LUND UNIVERSITY
School of Economics and Management
Master's Programme in Data Analytics & Business Ecnomics

Evaluating LSTM Neural Networks for Energy Forecasting in CHP Plants

A Comparative Approach

by

Felix Hult & Evagelos Ifantidis

Abstract: This paper evaluates the performance of the Long Short-Term Memory (LSTM) Neural Network for forecasting energy production from combined heat and power (CHP) plants in Stuttgart, Germany. The dataset consists of high frequency time series data and exhibits multiple seasonal patterns. A key challenge addressed in this study is the presence of a systematic reporting delay, where the most recent four days of data are not available for forecasting. To handle this, a hybrid approach is adopted, using a LightGBM model to impute the missing data caused by the reporting delay. This model also serves as a benchmark, alongside a dynamic harmonic regression model. The results show that the LSTM model is neither more accurate nor more computationally efficient than the LightGBM model. However, both models outperform the statistical and naïve benchmarks. Increasing sequence length for an LSTM model does not necessarily increase forecasting accuracy due to excessively long input sequences with high frequency data. This paper adds to the existing literature regarding advanced machine learning models' role in energy forecasting.

DABN01
Master's Thesis (15 credits ECTS)
May 2025
Supervisor: Dr. Simon Reese

Acknowledgements

We would like to extend our gratitude to Rickard Green at Energy Quant Solutions for providing us with this case and the data. With immense knowledge in the field of machine learning as well as domain knowledge in the energy sector, Rickards insights have been invaluable for us. We would also like to thank our supervisor Simon Reese for his support and guidance whenever questions arose during the writing of this thesis.

Contents

1

Introduction

1.1 Background

Following the expansion of renewable energy and recent geopolitical events, particularly the 2022 Russian invasion of Ukraine, European countries have made energy security a top priority (International Energy Agency, 2023). As a result, the need for grid stability and reliable energy supply has become increasingly important. For grid operators, ensuring stability, safety, and cost efficiency in power systems has become a key priority (Wan et al., 2023). In light of this, the need for accurate and reliable forecasting has emerged as a priority, not just for European countries, but for researchers and academics alike.

1.2 The Value of Forecasting Energy

Accurate forecasting enables grid operators and energy producers to anticipate demand fluctuations, optimize resource allocation and better deal with risks associated with supply and demand imbalances (Teixeira et al., 2024). Wan et al. (2023) note that power grids face many challenges due to elastic energy demand and the integration of more renewable energy sources, and to ensure a system's safety and cost-efficiency, short-term load forecasting has emerged as an important area of research and development. This need for forecasting extends beyond the common power grid to CHP plants, where both heat and power must be optimized simultaneously. Szega et al. (2022) states that the process of optimal planning of heat and power generated from CPH is one of the most important aspects influencing economic efficiency.

1.3 Combined Heat and Power

Combined Heat and Power (CHP), or cogeneration, simultaneously generates electricity and heat from a single fuel source. Instead of wasting heat during power generation, CHP systems recover it for heating, hot water, cooling or industrial use (CHP Alliance, n.d.) CHP is commonly used for facilities with steady electricity and thermal demand such as hospitals, universities, and manufacturing sites, and can be scaled from individual buildings to large utility managed installations. With efficiencies of 65-85%, CHP systems outperforms conventional separate generation

methods, reducing fuel and emissions (CHP Alliance, n.d.). While many use natural gas, newer systems increasingly incorporate low carbon fuels like biogas or hydrogen. CHP also enhances energy resilience in microgrids, localized networks that can operate independently during grid outages, making it valuable for reliable, decentralized energy supply.

1.4 Aim and Scope

Having access to energy production data from combined heat and power (CHP) plants in Stuttgart, Germany, this study aims to evaluate the viability of modern machine learning techniques in time series forecasting of energy production. Unlike traditional academic forecasting settings, this study incorporates several practical constraints that must be considered for real-time forecasting applications.

First, there is a data latency of approximately four days, which means that CHP plant measurements are not available for immediate use and analysis. The proposed methods and models must take this into account. Second, the data is recorded at fifteen-minute intervals, resulting in a high frequency time series with a large number of observations. This introduces significant complexity in terms of input sequence scaling for deep learning models. Third, the dataset contains daily, weekly, and annual seasonal patterns, and the high temporal granularity makes these patterns more challenging to model.

To address these challenges and add to the existing energy forecasting literature, this study proposed a hybrid forecasting framework that makes use of forecast imputations from a LightGBM model into the input sequence of an LSTM neural network. The objective is to assess and evaluate how well this approach handles the latency problem, sequence scaling, and complex seasonality by comparing and benchmarking against a naive baseline and a traditional statistical model.

2

Previous Research

In the context of energy forecasting, and forecasting in general, the most common models usually fall into two categories, those being traditional time-series methods (statistical methods) and machine learning methods (Elsworth and Güttel, 2020). The debate of whether machine learning methods such as neural networks outperform traditional statistical models has evolved over the years. Earlier studies usually favored statistical approaches, but the advancement in the field of machine learning, neural network architecture, and computational power is shifting the perspective. A study by Makridakis et al. (2018b) showed that traditional statistical models outperformed machine learning models both in terms of precision measured in symmetric mean absolute percentage error (sMAPE) and mean absolute scaled error (MASE), and computational efficiency measured as the computational time for the model tested divided by the computational time of a baseline model. Makridakis et al. (2018b) concluded the paper by stating that even though many studies find success in implementing machine learning models, they need to be compared to other benchmark models to avoid biases.

The shift from the dominance of statistical models to machine learning models can be showcased by examining the Makridakis Competitions throughout the years, a competition specifically made for testing forecasting models. The results from the M4 competition, which ended on may 31, 2018, likewise showed that the top performing models were a combination of statistical models while the purely machine learning submissions all performed poorly (Makridakis et al., 2018a). However, the following competition, the M5, where the objective was to predict the hierarchical unit sales for Walmart, the top performing models were all machine learning based. The majority of the top performers utilized an average of different LightGBM model predictions as well as combinations of neural networks, showcasing the promise of machine learning models and ensemble predictions in the field of forecasting (Makridakis et al., 2022).

In the realm of classical time series models, also commonly referred to as statistical models, the Autoregressive integrated Moving Average (ARIMA), first proposed by Box & Jenkins in 1976, and its extension SARIMA and SARIMAX has long been used in energy forecasting and are still highly competitive due to their low complexity and interpretability (Spiliotis, 2023). Studies utilizing these statistical models usually show great results such as the study from Makridakis et al. (2018b) which showed that models such as ETS (Exponential Smoothing) and ARIMA consistently outperformed machine learning models across different forecasting horizons.

Despite continuously showing good performance, traditional time series models such as ARIMA usually face several limitations such as not being able to handle multiple seasons, limited ability to capture non-linear patterns and high computational cost for large datasets (Bansal et al., 2025). This has led to the adoption of machine learning methods. Neural networks particularly show promise as recurrent neural networks has become a competitive forecasting method.

Zhou et al. (2019) utilized an ARIMA, a back-propagation neural network, and an LSTM model to forecast the daily and hourly energy consumption of air conditioning units in a university library in Guangzhou, China. They found that the LSTM model reduced the Mean Absolute Percentage Error (MAPE) by 11,2% and 49% for daily forecasts compared to ARIMA and BP.

LSTM models can be employed in combination with other architectures to form a hybrid model. A study done by Wan et al. (2023) combined a CNN and a LSTM combined with an attention mechanism for forecasting of CHP systems and increased accuracy by 7.3% compared to a regular LSTM. This approach was proposed to address the issue of information loss when the time series input data is excessively long, which a singular LSTM model struggles with (Wan et al., 2023). The CNN layers help extract the most relevant features while the LSTM captures the temporal dependencies in the data.

A recent advancement in the field of neural networks has sparked interest in what is called the attention mechanism, first introduced in the paper "Attention is all you need" by Vaswani et al. (2023). These mechanisms influenced fields such as natural language processing as well as time series forecasting. Lin et al. (2020) as well as Wan et al. (2023) utilized attention in their LSTM models. Lin et al. (2020) used this to forecast short-term electricity consumption, where the attention mechanisms assign weights to different time steps in an input sequence allowing the model to focus on the most important patterns. This model achieved a 6.5 % increase in prediction accuracy compared to a vanilla LSTM.

3

Models

The following section provides a brief introduction and discussion about the models that will be used in this paper.

3.1 Statistical Models

The ARIMA model, its seasonal extensions SARIMA and SARIMAX (for exogenous variables) have been used frequently in the forecasting literature. A non-seasonal ARIMA model is a combination of time series differencing, autoregression and a moving average model, giving it the name Autoregressive Integrated Moving Average, taking the form as shown in Eq. 3.1.

$$y'_t = c + \sum_{i=1}^{p} \phi_i y'_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{3.1}$$

where y'_t is the differenced series, ϕ_i are the AR coefficients, θ_j are the MA coefficients, and ε_t is the white noise error term. This is usually called an ARIMA (p,d,q) model where p is the order of the autoregressive part, d the degree of first differencing involved and q the order of the moving average part. The SARIMA model introduces seasosal AR, I and MA components with seasonal periods s, denoted SARIMA(p,d,q)x(P,D,Q)s. When including exogenous variables as regressors it becomes SARIMAX.

A drawback of the seasonal ARIMA models is that they are designed for shorter periods such as 12 for monthly data and 4 for quarterly (Hyndman and Athanasopoulos, 2018). Shorter time steps, such as hourly data, usually exhibit more than one seasonal pattern. Hyndman and Athanasopoulos (2018) points out that it doesn't make sense to do seasonal differencing of high order and instead suggest a dynamic harmonic regression approach.

A Dynamic Harmonic Regression model (DHR) includes fourier terms to handle data with multiple seasonal cycles. By including fourier terms of different frequencies the seasonality in the data can be modelled and be further tuned by the hyperparameter K which controls the number of fourier sin and cos pairs. The short term dynamics are modelled with simple ARIMA errors, (see Eq. 3.2. A drawback of this approach is that seasonality is assumed to be fixed, as compared to a SARIMA model. However, in practice, seasonal cycles are usually somewhat constant for long time series (Hyndman and Athanasopoulos, 2018).

$$y_t^* = \sum_{k=1}^{K} \left(\alpha_k \cos\left(\frac{2\pi kt}{T}\right) + \gamma_k \sin\left(\frac{2\pi kt}{T}\right) \right) + \varepsilon_t, \quad \varepsilon_t \sim \text{ARIMA}(p, d, q) \tag{3.2}$$

Where:

y_t^*	is the predicted value at time t,
α_k, γ_k	are Fourier coefficients for the cosine and sine terms,
K	is the number of Fourier terms (harmonics),
T	is the seasonal period,
t	is the time index,
ε_t	is the residual term, modeled as ARIMA(p, d, q).

3.2 Machine Learning Models

3.2.1 Feed Forward Neural Networks

At their simplest, neural networks take the form of a Feed-Forward Neural Network (FFNN), also known as a Multilayer Perceptron (MLP). The objective of these networks is to approximate some unknown function f^* by learning from data (Goodfellow et al., 2016). FFNNs, and neural networks in general, consist of multiple layers of interconnected neurons, which are individual units that apply a learned weight to their inputs and pass the results through a non-linear activation function. The composition of these layers becomes a chain of functions where the number of layers is usually referred to as the depth of the network (*hence the name deep learning*).

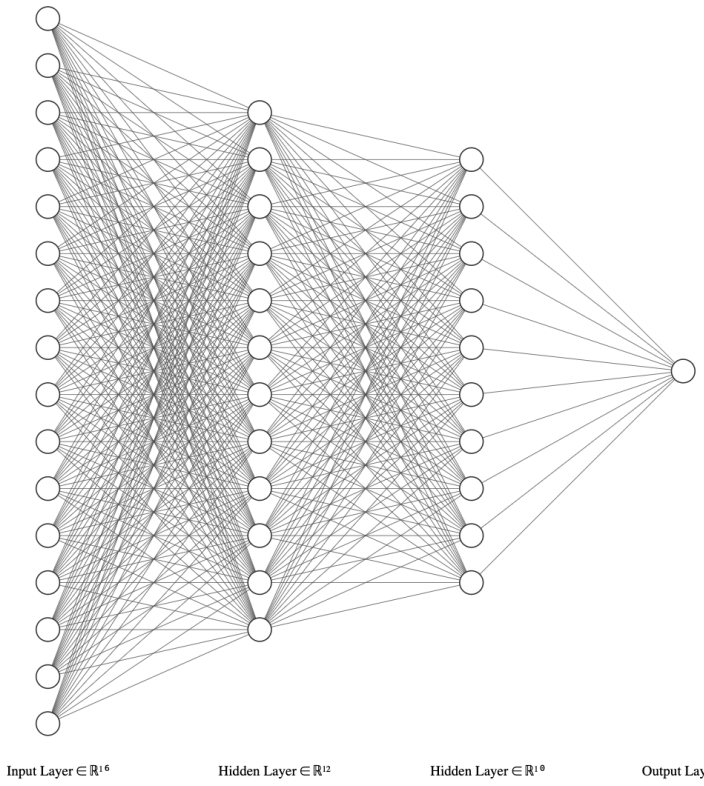

Figure 3.1: Feed forward neural network architecture

An FNN processes the data in one direction without feedback or memory of previous inputs. In contrast to linear models, such as linear regression, there is no closed-form solution to find the best parameters in a neural network due to their non-linear structure. NNs are instead trained iteratively through gradient-based optimizers.

