**	**551.29**	**303,921.16**	193.52

Table 5.2: Evaluation results for one day prediction. RMSE and MAE are in kWh.

The evaluation results for the next day prediction can be seen in Tab. 5.2. Details on the model configurations can be found in A.2. The results show that, overall, the xLSTM performs best on the task of one day prediction. With a RMSE of 551.29 kWh it beats the second best model, the LSTM, by 1.9%. Yet, the best MAE score (168.68 kWh) is achieved by the LSTM. All models are able to beat the baseline. A forecast for an example building is displayed in Fig. 5.2 for each model.

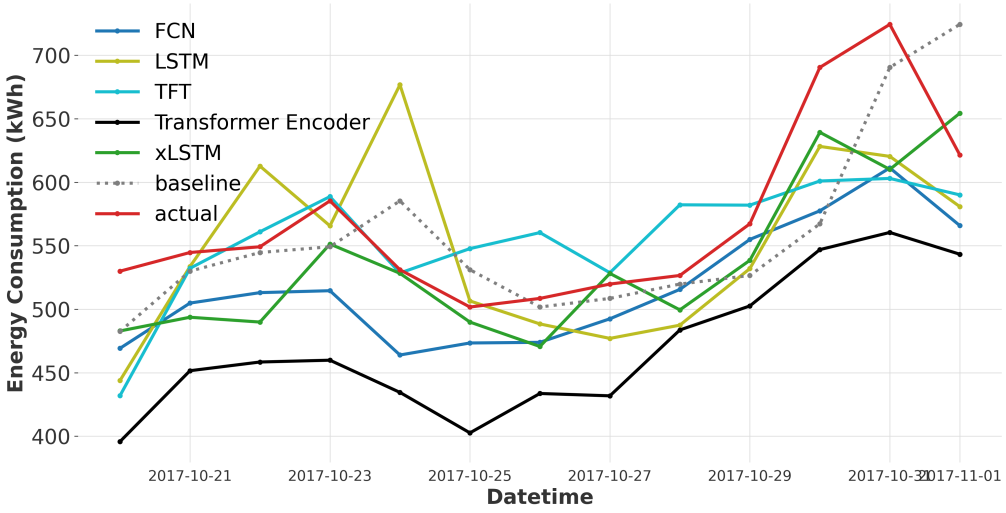

Figure 5.2: One day forecasts for all models for one building in the test set.

The xLSTM achieves the lowest RMSE of 551.29 kWh, this error is averaged over all buildings in the dataset. One can calculate the Rooted Squared Error (RSE), of which the RMSE is the mean, which is expected to be larger the higher the consumption of the building is. Therefore, the RSE is normalized by the average consumption of the building in the test set, which yields the normalized RSE (nRSE). For each of the 70 buildings in the dataset, the nRSE distribution is plotted in Fig. 5.3. The plots are sorted by average nRSE of the test series belonging to one building, ascending from left to right. The distributions are, except for a few outliers, on the same level. This means that the xLSTM is able to extract magnitude information from the provided input features, and use it to make accurate forecasts for buildings of different sizes.

Tab. 5.3 shows evaluation results for the seven day forecast experiment. Details on the single models used can be found in A.2. The best RMSE and MSE score can be achieved by the FCN (719.4385 kWh, 520,106.4685), which outperforms more complex

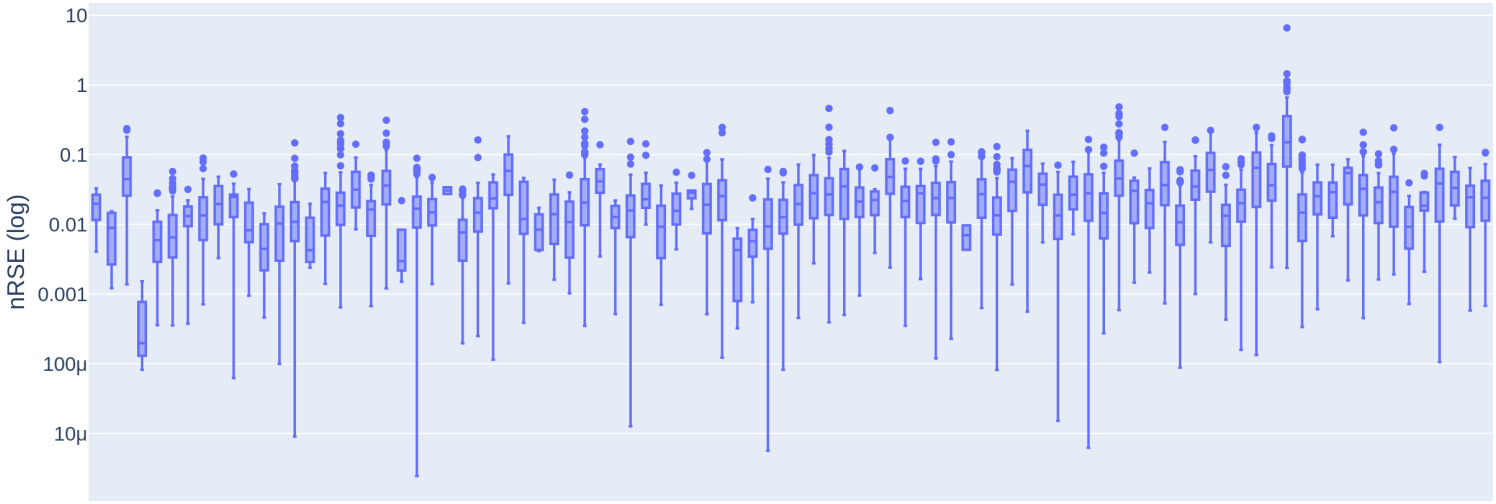

Figure 5.3: Box plot of nRSE (logarithmically scaled) for xLSTM for one day prediction for each of the 70 buildings in the dataset.

Model	nRMSE	RMSE	MSE	MAE
Baseline	0.2701	1399.18	1,958,064.18	524.3031
FCN	0.1527	**757.08**	**573,809.51**	247.95
LSTM	0.16098	797.77	636,612.42	250.898
TE	0.1563	809.86	656,077.36	288.47
TFT	**0.1508**	762.503	582,491.84	255.78
xLSTM	0.1577	790.498	627,413.24	**245.898**

Table 5.3: Evaluation results for seven day prediction. RMSE and MAE are in kWh.

models TFT (762.503 kWh, 582,491.8370) and xLSTM (790.498 kWh, 627,413.2446) on forecasting seven days ahead on those metrics. Yet, the TFT has the lowest nRMSE score with 0.15076, and the xLSTM the lowest MAE score of 245.1648 kWh.

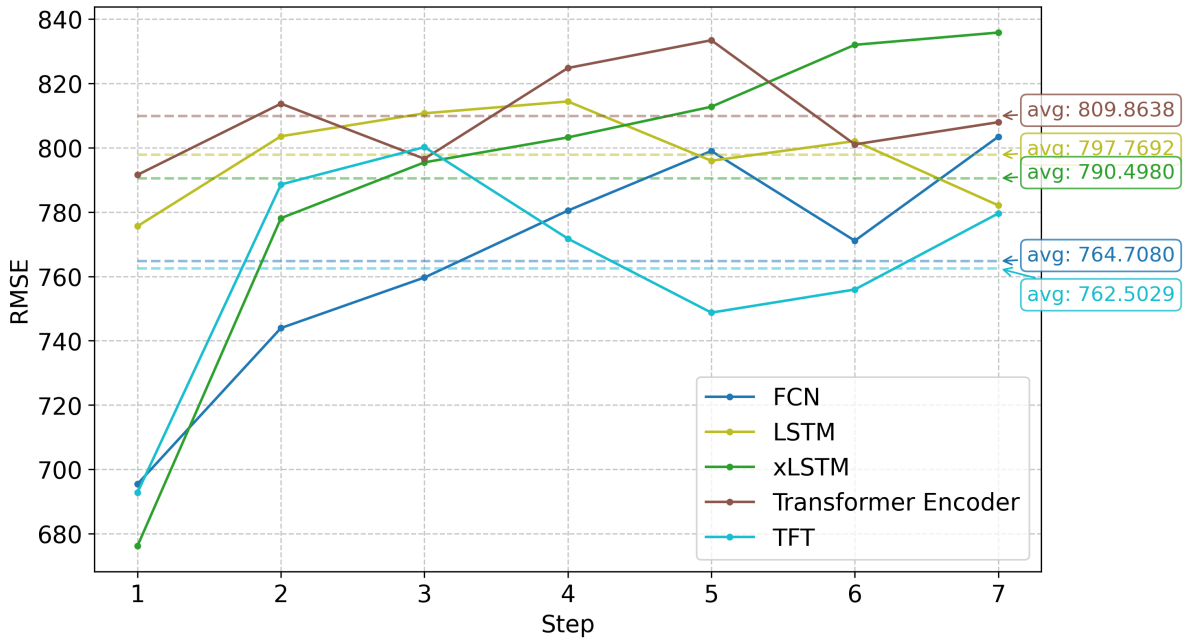

Figure 5.4: RMSE (kWh) for each step of the forecast horizon of 7 days per model.

The multi-step forecast can be analyzed further by investigating the RMSE for each step of the forecast horizon. Fig. 5.4 illustrates how each model performs on average for each step. It can be seen that the first output's error is far below its mean for xLSTM, TFT, and FCN, but increases drastically for the second forecasting step. The FCN and xLSTM are the only models that can stay below their mean RMSE for two steps, the FCN is the only model surpassing their mean RMSE after the third forecasting step. Yet, the models accuracy does not decrease linearly over time, for example the TFT's second best average RMSE is reached for step five. The TE and LSTM have the highest average RMSEs, but their RMSE per step does not change as drastically as for FCN, xLSTM, and TFT over the span of seven forecasting steps.

The RMSE of the FCN model on the task of seven day forecasting can be further analyzed by weekday, shown in Fig. 5.5. The error is averaged per weekday, showing that the days Monday until Wednesday have the lowest error of around 430 kWh. These days have the highest average consumption of heat load in the dataset. The rest of the week, Thursday until Sunday, the RMSE is higher with about 780 kWh on

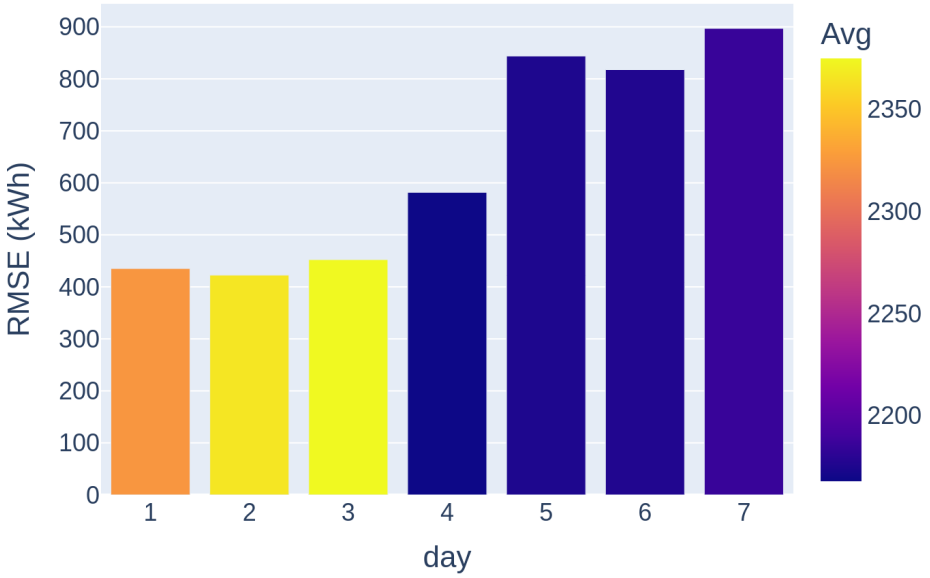

Figure 5.5: Average RMSE on the day of the week for the FCN model for seven day prediction, which has an overall RMSE of 757.08 kWh on this task. The color indicates how high the average energy consumption on this weekday in the test dataset was.

average. These days have a lower average heat load consumption, compared to the beginning of the week.

5.2.2 Hourly Heat Data

The models are evaluated on the hourly data for forecasting three hours and 24 hours. Forecasting of three hours can be helpful for grid operators to make very short-term decisions on the intraday market. A 24-hour prediction is useful for estimating the course of the next day in more detail, when performed at the start of the day.

Model	nRMSE	RMSE	MSE	MAE
Baseline	0.6981	35.6050	1,267.7481	15.6114
FCN	0.6485	22.4679	504.9629	10.5686
LSTM	0.6628	22.7919	519.5272	10.3654
TE	0.6685	22.9671	527.4964	11.0534
TFT	**0.5221**	21.8016	475.5771	**9.1612**
xLSTM	0.5535	**19.8792**	**395.2333**	10.3044

Table 5.4: Evaluation results for three hour prediction. RMSE and MAE are in kWh.

Results for the forecasting of three hours can be seen in Tab. 5.4. An overview of the model hyperparameters can be found in A.6. The results show how the TFT and xLSTM are outperforming the other models on three hour prediction. The TFT is able to achieve the lowest nRMSE (0.5221) and MAE (0.1612), while the xLSTM reaches the minimal RMSE (19.8792 kWh) and MSE (395.2333). It beats the second best model, the TFT, by 8.8% on the RMSE metric.

Model	nRMSE	RMSE	MSE	MAE
Baseline	**0.5156**	24.9731	623.6585	11.6490
FCN	0.7405	24.8871	619.5878	11.4869
LSTM	0.7188	24.9085	620.5474	11.3159
TE	0.9728	29.1964	852.5017	13.8039
TFT	0.5820	24.2263	587.6394	**10.5753**
xLSTM	0.5723	**21.4712**	**461.1092**	11.4327

Table 5.5: Evaluation results for 24-hour prediction. RMSE and MAE are in kWh.

The results of the 24-hour prediction are shown in Tab. 5.5. A more detailed view of the hyperparameters for each model are available in A.7. Again, the xLSTM is able to outperform all other tested models for RMSE (21.4712 kWh) and MSE score (461.1092).

An example of 24-hour predictions of the xLSTM for a time-series in the test set is given in Fig. 5.6. It beats the second best model, TFT by about 11,4% on the RMSE metric. Still, the TFT achieves the lowest MAE of 10.5753 kWh. In this experiment, no model was able to beat the baseline for the nRMSE (0.5156), which means that this experiment was particularly difficult, or that the baseline works especially well for 24-hour prediction. Also, the MAE score of the baseline (11.6490 kWh) could only be beaten by FCN (11.4869 kWh), LSTM (11.3159 kWh), and TFT (10.5753 kWh). The TE is not able to beat the baseline in this experiment for any metric.

