Power Grid Load Forecast - Documentation

Andre Baron
Rinchyen Bayarsaikhan
Kacper Dłubała
Filip Duda
Michał Dworniczak
Mateusz Działowski

October 2025

Contents

1 Introduction

In this project we aim to replicate the results of the paper Bak, M., Bielecki, A. (2007). Neural systems for short-term forecasting of electric power load. In B. Ribeiro, B. Beliczynski, A. Dz ielinski, M. Iwanowski (Eds.), ADAPTIVE AND NATURAL COMPUTING ALGORITHMS, PT 2 (Vol. 4432, p. 133). Springer Nature., which explores neural network–based methods for short-term power grid load prediction. In addition to reproducing the published results, we attempt to develop and evaluate a custom forecasting approach to compare its performance against the original models.

2 Project description

2.1 Project structure

1. **analysis** - module containing classes and methods used for data and model performance visualization, as well as the images of generated plots
2. **data** - data used for model training and evaluation
3. **doc** - project documentation
4. **eval** - module used for models' evaluation and keeping their MAPE over horizon and average MAPE in .txt files
5. **models** - module used for models' training and for storing trained models in .keras files
6. **utils** - module used for data preparation and cleanup utilities

3 Data analysis

3.1 Data used

In order to train our predicting models we gathered and cleaned the following collections of data:

1. **Total load for Germany** from transparency.entsoe.eu ("Total Load - Day Ahead / Actual" dataset to be exact). It spans from 00:00 on 01.01.2015 to 23:45 on 31.12.2024. The measurements of total load at the given time are expressed in MW and are taken every 15 minutes.
2. **Average temperature in the region of Germany** from Climate Data Store's dataset Thermal comfort indices derived from ERA5 reanalysis (https://cds.climate.copernicus.eu/datasets/derived-utci-historical?tab=download).

   We used sub-region extraction with values $55°N$, $5°E$, $15°E$ and $47°N$, which are rounded up values for geographical extreme points in Germany. Obtained dataset covers the same time span as the previous one, however the data points are in between 1 hour long intervals instead of 15 minute ones.

3.2 Parameters description

From the raw datasets we extracted [num] of features that were later used during training and evaluation of our models. Below is the list of those parameters with concise descriptions.

- **load** – represents the electrical load during a given moment. It is the value our models are trying to predict.
- **load_timestamp_−1, load_timestamp_−2, load_timestamp_−3** – electrical load values in three data points immediately prior to the current one.
- **load_previous_day_timestamp_−2, load_previous_day_timestamp_−1, load_previous_day_timestamp_0, load_previous_day_timestamp_1, load_previous_day_timestamp_2** – electrical load values from the same hour of the previous day, and those within two data points radius.
- **prev_3_temperature_timestamps_mean** – average air temperature from three previous data points.
- **prev_day_temperature_5_timestamps_mean** – average temperature from the same data points as **load_previous_day_timestamp**.
- **hour_of_day_sin, hour_of_day_cos, day_of_week_sin, day_of_week_cos**, **day_of_year_sin, day_of_year_cos** – cyclical encoding of time features representing hour, day of week, and day of year.
- **timestamp_day** – number of days that have passed since the first day in our dataset (01.01.2015).

The features (apart from naturally standardized features and the linear **timestamp_day** feature, which was normalized by dividing by it's maximum value), as well as the target value, where standardized to have means of 0 and standard deviations of 1.

We also tried to include a feature that would let the model differentiate between summer and winter time, as the same hour could mean different things in those cases, especially at times close to the DST switch. However, this feature made the models slightly worse, supposedly due to it's insignificance for timestamps far from the DST switch.

3.3 Data cleaning

The obtained data was already relatively clean, with no unexpected anomalies. All of the detected missing values fall into one of two categories:

3.3.1 Daylight saving time

The original dataset contains non-existent timestamps that appear due to the practice of setting clocks forward by one hour in spring. These rows we discarded, as they don't correspond to real-world data.

Another problem arises when we take autumn winter time into account, as that switch introduces duplicate timestamps. However, we decided to keep them in the dataset, as they correspond to real observations.

3.3.2 Missing previous-day values

Generating lag features for grid load (e.g., load at $t-1$, $t-2$) inevitably creates NaNs at the start of the time series. We imputed these initial missing values using the mean of their corresponding feature.

3.4 Patterns and correlations within data

As part of our exploratory data analysis process we created various plots in order to find patterns and correlations within the datasets easier. To do this we used Python libraries for data visualization - Matplotlib and Seaborn. In this section we present charts we deemed the most significant to our project, while also describing the meaning behind them.

3.4.1 Line plots

Line plots are the most straight-forward method of data visualization we've implemented in this project. It was used to better see the similarities between the changes in load and temperature over time and how they relate to one another.

To achieve the desired effect we normalized both values to 0-1 range, and plotted those for each month of each year in our training data set (charts for longer period would be to hard to read).

After analyzing the results we found that during warmer months, mainly the May-September period, there is a lot of overlap between the features, whereas in wintertime, especially in January and December, the relation between load and temperature is a lot less clear. To illustrate this, we selected plots of six months from our dataset and put them below.

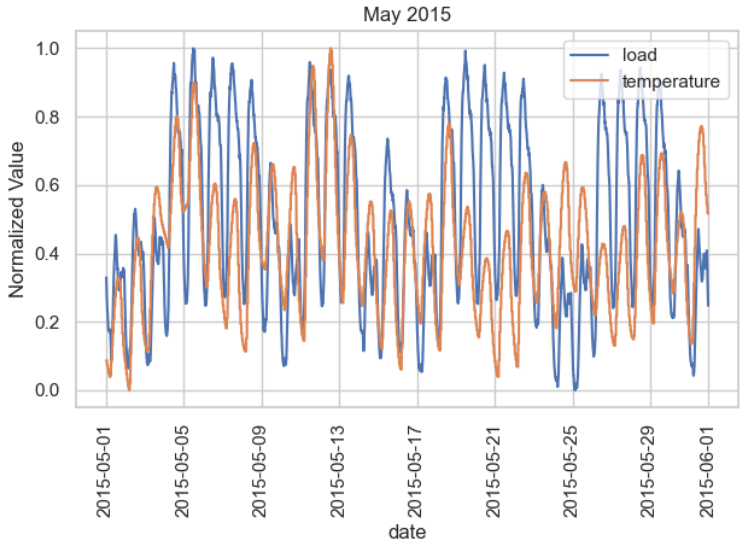

Figure 1: May 2015

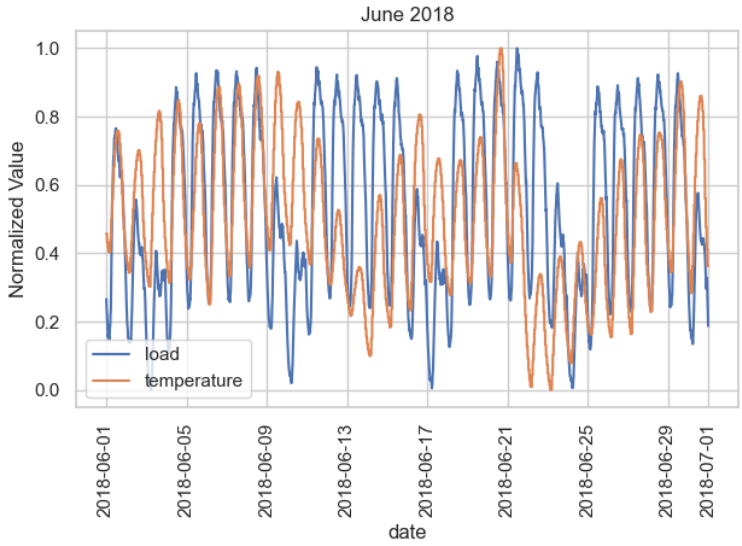

Figure 2: June 2018

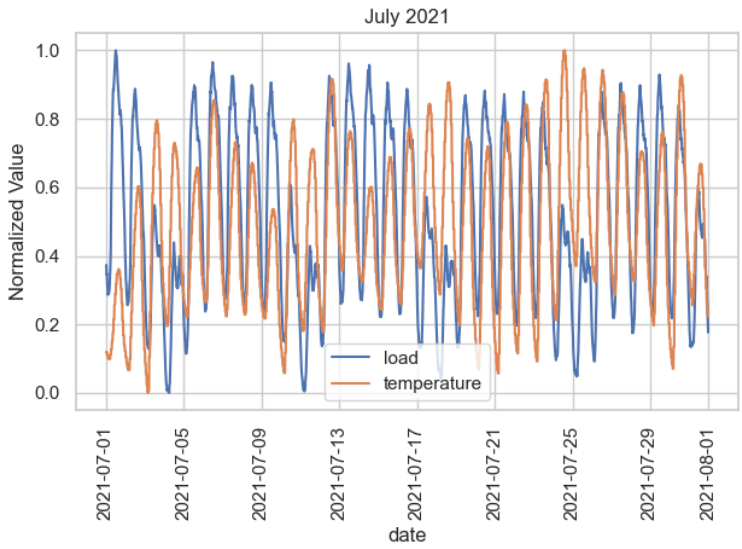

Figure 3: July 2021

There is a clearly visible relation between temperature and load. Day and night cycle of increasing and declining value can be seen in both, and after normalization, their spikes seem to be of similar sizes as well.

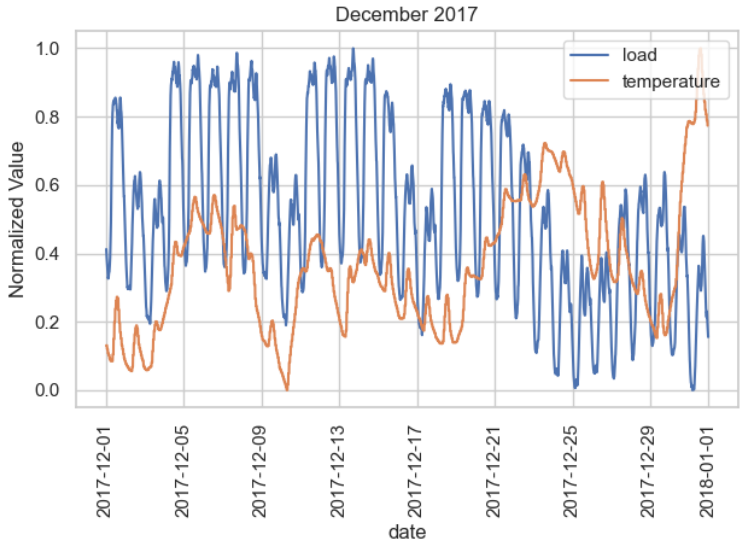

Figure 4: December 2017

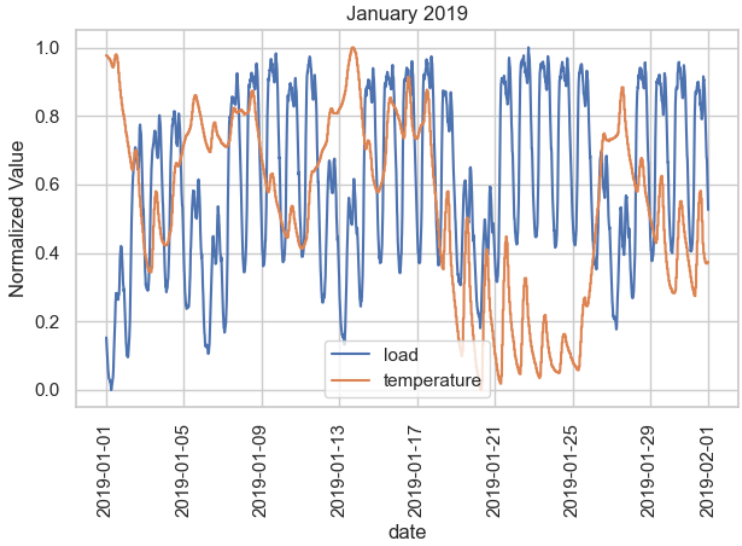

Figure 5: January 2019

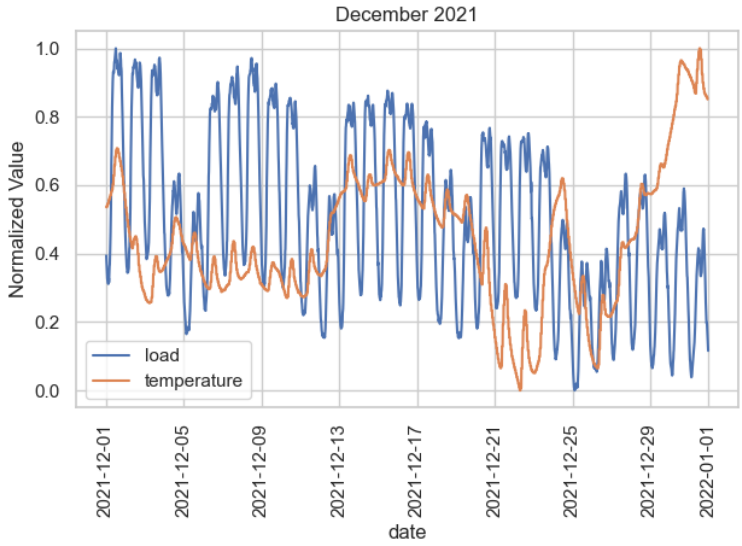

Figure 6: December 2021

However, it is not exactly the case for colder months. There is not nearly as much overlap between the chosen features, yet there is still some kind of relation. We can sometimes see, that load goes up as temperature goes down and vice versa (which might be due to lesser demand for energy for heating). Similarly to summer months, both load and temperature follow a day-night cycle.