3.2.2 Recurrent Neural Networks

Several studies point toward the superiority of Recurrent Neural Networks (RNN) for sequential data such as time series (Dubey et al., 2021; Siami-Namini et al., 2018). RNNs are a kind of neural network that specializes in processing sequences of values (Goodfellow et al., 2016). In contrast to regular feedforward neural networks, RNNs can scale to much longer sequences, which would not be practical for a feedforward network (Goodfellow et al., 2016).

The main difference between an FFNN and an RNN is that in RNNs, parameters are shared across different parts of the model. Furthermore, RNNs operate on sequences $x_{(t)}$ indexed by time step t, and maintain what is called a hidden state h_t, which contains memory from previous time steps. This is expressed as:

$$h_t = f(h_{t-1}, x_t; \boldsymbol{\theta}) \tag{3.3}$$

In an RNN without an output layer, the structure takes the form shown in Figure 3.3.

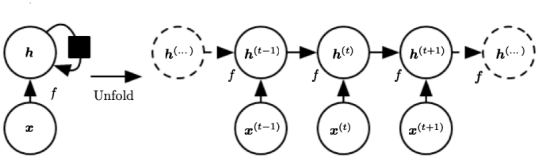

Figure 3.2: Unfolded Recurrent Neural Network without output layer (Goodfellow et al., 2016)

3.2.3 Long Short-Term Memory Recurrent Neural Network

Despite the theoretical capacity, vanilla RNNs in their base form exhibit the problem of vanishing gradients when sequences get too long. To combat this problem of the vanishing gradient a variant of a RNN was introduced called LSTM (Long Short-Term Memory) by Hochreiter and Schmidhuber (1997). By introducing a cell state and gating units that regulate the flow of information into the RNN architecture, the LSTM model can remember information over longer periods of time and can effectively capture short and long-term dependencies.

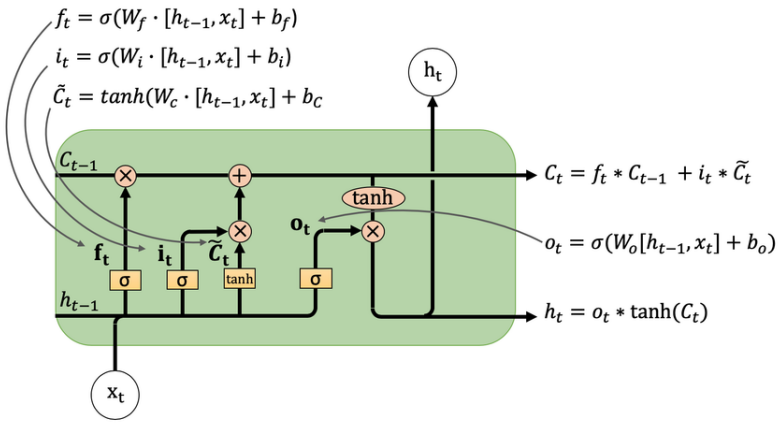

Figure 3.3: The LSTM cell architecture (Kizito et al., 2021).

Table 3.1: LSTM cell computations at time step t.

Forget Gate	$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$
Input Gate	$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$
Candidate Cell State	$\tilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$
Cell State Update	$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$
Output Gate	$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$
Hidden State	$h_t = o_t \odot \tanh(c_t)$

The LSTM architecture involves a memory cell in which three gates control the flow of information for each time step. These include the input gate, the forget gate, and the output gate. These gates control what information to add or remove from the memory cell, ultimately allowing the network to retain information from long-term dependencies.

At each time step, the LSTM model receives the current input x_t, the previous hidden state h_{t-1}, and the previous cell state c_{t-1}, while using the gates to control the information flow. This gating mechanism allows LSTMs to retain information over longer periods while mitigating the vanishing gradient issue, ultimately making them effective for time series forecasting (Ahmed et al., 2022).

3.2.4 LightGBM

The LightGBM model (Light Gradient Boosting Machine) was introduced by Ke et al. (2017) and is based on the Gradient Boosting Decision Tree method (GBDT) but built for scalability and efficiency. GBDT methods build ensembles of decision trees in a stage-wise fashion where each tries to correct the mistakes of the previous tree. With the emergence of big data, Ke et al. (2017) claims that traditional GBDT methods face a challenge in the trade-off between accuracy and efficiency, since for each feature, it needs to scan all data instances to estimate the information gain for all split points.

To combat this problem Ke et al. (2017) introduces what is called Gradient Based One Side Sampling (GOSS). GOSS keeps instances with large gradients since these carry more information about model error, and randomly samples from those with small gradients. This ultimately leads to faster training and lower computational cost. Another innovation found in the LightGBM model is Exclusive Feature Bundling (EFB). This technique bundles features into a single feature, essentially reducing the dimensions, which helps further the computational savings.

The study demonstrated that compared to a related GBDT model, XGBoost, LightGBM achieved significantly better computational speed and memory efficiency, while maintaining comparable accuracy.

4

Data

4.1 Combined Heat and Energy Production

This paper utilizes aggregated energy production data from several combined heat and power (CHP) plants in Stuttgart, Germany. The dataset covers the period from 1 January 2020 to 16 December 2024 and records measurements at fifteen-minute intervals, resulting in a total of 174,121 observations.

As is typical for energy production data, the CHP output exhibits clear seasonal patterns, including yearly, weekly, and daily cycles. On a yearly scale, production is generally higher in winter and lower in summer, as shown in Figure 4.1, which presents the complete dataset. This pattern is likely driven by increased heating and lighting demands during the colder months. In addition, as Germany presumably has relatively low cooling demand in summer, energy production tends to remain lower during this period.

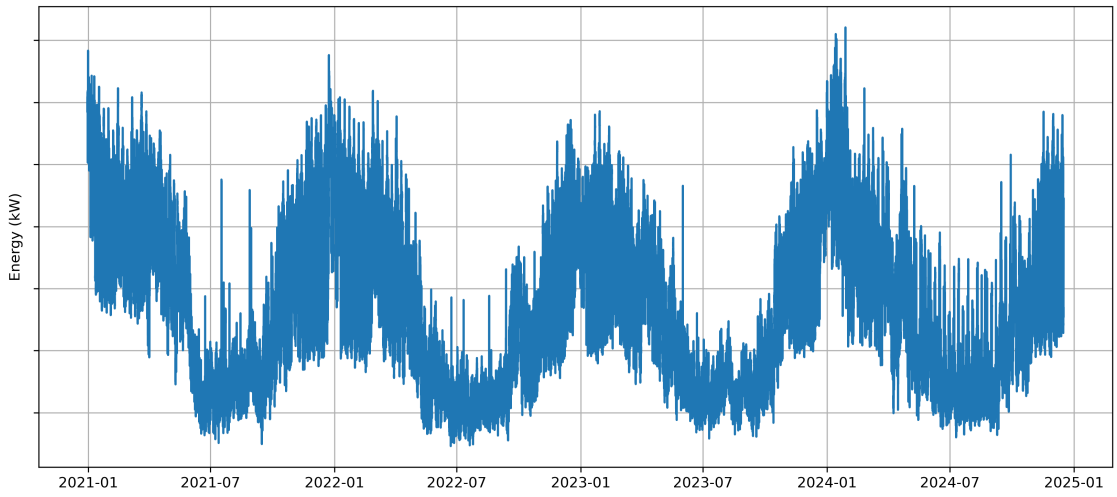

Figure 4.1: Quarter-hourly CHP energy production from 1 January 2020 to 16 December 2024.

The data also suggests that production variability is greater in winter than in summer, although occasional output spikes occur during the summer months. This

is illustrated in the box plot in Figure 4.2, which shows the monthly distribution of the dataset. A wider spread is evident during the winter months, while the summer months exhibit lower variability. The summer spikes appear as outliers in the box plot.

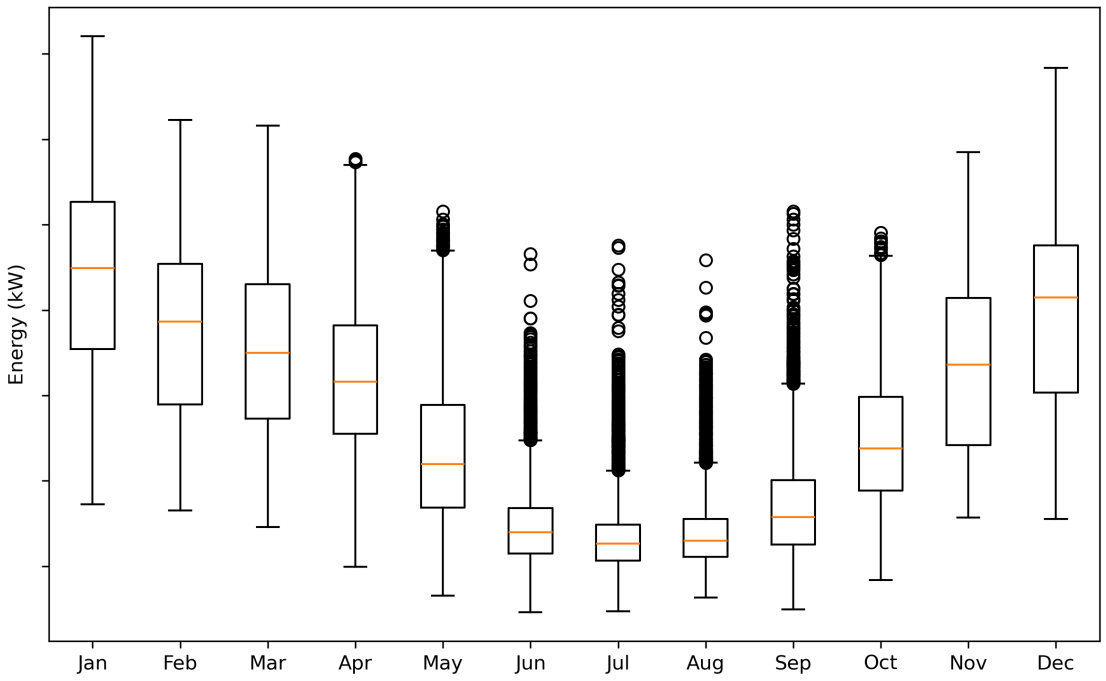

Figure 4.2: Distribution of CHP energy production across each month.

The weekly and daily cycles are illustrated in Figure 4.3, which shows the average quarter-hourly CHP production for each day of the week. Energy output typically peaks around 04:30 in the early morning before declining rapidly to its lowest point at approximately 09:00. Production then gradually increases throughout the day until about 18:30, after which it rises more sharply. Although the exact cause of the early morning peaks is not entirely clear, one possible explanation is the influence of off-peak heating contracts. Under these agreements, heating systems operate during off-peak hours to take advantage of lower energy prices, leading to increased CHP production during the early morning hours. The graph also highlights a clear distinction between weekdays and weekends, as energy production does not drop as sharply on weekends as it does during weekdays.

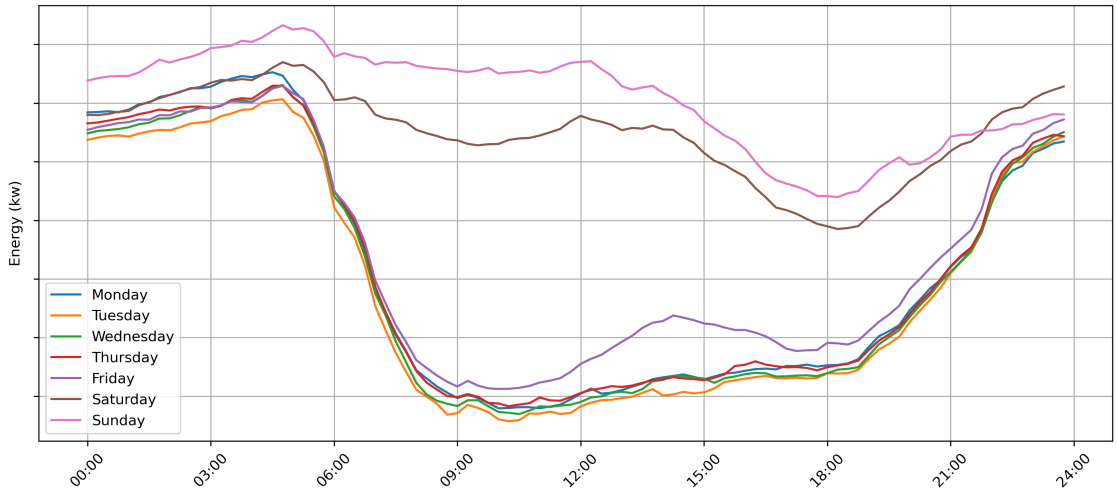

Figure 4.3: Average quarter-hourly CHP energy production by day of the week.