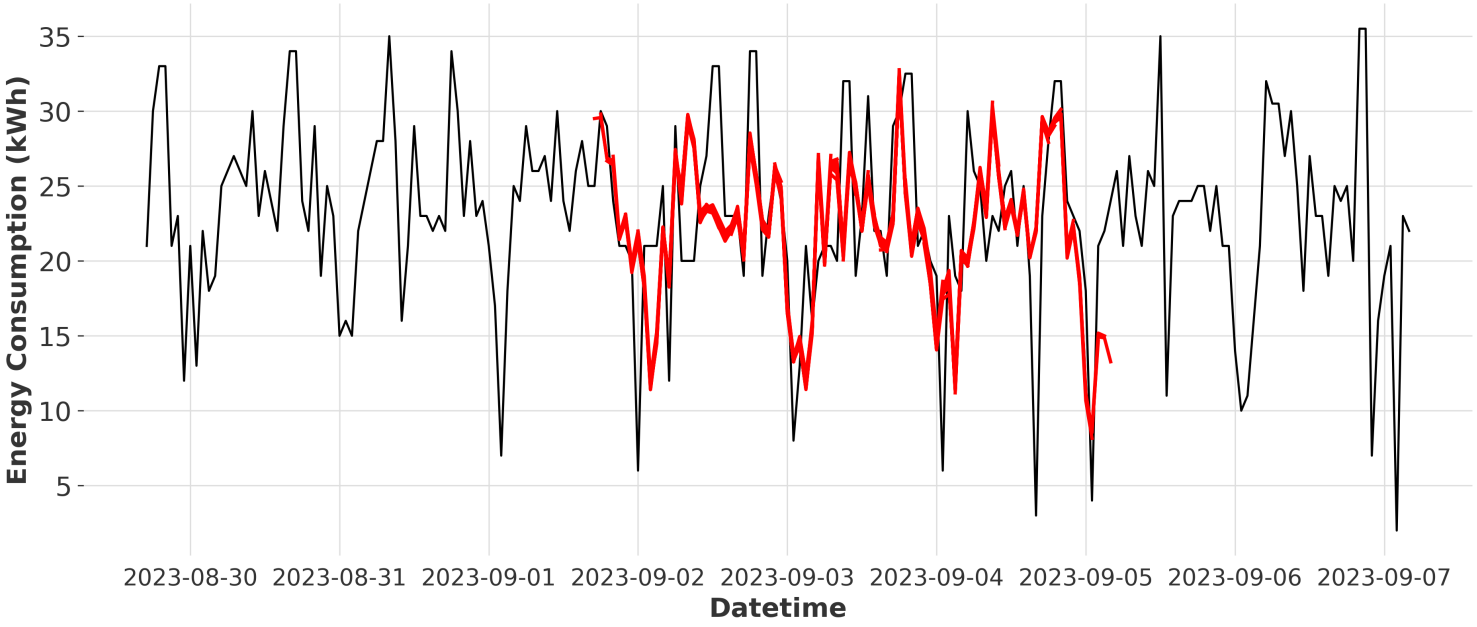

Figure 5.6: Example of 24-hour forecasts of the xLSTM for one time-series of the test set. The black line shows the recorded values, while the red lines show the 24-hour forecasts.

The xLSTM for 24-hour prediction achieves a RMSE of 21.4712 kWh, the value is averaged over all test-series in the dataset. The per-building analysis is visualized in Fig. 5.7, where the distribution of the nRSE is drawn for each series in the test dataset individually. For the 24-hour prediction, one can see that the distributions of the nRSE are more heterogenous than when predicting seven days. There is no immediate correlation between increasing size of the building, determined by the average consumption in the test set, and the nRSE distributions.

The performance of the xLSTM on 24-hour prediction can also be analyzed by hour of day. The result of grouping the test set by hour and calculating the nRSE is

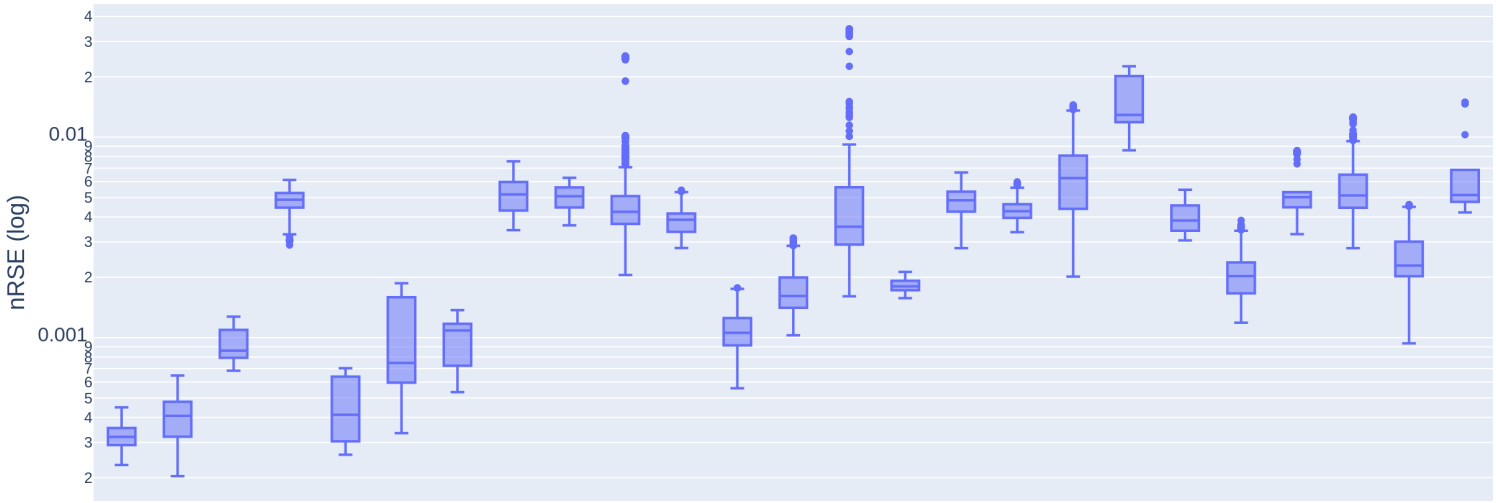

Figure 5.7: Distribution of nRSE for each time-series in the hourly test dataset, logarithmically scaled. The series are sorted by average energy consumption, ascending from left to right.

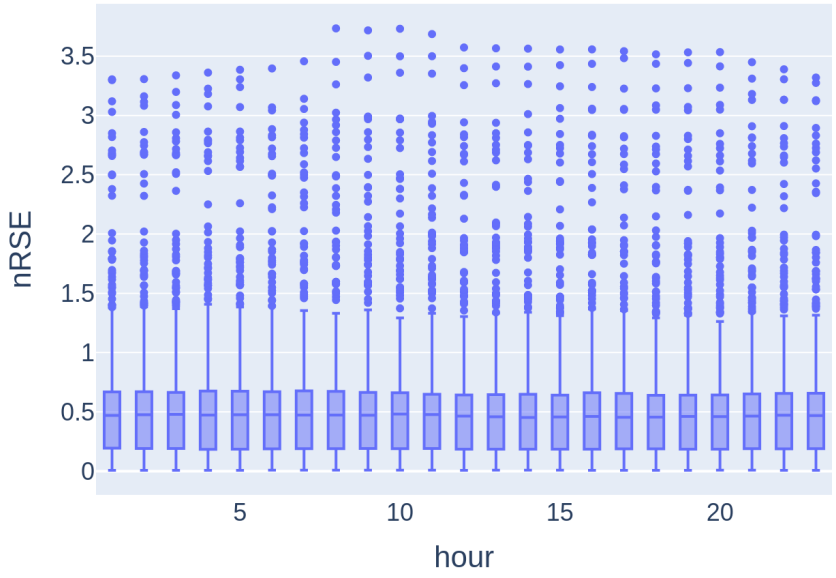

Figure 5.8: Distribution of nRSE per hour of the day of the xLSTM model for 24-hour prediction.

displayed in Fig. 5.8. The means of the distributions of the nRSE are about 0.5 for every hour in the day. Yet, there are a lot of outliers larger than 1.5 for every hour in the day as well.

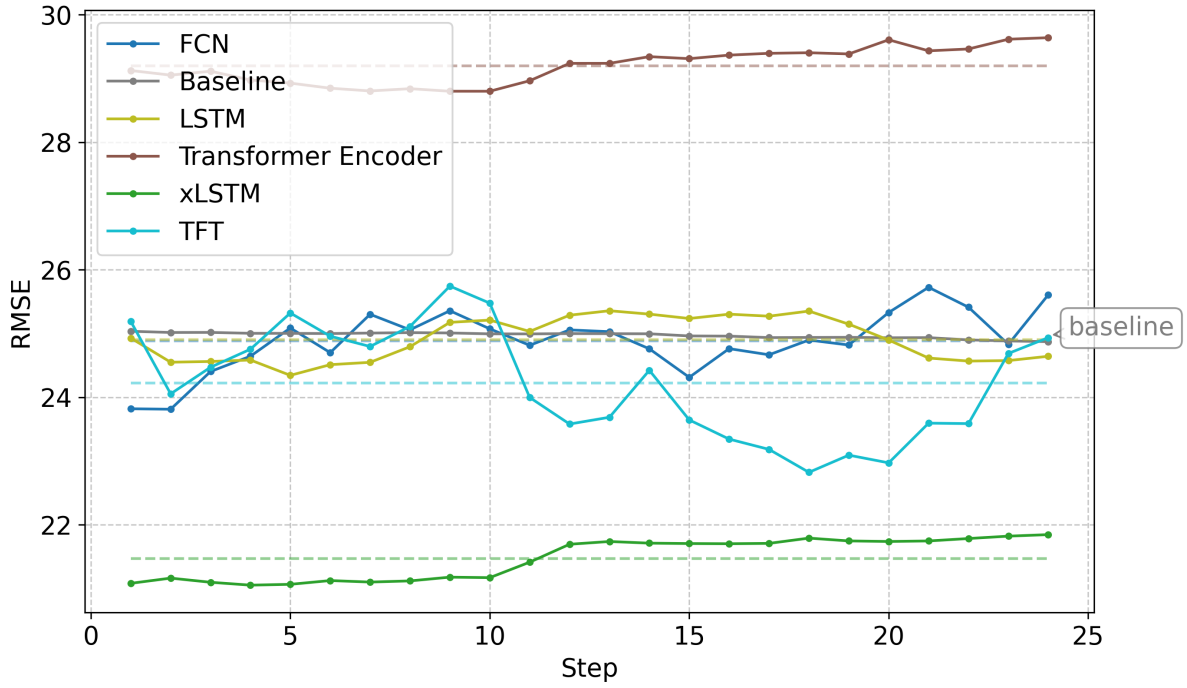

Figure 5.9: Average RMSE (kWh) per 24 forecasting steps for each model.

The performance of all models over the span of the forecasting horizon of 24 hours is depicted in Fig. 5.9. The mean RMSE is crossed from better to worse by TE, xLSTM, and LSTM for step twelve, eleven, and nine respectively. This can indicate the models' inability to create forecasts longer than those step sizes. TFT, however, shows its best performances for steps 11-22. The FCN's predictions accuracy also improves for steps 14-15, but then worsens again. Overall, the xLSTM is able to outperform the other models for every step in the forecasting horizon for the RMSE, while the TE shows the worst performance of all models.

5.2.3 Influence of Amount of Training Data

In another experiment, the influence of the amount of training data is investigated. It will create insights about the model performance under resource constraints. Also, the relationship between data amount and model prediction accuracy will be investigated.

This understanding is crucial for developing reliable ML solutions, even when the access to training data is limited.

The timing of model implementation represents a critical decision point in the deployment lifecycle. Early implementation enables more immediate realization of benefits, such as reduced heat load consumption. However, this advantage must be weighed against the potential reduction in forecasting accuracy due to limited training data availability. Therefore, it is valuable to know about the models ability to handle shorter training time-series.

To realize this, 20% and 50% of the beginning of the training time-series is removed, to simulate a later start of recording sensor readings. The amount of data for each resolution format can be found in Tab. 5.6. The models are trained with the exact same parameters as during experiments in Sec. 5.2.1 and Sec. 5.2.2.

Percentage of Data	Daily	Hourly
50%	20.450	50.843
80%	33.441	86.950
100%	42.147	105.274

Table 5.6: Training data point count for reduced data evaluation for each format of resolution.

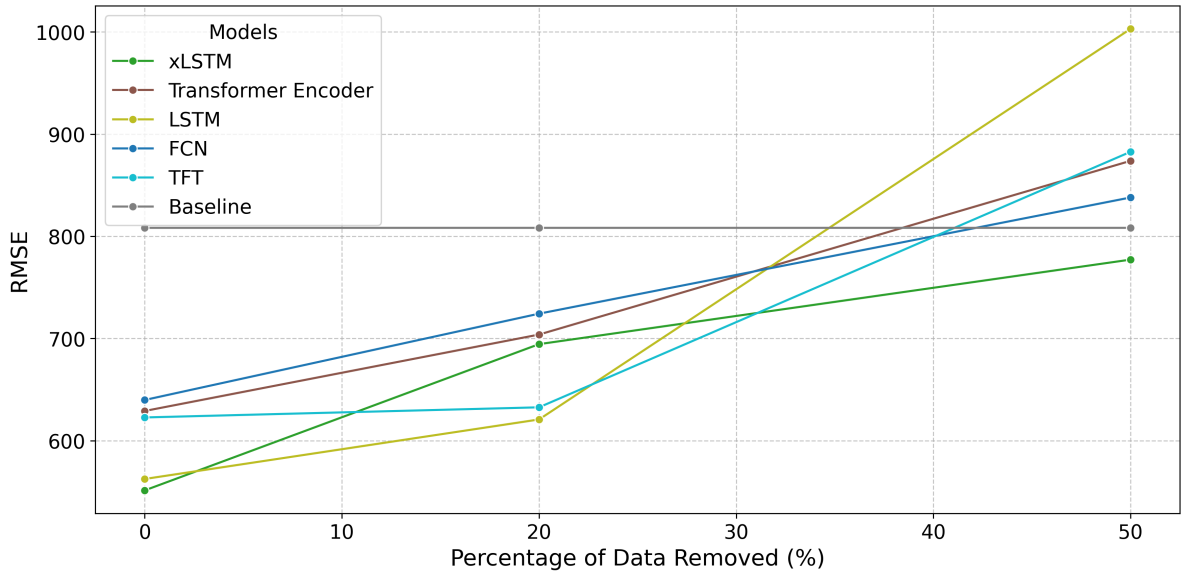

Figure 5.10: RMSE (kWh) results for the best models for prediction of one day with all training data available, 20% removed, and 50% removed training data.