3.4.2 Bar charts

We utilized bar charts to aggregate values over selected date components (year, month, day of week) using average values of load.

Firstly, we prepared plots (one with original data, one normalized) showing the average value of electrical load over the years. They all seem to achieve around 57,000 MW. That strongly indicates there are similarities in load distribution over the years, which is a very good sign from model training perspective, since simpler data allows even simple models to yield solid results. There only seem to be a slight decrease around early 2020s, especially 2020 and 2022 (possibly due to events like COVID-19 Pandemic or Ukraine War), which are clearly visible on the normalized version of the plot.

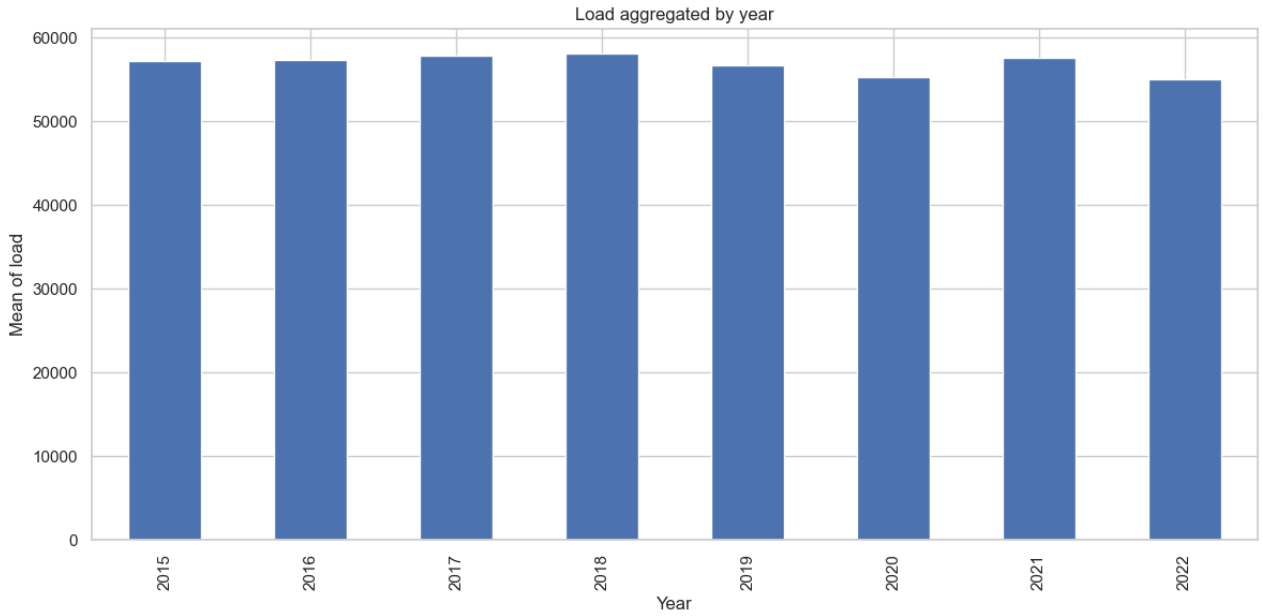

Figure 7: Average load over the years

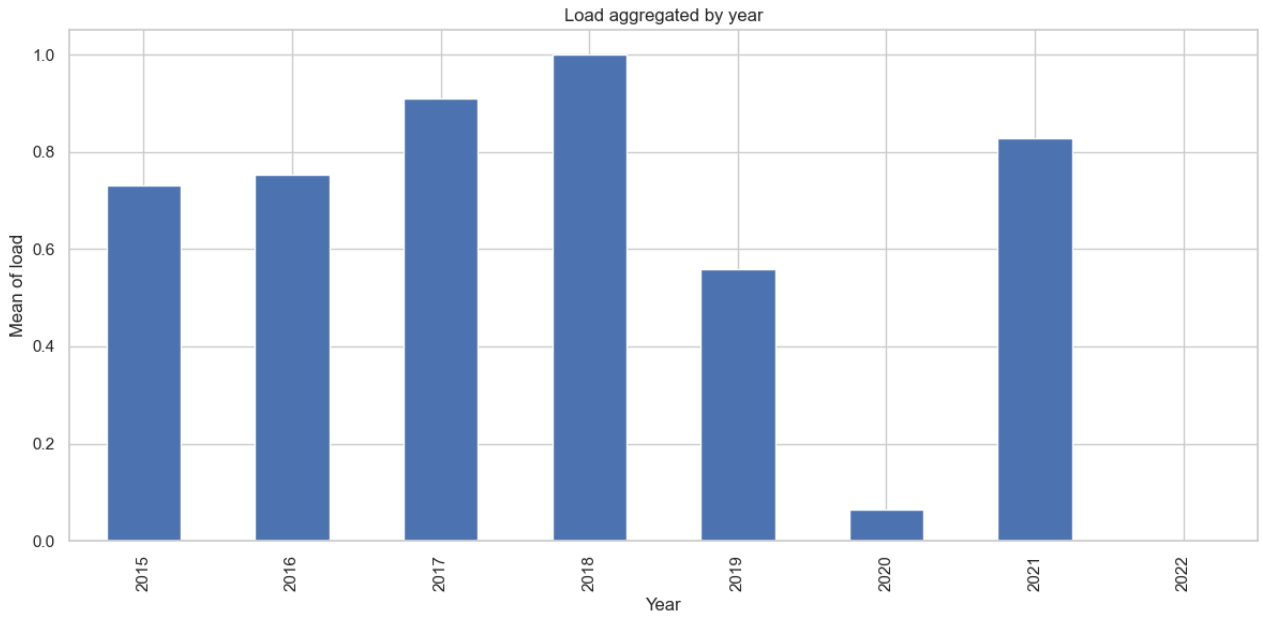

Figure 8: Average load over the years - normalized

After that, we checked how the distribution of load looks like when aggregated over days of week. We found out, that averages for Saturday and Sunday are significantly lower than those for workdays. This pattern was also visible on the line plots, appearing as five spikes of similar value, followed by two with lower ones. We can see the weekend load decrease on the following chart:

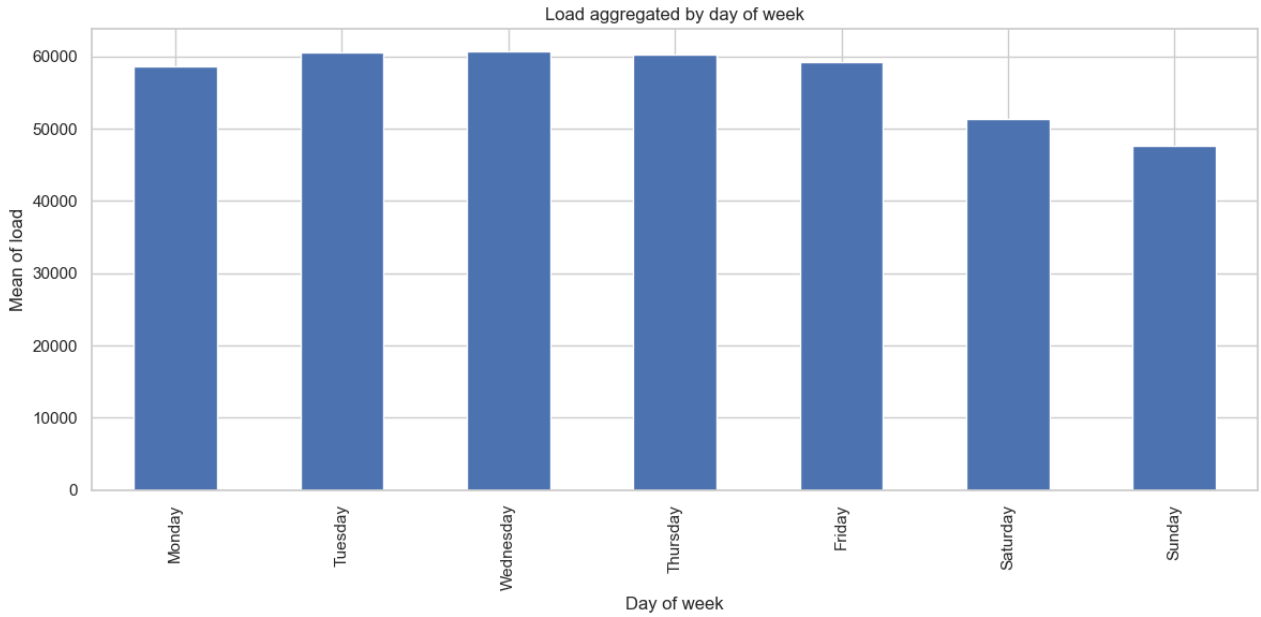

Figure 9: Average load over the days of week

Lastly, we prepared a bar chart showing the average load in each month. The bars appear to be the tallest in January and February, gradually decreasing in size till Summer. After that they begin to get taller again. This supports the intuition, that there is more electrical load during the colder parts of the year, likely due to heating, bigger demand for lighting or people spending more time indoors. The results are depicted below.

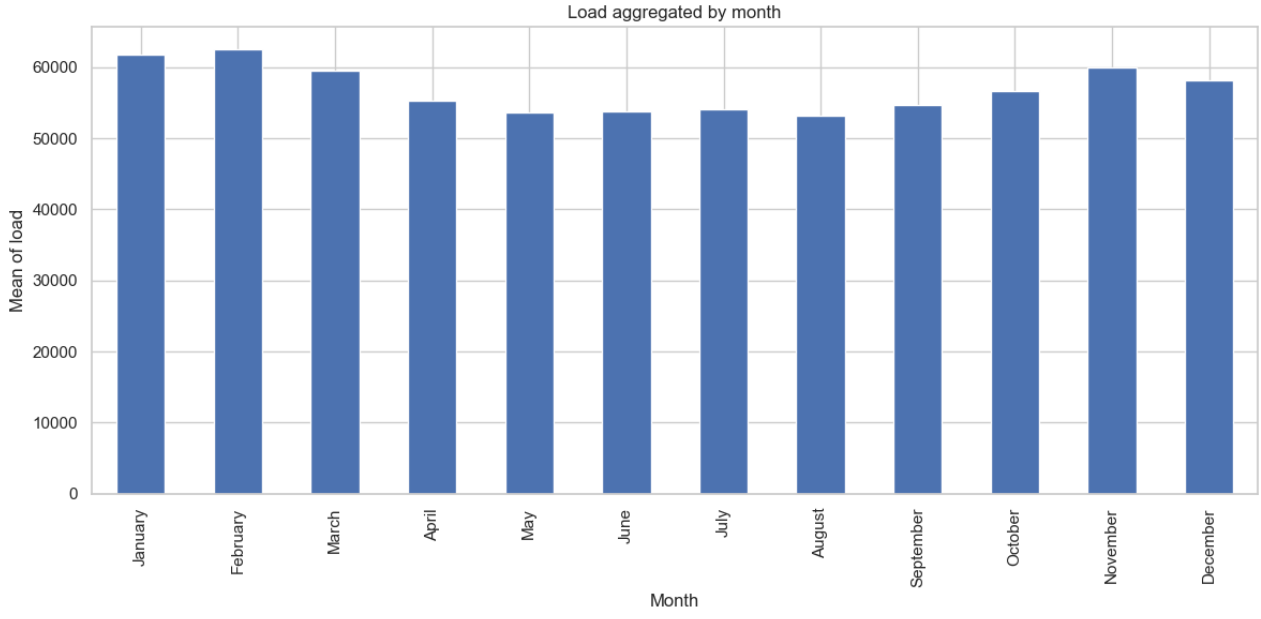

Figure 10: Average load over months

3.4.3 Histograms

To gain a better insight into the load and temperature data we created histograms for both of them. This allowed us to clearly see their distributions. The charts are presented below:

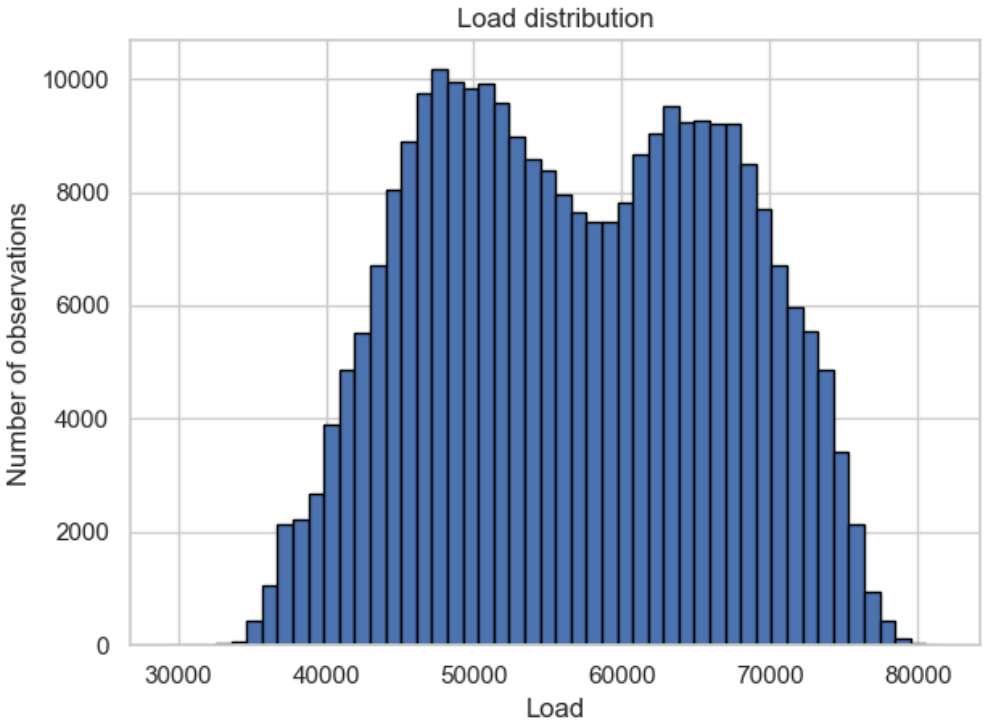

Figure 11: Electrical load distribution

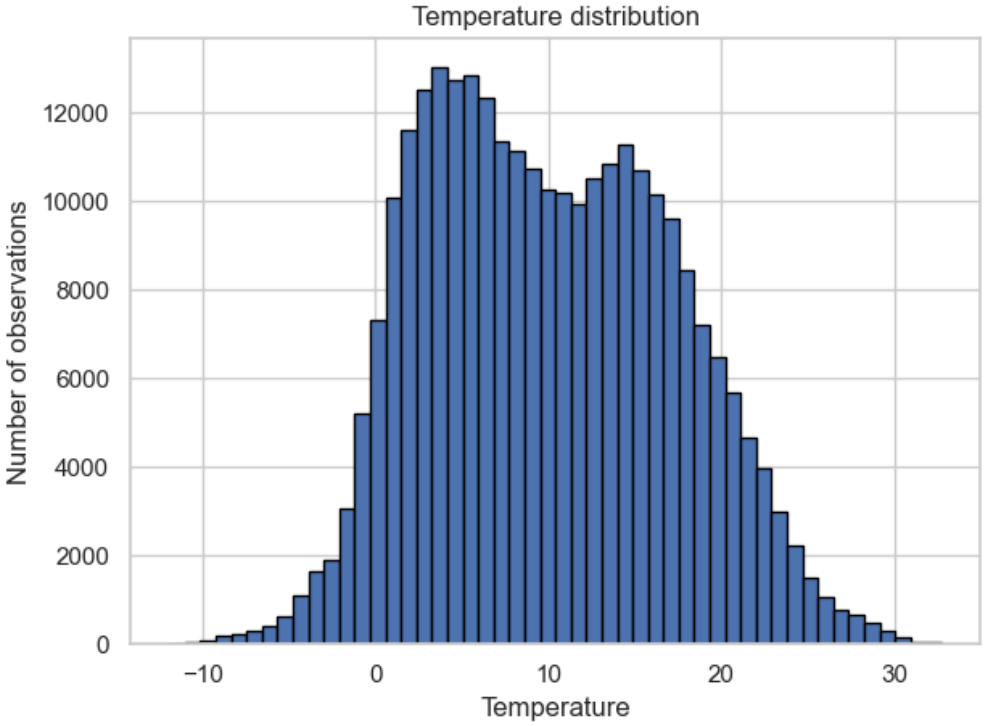

Figure 12: Temperature distribution

It is easy to observe, that both of those distributions are bimodal (they have two local maxima, unlike normal distribution). We think it is either due to both values changing during day-night cycle, or the changes in values over seasons.

3.4.4 Heatmaps

Heatmaps are a good way to visualize the fluctuations in data over time, so we decided to incorporate them into our project. We used them to represent the average value for each hour in a week, taking into account all data points in out training dataset. You can see the heatmap for electrical load below.

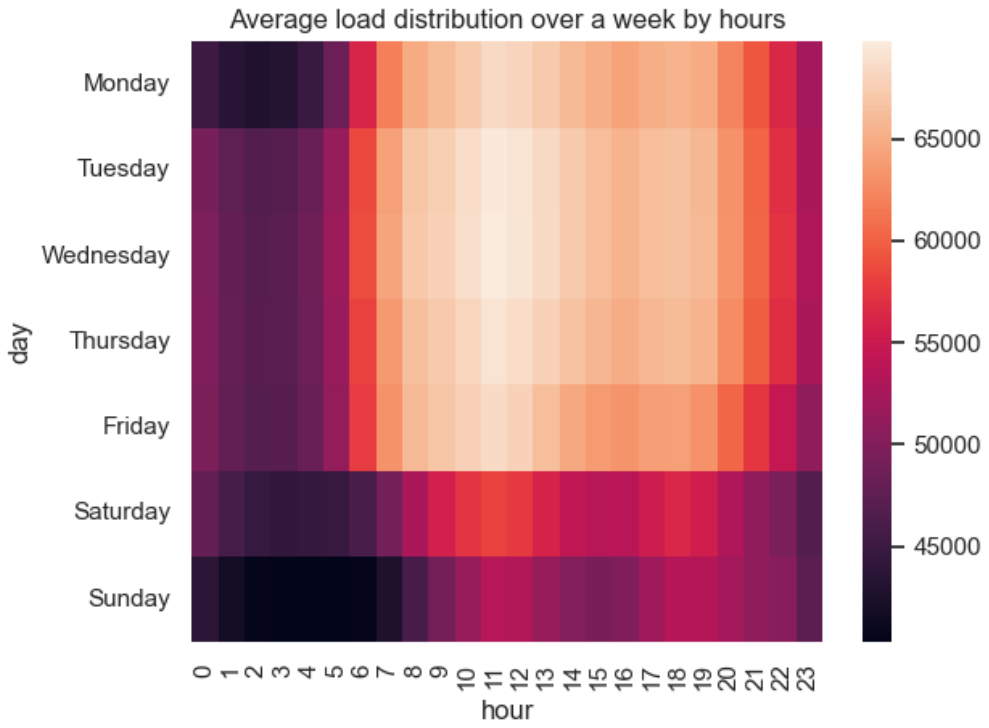

Figure 13: Load weekly heatmap

From what we see here, it appears that load is unsurprisingly at its lowest during night time (23:00 - 5:00), especially on weekends and Mondays, and reaches its highest levels around 11:00. We can also notice that it is significantly lower on weekends than workdays, as we have already established.

The same plot for temperature did not show anything noteworthy (lowest during night, highest in the early afternoon) and was therefore not included.

3.4.5 Correlation matrices

Correlation matrices are used to tell if there is a relation (in our case linear relation) between data. For our project we prepared correlation matrices between load and temperature for each year as a whole, for each month over all years, as well as for each month of each year separately.

All of the correlation matrices taken for whole years yielded similar results - correlation between load and temperature around -0.15 to 0, which suggest negligible linear relation between them. However, after analyzing matrices created for months we realized that those values are much higher during warmer months. Below we present a few chosen matrices for demonstration.

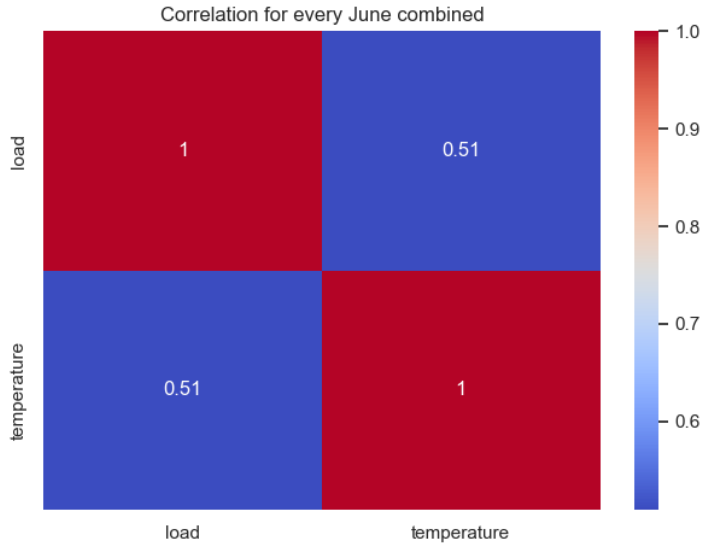

Figure 14: Correlation values for every June in dataset

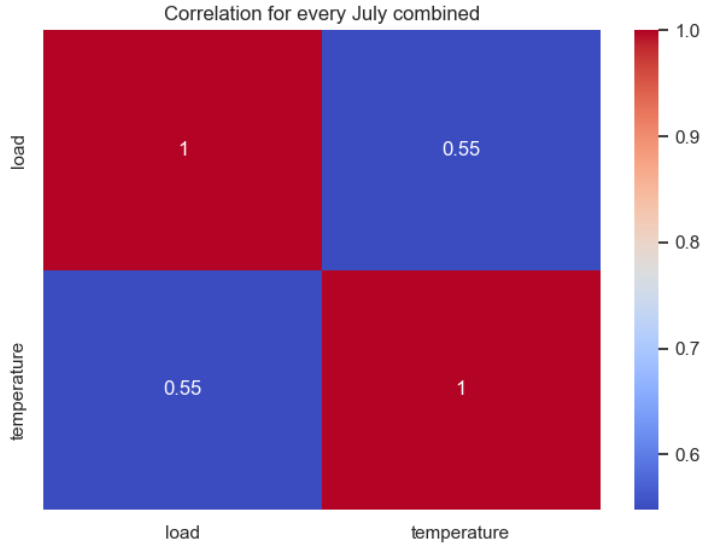

Figure 15: Correlation values for every July in dataset

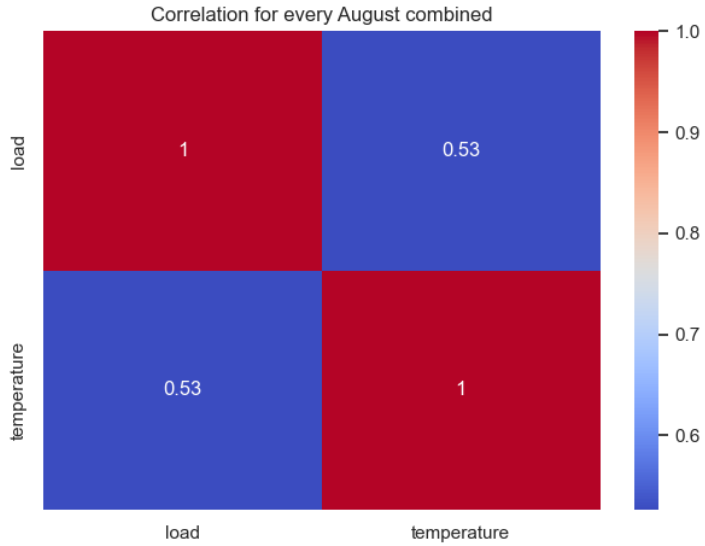

Figure 16: Correlation values for every August in dataset

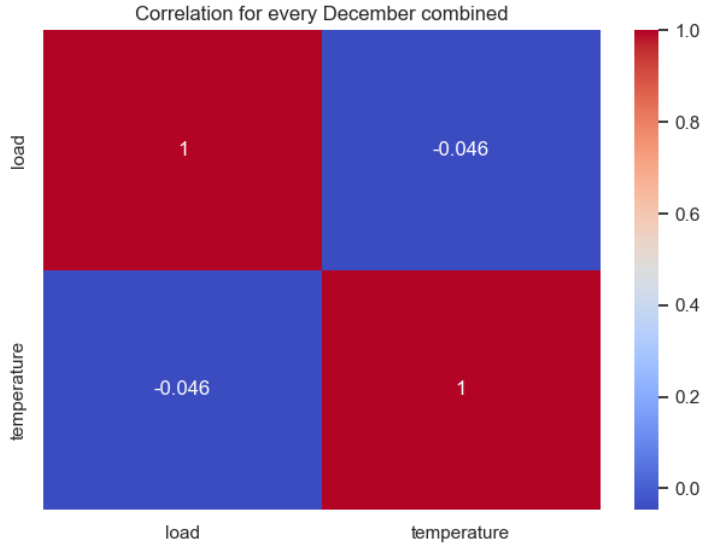

Figure 17: Correlation values for every December in dataset

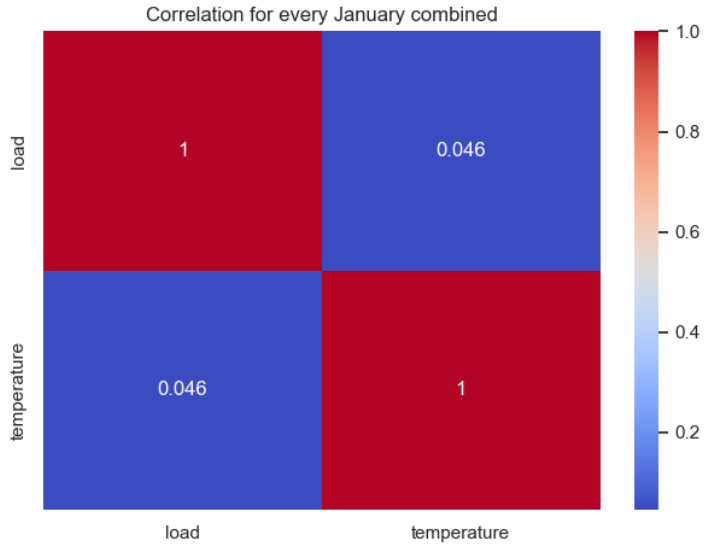

Figure 18: Correlation values for every January in dataset

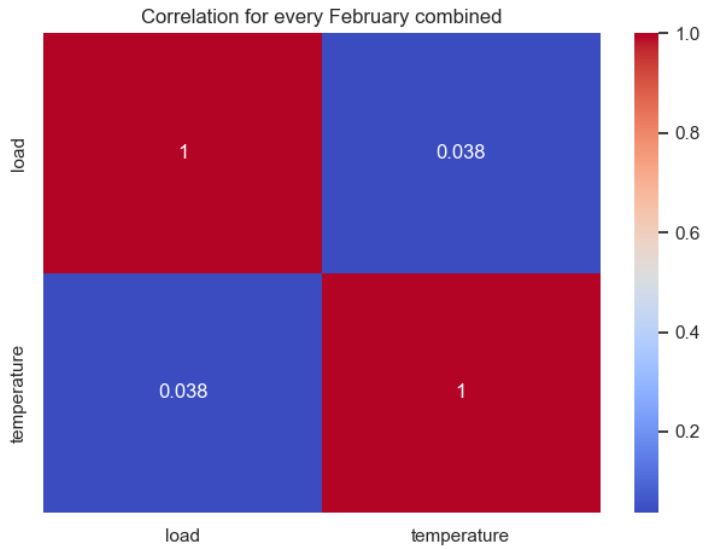

Figure 19: Correlation values for every February in dataset

To show this trend better we also included the correlation matrices for each month of three selected years from our dataset. You can see the results below:

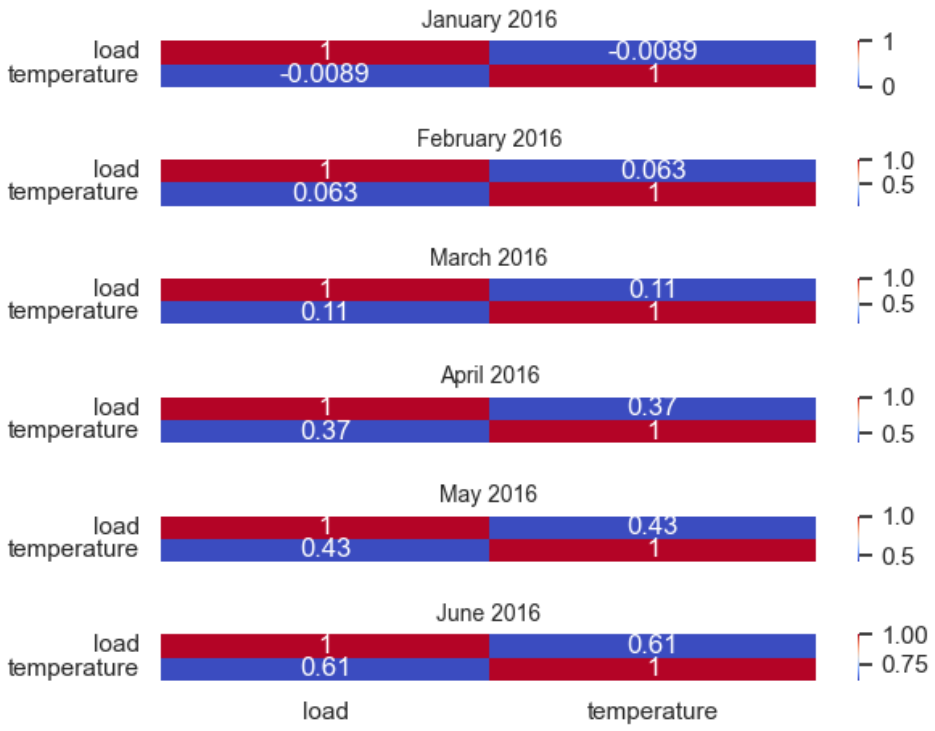

Figure 20: Correlation values for the first 6 months in 2016

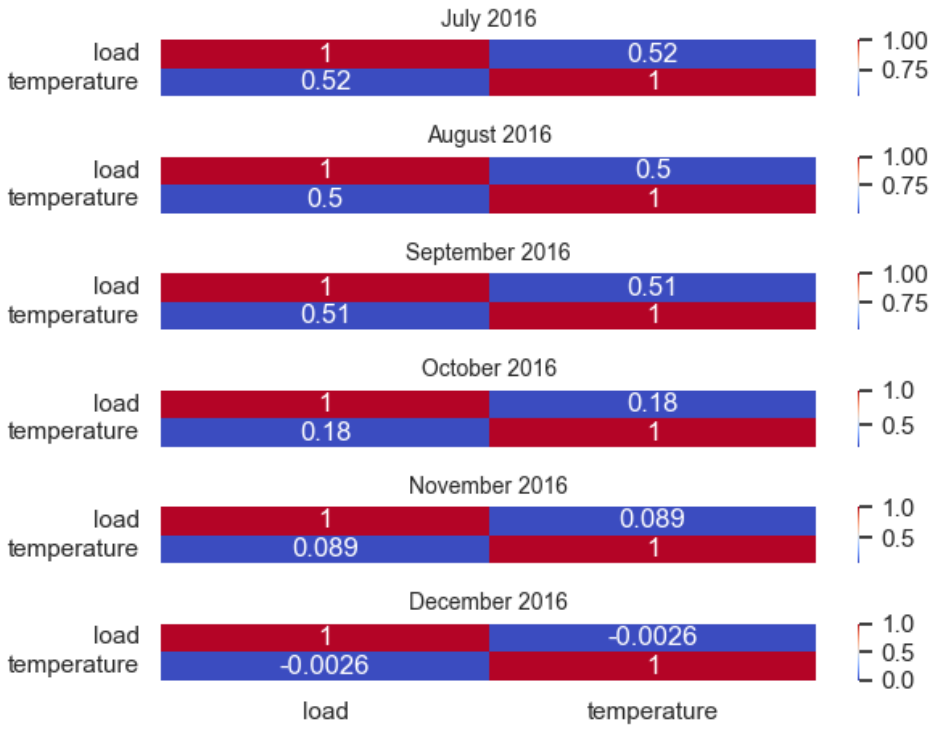

Figure 21: Correlation values for the last 6 months in 2016

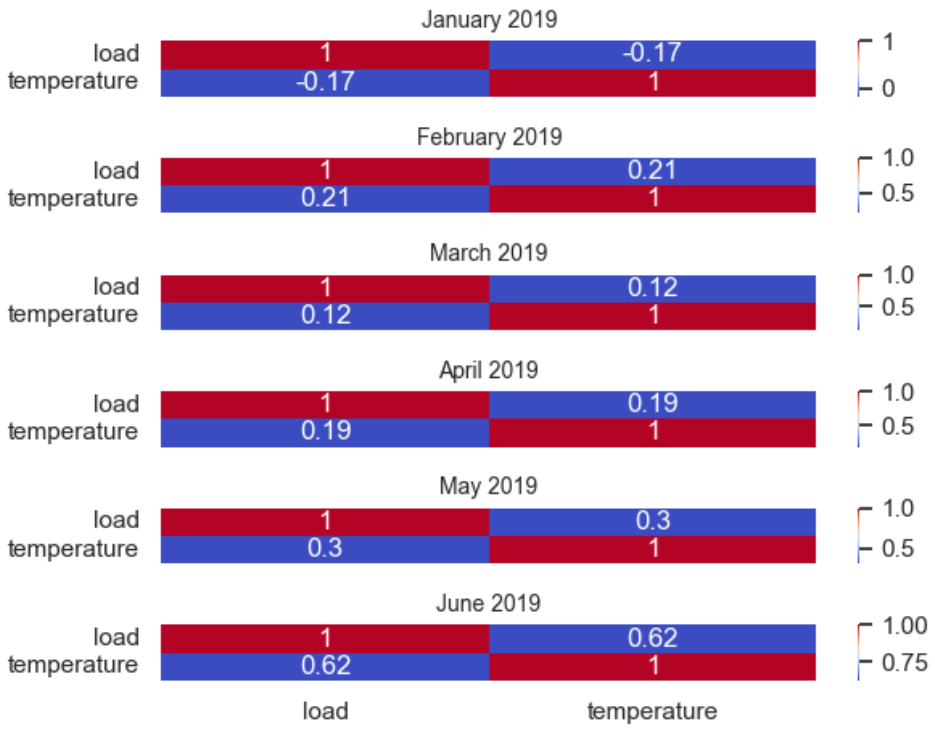

Figure 22: Correlation values for the first 6 months in 2019

Figure 23: Correlation values for the last 6 months in 2019

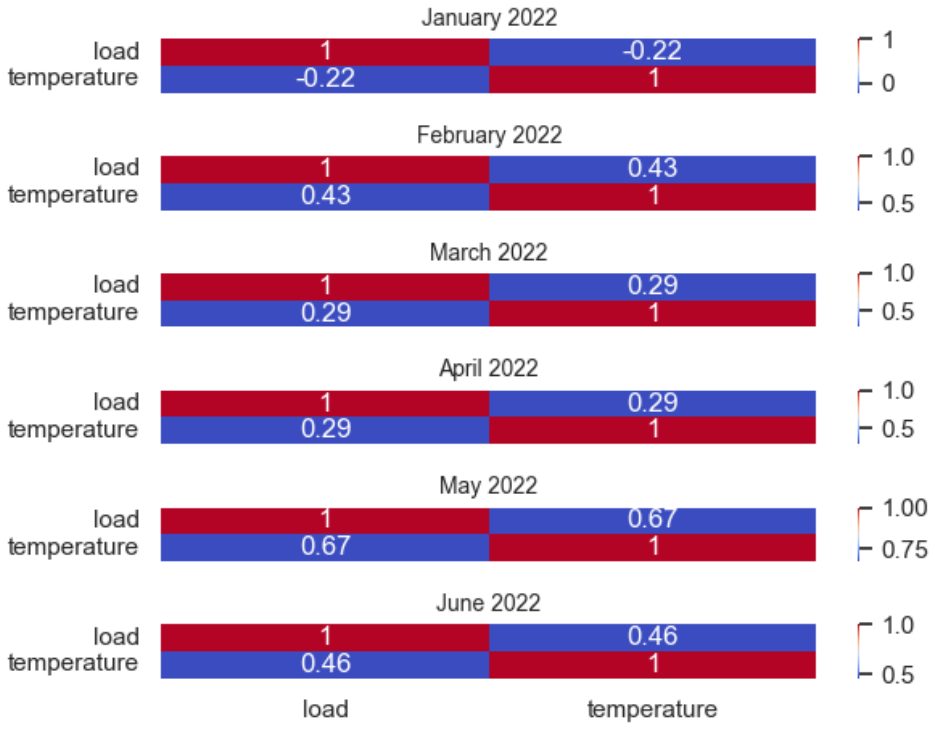

Figure 24: Correlation values for the first 6 months in 2022

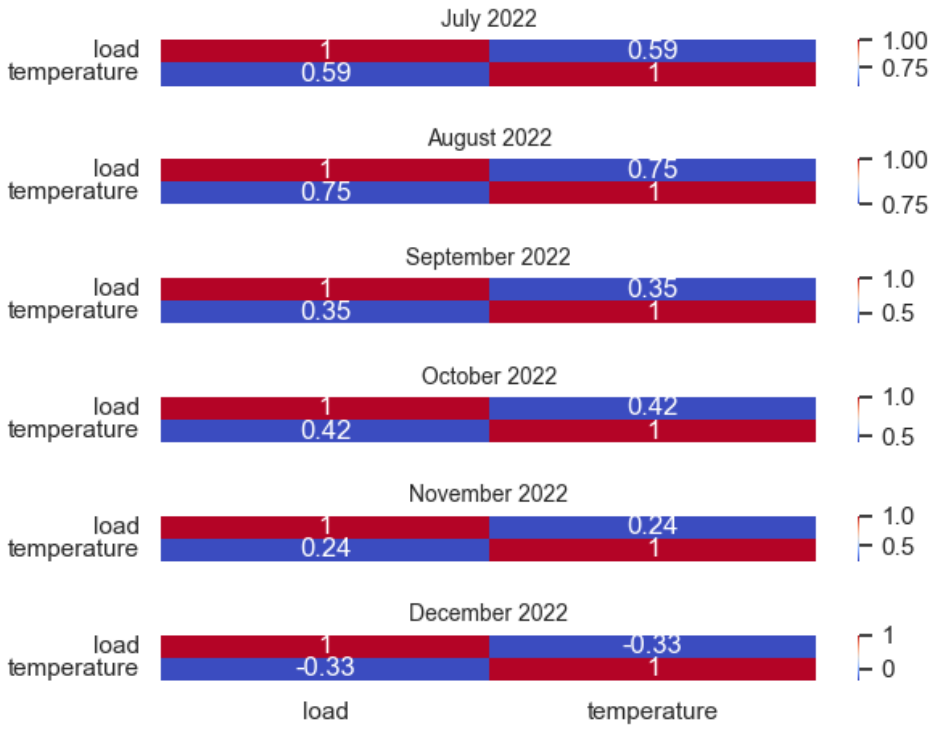

Figure 25: Correlation values for the last 6 months in 2022

It is clear that the linear relation between load and temperature gets higher when temperatures rise, and gets lower when it gets colder. It is likely related to what we have seen when examining line plots - strong overlap between both features between May and September, and barely any overlap, or even plots going in opposite directions at times during the winter season.