4.2 Temperature

Temperature data is included as an external variable, representing the hourly temperature in Stuttgart, Germany. As mentioned earlier, CHP energy production is generally higher during colder months and lower during warmer months. This relationship is clearly shown in Figure 4.4, which presents a strong negative correlation between CHP energy production and temperature, with a correlation coefficient of -0.769. This indicates that lower temperatures lead to increased energy demand and, in turn, higher CHP production, while higher temperatures reduce energy demand and lower production. The strong correlation also suggests that temperature is an important factor to consider when forecasting CHP energy output.

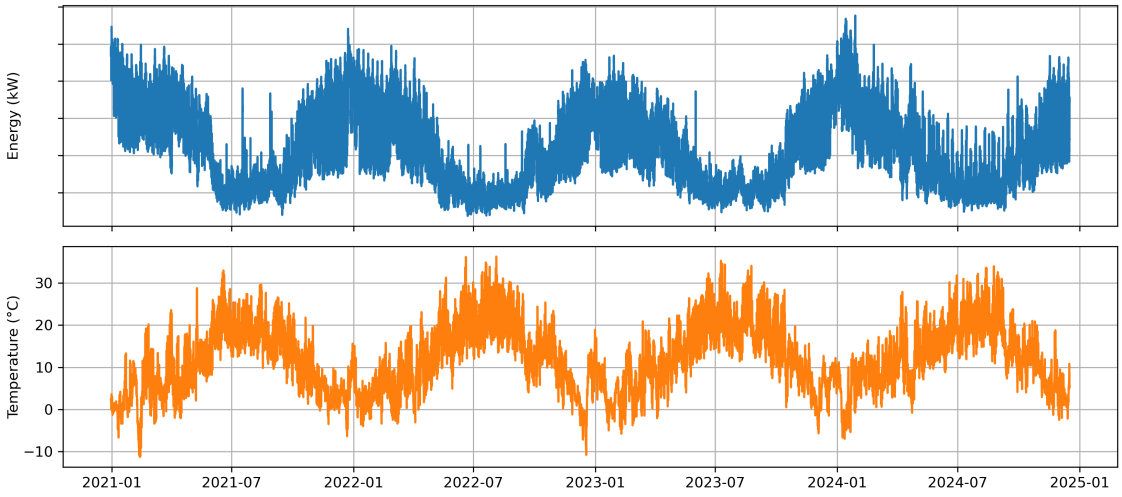

Figure 4.4: CHP energy production and temperature over time.

4.3 Reporting Delay

An important characteristic of the dataset is the presence of a reporting delay, where realized CHP production data becomes available only after four days. As a result,

the most recent historical data cannot be used when generating forecasts. For example, to forecast production for 18 December at 14:30, only data up to 14 December at 23:45 is available. This delay creates a challenge, as valuable recent information is lost, and short-term lags or rolling statistics within this four-day period cannot be used to inform the forecasts. The strategy for addressing this limitation and compensating for the missing information is further detailed in Section 5.6.2.

5

Methodology

This chapter outlines the methodological choices made in the study, beginning with the overall modeling approach in Section 5.1. Data preprocessing is discussed in Section 5.2, followed by feature extraction in Section 5.3. The design choices for each model are presented in Section 5.4, Section 5.5, and Section 5.6. The chapter concludes with the evaluation strategy, including error metrics and residual diagnostics, in Section 5.7.

5.1 Research Approach

The models and techniques used in this paper perform one step ahead deterministic forecasting. The main model examined is the long short-term memory (LSTM) neural network. However, due to the data availability delay discussed earlier, the models cannot use the most recent four days of input data, which presents a challenge. To ensure the forecasting procedure can operate under realistic, real time conditions, this constraint must be respected during model design.

To address this issue, an imputation procedure is employed. First, a separate model is trained to generate forecasts for the values missing due to the reporting delay. These forecasts are then used to replace the unavailable portion at the end of each input sequence, allowing the LSTM model to operate with complete inputs despite the data delay.

Two strategies are tested for the imputation procedure. The first is test-time imputation only, where the LSTM is trained on unaltered historical data, and imputation using external predictions is applied only at test time. This simplifies the training process, as the external model only needs to generate forecasts for the test period. However, it may introduce a mismatch between training and testing conditions, as the model has not encountered imputed values during training and may become overly reliant on recent lags that are fully observed in training but are replaced with less reliable external predictions during testing. The second strategy is train-time imputation, where the LSTM is trained and evaluated on data with the most recent four days imputed using external predictions. This allows the model to learn from the same type of inputs it will encounter in real-time forecasting and may help it adjust to errors in the imputed values. However, this approach requires more external forecasts, as the external model must generate predictions for both the training and testing periods to provide a complete imputed input for the LSTM.

Based on the findings of Ke et al. (2017) and Makridakis et al. (2022), this paper

uses LightGBM to impute the missing data caused by reporting delays. LightGBM is known for its computational efficiency and delivers prediction accuracy comparable to other gradient boosting decision tree (GBDT) models, making it well suited for scenarios where fast and reliable predictions are required. Since LightGBM operates on cross sectional data rather than sequences, it is not constrained by the four day data latency. Each input vector at time t, denoted X_t, is used independently to predict the corresponding energy output Y_t. This makes LightGBM a practical choice for filling in the unavailable values that occur at the end of the LSTM input sequences.

In addition to the LSTM model, this paper evaluates several benchmark models for comparison. As a simple baseline, a seasonal naïve forecasting model is used, which predicts each value using the corresponding value from exactly one week earlier. Hyndman and Athanasopoulos (2018) argue that such simple models are essential benchmarks, as any advanced model, whether statistical or machine learning, that fails to outperform them is not worth considering. To provide a traditional statistical comparison, a dynamic harmonic regression (DHR) model is also included. Makridakis et al. (2018b) highlight the importance of comparing modern machine learning models with established statistical methods, making the inclusion of DHR relevant for evaluating the LSTM's added value.

Finally, since LightGBM is already used to impute missing values caused by reporting delays, it is also evaluated as a standalone forecasting model. Comparing LightGBM directly with the LSTM offers insight into the trade-off between accuracy and computational efficiency. This is especially important given that all models in this study are developed with the goal of supporting real time forecasting.

5.2 Data Preprocessing

Before the energy production and temperature data could be used for analysis, preprocessing steps were necessary. The first step involved correcting for daylight saving time (DST) in the CHP energy production data. Since the dataset lacked time zone information, the end of DST, when clocks are set back, resulted in duplicate timestamps. To address this, the data was adjusted to Central European Time (CET), and the correct chronological order during the DST transition was inferred to ensure a consistent time sequence.

Second, the temperature data was upsampled from hourly to quarter-hourly frequency to match the energy data. Each hourly temperature reading was duplicated across the four corresponding quarter-hour intervals. Third, the energy and temperature datasets were merged based on their timestamps, ensuring that each energy measurement was paired with the appropriate temperature value. Finally, the merged dataset was checked for missing values, and none were found.

5.3 Feature Extraction

Apart from temperature, several other features are included to help the models capture time-based patterns and dependencies in the data. These features include lagged values of the target variable, rolling window statistics, holiday indicators, and

cyclical encodings of temporal features. All features used in the modeling process are described in Table 5.1 below.

5.3.1 Holiday Encoding

Holidays often disrupt typical energy consumption patterns, which can significantly affect CHP energy production. To account for this, a binary holiday indicator variable was created and included as a feature in the dataset. Using a predefined calendar of public holidays in Germany, specifically for the state of Baden-Württemberg (where the CHP plants are located), each timestamp in the dataset was checked against the holiday calendar. If a given date was a national or regional holiday, the corresponding entry was marked with a value of 1; otherwise, it was marked as 0. This encoding allows the forecasting models to learn and adjust their predictions based on whether a given observation falls on a holiday.

5.3.2 Addressing Seasonality with Cyclical Encoding

As previously mentioned, the CHP energy data exhibits clear seasonal patterns across daily, weekly, and yearly cycles, as well as nonconstant variance, with notably higher variance during winter months compared to summer. While there is no clear consensus on whether data must be deseasonalized before being fed into a neural network, some studies argue that removing strong seasonal signals may help the model focus on learning more complex patterns (Zhang and Qi, 2005).

Conversely, other research suggests that neural networks can learn seasonal dynamics directly from raw data, provided that temporal features are appropriately encoded. For example, Bansal et al. (2025) addressed this by applying cyclical encodings (sine and cosine transformations) to temporal variables such as hour of day, day of week, and month of year, which would otherwise be represented as categorical or dummy variables.

Given the presence of seasonality with varying variance in the dataset, this paper chooses not to deseasonalize the data prior to modeling. Instead, cyclical encodings are included to allow the models to learn seasonal patterns directly from the data.

To implement this, the hour of day, day of week, and month of year were first extracted from the timestamps. Each of these variables was then transformed into two continuous features using standard sine and cosine functions to preserve their cyclical structure. The transformations follow the formulas shown in Eq. 5.1 and Eq. 5.2.

$$x_{\sin}(t) = \sin\left(2\pi \cdot \frac{t}{P}\right) \tag{5.1}$$

$$x_{\cos}(t) = \cos\left(2\pi \cdot \frac{t}{P}\right) \tag{5.2}$$

where t is the original time-based variable and P is the period (24 for hours, 7 for days of the week, and 12 for months). For example, the hour variable, ranging from 0 to 23, was encoded using a period of 24, making sure that hour 0 and hour 23 are treated as close to each other in the transformed data. The same method was applied to the day of week and month variables using their respective periods. This encoding allows the model to learn smooth temporal patterns and seasonal effects

without introducing discontinuities that would arise from treating these variables as linear or categorical.

5.3.3 Time Series Features

While Long Short-Term Memory (LSTM) networks are designed to capture temporal dependencies directly from sequences of input data, gradient boosting models such as LightGBM do not have built in sequence handling. Therefore, to provide LightGBM with information about past values, a set of lag features and rolling window statistics was created.

Lag features were generated by shifting the target variable, energy output, by fixed intervals corresponding to one week, one month, and one year. These intervals were chosen to reflect typical seasonal patterns and recurring behaviors in the energy production data, and were defined based on the quarter hourly frequency: 672 steps for one week, 2,880 steps for one month, and 35,040 steps for one year. To comply with the data availability constraint mentioned earlier, where the most recent 384 time steps (corresponding to the last four days) are not available at prediction time, short term lags within this period were deliberately excluded to avoid data leakage.

In addition to lagged values of the target variable, rolling statistics were computed for the temperature variable. These included rolling means, maximums, and minimums over both one day and seven day windows (96 and 672 time steps, respectively). These features were intended to capture recent trends and extremes in temperature, which are likely to influence energy demand and CHP operation.

Finally, all rows with missing values resulting from the lag and rolling window calculations were removed, leaving 138,912 observations available for modeling.

Table 5.1: Variable description of the features used in this paper. Variables marked with (*) are used only for the LightGBM model.

Variable	Description
`temperature`	Hourly temperature in Stuttgart, upsampled to quarter-hourly frequency.
`is_holiday`	Indicator for public holidays in Baden-Württemberg (1 = holiday, 0 = non-holiday).
`hour_sin/hour_cos`	Cyclical encoding of hour of day (0 to 23).
`dayofweek_sin/dayofweek_cos`	Cyclical encoding of day of the week (Monday to Sunday).
`month_sin/month_cos`	Cyclical encoding of month of the year (January to December).
`lag_1w*`	Target variable lagged by one week (672 steps).
`lag_1m*`	Target variable lagged by one month (2880 steps).
`lag_1y*`	Target variable lagged by one year (35040 steps).
`temp_roll_mean_1_day*`	Mean temperature over the past 24 hours.
`temp_roll_max_1_day*`	Max temperature over the past 24 hours.
`temp_roll_min_1_day*`	Min temperature over the past 24 hours.
`temp_roll_mean_7_day*`	Mean temperature over the past 7 days.
`temp_roll_max_7_day*`	Max temperature over the past 7 days.
`temp_roll_min_7_day*`	Min temperature over the past 7 days.

5.4 LightGBM

LightGBM is a cross sectional model, meaning it does not process temporal sequences internally. Instead, it treats each observation independently and relies entirely on the features provided at each time step. As a result, any information about past behavior must be explicitly captured through engineered features. In this case,

the lagged values of the target variable and rolling window statistics described in Section 5.2 are included to provide the model with historical context.

Two separate instances of LightGBM model training and train-test splitting were conducted to support the different LSTM imputation strategies. For the test-time imputation approach, an 80/20 split was used, as LightGBM predictions were only needed to impute the test set. For the train-time imputation approach, a 50/50 split was used to ensure that enough LightGBM predictions were available to impute both the training and test sets, allowing for a meaningful evaluation of the LSTM model. The exact number of observations and date ranges for the training and test sets in each strategy are summarized in Table 5.2.

Table 5.2: Training and test splits for LightGBM models under different LSTM imputation strategies.