One Day Prediction Fig. 5.10 illustrates the RMSE of the models for one day heat load prediction under different data reduction scenarios. The graph plots the RMSE against the percentage of data removed (0%, 20%, and 50%), showing how prediction accuracy degrades as training data is reduced.

When examining the full dataset (0% reduction), the xLSTM and LSTM models demonstrate superior performance with the lowest RMSE (551.29 kWh and 562.47 kWh), indicating their capability in capturing the temporal dependencies crucial for accurate day-ahead heat load forecasting. The TE, FCN, and TFT models show moderately higher error rates, while the baseline model consistently maintains an RMSE of 808.16 kWh regardless of data reduction.

Looking at results for 20% data reduction, several notable patterns emerge. The LSTM, FCN, and TE models exhibit resilience, with their error increasing at a similar rate. In contrast, the xLSTM model begins to show vulnerability to the initial data reduction, with its error increasing more rapidly. Interestingly, the TFT model maintains nearly consistent performance between 0% and 20% data reduction, suggesting it possesses robust feature extraction capabilities even with reduced datasets for day-ahead predictions.

At 50% data reduction, the performance degradation becomes more pronounced across all models. The LSTM model experiences the most dramatic deterioration, with its RMSE increasing by approximately 80% from the full dataset scenario, reaching about 1120 kWh. In contrast, the xLSTM model maintains the lowest RMSE (approximately 775 kWh) even with half the training data removed, demonstrating its superior generalization capabilities in data-constrained environments. It is the only model that can beat the baseline with only training on half of the dataset.

Seven Day Prediction The data reduction experiment results for seven day ahead forecasting are shown in Fig. 5.11. On the complete training dataset, the FCN demonstrates the best performance with the lowest RMSE of 757.08 kWh, closely followed by the TFT with an RMSE of 762.503 kWh. The xLSTM and LSTM models show similar performance levels (790.498 kWh and 797.77 kWh), while the TE starts with a higher error (809.86 kWh). The baseline model maintains a consistently high RMSE of about 1400 kWh across all data reduction scenarios.

At 20% data reduction, some notable patterns can be observed. The LSTM shows a slight improvement in performance, with its RMSE decreasing to approximately 770 kWh. This counterintuitive result might suggest that some redundant or noisy data

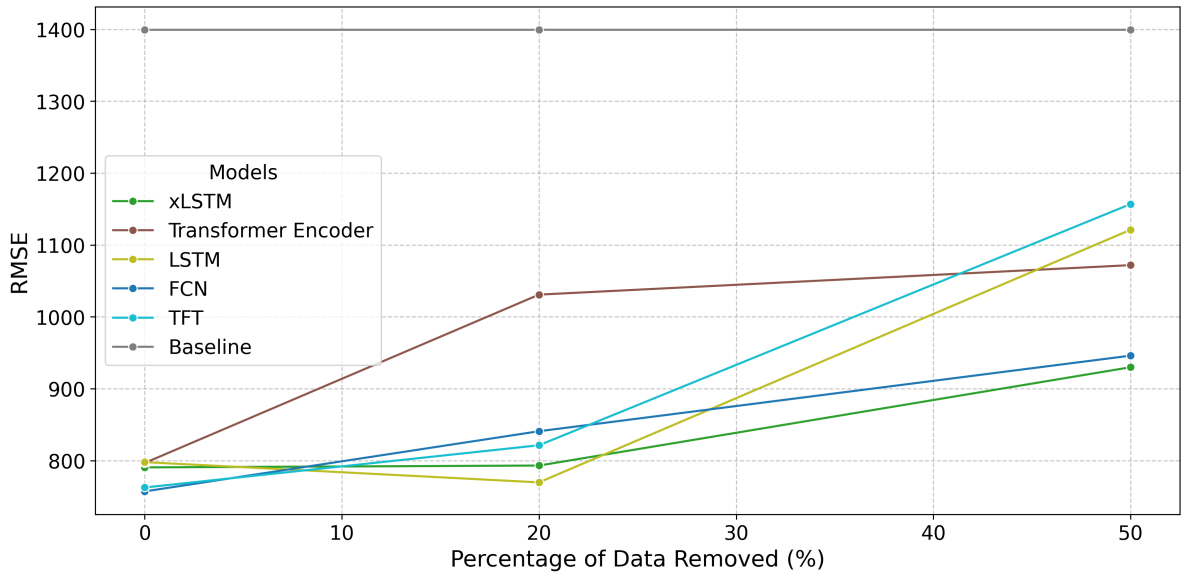

Figure 5.11: RMSE (kWh) results for the best models for prediction of seven days with all training data available, 20% removed, and 50% removed training data.

points were eliminated, actually enhancing the model's generalization capability for seven day predictions. The xLSTM maintains relatively stable performance with only a slight increase in RMSE. In stark contrast, the TE exhibits a dramatic degradation, with its RMSE increasing to about 1030 kWh, indicating a particular sensitivity to data reduction for seven day forecasting. Also, FCN and TFT, the best models on 100% of the training data, fall slightly behind LSTM and xLSTM, with a RMSE of 769.66 kWh and 821.29 kWh, respectively.

When implementing 50% data reduction, all models show performance deterioration, though at varying rates. The TFT and LSTM experience the most severe degradation, with their RMSEs increasing to above 1100 kWh, which makes them the worst two performers on this data split. The TE continues its upward trend to an RMSE of right below 1100 kWh, although with a much slower rate than before. Notably, the xLSTM and FCN demonstrate the greatest resilience to severe data reduction, maintaining the lowest RMSE values of 929.83 kWh and 945.97 kWh, even with half the training data removed. In this experiment, all models are able to beat the baseline being trained on only half of the training data.

Three Hour Prediction The short-term prediction of three hours and the influence of data reduction is displayed in Fig. 5.12. On all the training data, the xLSTM

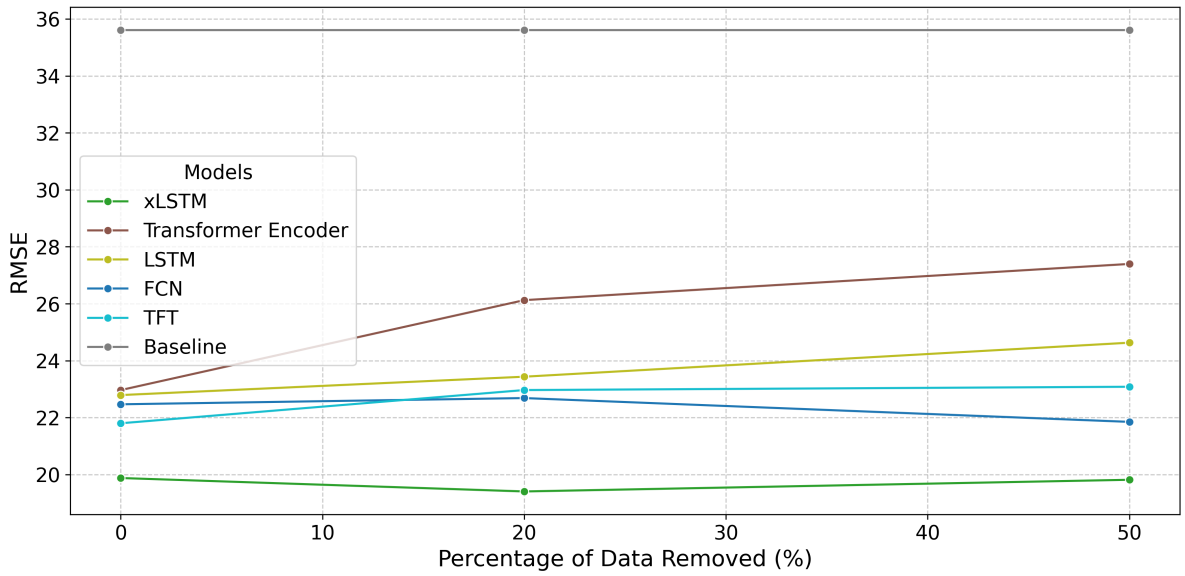

Figure 5.12: RMSE (kWh) results for the best models for prediction of three hours with all training data available, 20% removed, and 50% removed training data.

demonstrates superior performance with the lowest RMSE of about 19.88 kWh, while all other models achieve comparable RMSEs around 22-23 kWh. The baseline model shows a consistently high RMSE of 35.61 kWh across all scenarios.

When removing 20% of the data, the performance of the xLSTM even increases with an RMSE of 19.41 kWh. Again, it is possible that redundant or misleading data was removed, which caused the xLSTM to train better. The accuracy of TFT, FCN and LSTM almost remains the same when removing 20% of training data, while the TE deteriorates further to an RMSE of 26.13 kWh.

At 50% data reduction, the models display resilience for this short prediction horizon. The xLSTM maintains its superior performance with virtually unchanged RMSE, demonstrating exceptional robustness to data limitations for this training szenario. The FCN model shows an unexpected improvement, with its RMSE decreasing to 21.85 kWh, potentially indicating that the reduced dataset better captures the essential patterns for three hour predictions. The TFT model maintains stable performance with minimal error increase, while the LSTM experiences a moderate increase to an RMSE of 24.63 kWh. The TE, which was initially severely affected by the data reduction, shows a similar decrease in performance as the LSTM, with its RMSE reaching 27.397 kWh. Again, the baseline is constantly outperformed by all models.

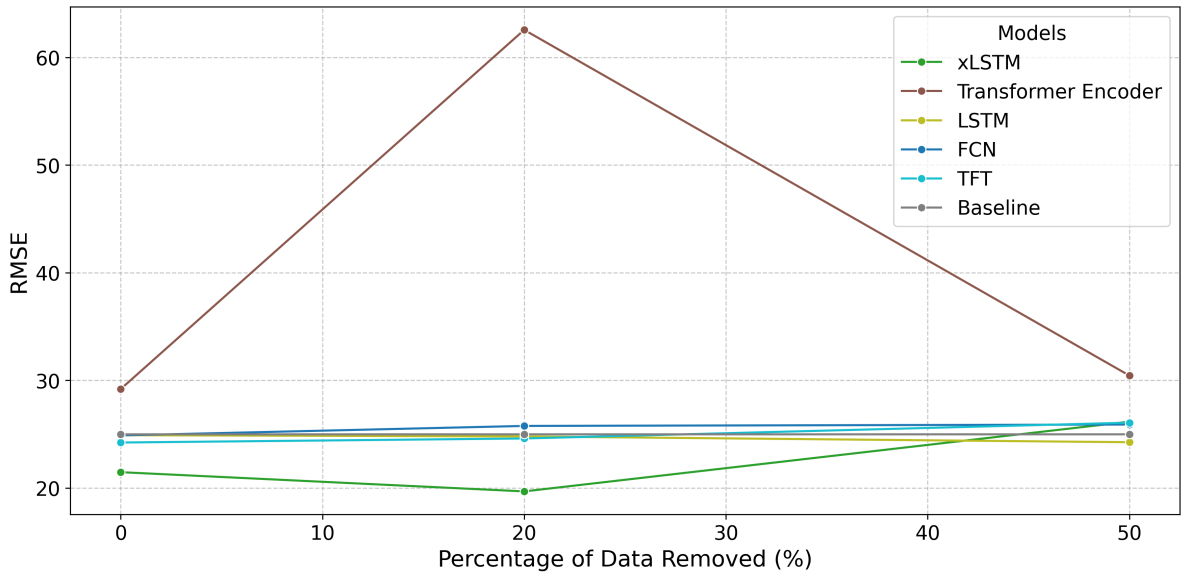

Figure 5.13: RMSE (kWh) results for the best models for prediction of 24 hours with all training data available, 20% removed, and 50% removed training data.

24 Hour Prediction The final scenario tests data reduction for the task of 24-hour prediction, shown in Fig. 5.13. The most notable occurrence is the jump in performance by the TE. From an RMSE of 29.2 kWh, it increases to about more than twice as much when removing 20% of the training data. It is also not able to beat the baseline for any amount of training data. This indicates that the TE architecture is not advanced enough to perform a one-shot prediction of 24 steps for our training dataset. Yet, it is also remarkable how well the baseline (24.97 kWh) performs in this experiment, showing results that are comparable to those of the FCN, LSTM, and TFT. Again, the best initial model is the xLSTM with a RMSE of 22.28 kWh, which improves to 19.69 kWh when removing 20% of the training data. For all other models but the TE, the reduction to 80% of training data has only a small negative impact on their RMSE. This is also true when removing 50% of the training data, which only has a large negative effect on the training ability of the xLSTM. Its RMSE increases to 26.16 kWh, worse than LSTM, FCN, TFT, and baseline. The only model able to beat the baseline on 50% of training data is the LSTM with an RMSE of 24.25 kWh.

The analysis of four different prediction horizons (three hour, 24-hour, one day, and seven day) under various data reduction scenarios (0%, 20%, and 50%) reveals several consistent patterns and important insights for heat load forecasting in data-constrained environments.

Summary The xLSTM model emerges as the most robust architecture across all prediction horizons and data reduction levels. It consistently maintains the lowest or near-lowest RMSE values, even when training data is reduced by 50%. Particularly noteworthy is its ability to occasionally show improved performance with moderate data reduction (by 20%) during three out of the four experiments, suggesting that it effectively captures essential patterns while being less susceptible to noise in the training data. This exceptional resilience makes xLSTM the recommended model for heat load forecasting applications where data availability may be limited or variable.

In stark contrast, the TE demonstrates the most volatile performance across different scenarios. While already outperformed with full training data, it shows extreme sensitivity to data reduction, particularly for the 24-hour prediction horizon where its RMSE more than doubled at 20% data reduction before partially recovering at 50% reduction. This non-monotonic behavior indicates complex interactions between the model architecture and specific data characteristics, indicating that the TE is a risky choice for applications where data availability cannot be guaranteed.

The traditional LSTM model shows good performance with complete datasets but degrades more rapidly than xLSTM as data is reduced, especially for the experiments on daily data. The FCN and TFT models generally maintain stable performance with moderate data reduction but show varying degrees of degradation at higher reduction levels depending on the format of the data. For hourly resolutions, the FCN and TFT remain a stable performance throughout data reduction steps, but not for the daily data. This indicates that these models can, in general, train better on larger amounts of data.

In practical solutions, reducing the data by 20% means removing data worth 8,706 days for the daily scenario, and 763.5 days for the hourly one. This is at least two years of data, spread to different energy meters. Still, the models, especially the xLSTM, perform reasonably well. This shows the importance of an early deployment of even a simple model like the FCN.

5.2.4 Influence of Accuracy of Future Weather Data

In the daily data forecasting experiments, FCN, LSTM, and TE improved when training with future covariates. Part of those features were weather data recordings, which are not available in real-life scenarios. Inspired by Potočnik et al. [21], we conduct an experiment that investigates the differences between testing the models

on weather forecast data versus using actual recorded future meteorological measurements. While weather forecasts provide valuable predictive information, they inherently contain noise and uncertainties that may impact model performance, especially when the model was trained on actual recorded weather data. Overcoming this gap is particularly important in practical applications, where only weather forecasts are available. This experiment addresses the question: How does model performance differ when tested with noisy forecast data versus actual recorded weather data?