3.4.6 Autocorrelation plots

We also prepared an autocorrelation plot for load. This plot shows the correlation between the original data point, and lagged (shifted) data points. In our case 1 lag = 15 minutes, so 96 lags = 24 hours. The chart we created covers the span of 672 lags, or a full week and is depicted below.

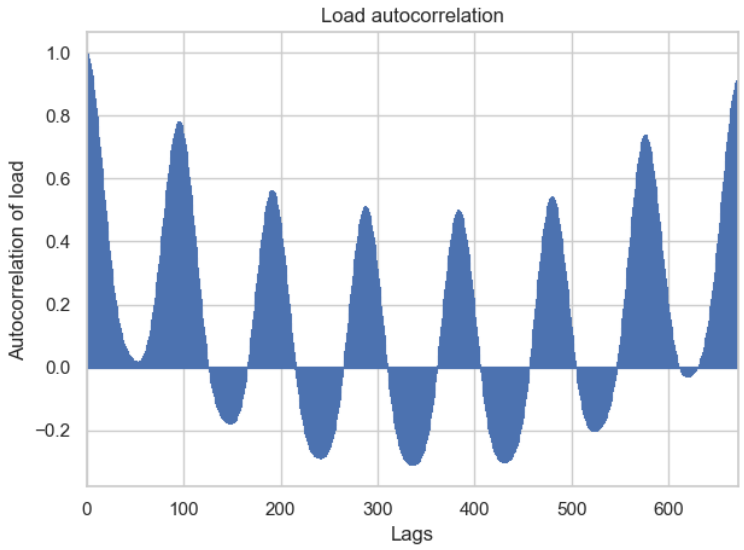

Figure 26: Load autocorrelation (1 lag = 15 minutes)

The 0 point represents the original point, hence the correlation is perfect (equal to 1). We can clearly see the cycles of declining and increasing correlation value. The peaks are spaced about 96 lags from one another, which suggest a strong correlation between data taken on different days, but the same hour. The rightmost value is the second tallest peak on the chart, and it's the data taken exactly one week from our original point, which indicates a significant weekly correlation as well.

3.4.7 Outlier days plots

We employed a visual outlier detection method by plotting a given day's hourly load pattern (mean load per hour) and comparing its deviation from three key baselines:

- The global mean load.
- The mean load for the corresponding day of the week.
- The mean load for the corresponding day of the month.

Special consideration was given to movable feasts, such as Easter-related holidays. As their dates are not fixed, they cannot be identified through simple date-part aggregation and were thus isolated for a separate analysis.

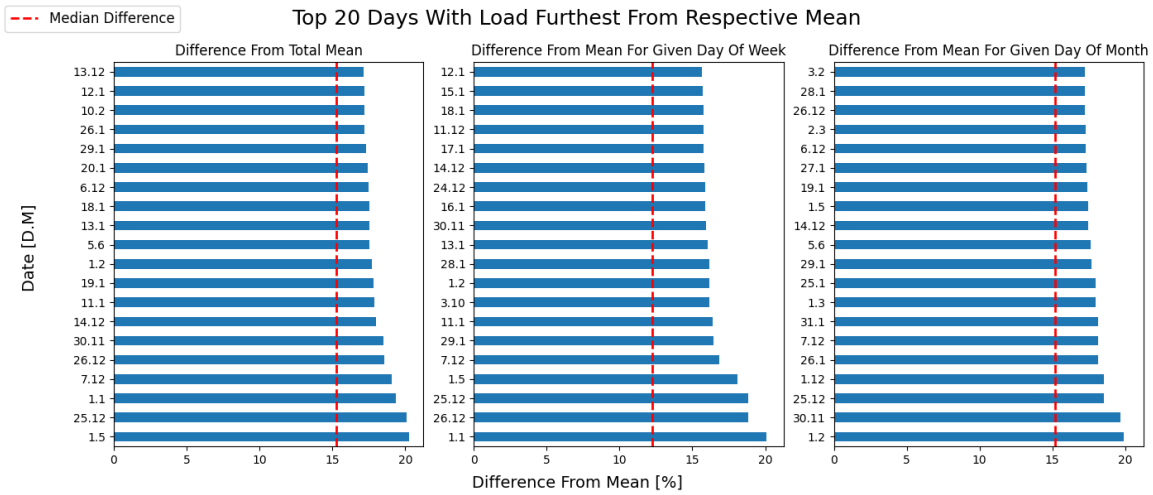

Figure 27: Biggest outliers by difference from mean

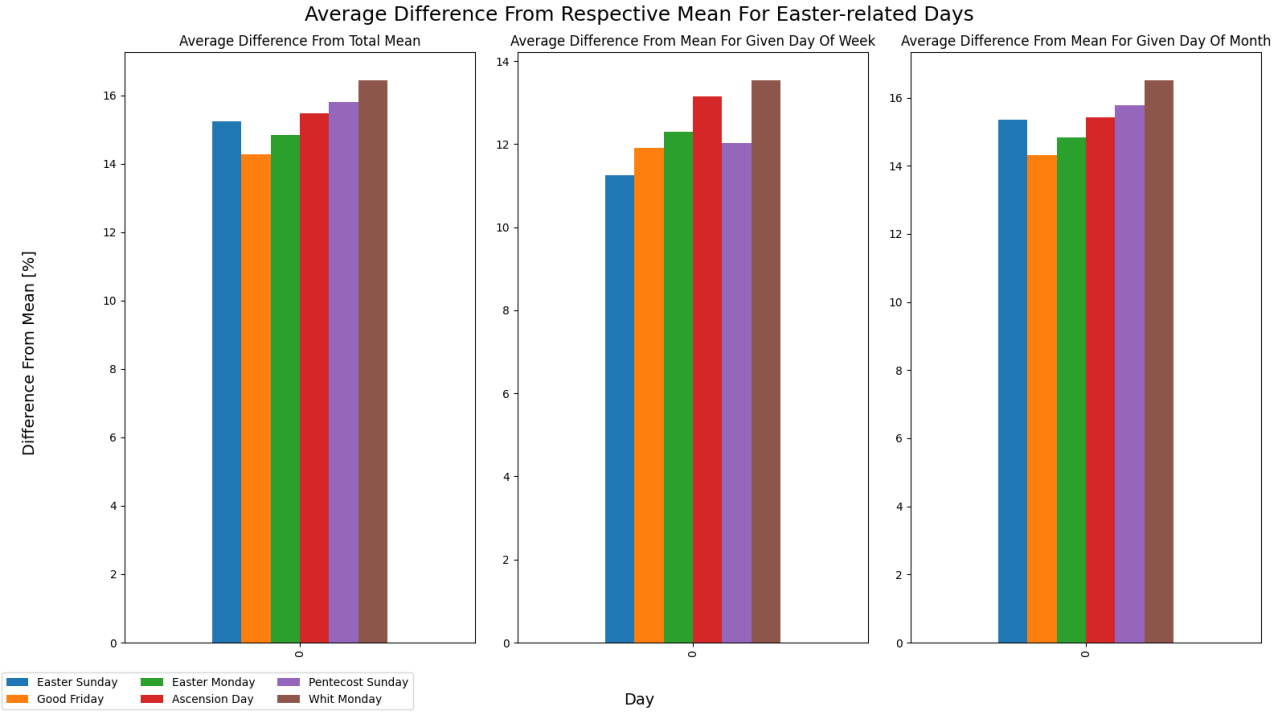

Figure 28: Difference from mean for Easter-related days

The analysis results align with the expectation that key outliers are mostly important holidays.

3.5 Conclusions

After careful analysis of our data set we came to the following conclusions, which might be helpful while developing our neural networks.

- There is a clear correlation between electrical load and temperature values, so temperature is likely to boost model's performance when included during training process.
- Lower average value of load on Saturdays and Sundays, as well as varying averages for different months should be taken into consideration.

- We noticed the power grid load is not exactly constant year by year. To account for this, we engineered a **timestamp_day** feature—a simple day counter starting from 01.01.2015. This continuous feature was chosen over a categorical year feature because it represents the trend as a smooth, linear progression. This avoids the artificial "jumps" a model would see on January 1st and makes it much easier to reliably forecast years that don't appear in the training data.
- Load is strongly correlated with its own value shifted by a full day (or multiple) - parameters like **load_previous_day_timestamp** or **prev_day_temperature** likely to enhance our models when included.
- Due to cyclic nature of electrical load we decided to include recurrent neural networks in our project, since they tend to excel on this kind of data.

4 Models used

4.1 Single Multi-layer Neural Network

4.1.1 Description

A single multi-layer neural network is the simplest model used in the project.

4.1.2 Inputs

The list of inputs [16] chosen for this model:

- **load_timestamp_-1**, **load_timestamp_-2**, **load_timestamp_-3**
- **load_previous_day_timestamp_-2**, **load_previous_day_timestamp_-1**, **load_previous_day_timestamp_0**, **load_previous_day_timestamp_1**, **load_previous_day_timestamp_2**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **day_of_year_sin**, **day_of_year_cos**
- **day_of_week_sin**, **day_of_week_cos**
- **hour_of_day_sin**, **hour_of_day_cos**

4.1.3 Network Architecture

The model consists of an input layer, one hidden dense layer with sigmoid activation, and a single linear output neuron. The hidden layer provides basic non-linear transformation of the input features, while the output layer produces a continuous load value. The network is optimized with Adam and trained using the mean squared error loss function.

4.2 Modular Neural System

4.2.1 Description

Modular neural system builds 24 separate neural networks, each one trained to predict one specific hour out of the 24-hour horizon.

4.2.2 Inputs

The list of inputs [15] chosen for this model:

- **load_timestamp_-1**, **load_timestamp_-2**, **load_timestamp_-3**
- **load_previous_day_timestamp_-2**, **load_previous_day_timestamp_-1**, **load_previous_day_timestamp_0**, **load_previous_day_timestamp_1**, **load_previous_day_timestamp_2**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **day_of_week_sin**, **day_of_week_cos**, **day_of_year_sin**, **day_of_year_cos**
- **timestamp_day**

4.2.3 Network Architecture

The modular system operates in a mode that uses 24 independent neural networks. Each network contains two hidden layers with 25 and 5 neurons, and each one is dedicated to predicting a specific hour in the 24-hour horizon.

4.3 Committee Neural System

4.3.1 Description

Using an ensemble system encapsulating multiple models and combining their outputs is a common method of improving a model's performance.

4.3.2 Inputs

The list of inputs [15] chosen for this model:

- **load_timestamp_-1**, **load_timestamp_-2**, **load_timestamp_-3**
- **load_previous_day_timestamp_-2**, **load_previous_day_timestamp_-1**, **load_previous_day_timestamp_0**, **load_previous_day_timestamp_1**, **load_previous_day_timestamp_2**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **day_of_week_sin**, **day_of_week_cos**, **day_of_year_sin**, **day_of_year_cos**
- **timestamp_day**

The features **hour_of_day_sin** and **hour_of_day_cos** were discarded like in the Modular Neural System, as every sub-model of the Modular System is trained to predict load for a different hour of day.

4.3.3 Network Architecture

A Committee System consisting of 10 Modular Neural Systems was used, with the model's output being the mean output of the 10 modular systems' outputs. Eeach Modular System consisted of 24 multi-layer perceptrons, with 2 hidden layers (25 and 5 neurons respectively), with RELU as the hidden layers' activation function. The model was trained to predict power grid load one step into the future and then used to recursively predict loads 24 hours ahead, while updating the values of features accordingly.

4.4 Rule-Aided Neural System

4.4.1 Description

The Rule-Aided Neural System aims to improve on the Committee Modular System's predictions by combining the machine learning model with special, hand-crafted rules.

4.4.2 Inputs

The inputs for this model are exactly the same as for the Committee Neural System and for the Modular Neural System.

4.4.3 Network Architecture

The Rule-Aided Neural System uses predefined rules to make predictions for special German holidays: it uses the mean power grid load trend for that day, shifted to fit with the previous day's load pattern. For normal days, predictions of the Committee Neural System are used.

Based on the data analysis described in 3.4.7, the special days taken into account are:

- 01.01 - New Year's Day.
- 02.01 - the day after New Year's.
- 25.12 - First day of Christmas.
- 26.12 - Second day of Christmas.
- 01.05 - Labour Day.
- 03.10 - German Unity Day.
- 01.11 - All Saints' Day.
- Easter Sunday
- Eeaster Monday
- Good Friday
- Ascension Day
- Pentecost Sunday
- Whit Monday
- 29.01 - One of the top outliers, reason unknown.
- 07.12 - One of the top outliers, close to. Nikolaustag.

4.5 Convolutional Neural Network

4.5.1 Description

CNN architecture is mostly known for being used in image processing, however a one-dimensional convolution layer is able to work with time series and learn patterns just like RNNs. We decided to include them in our project to compare their capabilities in handling such data to the architectures that are more commonly associated processing serial inputs.

4.5.2 Inputs

The list of input features [10] chosen for this model:

- **load**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **hour_of_day_sin, hour_of_day_cos, day_of_week_sin, day_of_week_cos, day_of_year_sin, day_of_year_cos**
- **timestamp_day**

4.5.3 Network Architecture

Described model was trained on 1 hour intervals. It takes **168** samples of **10** previously listed features as an input and returns **24** predicted values. When it comes to the hidden layers, the first one is **Convolution layer with 32 units**. It is followed by **Spatial Dropout layer with rate 0.1** and an **Average Pooling layer with size of 2**. At the end there are two **Dense layers with 32 and 16 units**, which makes a total of 5 hidden layers.