Imputation Strategy	Dataset	Date Range	Observations
Test-time imputation	Training	31 Dec 2020 – 2 Mar 2024	111,129
	Testing	2 Mar 2024 – 16 Dec 2024	27,783
Train-time imputation	Training	31 Dec 2020 – 24 Dec 2022	69,456
	Testing	24 Dec 2022 – 16 Dec 2024	69,456

5.4.1 Hyperparameter Tuning

Considering the number of parameters that can be adjusted in the LightGBM model, hyperparameter tuning is necessary to improve performance. Several approaches exist for tuning, including grid search and random search. In this study, the Optuna framework is used to optimize the LightGBM model's hyperparameters. Optuna is a free, open source framework for automated hyperparameter optimization. It uses a Tree-Structured Parzen Estimator (TPE) algorithm to efficiently explore the search space and adaptively guide the optimization process based on past evaluations (Akiba et al., 2019). The hyperparameters selected for tuning, along with the final values identified through the optimization process, are presented in Table 5.3.

Table 5.3: Hyperparameter tuning results for the LightGBM model, performed using Optuna. The best configuration was selected based on the lowest MAE.

Hyperparameter	Description	Optimal Value	
		Test-Time	Train-Time
num_leaves	Number of leaves (terminal nodes). Controls complexity of the tree.	56	160
max_depth	Maximum depth for the tree.	10	4
min_child_samples	Minimum number of data points in a leaf.	40	35
learning_rate	Boosting learning rate.	0.01975	0.01998
subsample	Subsample ratio of training instances.	0.67287	0.93950
colsample_bytree	Subsample ratio of columns when building each tree.	0.67400	0.59703
max_bin	Maximum number of bins for bucketing features.	175	100
reg_alpha	L1 regularization term on weights.	0.00036	0.02182
reg_lambda	L2 regularization term on weights.	0.00048	0.00374

5.5 Dynamic Harmonic Regression

Although many studies previously mentioned in this paper utilized an ARIMA model or a version of it as a benchmark model to compare any machine learning models towards, this paper chooses not to directly use an ARIMA-based model. The reason is simply that the fifteen-minute granularity introduces more than one seasonality, which a SARIMA model can't handle. As stated by Hyndman and Athanasopoulos (2018), ARIMA models are designed for shorter seasonal periods such as 12 for monthly data. If an ARIMA model were to be used in this case to capture a yearly seasonality that would amount to 35,040 time steps, it would be infeasible for an ARIMA model, which usually runs out of memory when a seasonal period is more than 200 (Hyndman and Athanasopoulos, 2018). Hyndman and Athanasopoulos (2018) also suggest that in such cases a Dynamic Harmonic Regression model (DHR) is to be preferred.

This paper uses a Dynamic Harmonic Regression model to act as the statistical model baseline when compared to the LSTM model. By using this model, we can capture the multiple seasonalities by use of Fourier terms while incorporating the time series dynamics by utilizing the autoregressive terms and moving averages.

$$y_t^* = \sum_{k=1}^{K} \left(\alpha_k \cos\left(\frac{2\pi kt}{T}\right) + \gamma_k \sin\left(\frac{2\pi kt}{T}\right) \right) + \varepsilon_t, \quad \varepsilon_t \sim \text{ARIMA}(p,d,q) \tag{5.3}$$

The model incorporates temperature as an exogenous regressor and utilizes Fourier terms to model seasonality. Specifically, daily, weekly, and yearly seasonalities are captured with seasonal periods $T_d = 96$, $T_w = 672$, and $T_y = 35136$, using $K_d = 4$, $K_w = 2$, and $K_y = 1$ harmonics, respectively. The residuals ε_t are modeled with an ARIMA(1,1,1) process. The target variable y_t is Box-Cox transformed prior to modeling.

K is the hyperparameter deciding how many fourier terms per seasonal period and controls the complexity and flexibility of the seasonal components of the model. Since the yearly season is fairly constant there is no need for a high K as it adds extra computational cost, rather the seasons that need more flexibility appear to be daily hence higher value of K chosen for daily harmonics. Variables used in the model is presented in Table 5.4

The ARIMA components are set to (1,1,1) meaning it uses an autoregressive component with lag p=1, an integrated component with order d=1 which applies first order differencing and a moving average component with lag q=1.

A common transformation of data in forecasting is what's called a *Box-Cox transformation* proposed by Box and Cox (1964). The motivation for it is to ensure that data conformed to assumptions about normality and constant error variance (Petropoulos et al., 2022). The transformation is applied to the data entering the DHR model (see Eq. 5.4).

$$y^{(\lambda)} = \begin{cases} \dfrac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y) & \text{if } \lambda = 0 \end{cases} \tag{5.4}$$

The model is implemented with a rolling window of 90 days (8640 time steps) in which a model is trained to forecast the day ahead (96 time steps) for the entire

Table 5.4: Model parameters used in the DHR model

Parameter	Description
`day_sin_1`, `day_cos_1`	First harmonic of daily seasonality.
`day_sin_2`, `day_cos_2`	Second harmonic of daily seasonality.
`day_sin_3`, `day_cos_3`	Third harmonic of daily seasonality.
`day_sin_4`, `day_cos_4`	Fourth harmonic of daily seasonality.
`week_sin_1`, `week_cos_1`	First harmonic of weekly seasonality.
`week_sin_2`, `week_cos_2`	Second harmonic of weekly seasonality.
`year_sin_1`, `year_cos_1`	First harmonic of yearly seasonality.
`temperature`	Hourly temperature, upsampled to 15-min frequency.
`ar.L1`	Autoregressive term at lag 1.
`ma.L1`	Moving average term at lag 1.
`sigma2`	Variance of the error term.

testing data (80%). These training and forecasting horizons were chosen to balance comparability with the LSTM model with computational feasibility. The last four days of each window thus consists of the LightGBM predictions, the same as the ones used for the LSTM.

5.6 LSTM

In contrast to the cross sectional nature of LightGBM, the LSTM model is designed to process sequential data. Rather than treating each observation independently, LSTM takes sequences of past values as input, allowing it to learn temporal dependencies directly from the data. As a result, there is no need to include manually engineered lagged features or rolling window statistics in the input. The LSTM model relies on its internal memory to capture temporal patterns, making these types of features redundant.

5.6.1 Input Sequences

To prepare the data for the LSTM model, the tabular time series is converted into a sequence-based format. This is done by sliding a fixed-size window across the dataset to extract overlapping sequences of past observations. Each sequence consists of a defined number of consecutive time steps and is used as one input sample. The corresponding target is the energy output value that occurs immediately after the end of the sequence. By shifting the window forward one step at a time and repeating this process across the dataset, a large number of input-target pairs are generated.

The length of the input sequence plays an important role in the model's ability to learn time based dependencies. As noted by Al-Selwi et al. (2023), longer sequences provide the model with more opportunity to learn relevant patterns and contextual information. This is particularly important for CHP energy data, which displays clear seasonal patterns across daily (day and night), weekly (weekdays and weekends), and yearly (winter and summer) cycles. Providing sufficient historical context helps the model capture these dynamics more effectively. However, the same study also shows that excessively long sequences can hinder model performance. As

sequence length increases, the LSTM may struggle to retain all relevant information, which can negatively impact its ability to make accurate predictions. In addition, because the data is recorded at fifteen minute intervals, longer sequences rapidly increase the input size. For instance, one week corresponds to 672 time steps, and a full year would result in 35040 time steps, significantly increasing model complexity and computational cost.

To address this, the time based encoding features described earlier are included, as suggested by Bansal et al. (2025). These are intended to help the model recognize repeating patterns without requiring excessively long input sequences, which has been identified as a limitation of LSTM models (Wan et al., 2023). In this study, sequence lengths of one, two, and three weeks (672, 1,344 and 2,016 time steps) are tested using cross validation to evaluate their effect on forecasting performance.

5.6.2 Data Imputation

As mentioned in Section 5.1, this paper explores two imputation strategies. In the test-time imputation strategy, only the test set is imputed, while in the train-time imputation strategy, both the training and test sets are imputed. The imputation procedure is illustrated in Figure 5.1 using a simplified example. To address the four day delay in data availability, which corresponds to 384 time steps at fifteen minute intervals, missing values at the end of each input sequence in the test set are replaced with forecasts generated by the separately trained LightGBM model. During imputation, only the final 384 time steps of each test sequence are substituted with LightGBM predictions, while the remaining portion of the sequence remains unchanged.

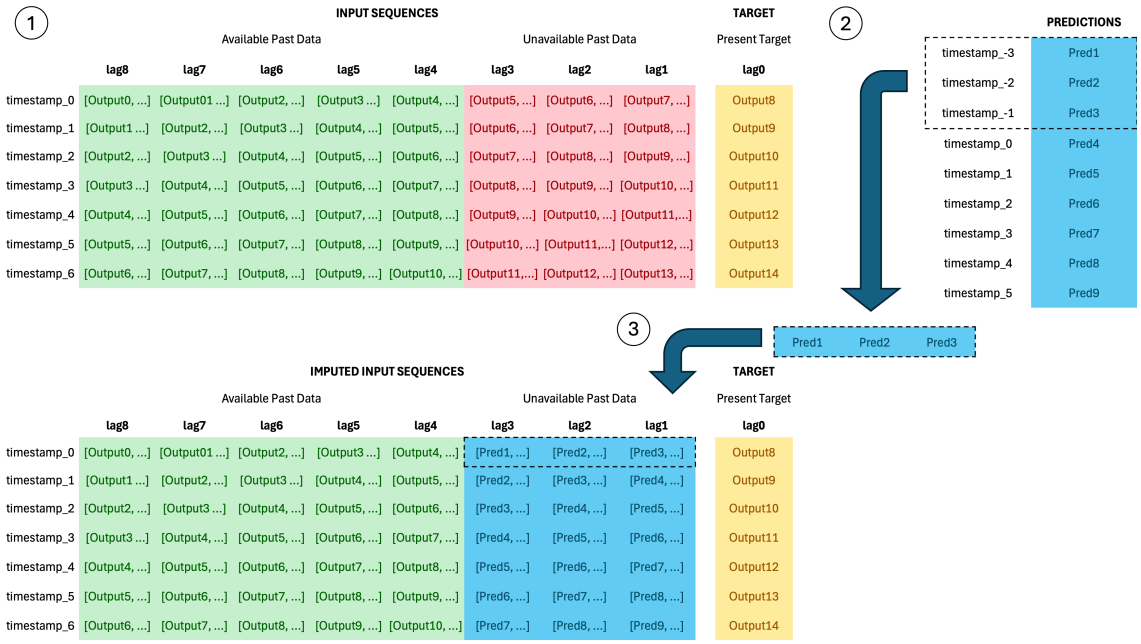

Figure 5.1: Illustration of the imputation procedure, where the last 384 time steps of each sequence are replaced with LightGBM predictions. For demonstration purposes, only the last 3 time steps are imputed in the illustration.

5.6.3 Data Preparation

A train-test split is performed to evaluate the LSTM model. For the test-time imputation strategy, the test period matches the time span of the LightGBM predictions. The training period consists of all data preceding the start of the LightGBM predictions. For the train-time imputation strategy, the same test period is used to ensure a fair comparison. This results in a smaller training set for the train-time strategy. The exact number of observations and date ranges for the training and test sets under each strategy are summarized in Table 5.5.

Table 5.5: Data split for the two imputation strategies used in this study.

Imputation Strategy	Dataset	Date Range	Observations
Test-time imputation	Training	31 Dec 2020 to 6 Mar 2024	111,129
	Testing	6 Mar 2024 to 16 Dec 2024	27,399
Train-time imputation	Training	24 Dec 2022 to 6 Mar 2024	41,443
	Testing	6 Mar 2024 to 16 Dec 2024	27,399

Before training the LSTM model, the input data X is normalized to improve training efficiency and model stability. Normalization scales the features to a consistent range, reducing the risk of vanishing or exploding gradients during training. The normalization is performed using Eq. 5.5:

$$X_{\text{norm}} = \frac{X - \mu}{\sigma} \tag{5.5}$$

where μ is the mean and σ is the standard deviation of each feature, computed from the training set. A normalization layer is fitted on the training data and applied consistently to all input sequences.

5.6.4 Model Specification

No extensive hyperparameter tuning is performed for the LSTM model, as it would require significant computational resources beyond the scope of this study. Instead, a limited search is conducted by testing a small set of values for the number of neurons in each hidden layer and the dropout regularization rate. The remaining components are selected based on heuristic choices: the model consists of two hidden layers, uses the Adam optimizer, and applies Mean Squared Error (MSE) as the loss function. To ensure efficiency, the model is trained for a maximum of 10 epochs, but training stops early if the model's performance on the validation set does not improve after four consecutive epochs. The full configuration of the LSTM architecture is summarized in Table 5.6.

The results of the manual hyperparameter tuning are presented in Table A.1 for the test-time imputation strategy and in Table A.2 for the train-time imputation strategy, both located in Appendix A. The first key observation from the tuning is that sequence length had no notable impact on the performance of the test-time imputation model. This is likely because the model, when trained on unaltered data, focuses too much on recent lags, making longer sequences unnecessary. This explanation is supported by the train-time imputation results, where longer sequences

Table 5.6: Components and configuration for the LSTM model. For components where hyperparameter tuning was performed, the tested values are shown in brackets.