For simulating weather forecasts, instead of recordings, one of the generalized Ornstein-Uhlenbeck processes [112] is used. These processes are usually used for financial modeling. The simulation process is defined as follows:

$$dW_t = \sqrt{dt} \cdot \mathcal{N}(0, 1) \tag{5.1}$$

$$X_t = X_{t-1} + \theta \cdot (\mu - X_{t-1}) \cdot dt + \sigma \cdot dW_t \tag{5.2}$$

, where t ranges from 1 to $N - 1$. X_t is the simulated feature at timestep t, θ is the rate of mean reversion, μ is the long-term mean of the process, σ is the volatility parameter, and dW_t is a standard Brownian motion [112]. The parameters are set to $\theta = 0.7$ and $\sigma = 0.3$ for the simulations. N is either one or seven, depending on the forecasting horizon. The mean μ is the mean of the single variable over all timesteps in the dataset. For each recorded weather feature in the dataset, N values are simulated and used to replace the actual future covariates. The average RMSE between the recorded and simulated weather features is 31.64, more details are given in A.8.

Model	Prediction Horizon	RMSE	RMSE + Noise	Difference (%)
FCN	one day	639.77	752.26	17.58
LSTM	one day	638.82	621.55	-2.70
TE	one day	731.49	700.39	-4.25
FCN	seven days	753.33	1466.21	94.63
LSTM	seven days	786.36	1381.29	75.65
TE	seven days	836.03	1303.89	56.08

Table 5.7: Comparison of model RMSE (kWh) for testing with recorded future weather data (RMSE) and testing with simulated future weather data (RMSE + Noise)

Results of the experiment can be seen in Tab. 5.7. When training the FCN on the task of one day prediction, the RMSE increases by about 20% when testing on simulated future covariates. Interestingly, the LSTM's and TE's RMSE decreases for this type of prediction. This could mean that the FCN relies heavier on the future covariates

for making an accurate prediction, than the sequential models. This changes for the task of forecasting seven days, where all models' performances drastically decrease when being tested on simulated future weather data. The RMSE of the FCN almost doubles, also surpassing the baseline RMSE of 1399.18 kWh. These results suggest that the future weather data was found essential for making accurate predictions over longer forecasting horizons during training, thus the decrease in performance for testing with simulated data. Therefore, it is important to feed the model forecast data during training as well, when training for predicting a long forecasting horizon.

5.3 Time and Memory Complexity Comparison

Neural network architectures vary significantly in their computational demands, making the understanding of their time and memory complexity crucial for effective model selection in practical applications like heat load forecasting. It is important that the training effort of a model is in relation to the performance gain it provides. This section gives a comprehensive comparison between the four neural network architectures: FCNs, LSTMs, xLSTMs, and Transformers. Analyzing their theoretical complexity bounds, empirical runtime performance during experiments discussed in Sec. 5.2, and memory requirements, aims at providing insights for making informed decisions when choosing an architecture for heat load forecasting.

5.3.1 Theoretical Complexity

Computational Complexity Comparison The computational complexity of FCNs is primarily determined by the number of operations required during the forward pass. For a network with L layers, the time complexity of the forward pass is $O\left(\sum_{i=0}^{L-1} k_i k_{i+1}\right)$, where k_i represents the number of neurons in layer i and the summation runs from $i = 0$ (input layer) to $i = L - 1$ (output layer). This quadratic relationship between adjacent layer sizes means that FCNs can become computationally expensive as the network width increases, especially for deep architectures with many neurons per layer. The FCN used in the experiments discussed in Sec. 5.2 only uses one layer, resulting in $L = 1$.

Unlike sequential models, FCNs process inputs simultaneously rather than temporally. This makes FCNs faster for fixed-size inputs but inefficient for variable-length sequences. While sequential models use parameter sharing across time steps, FCNs

excel in parallelization on GPUs. FCNs avoid the vanishing gradient issues of basic RNNs, but are not equipped with structures that can model sequential dependencies critical for time-series analysis.

The standard LSTM architecture has a computational complexity of $O(W)$, where W represents the number of weights. More precisely, $W = KH + KCS + HI + CSI$, with K the number of output units, C is the number of memory cell blocks, S the size of the memory cell blocks, H the number of hidden units, and I the number of units foward connected to memory cells [10]. Therefore, the computation time of the LSTM is linear in the number of weights, but the architecture can not run in parallel. The computation time of xLSTM, presented in Sec. 2.1.4.2, is linear as well, and its mLSTMs blocks can be parallelized [14]. This makes it possible to run the computations faster on a GPU. The sLSTM blocks, however, are not parallelizable, but can be run on the GPU typically less than two times slower than mLSTM [14].

The self-attention mechanism is the core component of Transformer models and also their primary computational bottleneck. For a sequence of length n and embedding dimension d, the time complexity of self-attention is $O(n^2 d)$ [12]. This quadratic dependency on sequence length arises from the need to compute attention scores between every pair of tokens in the sequence. When using multi-head attention, each head of the model has the same computational complexity of $O(n^2 d)$, but these can run in parallel, which does not increase the overall complexity. The TFT [31] introduces multiple architectural enhancements which increase computational cost in theory. The VSN compute importance weights for v variables in $O(vd)$ time, but provide an efficiency gain by filtering for the most relevant features. Each GRN layer adds $O(d^2)$ operations. Overall, the Transformer and TFT share the same computational complexity of $O(n^2)$.

Memory Complexity Comparison The primary memory consumption in FCNs comes from storing the weight matrices that connect adjacent layers. For a network with L layers, where each layer i has k_i neurons, the memory complexity for storing all weight matrices is $O\left(\sum_{i=0}^{L-1} k_i k_{i+1}\right)$. In addition to weight matrices, each neuron typically has an associated bias term which requires $O\left(\sum_{i=0}^{L-1} k_i\right)$ of memory. Also, FCNs need to store activation values for each layer, which has the same memory complexity.

The LSTM maintains two primary vectors for each time step: the hidden state h_t and the cell state c_t. For a model with hidden dimension d and sequence length

n, this results in a memory complexity of $O(nd)$. During training, LSTMs need the same amount of memory for storing activations. Overall, LSTMs are more parameter efficient as FCNs and Transformers, due to their ability to share weigths. The fundamental innovation of xLSTM is replacing scalar hidden states with matrix representations for the mLSTMs blocks. The matrix memory has a size of dxd with a dxd update complexity [14]. Yet, these updates can be run in parallel on the GPU.

The memory of Transformer models scales quadratically with input sequence length. The attention mechanism requires storing an attention matrix of size $O(n^2)$. Beyond attention matrices, transformer memory complexity is also influenced by parameter count. These parameters include embedding dimension d, number of layers L, and number of attention heads h. A vanilla Transformer requires approximately $O(4Ld^2 + 2Ld)$ parameters for the self-attention and feed-forward components across L layers [113]. TFTs introduce additional memory requirements beyond standard transformer architectures. They add VSNs, static covariate encoders, LSTM encoder and decoder structures, which all add to the total memory needed for this type of model.

5.3.2 Empirical Performance Analysis

The trainable parameters for each model for either forecasting seven days or 24 hours can be seen in Fig. 5.14. Overall, the FCN has by far the smallest amount of trainable parameters, with 16,027 and 8,277. For the experiments, we chose a rather small implementation, with only one fully-connected layer and one dropout layer. The neuron size is also kept relatively small, with 100 and 137, respectively. The reason that the FCN for 24-hour forecasting is even smaller than that for seven day forecasting is that it is only trained on the past consumption values. Therefore, the model does not need to process the feature time-series, resulting in a slimmer architecture. The next larger model is the LSTM, with 51,007 and 71,296 trainable parameters. This is more than three times the amount of the trainable parameters of the bigger FCN. xLSTM, Transformer, and TFT are by far the biggest models, with all of them having at least two million trainable parameters. The overall largest model is the TFT for seven day forecasting, with 6,025,296 trainable parameters, while the smallest out of the group is the xLSTM for 24-hour forecasting, having 2,153,376 trainable parameters.

Interestingly, the amount of trainable parameters does not stand in correlation with the performance. The FCN for 24-hour prediction has more than eight times less

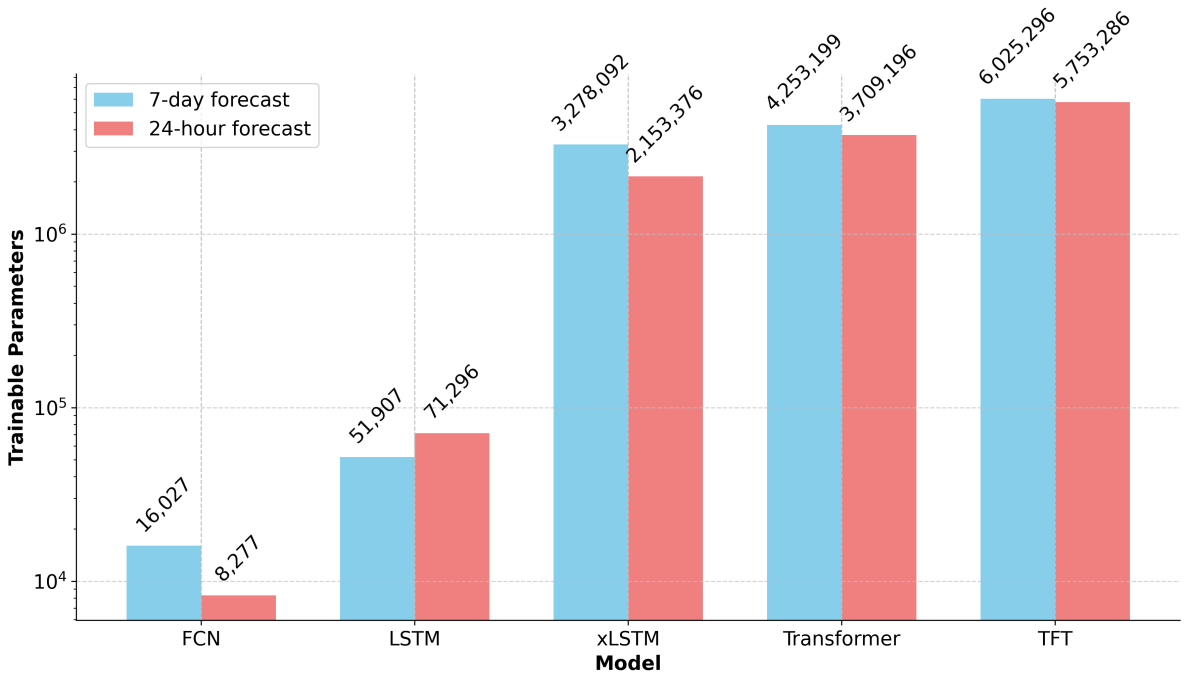

Figure 5.14: Comparison of the number of trainable parameters for each model and the forecasting horizons of seven days or 24 hours. The y-axis is logarithmically scaled.

trainable parameters than the LSTM, and is able to achieve a slightly lower RMSE. The TFT, being the largest model for 24-hour prediction, can only achieve a comparable RMSE to FCN and LSTM, but has the lowest MAE score of all. The FCN for seven day prediction outperforms all other models measured by the RMSE, despite being the smallest one. As a result, the FCN seems to be the best choice for resource constrained environments, while still achieving top results.

Model	one day forecast	seven day forecast	three hour forecast	24 hour forecast
FCN	1m 34s	55s	1m	1m 11s
LSTM	5m 59s	9m 43s	35m 19s	1h 5m 25s
TE	30m 29s	52m	2h 39m 25s	2m 33s*
TFT	37m 9s	39m	6h 23m	7h 9m
xLSTM	12m 18s*	11m 57s*	9m 54s*	19m 50s*

Table 5.8: Training runtime comparison of time-series prediction models. Runtimes with a * result from trainings on the GPU, rather than the CPU.

Based on the runtime comparison presented in Tab. 5.8, significant variations in computational efficiency among the models can be observed. The FCN demonstrates remarkable efficiency across all forecasting horizons, with training times consistently under two minutes, even on the CPU. This makes FCN particularly suitable for applications requiring rapid model development and deployment.

In contrast, the TFT exhibits substantially longer training times, particularly for hourly prediction, where training duration extends to over six hours. This significant computational cost must be weighed against the potential accuracy benefits when considering TFT for practical applications. Yet, it is an unfair comparison, because the xLSTM was trained on the GPU, which sped up its training significantly. Unfortunately, the TFT was not able to train on the GPU due to hardware restrictions. A visualization of xLSTM training time on the CPU compared to training a subset of the other models on CPU can be found in A.7.

The LSTM shows moderate performance for shorter forecasts (one day: 5m 59s, seven days: 9m 43s) but experiences significant slowdown for the three hour forecast (35m 19s) and especially the 24-hour forecast (1h 5m 25s). This pattern complies with the assumption that LSTM's computational demands increase with the complexity of the temporal patterns it needs to learn.

For practical applications, these findings suggest that, when computational resources are limited or rapid model development is required, the simple FCN offers an excellent balance of efficiency and performance. For applications where forecast accu-

racy is paramount and computational resources are abundant, more complex models like TFT and xLSTM may be justified despite their longer training times. Hardware acceleration can substantially reduce training times for complex models, potentially making them viable for applications where they would otherwise be too computationally expensive. Ultimately, the choice of time-series prediction model should consider both the computational constraints of the application environment and the specific forecasting requirements of the task at hand.

5.4 Discussion

In all experiments, at least one of the tested models was able to beat the baseline. That shows that they are able to learn on the dataset, even when it consists of multiple buildings. The most time and memory efficient model is the FCN, which also achieves top results for seven day prediction, and is competitive during all other experiments. The LSTM is able to outperform all models on the MAE for one day prediction, but, compared to the FCN, is bigger in size and takes longer to train. The attention based-structures, the TE and TFT, are the largest two architectures. Their training takes the longest, but can theoretically be accelerated on the GPU. Despite its size and training effort, the TE performs worst in almost any scenario. The TFT, however, achieves the best nRMSE value for seven day and three hour prediction, as well as the lowest MAE score for hourly predictions. The xLSTM is able to outperform all other models in three out of the four experiments, showing a strong ability to capture knowledge from the daily and hourly datasets. Additionally, it is smaller compared to TE and TFT and could be trained fast on the GPU.

Over the course of the experiments, the xLSTM shows a strong performance regarding the RMSE metric, but is almost constantly outperformed on the MAE metric. This suggests that the xLSTM is good at avoiding large errors, but is making smaller ones throughout its predictions. Yet, the model is trained on the MSE. If the MAE was the preferred metric in a scenario, training the xLSTM on that error might yield different results.