4.6 Recurrent Neural Network

4.6.1 Description

SimpleRNN represents the most fundamental recurrent neural network architecture and serves as the baseline for many more advanced sequence-processing models. Unlike convolution-based approaches, SimpleRNNs explicitly model temporal dependencies by maintaining a hidden state that is updated at each time step. While they are prone to issues such as vanishing gradients, they remain useful for capturing short-term temporal patterns in sequential data.

4.6.2 Inputs

The list of inputs [10] chosen for this model:

- **load**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **hour_of_day_sin, hour_of_day_cos, day_of_week_sin, day_of_week_cos, day_of_year_sin, day_of_year_cos,**
- **timestamp_day**

4.6.3 Network Architecture

The SimpleRNN model described above consists of one **SimpleRNN** layer with **64 units** and one **Dense** layer with **24 units** . The model is trained on 1 hour intervals. It takes **48** samples (data collected over 2 days) of **10** features (mentioned in 4.6.2) as input and returns **24** predicted values (one for each hour of the day).

4.7 Recurrent Neural Network with LSTM

4.7.1 Description

LSTM is an improved version of a classical RNN architecture, designed to deal with exploding/-vanishing gradient problem. They are commonly used to make predictions using time series, and tend to do especially well on highly cyclical data, such as the electrical load in our case. For that reason we included them in this project, expecting low MAPE results.

4.7.2 Inputs

The list of input features [10] chosen for this model:

- **load**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **hour_of_day_sin, hour_of_day_cos, day_of_week_sin, day_of_week_cos, day_of_year_sin, day_of_year_cos**
- **timestamp_day**

4.7.3 Network Architecture

Described model was trained on 1 hour intervals. It takes **168** samples of **10** aforementioned features as an input (data collected over one week) and returns **24** predicted values (1 for each hour of the day the predictions were made for). As for the hidden layers, the first two of them are **LSTM layers with 64 and 32 units** respectively. They are followed by another two **Dense layers with 32 and 16 units**, making 4 the total number of hidden layers.

4.8 CNN-LSTM Hybrid

4.8.1 Description

Hoping for an increase in performance we decided to combine both CNN and LSTM in one model, since they both operate on the same serial data, and are likely able to complement each other.

4.8.2 Inputs

The list of input features [10] chosen for this model:

- **load**
- **prev_3_temperature_timestamps_mean**

- **prev_day_temperature_5_timestamps_mean**
- **hour_of_day_sin, hour_of_day_cos, day_of_week_sin, day_of_week_cos, day_of_year_sin, day_of_year_cos**
- **timestamp_day**

4.8.3 Network Architecture

Described model was trained on 1 hour intervals. It takes **168** samples of **10** features as an input and returns **24** predictions. As for the hidden layers, the first one is **Convolution layer with 64 units**. It is followed by **Spatial Dropout layer with rate 0.1** and an **Average Pooling layer with size of 2**. After that there is an **LSTM layer with 32**. At the end there are two **Dense layers with 32 and 16 units**, which makes a total of 6 hidden layers.

4.9 Sequence-To-Sequence Encoder-Decoder Network

4.9.1 Description

The Sequence-to-sequence model uses a simple encoder-decoder architecture, with one set of LSTM layers outputting a 'context vector' that is later passed to another set of LSTM layers as input at every time step, producing a 24-hour forecast. Unlike in the classical 'seq2seq' models used in natural language processing, this model doesn't make use of initializing the decoder's initial hidden state with the context vector or running the decoder in an autoregressive loop.

The idea behind using this model is that, as opposed to standard LSTM networks, its outputs for different time steps are not independent of each other.

4.9.2 Inputs

The list of input features [10] chosen for this model:

- **load**
- **prev_3_temperature_timestamps_mean**
- **prev_day_temperature_5_timestamps_mean**
- **hour_of_day_sin, hour_of_day_cos, day_of_week_sin, day_of_week_cos, day_of_year_sin, day_of_year_cos**
- **timestamp_day**

4.9.3 Network Architecture

The Sequence-To-Sequence Encoder-Decoder Network was to predict 24-hour **load** sequences from 168 hours of past data for every feature.
The network constisted of 2 LSTM layers, one for the encoder and one for the decoder, with 32 cells in each of them. A separate dense layer was used to map each time step's outputs to a single value representing the **load**.

5 Models' performance

5.1 Single Multi-layer Neural Network

5.1.1 MAPE Results

In 20-hour forecasts, the Single Multi-layer Neural Network achieved a **8.69% MAPE** as its best result, using 25 hidden neurons and 20 training epochs, with detailed MAPE values presented in the table below. One o'clock was the starting hour.

12 neurons in the hidden layer								
Epochs	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Mean
15	19.90	8.70	10.15	6.97	6.39	5.94	16.17	10.59
20	17.45	8.35	9.82	7.30	6.35	5.47	13.31	9.71
25 neurons in the hidden layer								
Epochs	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Mean
15	17.96	7.97	8.81	5.62	6.29	5.40	13.49	9.37
20	17.41	8.63	8.39	6.01	6.72	4.97	8.60	8.69

5.1.2 Predictions

Plots showcasing the Single Multi-layer Neural Network predictions for a few selected 20-hour ranges are shown below:

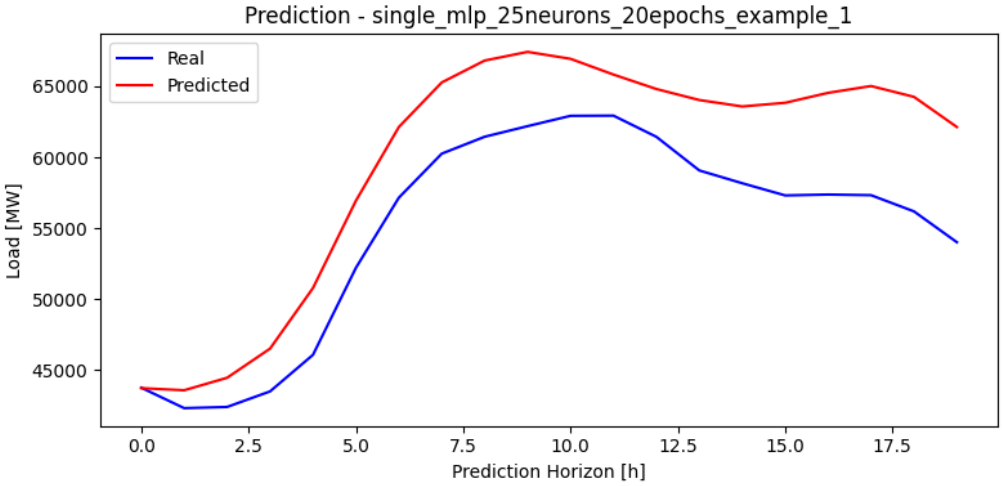

Figure 29: MLP Example 1

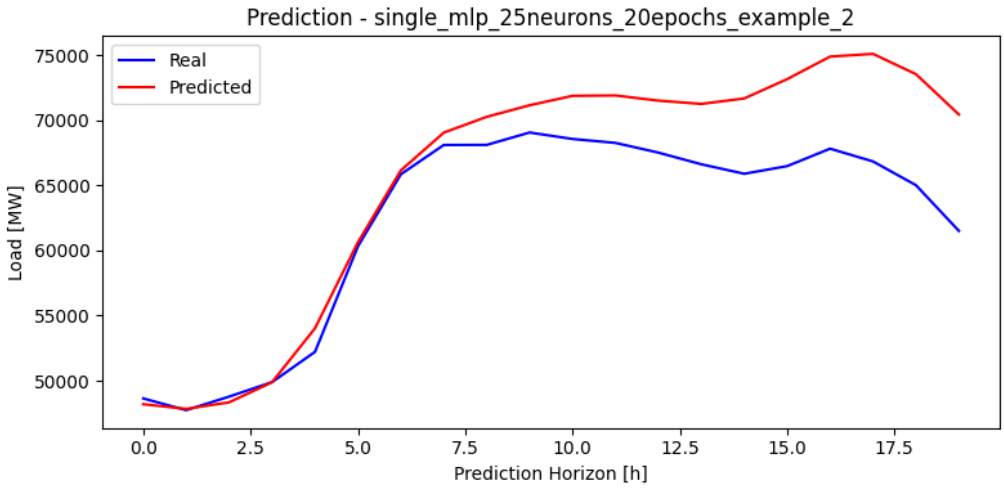

Figure 30: MLP Example 2

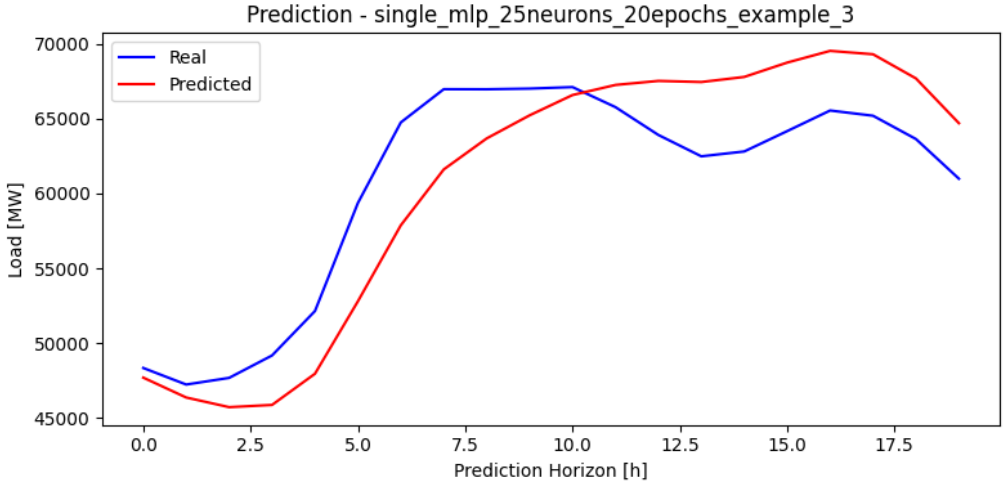

Figure 31: MLP Example 3

5.2 Modular Neural System

5.2.1 MAPE Results

In 24-hour forecasts, the Modular System achieved a **4.28% MAPE** for the best starting hour with MAPE scores for each starting hour shown in table below:

Starting hour	MAPE
0	6.94%
1	6.95%
2	7.06%
3	4.59%
4	4.53%
5	4.31%
6	4.28%

Continued on next page

Starting hour	MAPE
7	4.30%
8	4.74%
9	5.28%
10	5.74%
11	5.91%
12	5.96%
13	6.30%
14	6.64%
15	6.96%
16	7.24%
17	7.58%
18	6.83%
19	7.01%
20	7.01%
21	7.09%
22	7.24%
23	7.00%
Mean	**6.20%**

The mean MAPE scores for each forecasted hour in a 24-hour forecast are shown on the chart below:

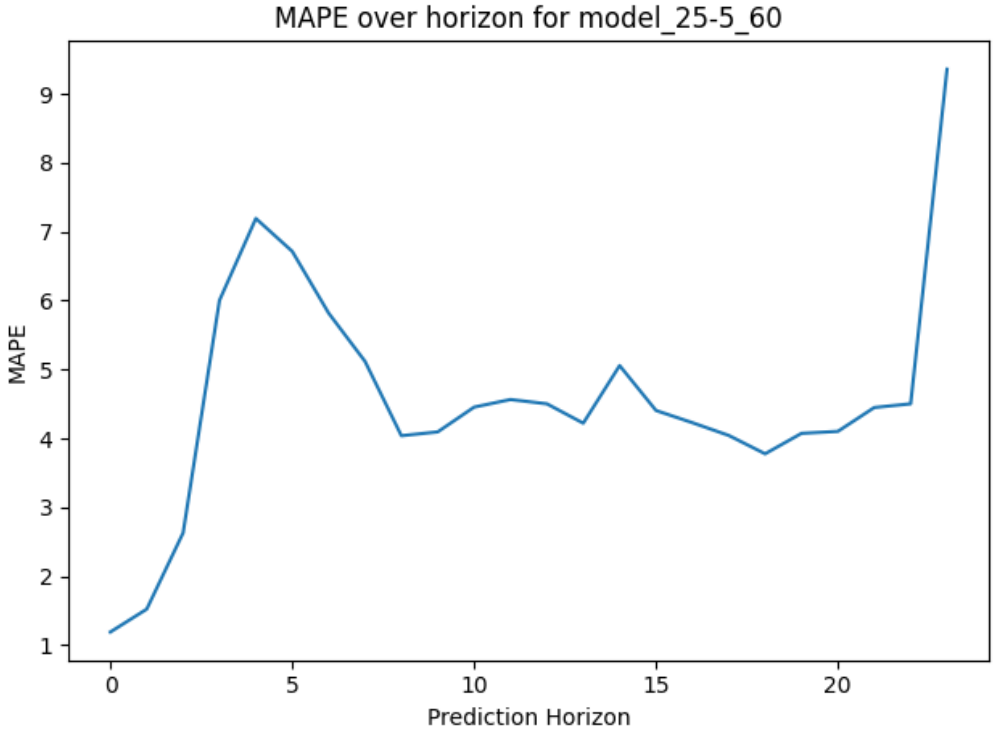

Figure 32: Module System Example 1

5.2.2 Predictions

Plots showcasing the Modular System's predictions for a few selected 24-hour ranges are shown below:

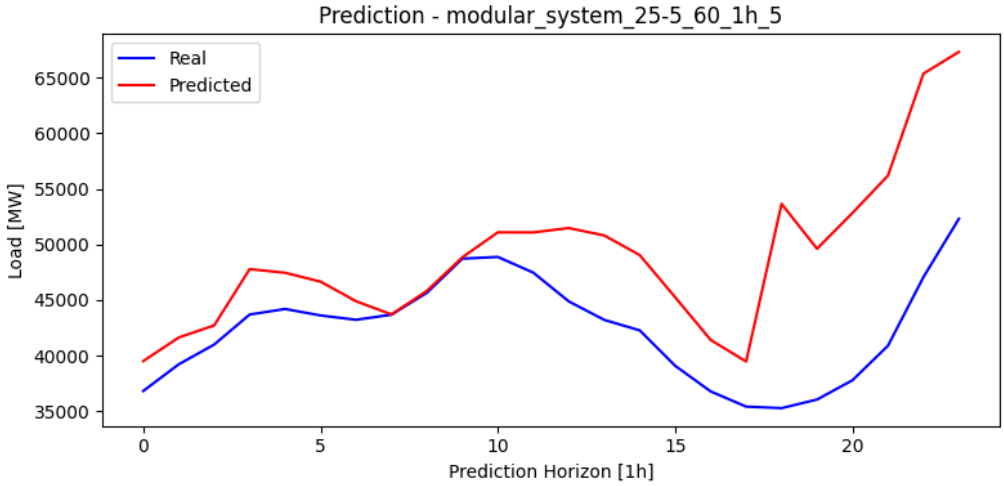

Figure 33: Module System Example 1

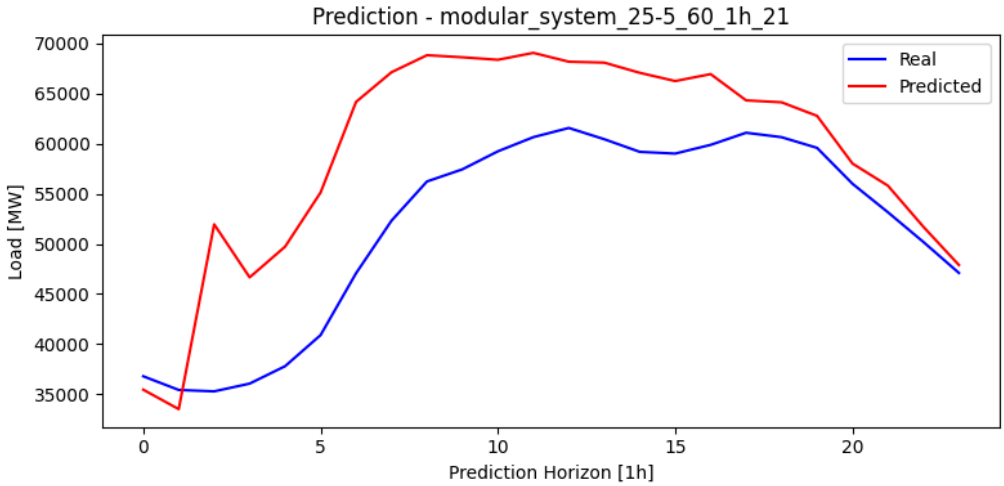

Figure 34: Module System Example 2

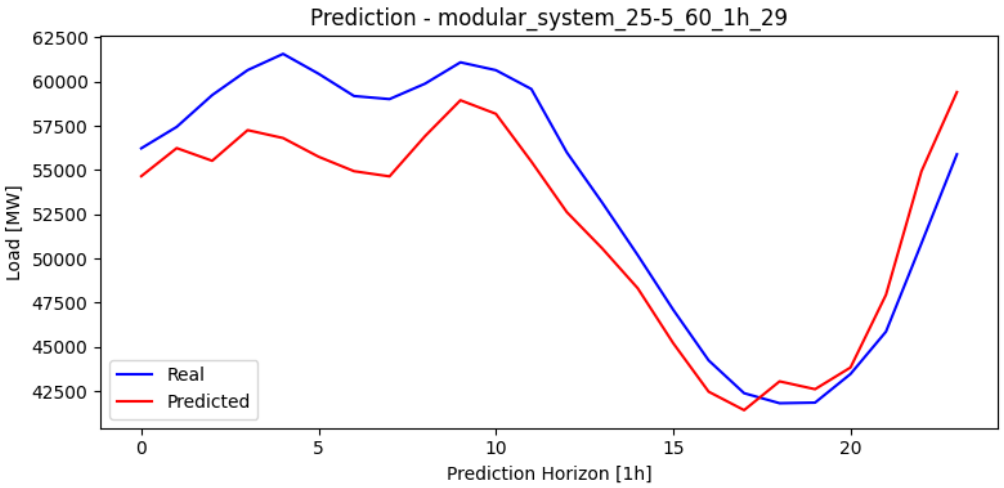

Figure 35: Module System Example 3

5.3 Committee Neural System

5.3.1 MAPE Results

In 24-hour forecasts, the Committee system achieved **2.73% MAPE** for the best starting hour, with MAPE scores for each starting hour shown in the table below:

Starting hour	MAPE
0	3.52%
1	3.25%
2	3.13%
3	3.22%
4	3.13%
5	2.87%
6	2.73%
7	2.84%
8	3.1%
9	3.25%
10	3.06%
11	2.96%
12	2.85%
13	2.96%
14	3.1%
15	3.23%
16	3.34%
17	3.43%
18	3.53%
19	3.46%
20	3.54%
21	3.55%
22	3.73%
23	3.35%

Continued on next page

Starting hour	MAPE
Mean	3.21%

The mean MAPE scores for each forecasted hour in a 24-hour forecast are shown on the chart below:

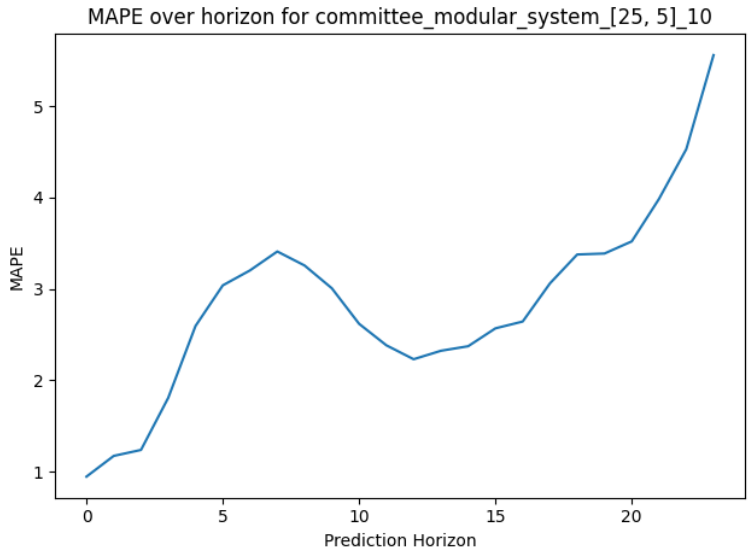

Figure 36: Average MAPE for time stamp in prediction horizon

5.3.2 Predictions

Plots showcasing the Committee Modular System's predictions for a few selected 24-hour ranges are shown below:

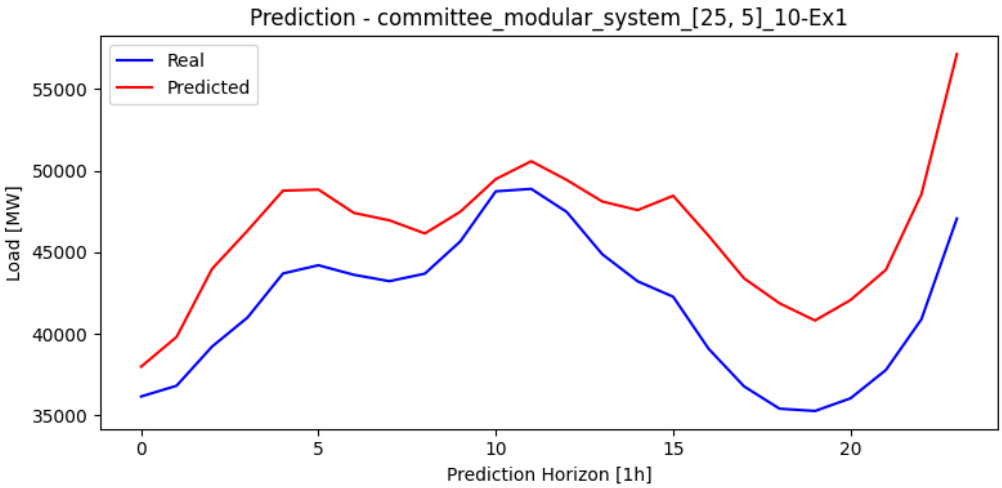

Figure 37: Committee Example 1

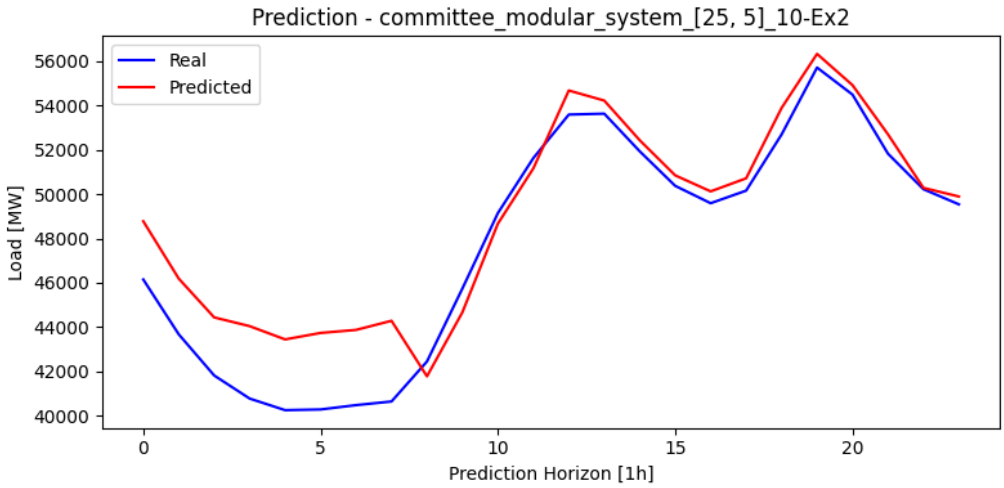

Figure 38: Committee Example 2

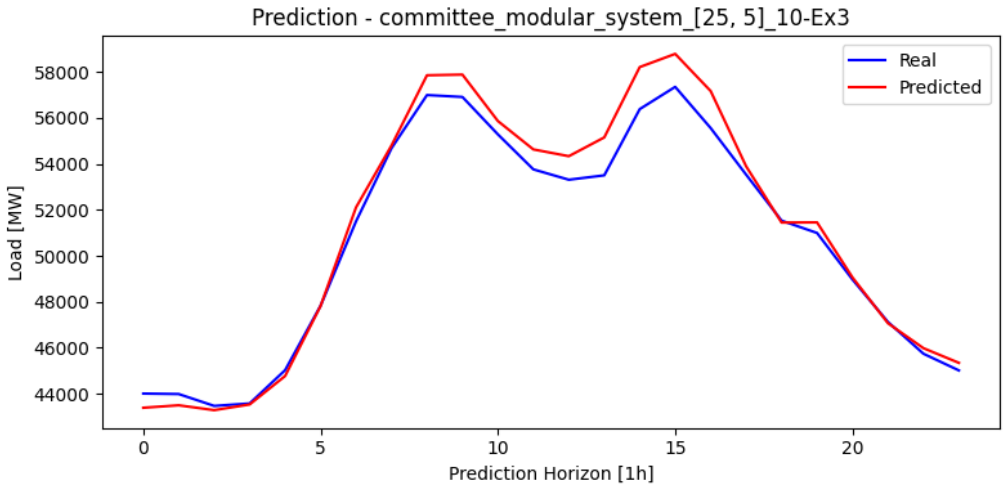

Figure 39: Committee Example 3

5.4 Rule-Aided Neural System

5.4.1 MAPE Results

For 24-hour forecasting, the Rule-Aided System's MAPE was **2.57%** for the best starting hour, with MAPE scores for each starting hour shown in the table below:

Starting hour	MAPE
0	2.78%
1	2.63%
2	2.63%
3	2.76%
4	2.74%
5	2.61%
6	2.61%

Continued on next page

Starting hour	MAPE
7	2.57%
8	2.65%
9	2.72%
10	2.69%
11	2.61%
12	2.63%
13	2.58%
14	2.64%
15	2.69%
16	2.76%
17	2.78%
18	2.76%
19	2.76%
20	2.77%
21	2.72%
22	2.74%
23	2.72%
Mean	**2.69%**

The Rule-Aided System was slightly better than the Committee System for the best starting hour, but showed significantly better results across all starting hours.