Component	Description	Configuration
Sequence length	Number of consecutive time steps used as input for the model to predict the next value	[672; 1344; 2016]
Number of hidden layers	Number of LSTM layers in the model	2
Neurons per layer	Dimensionality of the output space for each LSTM layer	[32/16; 64/32; 128/64]
Dropout rate	Regularization method where a fraction of units are randomly set to zero during training	[0; 0.2; 0.4]
Activation function	Nonlinear transformation applied within the LSTM units	Tanh
Optimizer	Algorithm that adjusts model weights to minimize the loss function	Adam
Loss function	Metric that quantifies the difference between predicted and actual values	Mean Squared Error (MSE)
Batch size	Number of samples processed before updating the model's weights	64
Epochs	Number of complete passes through the training dataset	10

led to a slight, though inconsistent, improvement in performance. Another key finding is that both models performed best with no dropout regularization. While the number of neurons in the hidden layers did impact performance, no clear pattern emerged. The final LSTM configurations are summarized in Table 5.7.

Table 5.7: Optimal values from the manual tuning of the LSTM model

Hyperparameter	Test-Time Model	Train-Time Model
Sequence Length	672	2,016
Neurons	128/64	64/32
Dropout	0	0

5.7 Evaluation Metrics

5.7.1 Error Metrics

The accuracy of the forecasting models is evaluated using three standard error metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

The Mean Absolute Error (MAE) is the average of the absolute forecast errors and is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{5.6}$$

where $e_t = y_t - \hat{y}_t$ is the forecast error at time t. The MAE is scale-dependent, meaning it is measured in the same units as the original data. It provides a simple and intuitive measure of accuracy, with lower values indicating better model performance.

The Root Mean Squared Error (RMSE) is the square root of the average of the squared forecast errors:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2} \tag{5.7}$$

The RMSE penalizes larger errors more heavily than the MAE, making it sensitive to outliers and useful for capturing situations where larger deviations from the true values are particularly important. Like the MAE, the RMSE is scale-dependent.

The Mean Absolute Percentage Error (MAPE) expresses forecast errors as a percentage of the actual values:

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{e_t}{y_t} \right| \tag{5.8}$$

The MAPE is unit-free and allows for comparison across different datasets, provided the series are strictly positive and have a meaningful zero. However, MAPE can be unstable when actual values approach zero, and it may overweight negative errors compared to positive ones (Hyndman and Athanasopoulos, 2018).

5.7.2 Residual Diagnostics

To assess the quality of the forecasting models used in this study, residual diagnostics are conducted to evaluate whether the models have adequately captured the information in the data. A good forecasting model should produce residuals that satisfy two key conditions: (1) the residuals should be uncorrelated, and (2) the residuals should have a mean close to zero. Formally, these conditions can be expressed as:

$$E[e_t] \approx 0 \tag{5.9}$$

$$\text{Cov}(e_t, e_{t-k}) \approx 0 \quad \text{for all lags } k \neq 0 \tag{5.10}$$

where $e_t = y_t - \hat{y}_t$ is the residual at time t, and $E[\cdot]$ denotes the expectation. If either condition is violated, it indicates that useful information remains in the residuals that the model has failed to capture, and the forecasting performance could potentially be improved.

To check these conditions, both graphical and statistical methods are applied. Time series plots and autocorrelation function (ACF) plots of the residuals are used to visually inspect for correlation and changing variance. The ACF at lag is calculated as:

$$r_k = \frac{\sum_{t=k+1}^{n} (e_t - \bar{e})(e_{t-k} - \bar{e})}{\sum_{t=1}^{n} (e_t - \bar{e})^2} \tag{5.11}$$

where $\bar{e}$ is the mean of the residuals. A histogram is also plotted to assess the distribution of the residuals and to check for strong deviations from a normal distribution. Although normality is not a strict requirement for accurate forecasts, it is helpful for constructing reliable prediction intervals.

In addition to these visual checks, the Ljung–Box test is used as a formal statistical test to detect autocorrelation in the residuals. The test statistic is computed as:

$$Q^* = T(T+2)\sum_{k=1}^{\ell}\frac{r_k^2}{T-k} \tag{5.12}$$

where T is the sample size, r_k is the autocorrelation at lag k, and ℓ is the number of lags tested. Following the recommendations of Hyndman and Athanasopoulos (2018, pp. 69–76), the number of lags ℓ is set to twice the seasonal period, unless this exceeds $T/5$, in which case $\ell = T/5$ is used.

For this study, weekly seasonality is considered the most relevant pattern, and the lag parameter is therefore set to $\ell = 2 \times 672 = 1{,}344$.

6

Empirical Analysis

This section presents the results of each model, including performance metrics and relevant plots. A residual analysis is also provided in Section 6.4.

6.1 Naive Model

The results for the baseline seasonal naïve model, presented in Table 6.1, show that it achieves an MAE of 936.33 kW, an RMSE of 1313.13 kW, and a MAPE of 22.11%. These metrics serve as the benchmark for evaluating the other forecasting models.

Table 6.1: Seasonal naive model performance on the test set.

Error Metric	MAE (kW)	MAPE (%)	RMSE (kW)	Time (sec.)
Result	936.33	22.11	1313.13	0.22

By definition, the seasonal naïve model simply repeats the values from the previous week, producing forecasts that follow past patterns, as shown in Figure 6.1. The weekly plots in Figure 6.2 show that the model aligns well with actual values when weekly patterns remain stable, particularly during the colder months when weekly cycles are more consistent. However, when consecutive weeks differ, the model's predictions quickly fall out of phase. Overall, the weekly patterns tend to weaken and become less consistent during the summer months. This is likely due to changes in work and school schedules, which make consumer demand more unpredictable.

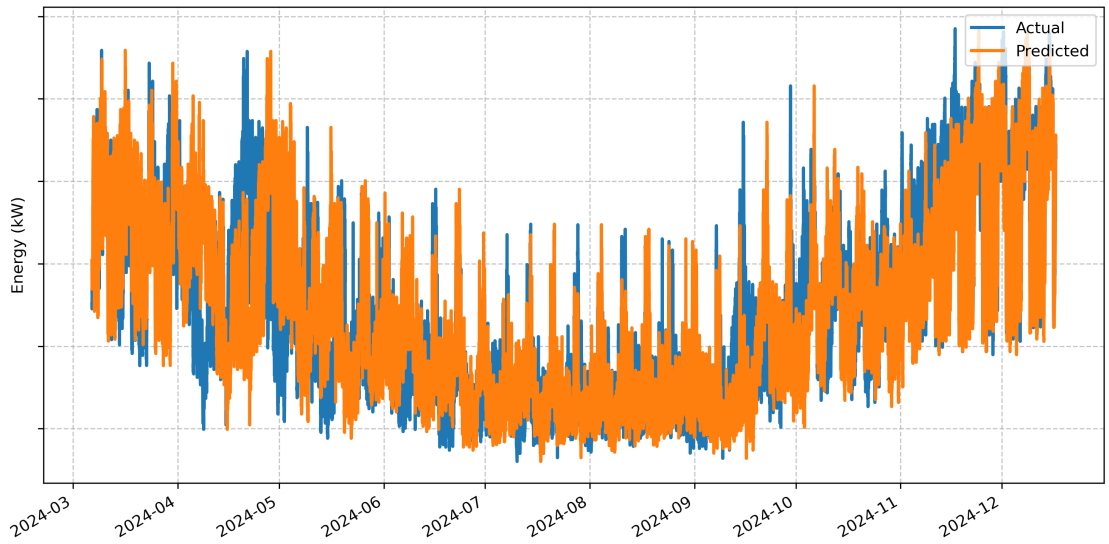

Figure 6.1: Comparison of actual CHP energy production and predictions generated by the Seasonal Naïve model.

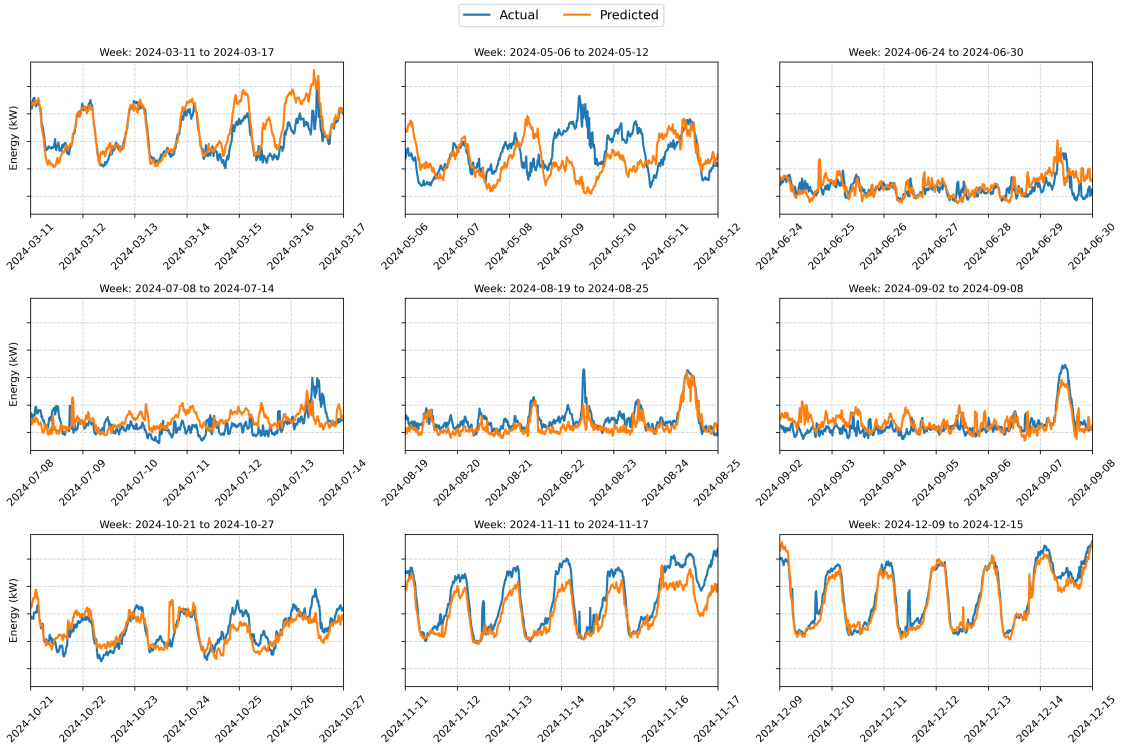

Figure 6.2: Comparison of actual CHP energy production and Seasonal Naïve model predictions for selected weeks in the test set.

6.2 Statistical Approach

The Dynamic Harmonic Regression (DHR) model offers only a marginal improvement over the naive baseline, with an MAE of 919.86 compared to 936.33 for the naive model (see Table 6.2). Notably, the DHR model requires significantly longer

computation time, whereas the naive model produces forecasts almost instantaneously.

Table 6.2: Dynamic Harmonic Regression performance on test set.

Error Metric	MAE (kW)	MAPE (%)	RMSE (kW)	Time (sec.)
Result	919.86	19.01	1279.47	5953.68

The relatively strong performance of the seasonal naive model likely reflects the presence of stable and regular seasonality in the data. When seasonal patterns are consistent, even complex models such as DHR, which are designed to extract and model seasonal components, provide limited additional benefit. In this case, the naive model already captures the dominant seasonal patterns, leaving little residual structure for DHR to exploit. As a result, the DHR model achieves only modest gains in accuracy.

Examining the predictions in 6.3, the DHR model captures both the yearly and weekly patterns in the data but consistently underestimates the actual values. This tendency is particularly evident during the summer months, where large spikes in production occur that the DHR model fails to capture. This pattern becomes even clearer when examining the weekly plots in 6.4. Overall, while the DHR model effectively follows the cyclical trends, it consistently underestimates the size of the fluctuations.

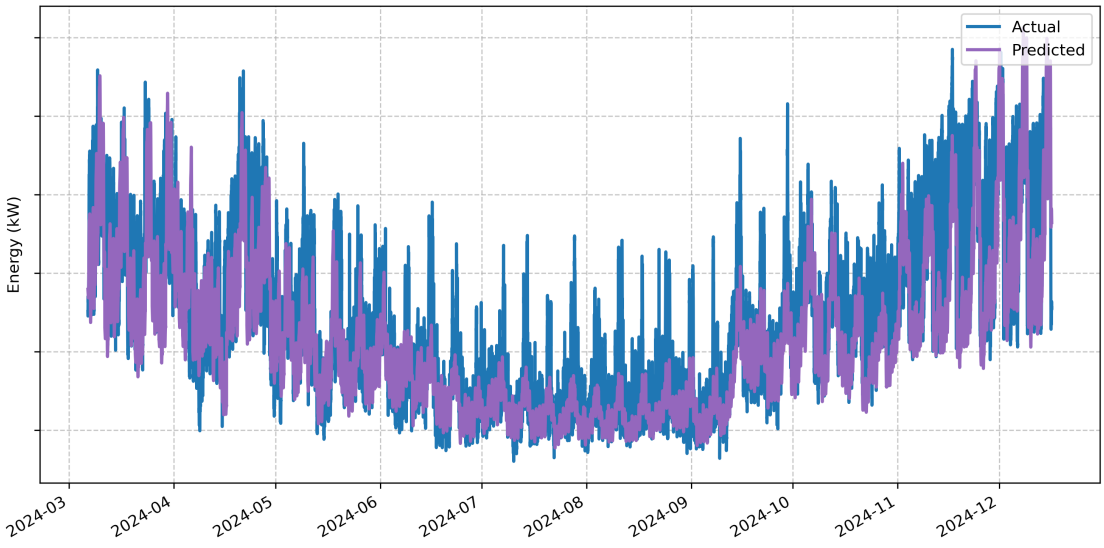

Figure 6.3: Comparison of actual CHP energy production and predictions generated by the DHR model.