The xLSTM is not trained with future covariates. Yet, it is still able to produce top results in all experiments, even compared to models that see future covariates during training and inference. Adapting the original xLSTM architecture to handle future covariates might improve its performance even further.

The training of the xLSTM often threw an Out-Of-Memory errors on available hardware. These could be avoided by reducing training epochs, but the xLSTM might have yielded even better results, if it had been possible to train it longer.

Smaller models, like FCN and LSTM, do not take as long to train. Therefore, they can undergo a lot of iterations in the hyperparameter search. For the TFT training, a hyperparameter search was conducted as well, but TFT training takes at least 30 minutes. During 12 hours pf hyperparameter search, about 20 configurations were tested. The TE and xLSTM hyperparameters were chosen as what was believed to be the best combination. For example, a maximum of input feature length was thought to be ideal. For hourly prediction, a maximum of features was set as well. This difference in choosing hyperparameters might favor models, where more hyperparameter sets could be tried.

The official TFT implementation by Google [97] only runs on TensorFlow version 1.15, which was published in October 2019. Unfortunately, the code was not compatible with available GPUs and the training could not be accelerated due to this issue. Another option for future experiments could be to use the darts implementation[2] of the TFT, which uses PytorchForecasting's model implementation[3].

The task of time-series forecasting was particularly difficult on the given dataset, which contained many heterogeneous time-series from different buildings. The series data was not recorded for one period of time, but rather each series had their own length and recording date. Therefore, it could happen that the model did not see any data from the heating period of a building, but was tested on it, and vice versa. An evaluation with those time constraints in mind could give a fairer representation of the models' abilities. However, it might reduce the available data even further, because not all time-series are covering at least two heating periods. Also, the given scenario might be more realistic for practical solutions.

[2] https://unit8co.github.io/darts/generated_api/darts.models.forecasting.tft_model.html

[3] https://pytorch-forecasting.readthedocs.io/en/latest/api/pytorch_forecasting.models.temporal_fusion_transformer._tft.TemporalFusionTransformer.html

6 Conclusion and Outlook

This work performed a benchmark of popular and new ML methods for time-series forecasting on a heat energy consumption dataset. Tested models were the FCN, LSTM, TE, TFT, and the more recently proposed xLSTM. Current trends in ML models for time-series forecasting for energy forecasting are researched . The data processing pipeline is explained in detail, as well as final model implementations. The forecasting abilities of the models are tested on daily data, for one and seven day prediction, and on hourly data, for three and 24-hour prediction. Also, the influence of amount of training data and impact of future covariate accuracy on the models' prediction quality are investigated. This is followed by a theoretical and empirical analysis of the models' runtime and memory complexity.

The best performing model of the experiments is the xLSTM, which is able to achieve the lowest RMSE score for three out of four experiments. The xLSTM is also able to keep a low RMSE for reduced training data, outperforming models even when only 50% of the data is available for training. Over a 24-step prediction horizon, it does increase in error for the eleventh step, but still in a moderate rate. It is only beaten for seven day prediction, by the FCN, which also provides competitive results for the other experiments. The FCN has the least amount of trainable parameters and is the fastest model to train, which makes it the best choice for hardware-restrained environments. The largest model is the TFT, which outperforms all models on the MAE for hourly forecasting tasks. For 24-hour prediction, its RMSE interestingly decreases for later steps, rather than decrease linearly, like the xLSTMs. Out of all the tasks, the models had the lowest difference to the baseline RMSE for 24-hour prediction. This suggests that the baseline either works comparably well on this task, or that it was particularly difficult for the models to find an accurate forecast for this horizon length and resolution.

The experiments showed how ML methods can be successfully used for time-series forecasting on an energy dataset comprised of multiple buildings with different energy consumption types and time-series lengths. The baseline, forecasting the value of the

step-ahead, could be beaten for almost all metrics and use-cases. The TFT provided methods of integrating continuous and static covariates, which helped it to achieve top MAE results for hourly predictions. The xLSTM, without considering features separately, still provided top RMSE results across daily and hourly data. The FCN was the most time and memory efficient model to train, still able to provide competitive to top results.

This work investigated point-wise predictions, but for decision-making, often interval estimates are preferred. These give thresholds, beneath which a future value will be for a given probability. The TFT is originally constructed to perform such quantile regression. Therefore, it would be interesting to see if the TFT can outperform the other models, adapted to this regression task, due to its original conception.

The experiments were performed on daily and hourly resolutions of the data, where the daily data was computed from an aggregated part of the hourly dataset. It could be investigated how the daily prediction performs, if hourly datapoints are used in training and results are aggregated, rather than aggregating hours beforehand.

The TFT is one of many proposed Transformer based models for time-series forecasting. The experiments could be repeated using models like the Time Augmented Transformer [34], Informer [13], Autoformer [70], FEDformer [71], or Crossformer [82]. Also, the xLSTM is not specifically designed for time-series forecasting. Two models, the xLSTMTime [87] and xLSTM-Mixer [89] have recently been proposed, adapting a xLSTM for time-series forecasting tasks. It would be interesting to evaluate if these models can further improve the results made by the xLSTM used in this work.

During the data preprocessing, more than half of the data was removed due to the missing feature heated area. To evaluate, if adding the building data to the covariate features increases the model's ability to train, only time-series with known heated area feature were used for training, whether or not the feature was included. As the results show that models can even work better without any features, another approach could be to train the architectures on the full dataset, without using building information.

Another recent approach is to use Foundation Models (FMs) for time-series forecasting. FMs are models trained on large-scale data and used in various domains, especially NLP and computer vision. The success of those models lies in their ability to perform well in any task, without being trained on it specifically. This results from the knowledge they gain by training on large amounts of data. Recently, the abilities of FMs have been noticed by the time-series forecasting community. Models like TimesFM [114] by Google or TiRex [115] by NX-AI have been proposed as FM for

time-series forecasting. After having performed an extensive survey and benchmark on the topic, Wild [116] concludes that practical applications of FMs with covariates need to rather fine-tune the models on the forecasting task at hand, than simply be used as zero-shot models. Therefore, investigating the forecasting accuracy of FMs after fine-tuning them on the heat load dataset would be an interesting future work.

The xLSTM performed the best out of all benchmarked models, without having had access to future covariates. The TFT implemented an LSTM-based Encoder-Decoder, for incorporating this specific information. Adding a similar mechanism to the xLSTM could improve its performance even further, which might be evaluated by future research.

A Appendix

A.1 Dataset

Process Step	Daily Count	Daily Sensors	Daily Series
Raw Data	110923	149	149
Remove Corrupt Sensors	110005	142	142
Remove Negative Consumption	109958	-	-
Filter Connection Errors	108757	-	-
Filter Outliers	108067	-	-
Filter Flat Lines	101868	141	141
Split Series	101868	141	364
Interpolate	104223	-	-
Drop Lag Feature Nulls	99416	-	-
Drop Feature Nulls	56955	70	132
Remove Short Series	54106	-	92

Figure A.1: (a) Daily Data Preprocessing Steps

Process Step	Hourly Count	Hourly Sensors	Hourly Series
Raw Data	870230	61	61
Remove Corrupt Sensors	870230	-	-
Remove Negative Consumption	870210	-	-
Filter Connection Errors	853488	-	-
Filter Outliers	823722	-	-
Filter Flat Lines	688766	61	61
Split Series	688766	57	921
Interpolate	834313	-	1127
Drop Lag Feature Nulls	729798	-	-
Drop Feature Nulls	142085	14	44
Remove Short Series	136043	-	25

Figure A.2: (b) Hourly Data Preprocessing Steps

Figure A.3: Preprocessing steps and data counts for daily and hourly formats

Figure A.4: Map of Germany, where the location of sensors is highlighted in green.

Table A.1: Variable Description and Availability

Variable	Description	Availability
coco [%]	Cloud cover	hourly
daily_avg [kWh]	Daily average energy consumption of the building	both
day_of_month_cos	Cosine transformation of day of month	both
day_of_month_sin	Sine transformation of day of month	both
diff [kWh]	Energy consumption	both
dwpt [°C]	Dew point temperature	hourly
heated_area [m^2]	Heated area of building	both
holiday [binary]	Holiday indicator	both
hum_avg [%]	Average humidity	both
hum_max [%]	Maximum humidity	daily
hum_min [%]	Minimum humidity	daily
pres [hPa]	Atmospheric pressure	both
prcp [mm]	Precipitation	both
primary_energy_dh [binary]	District heating energy	both
primary_energy_gas [binary]	Primary energy gas	both
rhum [%]	Relative humidity	hourly
snow [cm]	Snow depth	both
tavg [°C]	Average temperature	daily
temp [°C]	Temperature	hourly
tmax [°C]	Maximum temperature	daily
tmin [°C]	Minimum temperature	daily
tsun [hours]	Sunshine duration	both
typ_0 [binary]	Building type: residential	both
typ_1 [binary]	Building type: museum	both
typ_2 [binary]	Building type: school	both
typ_3 [binary]	Building type: student housing	daily
typ_4 [binary]	Building type: social housing	both
wdir [degrees]	Wind direction	both
weekday_cos	Cosine transformation of weekday	both
weekday_sin	Sine transformation of weekday	both
weekend [binary]	Weekend indicator	both
wpgt [m/s]	Wind peak gust	both
wspd [m/s]	Wind speed	both

Table A.2: p-Value Analysis Daily Dataset

Variable	**p-value**	**Coefficient**
primary_energy_gas	0.0000000000	1.0068
wdir	6.4261e-31	-222.3362
daily_avg	1.8864e-28	4171.0464
pres	4.5782e-26	2816.9578
diff	2.2676e-25	7806.7107
heated_area	6.0753e-22	3635.6643
typ_4	7.5535e-18	1764.6553
weekday_sin	8.9271e-18	1698.3419
typ_1	9.6472e-16	-10.1126
day_of_month_sin	7.6907e-15	1526.7557
weekend	6.7005e-14	19.2167
hum_avg	7.9327e-13	-163.5859
tmax	1.8665e-06	-0.6387
typ_0	2.7027e-06	-20.2157
wpgt	2.4561e-05	16.0876
wspd	5.9137e-05	-0.3321
snow	1.2857e-03	39.1406
holiday	1.3870e-03	-49.5990
tmin	4.2061e-03	-27.4088
prcp	4.6730e-02	-79.7678
hum_max	5.0213e-02	-3.1269
typ_2	7.2797e-02	43.0864
tavg	1.9282e-01	-19.6758
hum_min	2.4154e-01	-3.7668
day_of_month_cos	3.4646e-01	16.5780
tsun	4.6332e-01	-1.1560
primary_energy_district heating	7.6232e-01	0.0010
weekday_cos	7.9884e-01	-0.7950

Table A.3: p-Value Analysis Hourly Dataset

Variable	**p-value**	**Coefficient**
heated_area	0.0000000000	1.0018
weekend	0.0000000000	-9.5350
weekday_cos	0.0000000000	1.7786
wdir	3.0112e-215	-0.5450
holiday	2.5139e-165	73.4497
snow	1.3110e-164	3.9720
typ_0	1.5238e-147	69.4139
temp	3.5327e-135	-0.4041
daily_avg	3.5379e-126	0.0021
day_of_month_cos	3.2348e-122	123.9734
diff	4.2912e-116	241.6272
primary_energy_district heating	9.4813e-110	117.6538
rhum	3.0383e-105	57.8139
typ_1	1.1531e-71	-18.1641
pres	7.4418e-49	40.9497
wspd	2.5921e-34	0.3289
typ_2	9.1992e-23	-7.5257
primary_energy_gas	2.5611e-09	0.0505
weekday_sin	5.5729e-09	-1.1653
day_of_month_sin	2.4158e-07	-1.7398
prcp	4.4323e-07	1.0436
wpgt	9.7311e-04	0.1964
typ_4	4.2488e-03	0.0044
tsun	1.2245e-02	-0.1200
dwpt	2.2697e-02	-3.2954
coco	1.7451e-01	-1.8691

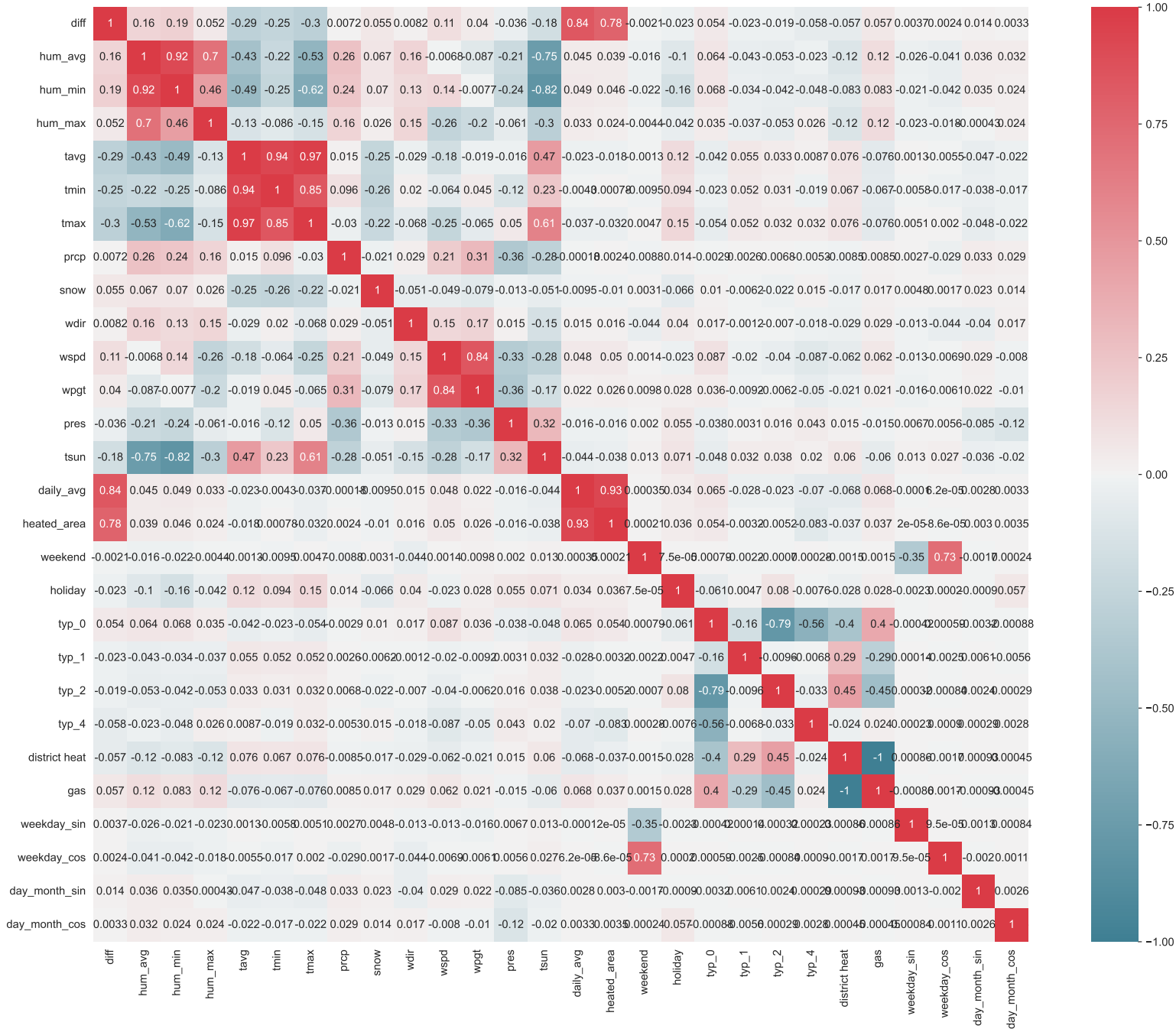

Figure A.5: Correlation matrix for daily dataset.

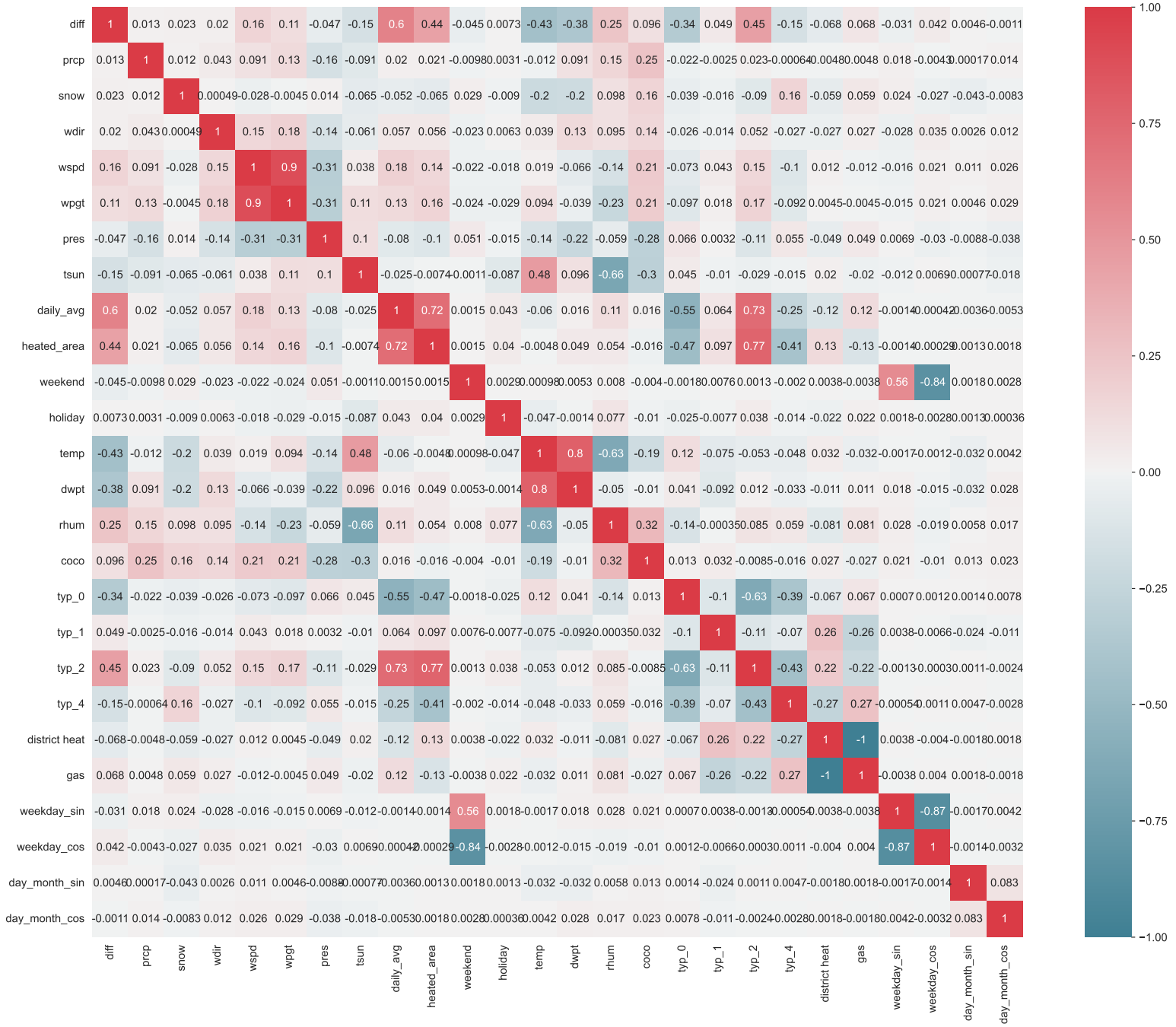

Figure A.6: Correlation matrix for hourly dataset.

A.2 Experiments

The following tables give a better understanding of which hyperparameter configurations were chosen for each model in each experiment. TE is short for Transformer Encoder and n_in is the length of input series, while n_future is the length of future covariate series.

Model	Epochs	Neurons/Heads	n_in	n_future	Feature
FCN	40	60	7	0	No building
LSTM	80	100	7	1	all
TE	25	4	7	1	No building
TFT	25	4	7	1	all
xLSTM	10	4	7	0	No building

Table A.4: Hyperparameter comparison for models for next day prediction

Model	Epochs	Neurons/Heads	n_in	n_future	Feature
FCN	40	100	7	7	No building
LSTM	80	100	7	7	all
TE	20	4	7	7	all
TFT	25	4	7	7	No building
xLSTM	10	4	7	0	No building

Table A.5: Hyperparameter comparison for models for next week prediction

Model	Epochs	Neurons/Heads	n_in	n_future	Feature
FCN	34	51	72	0	only diff
LSTM	28	121	59	0	only diff
TE	10	8	24	0	all
TFT	20	4	72	3	all
xLSTM	10	4	72	0	all

Table A.6: Hyperparameter comparison for models for three hour prediction

Model	Epochs	Neurons/Heads	n_in	n_future	Feature
FCN	78	131	38	0	only diff
LSTM	49	118	38	24	all
TE	10	4	72	24	all
TFT	20	4	72	24	all
xLSTM	30	4	72	0	all

Table A.7: Hyperparameter comparison for models for 24-hour prediction

Feature	RMSE
prcp	3.97
tsun	228.20
hum_min	13.73
tmin	4.47
wspd	5.15
hum_avg	9.27
hum_max	5.65
tmax	5.62
wpgt	12.73
tavg	4.72
wdir	78.91
pres	7.21
Average	**31.64**

Table A.8: RMSE values for different weather features actual and simulated values.

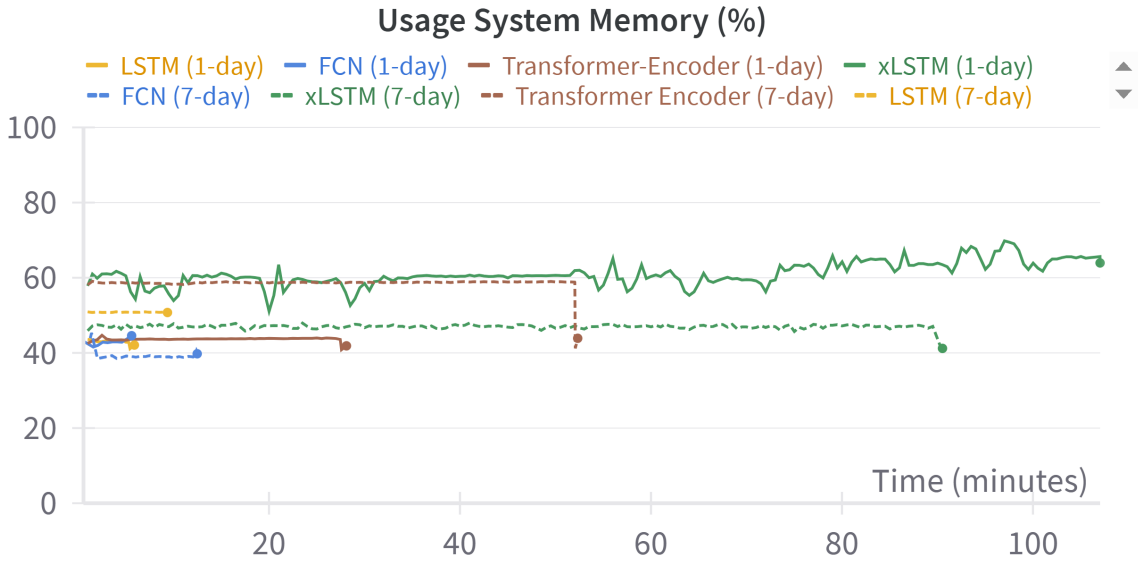

Figure A.7: Usage of the system memory in % for each model for seven day prediction over training time. The xLSTM training displayed here does not belong to the best xLSTM model, but an xLSTM trained on the CPU. The TFT is missing in this summary.

Acronyms

Adam Adaptive Moment Estimation. 9

AI Artificial Intelligence. 1, 5, 34–36, 93

ANN Artificial Neural Network. 2, 8, 17, 19, 93

AR autoregressive. 7

ARIMA Autoregressive Integrated Moving Average. 1, 7, 23–25

ARMA Autoregressive Moving Average. 19

BERT Bidirectional Encoder Representations from Transformers. 12

CNN Convolutional Neural Network. 19, 20, 23, 24

CPU Central Processing Unit. 53, 72, 73, 89, 95

DHC District Heating and Cooling. 19

DNN Deep Neural Network. 18

EMD Empirical Mode Decomposition. 17

FCN Fully-connected Network. 8, 19, 26–29, 49, 50, 52–60, 63–74, 76, 87, 88, 93, 94

FM Foundation Model. 77, 78

GJ Giga-Joule. 19

GLU Gated Linear Unit. 30, 31

GPR Gaussian Process Regression. 17, 18

GPT Generative Pre-trained Transformer. 12

GPU Graphics Processing Unit. 53, 69, 70, 72–74, 95

GRN Gated Residual Network. 30–32, 70

GRU Gated Recurrent Unit. 11, 23

HVAC heat, ventilation, air, conditioning. 17, 18

ICA Imperialistic Competitive Algorithm. 17

IQR Interquartile Range. 38, 39, 93

LLM Large Language Models. 2, 12

LoD Level of Detail. 48

LSTM Long short-term memory. 2, 3, 10–14, 19–21, 23–27, 29, 32, 49, 50, 52–56, 58–60, 63–74, 76, 87, 88, 93

MAE Mean Absolute Error. 16, 18, 53, 54, 58, 59, 72–74

MANE Mean Absolute Normalized Error. 18

MAPE Mean Absolute Percentage Error. 16–19, 23–25

ML machine learning. 1–3, 5, 26, 52, 60

MLP Multi-Layer Perceptron. 18

mLSTM matrix LSTM. 14, 15, 32, 33, 70, 93

MSE Mean Squared Error. 15, 16, 19, 32, 54, 58, 74

NLP Natural Language Processing. 2, 32, 77

nRMSE normalized RMSE. 15, 54, 58, 73

nRSE normalized RSE. 54, 55, 59–61, 94

RelU Rectified Linear Unit. 8, 28

RMSE Root Mean Squared Error. 15–17, 20, 23, 52–54, 56–60, 62–68, 72, 74, 76, 77, 88, 94–96

RNN Recurrent Neural Network. 9, 10, 12, 13, 19, 23, 26, 28, 29, 69

RSE Rooted Squared Error. 54

SARIMA seasonal ARIMA. 7, 19

SLPs Synthesized Load Profiles. 2

sLSTM scalar LSTM. 14, 15, 25, 26, 32, 33, 70, 93

SMAPE Symmetric MAPE. 23

SSM State Space Model. 22

SVM Support Vector Machine. 17, 18, 23

SVR Support Vector Regression. 18, 24

TE Transformer-Encoder. 27, 49, 50, 53, 55, 56, 58–60, 63–68, 72–74, 76, 87, 88

TFT Temporal Fusion Transformer. 7, 21, 22, 24, 25, 27, 28, 30–32, 49, 50, 52–56, 58–60, 63–67, 70, 72–74, 76–78, 87–89, 93, 95

VSN Variable Selection Network. 31, 70, 71

XGBoost eXtreme Gradient Boosting. 18

xLSTM extended Long short-term memory. v, 2, 3, 5, 12, 14, 15, 25–28, 32, 33, 49, 53–56, 58–61, 63–67, 69, 70, 72–74, 76–78, 87–89, 93–95

Figures

2.1 An example ANN [42] . . . 9
2.2 Recurrent Neural Network cell with feedback loop, input x_t and hidden state h_t [44] . . . 10
2.3 The LSTM and LSTM cell architecture [44] . . . 11
2.4 The Transformer architecture [12] . . . 13

3.1 High-Level overview of model training process. . . . 28
3.2 Architecture of FCN with one hidden layer . . . 29
3.3 Architecture of Transformer encoder with N encoder blocks and preceding up-projection layer, as well as successive down-projection layers. 30
3.4 Architecture of TFT as presented by Lim et al. [31]. . . . 31
3.5 Architecture of the adapted xLSTM. mLSTM and sLSTM are like in the original xLSTM paper, added layers are described with blue text box. . 34

4.1 Time-series recording span overview for the raw daily dataset. Time-series are from following sources: Legacy (82, dark blue), KI-NERGY (14, brown), AI in district heating (47, cyan) . . . 35
4.2 Number of Sensors Overview . . . 36
4.3 Number of Daily Data Points Overview . . . 36
4.4 Number of Hourly Data Points Overview . . . 36
4.5 Overview of raw data counts and sources by category . . . 36
4.6 Overview of types of buildings in the dataset. . . . 37
4.7 Count of data points after each preprocessing step in daily and hourly format. The numbers in brackets indicate the amount of energy meters . 38
4.8 Example time-series, where outliers detected by the IQR method are colored in red. These points are removed for further processing. . . . 40
4.9 Plot of a time-series recording daily energy consumption. The series shows a flat line starting around June, 2024. Missing data is signaled by blue color. The consumption is shown in kWh. . . . 41

4.10 Number of missing data points compared to available data points per series for raw daily data. 41
4.11 Example time-series that was split along missing data periods and interpolated. The different colored sections are new time-series. 43
4.12 Result of clustering algorithm on mixed dataset. Each sensor belongs to one cluster and is visualized using the average of its daily consumption values and the standard deviation. 47