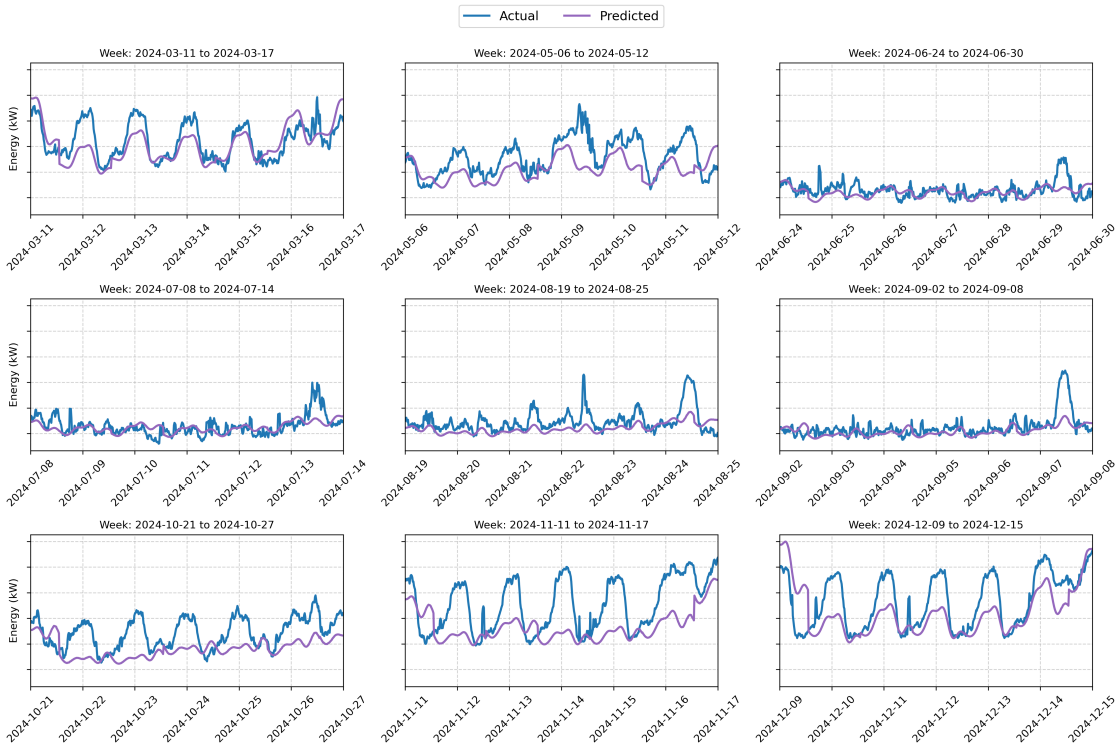

Figure 6.4: Comparison of actual CHP energy production and DHR model predictions for selected weeks in the test set.

6.3 Machine Learning Approach

6.3.1 LightGBM

The results for the LightGBM model are presented in Table 6.3. LightGBM substantially outperforms both the naive baseline and the DHR model, achieving an MAE of 524.57, which represents an approximately 44 percent reduction compared to the naive model. The MAPE indicates that the LightGBM forecasts deviate from the actual energy production by an average of 11.69 percent. Furthermore, LightGBM demonstrates a significant improvement in computational efficiency, requiring only 5.02 seconds to train compared to 99 minutes for the DHR model.

Table 6.3: LightGBM performance on the test set for both imputation strategies used in this paper.

Imputation Strategy	MAE (kW)	MAPE (%)	RMSE (kW)	Time (sec.)
Test-time[1]	524.57	11.69	776.70	5.02
Train-time[2]	533.77	10.69	757.11	2.06

[1] Uses an 80/20 train-test split.

[2] Uses a 50/50 train-test split.

Compared to the DHR model, which can only incorporate a limited number of external variables, LightGBM is better suited to handle a broader set of inputs and capture more complex relationships within the data. By including the sine-cosine temporal encodings and rolling window statistics for temperature, LightGBM is

better able to model the structure of the energy production data. This is supported by the feature importance plot in Figure 6.5, which highlights the weekly rolling window statistics as the most important predictors for reducing the model's loss function during training.

An additional observation from Figure 6.5 is that that features based on recent values, such as the one-week lag of the target and the one-day and one-week rolling window statistics of temperature, have a greater influence on the model's predictions than variables representing general trends, such as the long-term lags and the temporal encodings. This finding is particularly noteworthy given the strong seasonality present in the energy production data. It would typically be expected that seasonal features would play a more substantial role in forecasting.

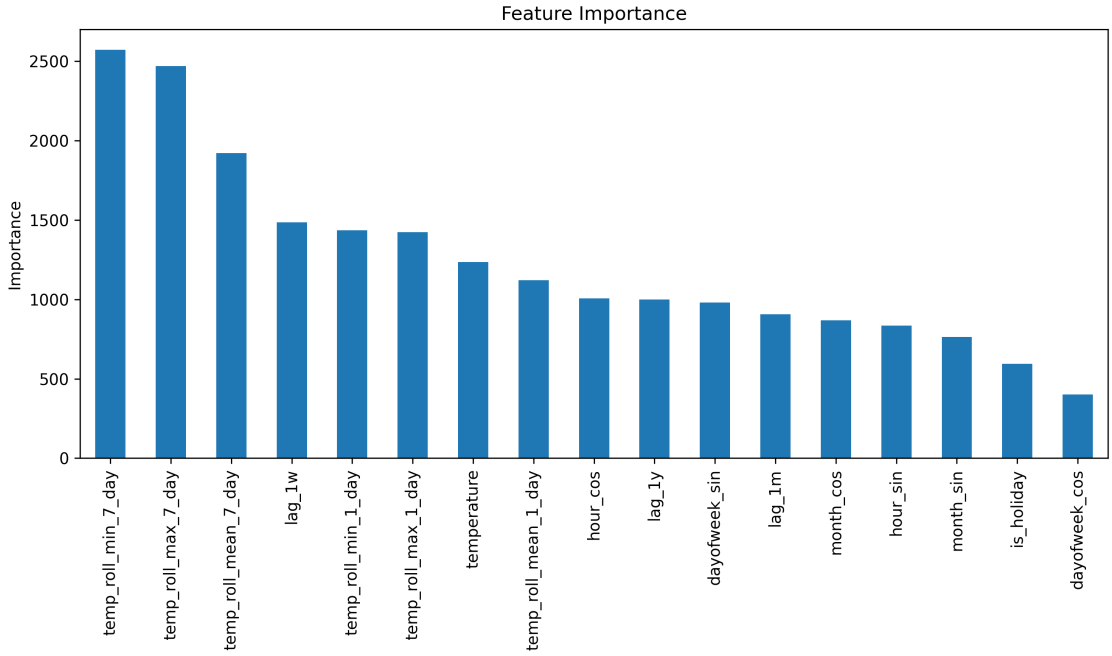

Figure 6.5: Feature importance of LightGBM, based on the total contribution of each feature to reducing prediction error during training.

Next, by observing the predictions in Figure 6.6, it is clear that LightGBM performs better during the colder months compared to DHR, but still struggles to capture the irregular production patterns of the warmer months. This becomes more evident when examining Figure 6.7, which shows that LightGBM does very well in predicting the weeks in March, October, November and December, but has trouble capturing the spikes and small fluctuations in the weeks from May to September. This suggests the model may need additional features to better account for these variations.

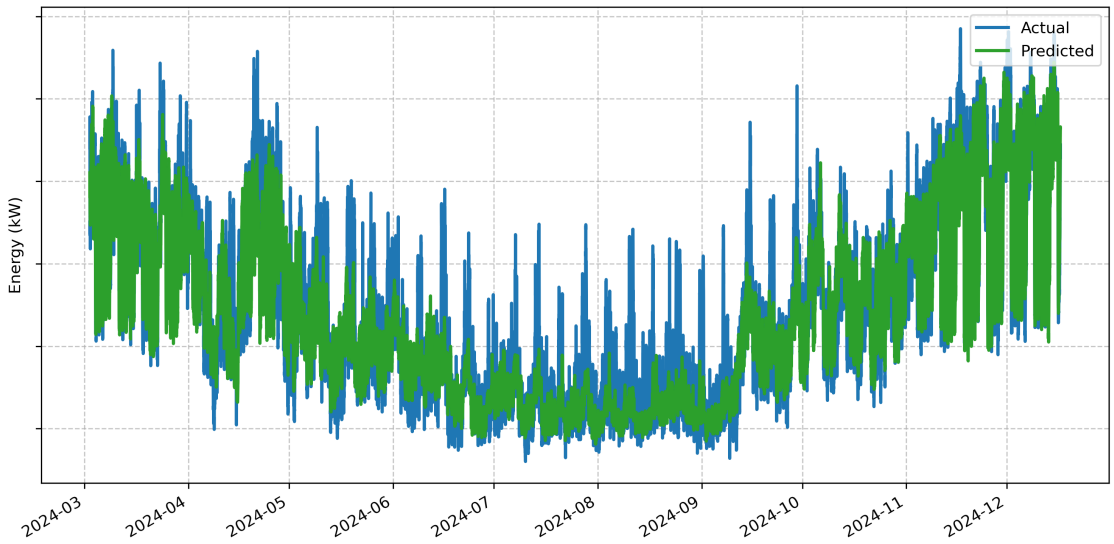

Figure 6.6: Comparison of actual CHP energy production and predictions generated by the LightGBM model using an 80/20 train-test split.

Figure 6.7: Comparison of actual CHP energy production and LightGBM model predictions for selected weeks in the test set.

6.3.2 LSTM

Table 6.4 presents the results of the LSTM models. The test-time model, trained on unaltered historical data, achieved a MAE of 525.97 kW. In comparison, the train-time model, trained on imputed data, had an MAE of 557.23 kW. Neither model shows any improvement compared to LightGBM, which provided an MAE of 524.57. Although the test-time model produced comparable results to LightGBM,

its computational time was substantially longer, with 10.27 minutes compared to LightGBM's 5.02 seconds. Meanwhile, the train-time model performed worse than both the test-time model and LightGBM in terms of MAE, but slightly better when evaluated using RMSE. This difference in MAE and RMSE suggests that the train-time model, while slightly worse on average, may be better at avoiding extreme errors, whereas the test-time model is generally more accurate but occasionally makes larger mistakes.

Table 6.4: LSTM performance on test set.

Strategy	**MAE (kW)**	**MAPE (%)**	**RMSE (kW)**	**Time (sec.)**
Test-time	525.97	11.77	779.00	616.48
Train-time	557.23	12.79	766.67	301.27[1]

[1] Early stopping at 5$^{\text{th}}$ epoch.

To address the concerns raised in Section 5.1 regarding whether the LSTM model with only test-time imputation may become overly reliant on recent lags, a comparison is made between the LSTM models and their respective LightGBM models used for imputation, as shown in Figure 6.8. The results indicate that the test-time LSTM model (top panel) places a strong dependence on the most recent lag, producing predictions that closely resemble a one-step "echo" of the LightGBM outputs, with minor adjustments from earlier dependencies. This behavior likely occurs because the 15 minute time series is highly autocorrelated, and a simple persistence strategy of repeating the most recent value is an effective way to minimize the loss. In this case, the LSTM appears to converge toward a persistence-like solution, as this strategy alone achieves a low error. This also implies that the model gains little additional benefit from the sine-cosine encodings. Moreover, when examining the LSTM model trained on imputed data (bottom panel), the predictions exhibit a smoother pattern. This suggests that the model attempts to adjust for errors in the imputation values by incorporating longer temporal dependencies. However, despite these adjustments, the train-time model still underperforms compared to the test-time model, and by extension the LightGBM model, as the test-time model closely mirrors the LightGBM predictions.

Figure 6.8: Focused view comparing LSTM and LightGBM predictions. Top: test-time LSTM. Bottom: train-time LSTM.

The predictions of both models are presented in Figure 6.9 for the entire test period and in Figure 6.10 for selected weeks. While both models generally follow the same overall patterns, the train-time model produces smoother predictions compared to the test-time model, as noted above. Similar to the LightGBM model, the LSTM also struggles to accurately capture the variation in the summer months.

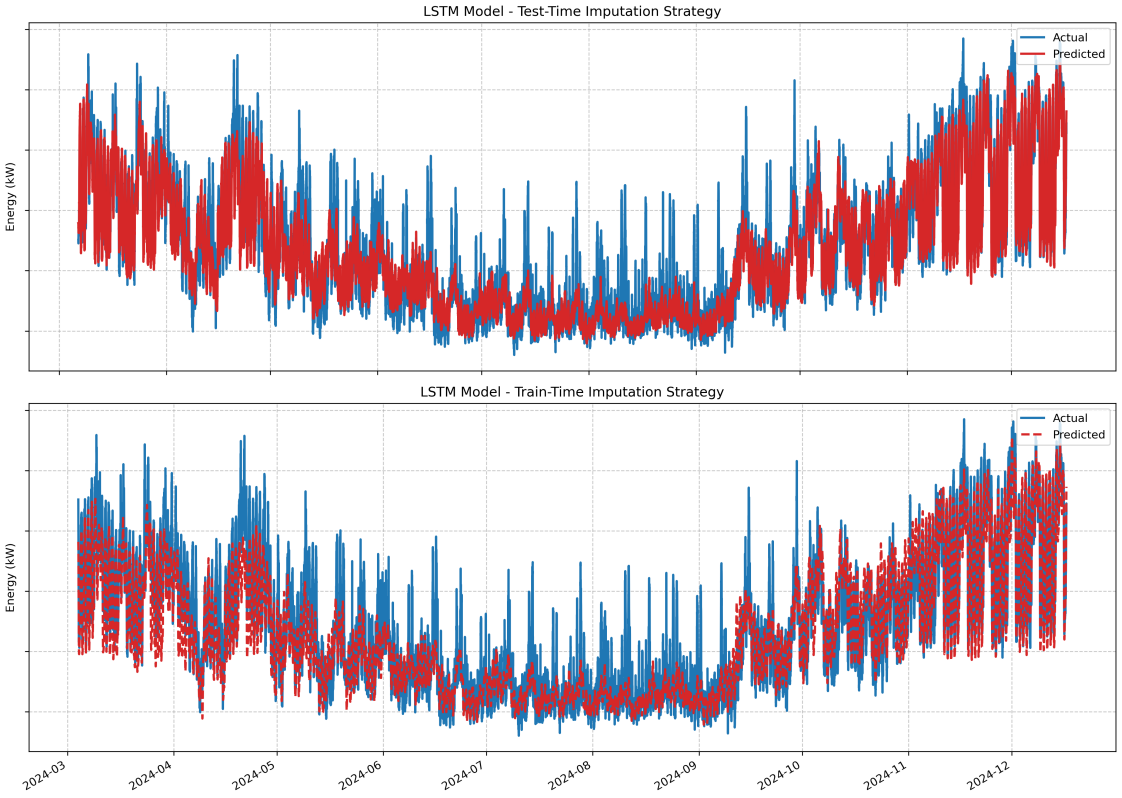

Figure 6.9: Comparison of actual CHP energy production and predictions generated by the LSTM models. Top: test-time LSTM. Bottom: train-time LSTM.

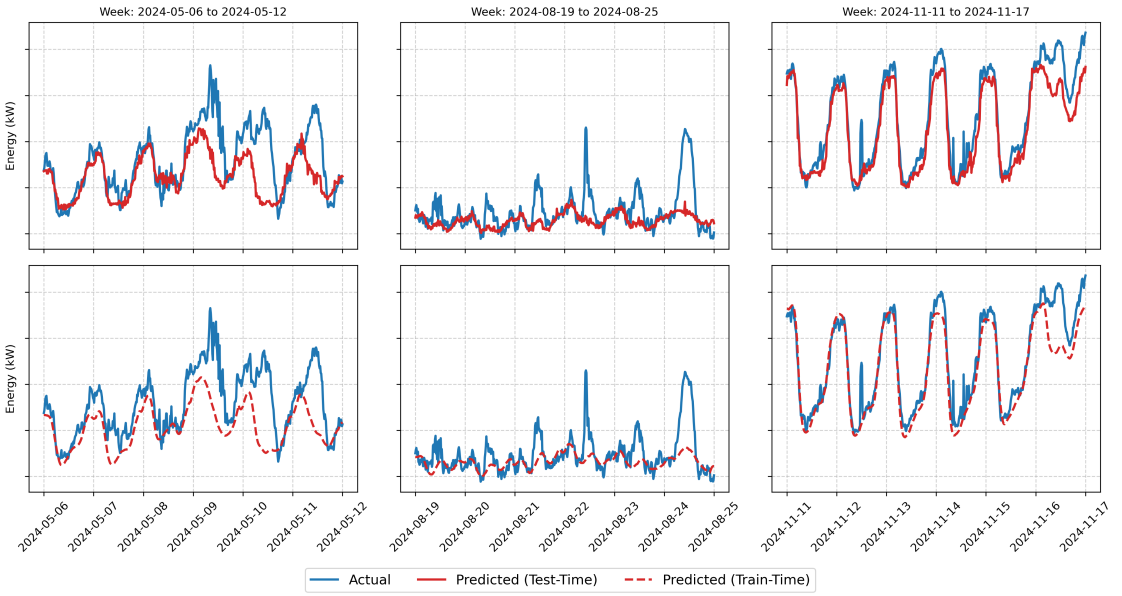

Figure 6.10: Comparison of actual CHP energy production and LightGBM model predictions for selected weeks in the test set. Top: test-time LSTM. Bottom: train-time LSTM.

6.4 Residual Analysis

Residual analysis is performed on the best performing model, the LightGBM model. As discussed in Section 5.7.2, a well-performing forecasting model should produce residuals that exhibit no autocorrelation and have a mean close to zero. If these conditions are not met, it suggests that there are patterns in the data that the model has not fully captured. Plots of the residuals, autocorrelation function (ACF) and histogram is presented in Figure 6.11.

The residuals average out to nearly zero, indicating no systematic over- or under-prediction. However, their spread varies throughout the year. Larger fluctuations appear in spring and late summer, while autumn shows more stable patterns. As illustrated in Figure 6.12, April and September have the widest spread, suggesting greater model uncertainty during these months. While summer months show a few extreme errors, most residuals are clustered near zero. In contrast, April and May have fatter tails, indicating more frequent large errors. This pattern suggests that higher load levels in spring amplify small relative fluctuations, leading to greater absolute errors.

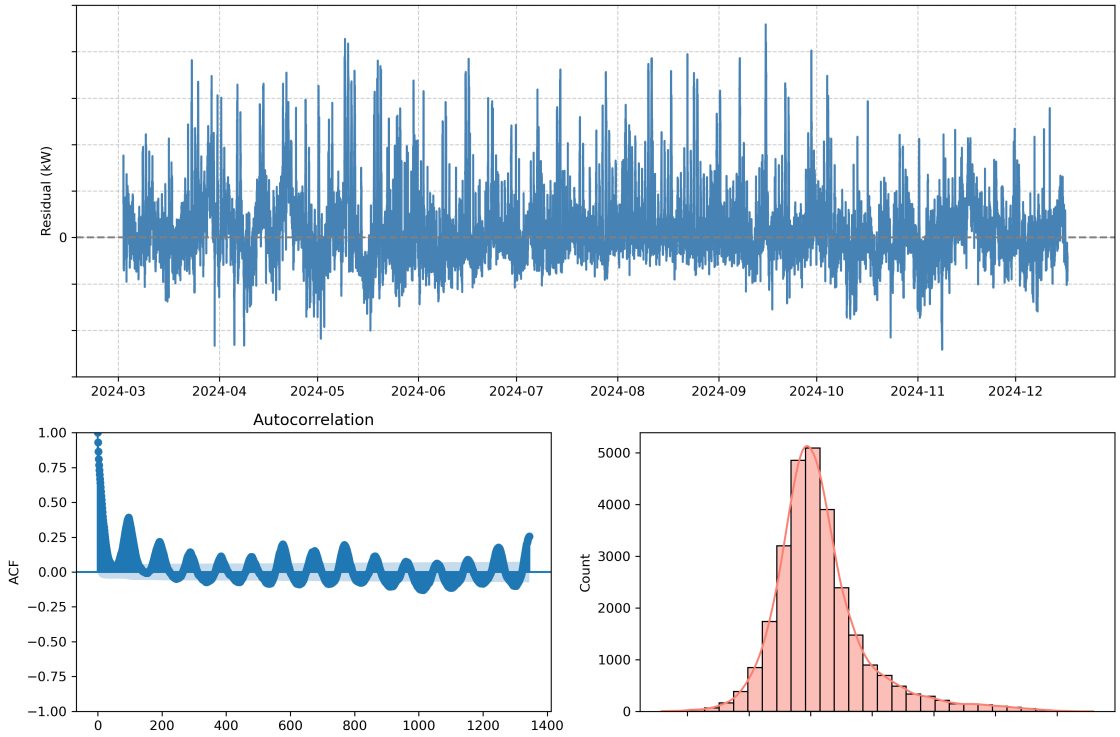

Figure 6.11: Residual analysis for the test-time LightGBM model, including the time series plot of residuals, autocorrelation function (ACF) plot, and histogram of residuals.

The ACF plot shows a strong initial spike at lag 1 that slowly declines, with recurring peaks every 96 lags (daily) and less prominently every 672 lags (weekly). This indicates that daily and weekly patterns persist in the residuals. The histogram supports this observation, with a sharp central peak and a longer right tail, pointing to occasional large underpredictions. This is also observed in the violin plot in Figure 6.12, which shows more positive extreme residuals than negative ones.

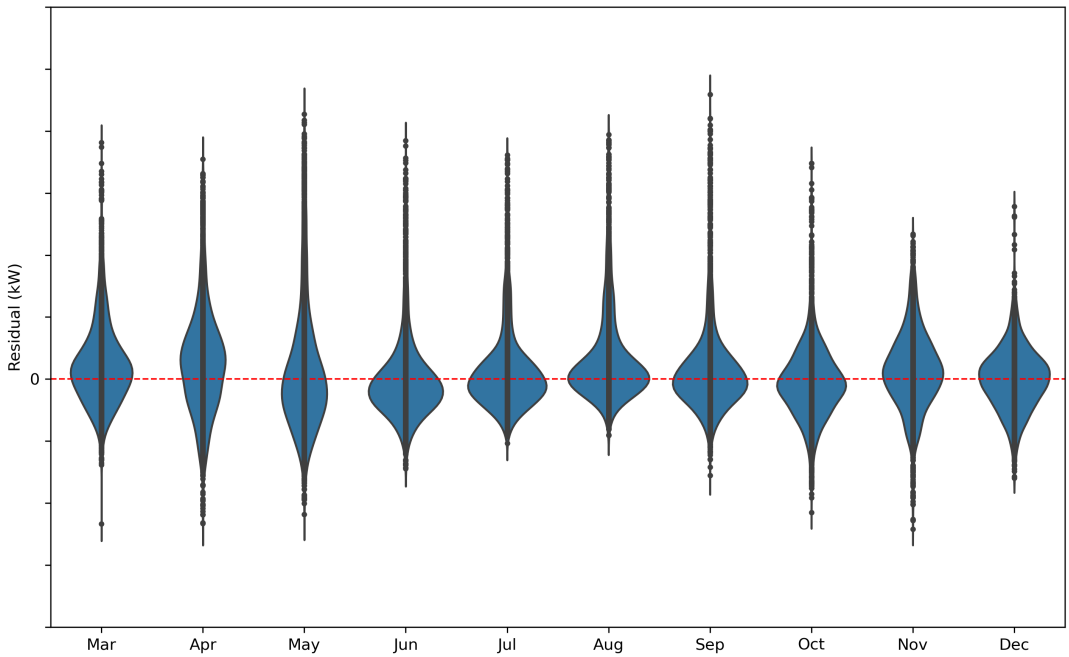

Figure 6.12: Distribution of residuals across each month in the test set. Width indicates error density, vertical range shows spread, and fat tails highlight the frequent occurrence of extreme errors.

The Ljung–Box test results, shown in Table 6.5, confirm the results from the ACF plot. The very low p-values ($p < 0.001$) at both 192 and 1344 lags provide strong evidence that the residuals are not random white noise but instead exhibit significant autocorrelation.

Taken together, these findings indicate that while the model is unbiased on average, it fails to fully capture the daily and weekly cycles. Incorporating additional features could help address these limitations.

Table 6.5: Ljung–Box test results for the test-time LightGBM model.

Lag Length	**Statistic**	**p-Value**
192	333,149.87	<0.001
1344	554,791.88	<0.001

7

Discussion

This section provides a deeper discussion of the results and their relation to findings from other studies. Section 7.2 outlines the limitations of this study and the insights that can be drawn. Section 7.3 explores the practical implications of the findings, while Section 7.4 suggests directions for future research.

7.1 Discussion of Main Findings

Table 7.1: Model performance comparison on the test set.