5.1 Sliding window technique with step size one [110] 50
5.2 One day forecasts for all models for one building in the test set. 55
5.3 Box plot of nRSE (logarithmically scaled) for xLSTM for one day prediction for each of the 70 buildings in the dataset. 56
5.4 RMSE (kWh) for each step of the forecast horizon of 7 days per model. 57
5.5 Average RMSE on the day of the week for the FCN model for seven day prediction, which has an overall RMSE of 757.08 kWh on this task. The color indicates how high the average energy consumption on this weekday in the test dataset was. 58
5.6 Example of 24-hour forecasts of the xLSTM for one time-series of the test set. The black line shows the recorded values, while the red lines show the 24-hour forecasts. 60
5.7 Distribution of nRSE for each time-series in the hourly test dataset, logarithmically scaled. The series are sorted by average energy consumption, ascending from left to right. 61
5.8 Distribution of nRSE per hour of the day of the xLSTM model for 24-hour prediction. 61
5.9 Average RMSE (kWh) per 24 forecasting steps for each model. 62
5.10 RMSE (kWh) results for the best models for prediction of one day with all training data available, 20% removed, and 50% removed training data. 63
5.11 RMSE (kWh) results for the best models for prediction of seven days with all training data available, 20% removed, and 50% removed training data. 65
5.12 RMSE (kWh) results for the best models for prediction of three hours with all training data available, 20% removed, and 50% removed training data. 66
5.13 RMSE (kWh) results for the best models for prediction of 24 hours with all training data available, 20% removed, and 50% removed training data. 67

5.14 Comparison of the number of trainable parameters for each model and the forecasting horizons of seven days or 24 hours. The y-axis is logarithmically scaled. 73

A.1 (a) Daily Data Preprocessing Steps 81
A.2 (b) Hourly Data Preprocessing Steps 81
A.3 Preprocessing steps and data counts for daily and hourly formats . . . 81
A.4 Map of Germany, where the location of sensors is highlighted in green. 82
A.5 Correlation matrix for daily dataset. 86
A.6 Correlation matrix for hourly dataset. 87
A.7 Usage of the system memory in % for each model for seven day prediction over training time. The xLSTM training displayed here does not belong to the best xLSTM model, but an xLSTM trained on the CPU. The TFT is missing in this summary. 90

Tables

4.1 Daily and hourly weather features provided by meteostat. Values with a (*) are calculated from hourly values. 44

5.1 Hyperparameter Search Space Configurations by Model 53
5.2 Evaluation results for one day prediction. RMSE and MAE are in kWh. 54
5.3 Evaluation results for seven day prediction. RMSE and MAE are in kWh. 56
5.4 Evaluation results for three hour prediction. RMSE and MAE are in kWh. 59
5.5 Evaluation results for 24-hour prediction. RMSE and MAE are in kWh. 59
5.6 Training data point count for reduced data evaluation for each format of resolution. 63
5.7 Comparison of model RMSE (kWh) for testing with recorded future weather data (RMSE) and testing with simulated future weather data (RMSE + Noise) 69

5.8 Training runtime comparison of time-series prediction models. Runtimes with a * result from trainings on the GPU, rather than the CPU. . 74

A.1 Variable Description and Availability 83
A.2 p-Value Analysis Daily Dataset 84
A.3 p-Value Analysis Hourly Dataset 85
A.4 Hyperparameter comparison for models for next day prediction 88
A.5 Hyperparameter comparison for models for next week prediction . . . 88
A.6 Hyperparameter comparison for models for three hour prediction . . . 88
A.7 Hyperparameter comparison for models for 24-hour prediction 89
A.8 RMSE values for different weather features actual and simulated values. 89

Bibliography

[1] "Energy and climate", Accessed: Jan. 1, 2025. [Online]. Available: https://www.bmz.de/en/issues/climate-change-and-development/energy-and-climate.

[2] T Pan. "Why heat is a challenge in the fight against climate change, and what we can do about it", Accessed: Jan. 1, 2025. [Online]. Available: https://www.weforum.org/stories/2023/03/heat-heres-why-its-the-elephant-in-the-room-for-decarbonization/.

[3] K. Djebko et al., "Design and implementation of a decision integration system for monitoring and optimizing heating systems: Results and lessons learned", *Energies*, vol. 17, no. 24, p. 6290, Jan. 2024, Number: 24 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en17246290. Accessed: May 23, 2025. [Online]. Available: https://www.mdpi.com/1996-1073/17/24/6290.

[4] A. Vandermeulen, B. van der Heijde, D. Patteeuw, D. Vanhoudt, and L. Helsen, "An operational strategy for district heating networks: Application of energetic flexibility", *Energy Informatics*, vol. 3, no. 1, pp. 1–17, 2020. DOI: 10.1186/s42162-020-00125-5.

[5] Y. Feng, "Optimizing energy efficiency: Predicting heating load with a machine learning approach and meta-heuristic algorithms", *Multiscale and Multidisciplinary Modeling, Experiments and Design*, vol. 7, pp. 3993–4009, 2024. DOI: 10.1007/s41939-024-00453-z.

[6] Y. Jiang, Q. Wan, X. Yang, Y. Xu, and Y. Wang, "Cooling, heating and electric load forecasting for integrated energy system based on lstm-cnn", in *2021 IEEE 4th International Electrical and Energy Conference (CIEEC)*, IEEE, 2021, pp. 1–6. DOI: 10.1109/CIEEC50170.2021.9635244.

[7] J. Zhu et al., "Time-series power forecasting for wind and solar energy based on the sl-transformer", *Energies*, vol. 16, no. 22, p. 7610, 2023.

[8] D. R. Bayer, F. Haag, M. Pruckner, and K. Hopf, "Electricity demand forecasting in future grid states: A digital twin-based simulation study", in *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*, IEEE, 2024, pp. 1–6.

[9] B. Lim and S. Zohren, *Time series forecasting with deep learning: A survey*, Sep. 27, 2020. DOI: 10.48550/arXiv.2004.13408. arXiv: 2004.13408. Accessed: Nov. 29, 2024. [Online]. Available: http://arxiv.org/abs/2004.13408.

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

[11] A. Sherstinsky, *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*, Jul. 31, 2023. DOI: 10.48550/arXiv.1808.03314. arXiv: 1808.03314. Accessed: Nov. 29, 2024. [Online]. Available: http://arxiv.org/abs/1808.03314.

[12] A. Vaswani et al., "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[13] H. Zhou et al., *Informer: Beyond efficient transformer for long sequence time-series forecasting*, Mar. 28, 2021. DOI: 10.48550/arXiv.2012.07436. arXiv: 2012.07436. Accessed: Nov. 29, 2024. [Online]. Available: http://arxiv.org/abs/2012.07436.

[14] M. Beck et al., *Xlstm: Extended long short-term memory*, 2024. arXiv: 2405.04517 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2405.04517.

[15] Y. Tay et al., "Long range arena : A benchmark for efficient transformers", in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=qVyeW-grC2k.

[16] Z. Zhao et al., "Short-Term Load Forecasting Based on the Transformer Model", en, *Information*, vol. 12, no. 12, p. 516, Dec. 2021, Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2078-2489. DOI: 10.3390/info12120516. Accessed: May 15, 2025. [Online]. Available: https://www.mdpi.com/2078-2489/12/12/516.

[17] F. Tang, A. Kusiak, and X. Wei, "Modeling and short-term prediction of HVAC system with a clustering algorithm", en, *Energy and Buildings*, vol. 82, pp. 310–321, Oct. 2014, ISSN: 03787788. DOI: 10.1016/j.enbuild.2014.07.037. Ac-

cessed: Mar. 21, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378778814005751.

[18] A. T. Eseye and M. Lehtonen, "Short-Term Forecasting of Heat Demand of Buildings for Efficient and Optimal Energy Management Based on Integrated Machine Learning Models", *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7743–7755, Dec. 2020, Conference Name: IEEE Transactions on Industrial Informatics, ISSN: 1941-0050. DOI: 10.1109/TII.2020.2970165. Accessed: Mar. 16, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8990012.

[19] P. C. Huy, N. Q. Minh, N. D. Tien, and T. T. Q. Anh, "Short-Term Electricity Load Forecasting Based on Temporal Fusion Transformer Model", *IEEE Access*, vol. 10, pp. 106 296–106 304, 2022, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3211941. Accessed: Mar. 16, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9910162?denied=.

[20] P. Xue, Y. Jiang, Z. Zhou, X. Chen, X. Fang, and J. Liu, "Multi-step ahead forecasting of heat load in district heating systems using machine learning algorithms", *Energy*, vol. 188, p. 116 085, Dec. 2019, ISSN: 0360-5442. DOI: 10.1016/j.energy.2019.116085. Accessed: Mar. 19, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544219317803.

[21] P. Potočnik, P. Škerl, and E. Govekar, "Machine-learning-based multi-step heat demand forecasting in a district heating system", *Energy and Buildings*, vol. 233, p. 110 673, Feb. 2021, ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2020.110673. Accessed: Mar. 16, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778820334599.

[22] *Tracking clean energy progress 2017*, https://www.iea.org/reports/tracking-clean-energy-progress-2017, Accessed: 23.05.2025, 2017.

[23] *Global energy transition statistics*, https://yearbook.enerdata.net/, Accessed: 23.05.2025.

[24] B. N. Silva, M. Khan, and K. Han, "Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities", *Sustainable cities and society*, vol. 38, pp. 697–713, 2018.

[25] M. Bourdeau, X. Q. Zhai, E. Nefzaoui, X. Guo, and P. Chatellier, "Modeling and forecasting building energy consumption: A review of data-driven techniques", en, *Sustainable Cities and Society*, vol. 48, p. 101 533, Jul. 2019, ISSN: 22106707. DOI: 10.1016/j.scs.2019.101533. Accessed: Mar. 21, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2210670718323862.

[26] X. Xu, J. E. Taylor, A. L. Pisello, and P. J. Culligan, "The impact of place-based affiliation networks on energy conservation: An holistic model that integrates the influence of buildings, residents and the neighborhood context", *Energy and Buildings*, Cool Roofs, Cool Pavements, Cool Cities, and Cool World, vol. 55, pp. 637–646, Dec. 1, 2012, ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2012.09.013. Accessed: May 23, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037877881200463X.

[27] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour, "Energy Forecasting: A Review and Outlook", *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 376–388, 2020, Conference Name: IEEE Open Access Journal of Power and Energy, ISSN: 2687-7910. DOI: 10.1109/OAJPE.2020.3029979. Accessed: Jan. 2, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9218967/?arnumber=9218967.

[28] *Ai in district heating*, https://future-energy-lab.de/projects/ki-in-fernwaerme/, Accessed: 23.05.2025.

[29] B. Auffarth, *Machine Learning for Time-Series with Python: Forecast, predict, and detect anomalies with state-of-the-art machine learning methods*. Birmingham, UK: Packt Publishing Ltd, 2021, p. 371, ISBN: 9781801819626.

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

[31] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, *Temporal fusion transformers for interpretable multi-horizon time series forecasting*, 2020. arXiv: 1912.09363 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1912.09363.

[32] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour, "Energy forecasting: A review and outlook", *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 376–388, 2020.

[33] I. A. Samuel, A. Emmanuel, I. A. Odigwe, and F. C. Felly-Njoku, "A comparative study of regression analysis and artificial neural network methods for medium-term load forecasting", *Indian Journal of Science and Technology*, vol. 10, no. 10, pp. 1–7, 2017.

[34] G. Zhang, C. Wei, C. Jing, and Y. Wang, "Short-Term Electrical Load Forecasting Based on Time Augmented Transformer", en, *International Journal of Computational Intelligence Systems*, vol. 15, no. 1, p. 67, Aug. 2022, ISSN: 1875-6883. DOI: 10.1007/s44196-022-00128-y. Accessed: Mar. 16, 2025. [Online]. Available: https://doi.org/10.1007/s44196-022-00128-y.

[35] E. Box George, M Jenkins Gwilym, C Reinsel Gregory, and M Ljung Greta, "Time series analysis: Forecasting and control", 1970.

[36] G. E. Box and G. C. Tiao, "Intervention analysis with applications to economic and environmental problems", *Journal of the American Statistical association*, vol. 70, no. 349, pp. 70–79, 1975.

[37] E. S. Gardner Jr, "Exponential smoothing: The state of the art", *Journal of forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[38] A. C. Harvey, "Forecasting, structural time series models and the kalman filter", 1990.

[39] X. Liu, C. Marnay, W. Feng, N. Zhou, and N. Karali, "A review of the american recovery and reinvestment act smart grid projects and their implications for china", 2017.

[40] *Metadata record for: The Building Data Genome Project 2, energy meter data from the ASHRAE Great Energy Predictor III competition*, en, Oct. 2020. DOI: 10.6084/m9.figshare.13033847.v1. Accessed: May 15, 2025. [Online]. Available: https://springernature.figshare.com/articles/dataset/Metadata_record_for_The_Building_Data_Genome_Project_2_energy_meter_data_from_the_ASHRAE_Great_Energy_Predictor_III_competition/13033847/1.

[41] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[42] *Artificial neural networks and its applications*, https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/, Accessed: 25.05.2025, Apr. 2025.

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

[44] C. Olah, *Understanding lstm networks*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/, Accessed: 23.05.2025, 2015.

[45] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm", *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[46] K. Cho et al., *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, arXiv:1406.1078 [cs] version: 3, Sep. 2014. DOI: 10.48550/arXiv.1406.1078. Accessed: May 23, 2025. [Online]. Available: http://arxiv.org/abs/1406.1078.

[47] S. Raschka, *Build a Large Language Model (From Scratch)*, en. Simon and Schuster, Oct. 2024, Google-Books-ID: uSUmEQAAQBAJ, ISBN: 978-1-63343-716-6.

[48] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[49] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., "Improving language understanding by generative pre-training", 2018.

[50] Q. Wen et al., *Transformers in time series: A survey*, May 11, 2023. DOI: 10.48550/arXiv.2202.07125. arXiv: 2202.07125. Accessed: Nov. 29, 2024. [Online]. Available: http://arxiv.org/abs/2202.07125.

[51] T. Kohonen, "Correlation matrix memories", *IEEE transactions on computers*, vol. 100, no. 4, pp. 353–359, 2009.

[52] G. Gong, X. An, N. K. Mahato, S. Sun, S. Chen, and Y. Wen, "Research on Short-Term Load Prediction Based on Seq2seq Model", en, *Energies*, vol. 12, no. 16, p. 3199, Jan. 2019, Number: 16 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en12163199. Accessed: May 15, 2025. [Online]. Available: https://www.mdpi.com/1996-1073/12/16/3199.

[53] S. Li et al., *Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting*, arXiv:1907.00235 [cs], Jan. 2020. DOI: 10.48550/arXiv.1907.00235. Accessed: Apr. 12, 2025. [Online]. Available: http://arxiv.org/abs/1907.00235.