Model	MAE (kW)	MAPE (%)	RMSE (kW)	Time (sec.)
Naive	936.33	22.11	1313.13	**1.88**
DHR	919.86	19.01	1279.47	5953.68
LightGBM	**524.57**	**11.69**	776.70	5.02
LSTM (Test-time)	525.97	11.77	779.00	616.48
LSTM (Train-time)	557.23	12.79	**766.67**	301.27[1]

Summarizing the results it is clear that both machine learning models used outperforms the simple seasonal naive and Dynamic Harmonic Regression (DHR) models. The LightGBM model delivers the most accurate forecasts and the fastest computational time. These findings are in line with those of Makridakis et al. (2022) and other recent studies comparing machine learning techniques to traditional forecasting methods such as Zhou et al. (2019) , further reinforcing the strengths and capacities of the LightGBM model proposed by Ke et al. (2017)

The main model evaluated in this paper, the long short-term memory model, did not outperform the simpler LightGBM model. Considering the high-frequency nature of the dataset it is reasonable to assume that the most recent datapoint carries the most predictive power. This leads the LSTM to emphasize short-term memory rather than capturing long-term dependencies adequately. As noted by Al-Selwi et al. (2023), sequence length is crucial in enabling LSTMs to learn temporal dependencies. However, excessively long sequences can actually worsen performance (Al-Selwi et al., 2023; Wan et al., 2023). There appears to be a trade-off between capturing long-term dependencies and keeping sequences short enough for effective learning, especially in high-frequency datasets. Furthermore, the reporting delay substantially limits the potential of the LSTM model for high-frequency data, as

it prevents the model from incorporating the most recent observations, which are often the most informative for short-term forecasting. Instead, the model must rely on imputed values that may introduce additional errors. Nevertheless, the LSTM still outperformed the naive baseline and statistical model, highlighting its potential for time series forecasting.

The residual analysis further indicates that while the LightGBM model does not exhibit systematic bias, some recurring daily and weekly patterns persist in the residuals. The residuals exhibit a strong autocorrelation at lag 1 that gradually diminishes, with recurring peaks visible on a daily basis and, to a lesser extent, weekly. This may reflect the persistence of energy production volatility over time, where periods of high variability tend to follow other periods of high variability, rather than occurring as isolated events. Notably, these larger deviations and error spikes are most prominent during the spring and late summer months. Incorporating additional features could help address these limitations. This may include further weather-related variables, such as wind and precipitation data, as well as indicators of consumer behavior during extended holiday periods, particularly in the summer.

7.2 Limitations and Insights

The primary model examined in this thesis has been the LSTM neural network with the aim of forecasting one-step ahead deterministic forecasting. In an academic setting this forecasting horizon serves as a good starting point for evaluating the effectiveness of the model. However, in real-world applications, a longer horizon may be more useful, particularly given the 15-minute frequency of the data.

A limitation to consider is the generalizability of these results. Forecasting tasks can take many forms, one of which is energy production output from CHP plants. The results of this thesis might point towards the superiority of machine learning models in that context but doesn't necessarily need to extend to other areas of forecasting. The same applies to the discussion around LSTM sequence length. Although our results are consistent to those of Al-Selwi et al. (2023), where extending the sequence did not lead to better results, this may be dataset-specific rather than model-specific.

Furthermore this study also chose not to do any prior deseasonalization to the data and instead utilize temporal encodings to make the models learn these patterns directly from the data. Though prior literature, such as those of Bansal et al. (2025); Hyndman and Athanasopoulos (2018) has proposed that these representations can help models model seasonality, for this paper to draw any conclusions the results needs to be compared to a model that has been trained on deseasonalized data, which was outside the scope of this paper.

7.3 Practical Implications

For real-time energy forecasting for CHP energy production, an LSTM model may not be ideal due to its relatively high computational cost and inferior accuracy compared to the LightGBM. Consistent with findings from the M5 competition, lightGBM delivered the best performance and lowest compute time. Furthermore,

LightGBM is not dependent on prior timesteps which eliminates the data latency constraint. This positions it as a practical choice for real-time forecasting.

7.4 Future Research

The problem of excessively long input sequences has been addressed by Wan et al. (2023), whereby using Convolutional layers as well as attention mechanisms helped mitigate this problem. The study did however use daily data, which makes scaling the sequence length to cover more cycles easier compared to data with fifteen minute frequency. Future research that has access to high frequency data such as the data used in this study should examine if the effectiveness of Wan et al. (2023)'s approach is transferable to data of higher frequency. Beyond convolutional structures, evaluating newer architectures such as attention-based neural networks or informers would be worthwhile. These models may provide better performance while mitigating the drawback of sequence length dependence. For instance, Zhou et al. (2021) introduced the Informer model, which is specifically designed for long-sequence time series forecasting. Informer uses an efficient self-attention mechanism that reduces computational complexity while effectively capturing long-range dependencies, making it a promising approach for future work in high-frequency energy forecasting.

8

Conclusion

This study has examined the effectiveness of a Long Short-Term Memory Neural Network for time series forecasting of energy output from CHP plants while comparing it to the light LightGBM model, a Dynamic Harmonic Regression and a seasonal Naive model. The results show that all machine learning methods presented significantly outperformed the naïve benchmark and the statistical model with the LightGBM model performing the best both in terms of forecasting accuracy and computational time. For the LSTM model, the sequence length did not influence the performance of the model significantly, showcasing that excessively long input sequences do not contribute to LSTM ability for accurate forecasting of energy output when the data interval is short. Rather, when energy time series data is recorded in short intervals, the most recent point carries the most weight while long term dependencies decrease in relevance. This weakness is further amplified when the data exhibits a reporting delay, removing the possibility for the network to leverage the most recent data point for predictions.

Bibliography

D. M. Ahmed, M. M. Hassan, and R. J. Mstafa. A review on deep sequential models for forecasting time series data. *Applied Computational Intelligence and Soft Computing*, 2022(1):19, 2022. doi: https://doi.org/10.1155/2022/6596397. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/6596397.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. http://arxiv.org/abs/1907.10902, 2019. Accessed 21 May 2025.

S. M. Al-Selwi, M. F. Hassan, S. J. Abdulkadir, and A. Muneer. Lstm inefficiency in long-term dependencies regression problems. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 30(3):16–31, 2023.

A. Bansal, K. Balaji, and Z. Lalani. Temporal encoding strategies for energy time series prediction, 2025. URL https://arxiv.org/abs/2503.15456. Accessed 13 May 2025.

G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964. doi: https://doi.org/10.1111/j.2517-6161.1964.tb00553.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1964.tb00553.x.

CHP Alliance. What is chp? — combined heat and power alliance. https://chpalliance.org/what-is-chp/, n.d. Accessed 21 May 2025.

A. Dubey, A. Kumar, V. García Díaz, A. Sharma, and K. Kanhaiya. Study and analysis of sarima and lstm in forecasting time series data. *Sustainable Energy Technologies and Assessments*, 07 2021. doi: 10.1016/j.seta.2021.101474.

S. Elsworth and S. Güttel. Time series forecasting using lstm networks: A symbolic approach, 2020. URL https://arxiv.org/abs/2003.05672.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2018.

International Energy Agency. World energy outlook 2023, 2023. URL https://www.iea.org/reports/world-energy-outlook-2023. Accessed: 27 May 2025.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. Accessed 10 May 2025.

R. Kizito, P. Scruggs, X. Li, M. Devinney, J. Jansen, and R. Kress. Long short-term memory networks for facility infrastructure failure and remaining useful life prediction. *IEEE Access*, PP:1–1, 05 2021. doi: 10.1109/ACCESS.2021.3077192.

Z. Lin, L. Cheng, and G. Huang. Electricity consumption prediction based on lstm with attention mechanism. *IEEJ Transactions on Electrical and Electronic Engineering*, 15, 01 2020. doi: 10.1002/tee.23088.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34, 06 2018a. doi: 10.1016/j.ijforecast.2018.06.001.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3):e0194889, 2018b. doi: https://doi.org/10.1371/journal.pone.0194889.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022.

F. Petropoulos, D. Apiletti, V. Assimakopoulos, M. Z. Babai, D. K. Barrow, S. Ben Taieb, C. Bergmeir, R. J. Bessa, J. Bijak, J. E. Boylan, J. Browell, C. Carnevale, J. L. Castle, P. Cirillo, M. P. Clements, C. Cordeiro, F. L. Cyrino Oliveira, S. De Baets, A. Dokumentov, J. Ellison, P. Fiszeder, P. H. Franses, D. T. Frazier, M. Gilliland, M. S. Gönül, P. Goodwin, L. Grossi, Y. Grushka-Cockayne, M. Guidolin, M. Guidolin, U. Gunter, X. Guo, R. Guseo, N. Harvey, D. F. Hendry, R. Hollyman, T. Januschowski, J. Jeon, V. R. R. Jose, Y. Kang, A. B. Koehler, S. Kolassa, N. Kourentzes, S. Leva, F. Li, K. Litsiou, S. Makridakis, G. M. Martin, A. B. Martinez, S. Meeran, T. Modis, K. Nikolopoulos, D. Önkal, A. Paccagnini, A. Panagiotelis, I. Panapakidis, J. M. Pavía, M. Pedio, D. J. Pedregal, P. Pinson, P. Ramos, D. E. Rapach, J. J. Reade, B. Rostami-Tabar, M. Rubaszek, G. Sermpinis, H. L. Shang, E. Spiliotis, A. A. Syntetos, P. D. Talagala, T. S. Talagala, L. Tashman, D. Thomakos, T. Thorarinsdottir, E. Todini, J. R. Trapero Arenas, X. Wang, R. L. Winkler, A. Yusupova, and F. Ziel. Forecasting: theory and practice. *International Journal of Forecasting*, 38(3):705–871, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.001. URL https://www.sciencedirect.com/science/article/pii/S0169207021001758.

S. Siami-Namini, N. Tavakoli, and A. Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018. doi: 10.1109/ICMLA.2018.00227.

E. Spiliotis. *Time Series Forecasting with Statistical, Machine Learning, and Deep Learning Methods: Past, Present, and Future*, pages 49–75. 09 2023. ISBN 978-3-031-35878-4. doi: 10.1007/978-3-031-35879-1_3.

M. Szega, P. Żymełka, and T. Janda. Improving the accuracy of electricity and heat production forecasting in a supervision computer system of a selected gas-fired chp plant operation. *Energy*, 239:122464, 2022. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2021.122464. URL https://www.sciencedirect.com/science/article/pii/S0360544221027134.

R. Teixeira, A. Cerveira, E. Pires, and J. Baptista. Advancing renewable energy forecasting: A comprehensive review of renewable energy forecasting methods. *Energies*, 17:3480, 07 2024. doi: 10.3390/en17143480.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

A. Wan, Q. Chang, K. AL-Bukhaiti, and J. He. Short-term power load forecasting for combined heat and power using cnn-lstm enhanced by attention mechanism. *Energy*, 282:128274, 2023.

G. Zhang and M. Qi. Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160(2):501–514, 2005. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2003.08.037. URL https://www.sciencedirect.com/science/article/pii/S0377221703005484. Decision Support Systems in the Internet Age.

C. Zhou, Z. Fang, X. Xu, Y. Ding, X. Jiang, and Y. Ji. Using long short-term memory networks to predict energy consumption of air-conditioning systems. *Sustainable Cities and Society*, 55, 12 2019. doi: 10.1016/j.scs.2019.102000.

H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021. URL https://arxiv.org/abs/2012.07436.

Appendix A

Hyperparameter Tuning Results for LSTM

Table A.1: Results from the manual tuning of the LSTM model with the test-time imputation strategy. The optimal value is marked with an asterisk (*).

Stage	Tested Values	Fixed Settings	MAE (kW)	Time (sec.)
1. Sequence Length	672	Neurons: 64/32, Dropout: 0	528.87	694.02
	1344		530.66	1290.24
	2016		529.33	940.87[1]
2. Neurons	32/16	Sequence Length: 672*, Dropout: 0	528.47	572.41
	64/32		528.87	694.02
	128/64		525.97	616.48
3. Dropout	**0**	Sequence Length: 672*, Neurons: 128/64*	**525.97**	**616.48**
	0.2		528.11	636.32
	0.4		527.97	635.08

[1] Early stopping at 5$^{\text{th}}$ epoch

Table A.2: Results from the manual tuning of the LSTM model with the train-time imputation strategy. The optimal value is marked with an asterisk (*).

Stage	Tested Values	Fixed Settings	MAE (kW)	Time (sec.)
1. Sequence Length	672	Neurons: 64/32, Dropout: 0	572.95	110.05[1]
	1344		584.27	246.13[2]
	2016		557.23	301.27[1]
2. Neurons	32/16	Sequence Length: 2016*, Dropout: 0	581.87	291.03[1]
	64/32		557.23	301.27[1]
	128/64		591.60	326.56[1]
3. Dropout	**0**	Sequence Length: 672*, Neurons: 64/32*	**557.23**	**301.27**[1]
	0.2		613.38	604.92
	0.4		565.82	304.69[1]

[1] Early stopping at 5$^{\text{th}}$ epoch
[2] Early stopping at 6$^{\text{th}}$ epoch

Appendix B

AI Statement

AI tools have been used during the writing of this thesis. Those include ChatGPT and Google Gemini (a coding assistant in Google Collab). These tools have been used primarily for coding related tasks in Python and laTeX such as syntax, error handling and idea generation. Considering the complexity of the methods implemented in this thesis, AI tools have been of great help to get quick answers which were then examined and further developed by the authors. AI has not been used for tasks relating to establishing the aim of the thesis, the main problem of the thesis or for any insights drawn from the results. The approach in which this thesis has tackled the research question is composed of the author's original ideas.