[54] M. López, C. Sans, S. Valero, and C. Senabre, "Classification of Special Days in Short-Term Load Forecasting: The Spanish Case Study", en, *Energies*, vol. 12, no. 7, p. 1253, Jan. 2019, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en12071253. Accessed: Mar. 19, 2025. [Online]. Available: https://www.mdpi.com/1996-1073/12/7/1253.

[55] S. Haben, G. Giasemidis, F. Ziel, and S. Arora, "Short term load forecasting and the effect of temperature at the low voltage level", *International Journal of Forecasting*, vol. 35, no. 4, pp. 1469–1484, Oct. 2019, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2018.10.007. Accessed: Mar. 7, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207018301870.

[56] R. K. Jain, K. M. Smith, P. J. Culligan, and J. E. Taylor, "Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial monitoring granularity on performance accuracy", *Applied Energy*, vol. 123, pp. 168–178, Jun. 2014, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2014.02.057. Accessed: Mar. 7, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261914002013.

[57] E. Giacomazzi, F. Haag, and K. Hopf, "Short-term electricity load forecasting using the temporal fusion transformer: Effect of grid hierarchies and data sources", in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 353–360.

[58] D. R. Bayer, F. Haag, M. Pruckner, and K. Hopf, "Electricity Demand Forecasting in Future Grid States: A Digital Twin-Based Simulation Study", en, in *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*, Bol and Split, Croatia: IEEE, Jun. 2024, pp. 1–6, ISBN: 978-953-290-135-1. DOI: 10.23919/SpliTech61897.2024.10612563. Accessed: Dec. 25, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10612563/.

[59] K. Kato, M. Sakawa, K. Ishimaru, S. Ushiro, and T. Shibano, "Heat load prediction through recurrent neural network in district heating and cooling systems", in *2008 IEEE International Conference on Systems, Man and Cybernetics*, ISSN: 1062-922X, Oct. 2008, pp. 1401–1406. DOI: 10.1109/ICSMC.2008.4811482. Accessed: Mar. 21, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/4811482/?arnumber=4811482.

[60] D. Geysen, O. De Somer, C. Johansson, J. Brage, and D. Vanhoudt, "Operational thermal load forecasting in district heating networks using machine learning and expert advice", en, *Energy and Buildings*, vol. 162, pp. 144–153, Mar. 2018, ISSN: 03787788. DOI: 10.1016/j.enbuild.2017.12.042. Accessed: Mar. 21, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378778817312070.

[61] J. Xie et al., "Analysis of Key Factors in Heat Demand Prediction with Neural Networks", *Energy Procedia*, 8th International Conference on Applied Energy, ICAE2016, 8-11 October 2016, Beijing, China, vol. 105, pp. 2965–2970, May 2017, ISSN: 1876-6102. DOI: 10.1016/j.egypro.2017.03.704. Accessed: Mar. 21, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610217307671.

[62] S. Muzaffar and A. Afshari, "Short-Term Load Forecasts Using LSTM Networks", en, *Energy Procedia*, vol. 158, pp. 2922–2927, Feb. 2019, ISSN: 18766102. DOI: 10.1016/j.egypro.2019.01.952. Accessed: Mar. 19, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1876610219310008.

[63] G. Suryanarayana, J. Lago, D. Geysen, P. Aleksiejuk, and C. Johansson, "Thermal load forecasting in district heating networks using deep learning and advanced feature selection methods", *Energy*, vol. 157, pp. 141–149, Aug. 2018, ISSN: 0360-5442. DOI: 10.1016/j.energy.2018.05.111. Accessed: Mar. 19, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544218309381.

[64] M. Dahl, A. Brun, and G. B. Andresen, "Using ensemble weather predictions in district heating operation and load forecasting", *Applied Energy*, vol. 193, pp. 455–465, May 2017, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2017.02.066. Accessed: Mar. 19, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261917302143.

[65] R. Petrichenko, K. Baltputnis, A. Sauhats, and D. Sobolevsky, "District heating demand short-term forecasting", in *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, 2017, pp. 1–5. DOI: 10.1109/EEEIC.2017.7977633.

[66] J. Zhu et al., "Time-Series Power Forecasting for Wind and Solar Energy Based on the SL-Transformer", en, *Energies*, vol. 16, no. 22, p. 7610, Jan. 2023, Number: 22 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en16227610. Accessed: Dec. 4, 2024. [Online]. Available: https://www.mdpi.com/1996-1073/16/22/7610.

[67] C. Johansson, M. Bergkvist, D. Geysen, O. D. Somer, N. Lavesson, and D. Vanhoudt, "Operational demand forecasting in district heating systems using ensembles of online machine learning algorithms", *Energy Procedia*, vol. 116, pp. 208–216, 2017, 15th International Symposium on District Heating and Cooling, DHC15-2016, 4-7 September 2016, Seoul, South Korea, ISSN: 1876-6102. DOI: https://doi.org/10.1016/j.egypro.2017.05.068. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610217322750.

[68] G. Li, X. Zhao, C. Fan, X. Fang, F. Li, and Y. Wu, "Assessment of long short-term memory and its modifications for enhanced short-term building energy predictions", en, *Journal of Building Engineering*, vol. 43, p. 103 182, Nov. 2021, ISSN: 23527102. DOI: 10.1016/j.jobe.2021.103182. Accessed: May 15, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352710221010408.

[69] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, *A Transformer-based Framework for Multivariate Time Series Representation Learning*, arXiv:2010.02803 [cs], Dec. 2020. DOI: 10.48550/arXiv.2010.02803. Accessed: Apr. 12, 2025. [Online]. Available: http://arxiv.org/abs/2010.02803.

[70] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting", in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 22 419–22 430. Accessed: Jan. 9, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

[71] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting", en, in *Proceedings of the 39th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jun. 2022, pp. 27 268–27 286. Accessed: Jan. 9, 2025. [Online]. Available: https://proceedings.mlr.press/v162/zhou22g.html.

[72] R. Child, S. Gray, A. Radford, and I. Sutskever, *Generating Long Sequences with Sparse Transformers*, arXiv:1904.10509 [cs], Apr. 2019. DOI: 10.48550/arXiv.1904.10509. Accessed: Apr. 15, 2025. [Online]. Available: http://arxiv.org/abs/1904.10509.

[73] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, *Compressive Transformers for Long-Range Sequence Modelling*, arXiv:1911.05507 [cs], Nov. 2019. DOI: 10.48550/arXiv.1911.05507. Accessed: Apr. 15, 2025. [Online]. Available: http://arxiv.org/abs/1911.05507.

[74] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*, arXiv:1901.02860 [cs], Jun. 2019. DOI: 10.48550/arXiv.1901.02860. Accessed: Apr. 15, 2025. [Online]. Available: http://arxiv.org/abs/1901.02860.

[75] I. Beltagy, M. E. Peters, and A. Cohan, *Longformer: The Long-Document Transformer*, en, arXiv:2004.05150 [cs], Dec. 2020. DOI: 10.48550/arXiv.2004.05150. Accessed: Apr. 15, 2025. [Online]. Available: http://arxiv.org/abs/2004.05150.

[76] N. Kitaev, Kaiser, and A. Levskaya, *Reformer: The Efficient Transformer*, arXiv:2001.04451 [cs], Feb. 2020. DOI: 10.48550/arXiv.2001.04451. Accessed: Apr. 15, 2025. [Online]. Available: http://arxiv.org/abs/2001.04451.

[77] S. Liu et al., "Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting", en, Oct. 2021. Accessed: Apr. 12, 2025. [Online]. Available: https://openreview.net/forum?id=0EXmFzUn5I.

[78] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial Sparse Transformer for Time Series Forecasting", in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 17 105–17 115. Accessed: Apr. 12, 2025. [Online]. Available: https://papers.nips.cc/paper_files/paper/2020/hash/c6b8c8d762da15fa8dbbdfb6baf9e260-Abstract.html.

[79] W. Chen, W. Wang, B. Peng, Q. Wen, T. Zhou, and L. Sun, "Learning to Rotate: Quaternion Transformer for Complicated Periodical Time Series Forecasting", in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22, New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 146–156, ISBN: 978-1-4503-9385-0. DOI: 10.1145/

3534678.3539234. Accessed: Apr. 12, 2025. [Online]. Available: https://doi.org/10.1145/3534678.3539234.

[80] B. Tang and D. S. Matteson, "Probabilistic Transformer For Time Series Analysis", in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 23 592–23 608. Accessed: Apr. 12, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/c68bd9055776bf38d8fc43c0ed283678-Abstract.html.

[81] Y. Lin, I. Koprinska, and M. Rana, *SSDNet: State Space Decomposition Neural Network for Time Series Forecasting*, arXiv:2112.10251 [cs], Dec. 2021. DOI: 10.48550/arXiv.2112.10251. Accessed: Apr. 12, 2025. [Online]. Available: http://arxiv.org/abs/2112.10251.

[82] Y. Zhang and J. Yan, "Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting", en, Sep. 2022. Accessed: Apr. 12, 2025. [Online]. Available: https://openreview.net/forum?id=vSVLM2j9eie.

[83] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. Accessed: May 16, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

[84] J. MacQueen, "Some methods for classification and analysis of multivariate observations", in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, University of California press, vol. 5, 1967, pp. 281–298.

[85] P. Li, S. He, P. Han, M. Zheng, M. Huang, and J. Sun, "Short-term load forecasting of smart grid based on long-short-term memory recurrent neural networks in condition of real-time electricity price", *Power System Technology*, vol. 42, no. 12, pp. 4045–4052, 2018.

[86] S. J. Taylor and B. Letham, "Forecasting at scale", *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[87] M. Alharthi and A. Mahmood, "Xlstmtime: Long-term time series forecasting with xlstm", *AI*, vol. 5, no. 3, pp. 1482–1495, 2024.

[88] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift", in *International conference on learning representations*, 2021.

[89] M. Kraus, F. Divo, D. S. Dhami, and K. Kersting, *xLSTM-mixer: Multivariate time series forecasting by mixing via scalar memories*, Oct. 23, 2024. DOI: 10.48550/arXiv.2410.16928. arXiv: 2410.16928[cs]. Accessed: May 27, 2025. [Online]. Available: http://arxiv.org/abs/2410.16928.

[90] Z. Li, S. Qi, Y. Li, and Z. Xu, "Revisiting long-term time series forecasting: An investigation on linear mapping", *arXiv preprint arXiv:2305.10721*, 2023.

[91] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, 2023, pp. 11 121–11 128.

[92] K. Djebko et al., "Integrated simulation and calibration framework for heating system optimization", *Sensors*, vol. 24, no. 3, 2024, ISSN: 1424-8220. DOI: 10.3390/s24030886. [Online]. Available: https://www.mdpi.com/1424-8220/24/3/886.

[93] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[94] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[95] J. Ansel et al., "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation", in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr. 2024. DOI: 10.1145/3620665.3640366. [Online]. Available: https://docs.pytorch.org/assets/pytorch2-2.pdf.

[96] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: https://www.wandb.com/.

[97] "Temporal fusion transformer repository", Accessed: Jun. 25, 2025. [Online]. Available: https://github.com/google-research/google-research/tree/master/tft.

[98] "Xlstm repository", Accessed: Jun. 25, 2025. [Online]. Available: https://github.com/NX-AI/xlstm.

[99] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[100] R. Raj, *Activation functions for output layer in neural networks*, https://www.enjoyalgorithms.com/blog/how-to-choose-activation-function-for-output-layer, Accessed: 31.05.2025.

[101] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[102] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks", in *International conference on machine learning*, PMLR, 2017, pp. 933–941.

[103] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)", *arXiv preprint arXiv:1511.07289*, 2015.

[104] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization", *arXiv preprint arXiv:1607.06450*, 2016.

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

[106] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster", *arXiv preprint arXiv:1711.11053*, 2017.

[107] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, *Applied linear statistical models*. McGraw-hill, 2005.

[108] K. Pearson, "Vii. mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia", *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, no. 187, pp. 253–318, 1896.

[109] D. Ruppert, *The elements of statistical learning: Data mining, inference, and prediction*, 2004.

[110] J. Wang, W. Jiang, Z. Li, and Y. Lu, "A new multi-scale sliding window lstm framework (mssw-lstm): A case study for gnss time-series prediction", *Remote Sensing*, vol. 13, no. 16, 2021, ISSN: 2072-4292. DOI: 10.3390/rs13163328. [Online]. Available: https://www.mdpi.com/2072-4292/13/16/3328.

[111] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review", *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, Jul. 2016, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2015.11.011. Accessed: Mar. 19, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207015001508.

[112] R. A. Maller, G. Müller, and A. Szimayer, "Ornstein–Uhlenbeck Processes and Extensions", en, in *Handbook of Financial Time Series*, T. Mikosch, J.-P. Kreiß, R. A. Davis, and T. G. Andersen, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 421–437, ISBN: 978-3-540-71296-1 978-3-540-71297-8. DOI: 10.1007/978-3-540-71297-8_18. Accessed: Jul. 6, 2025. [Online]. Available: https://link.springer.com/10.1007/978-3-540-71297-8_18.

[113] D. Nikolaiev, "How to estimate the number of parameters in transformer models", *Towards Data Science*, 2023, Accessed: July 2025. [Online]. Available: https://towardsdatascience.com/how-to-estimate-the-number-of-parameters-in-transformer-models-ca0f57d8dff0.

[114] A. Das, W. Kong, R. Sen, and Y. Zhou, *A decoder-only foundation model for time-series forecasting*, 2024. arXiv: 2310.10688 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.10688.

[115] A. Auer, P. Podest, D. Klotz, S. Böck, G. Klambauer, and S. Hochreiter, "TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning", *ArXiv*, vol. 2505.23719, 2025.

[116] M. Wild, "Foundation models for time series forecasting with exogenous variables: A survey and benchmark", Department of Media, M.S. thesis, Fachhochschule Kiel, Kiel, Germany, Jun. 2025. [Online]. Available: https://github.com/martinwild97/fm-tsf-with-covs.

Erklärung

Ich versichere, die vorliegende Masterarbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur verfasst zu haben. Diese Erklärung erstreckt sich auch auf die in der Arbeit enthaltenen Abbildungen. Darüber hinaus versichere ich, die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde zur Erlangung eines akademischen Grades vorgelegt zu haben.

Im Rahmen dieser Arbeit wurden KI-Tools für die folgenden Aufgaben angenwendet:

- Formulierungshilfen (z.B., Verbesserung von Satzbau, Wortwahl, Stilistik usw.)
- Automatische Textgenerierung aus Stichpunkten

Die generierten Inhalte wurden von mir auf ihre Korrektheit hin geprüft. Ich übernehme für alle generierten Inhalte die vollumfängliche Verantwortung.

Würzburg, den 21.07.2025

(Marja Wahl)

This document was created using LaTeX.
The template is based on the LaTeX template
of the Chair of Computer Science 11, version 1.4.