The mean MAPE scores for each forecasted hour in a 24-hour forecast are shown on the chart below:

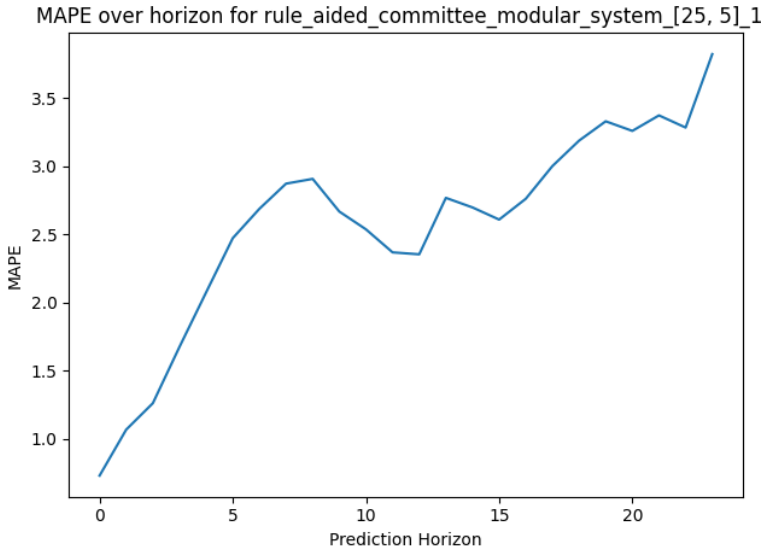

Figure 40: Average MAPE for time stamp in prediction horizon

5.4.2 Predictions

Plots showcasing the Rule-Aided Committee Modular System's predictions for a few selected 24-hour ranges are shown below:

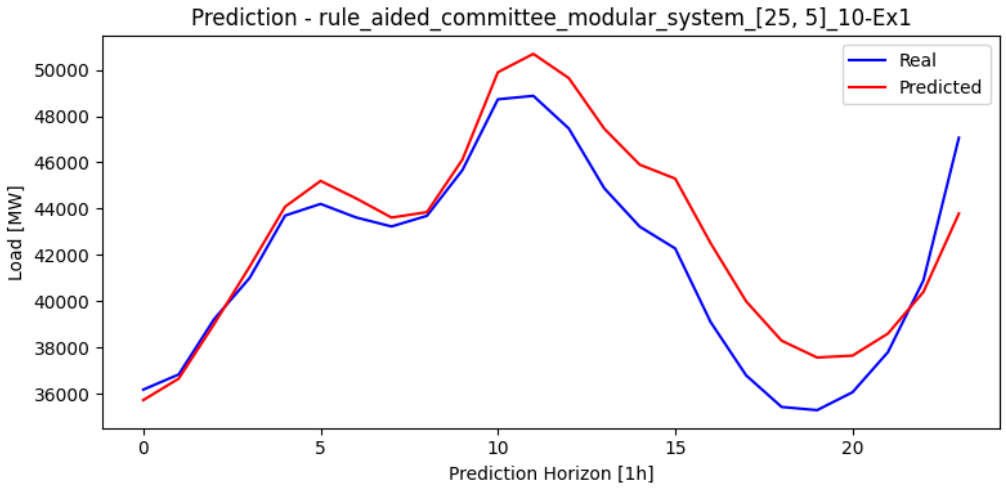

Figure 41: Rule-Aided Example 1

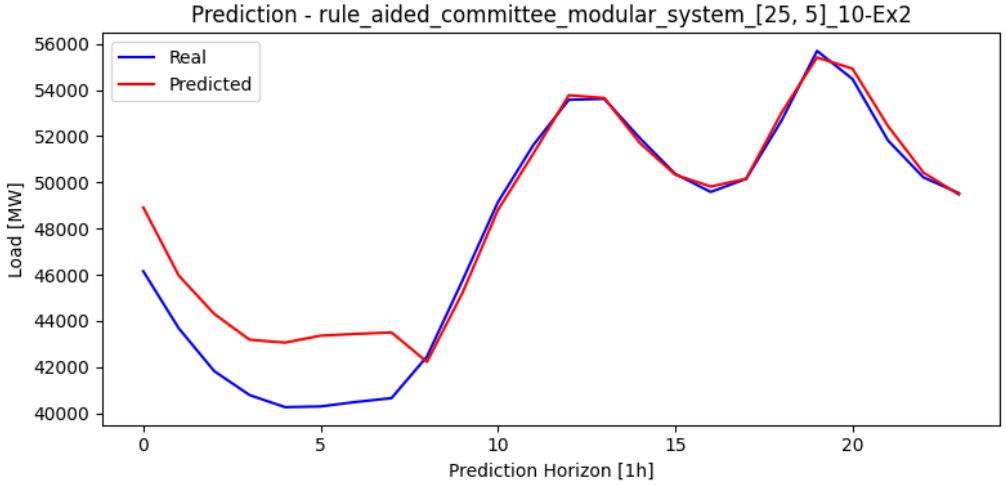

Figure 42: Rule-Aided Example 2

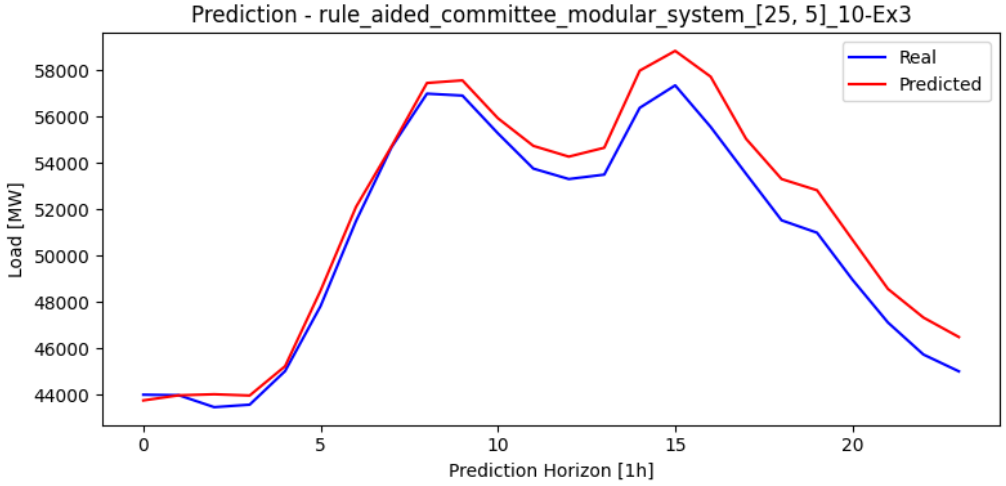

Figure 43: Rule-Aided Example 3

5.5 Convolutional Neural Network

5.5.1 MAPE Results

Due to model's small size, extremely short training time (about 5-10 minutes) and CNNs not being generally associated with serial data processing, we were suprised to see that it scored **2.80** MAPE on our test set. This low score was likely achieved due to the simple nature of electrical load patterns, which our CNN was seemingly able to pick up very well. As usual, the further the prediction the worse MAPE gets, which is visible on the following plot:

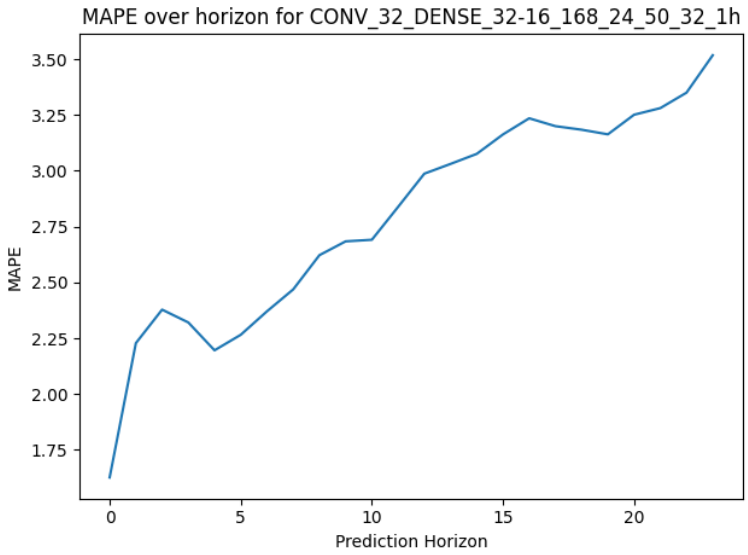

Figure 44: Average MAPE for time stamp in prediction horizon

5.5.2 Predictions

Since test sets spans over two years worth of data, we decided to only showcase a few randomly selected days for the sake of readability. Below you can find plots comparing real values to the predictions made by our CNN.

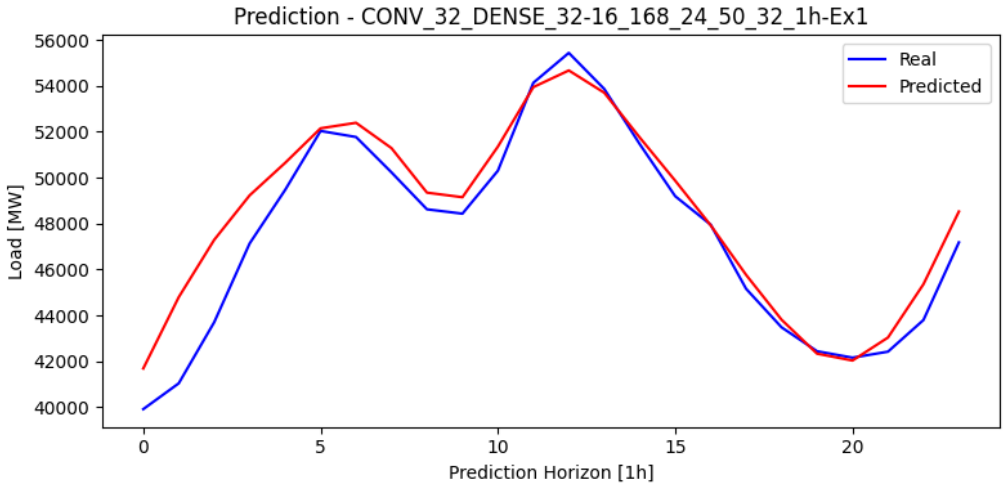

Figure 45: CNN Example 1

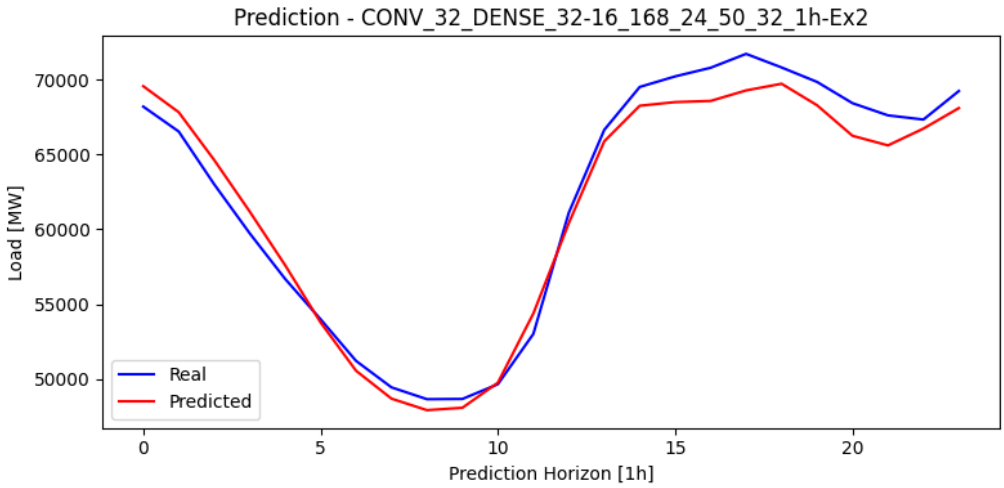

Figure 46: CNN Example 2

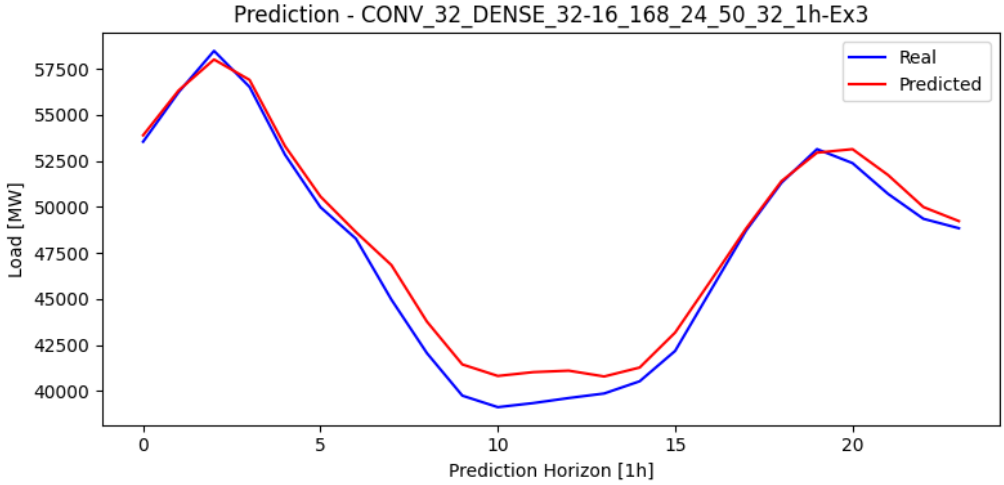

Figure 47: CNN Example 3

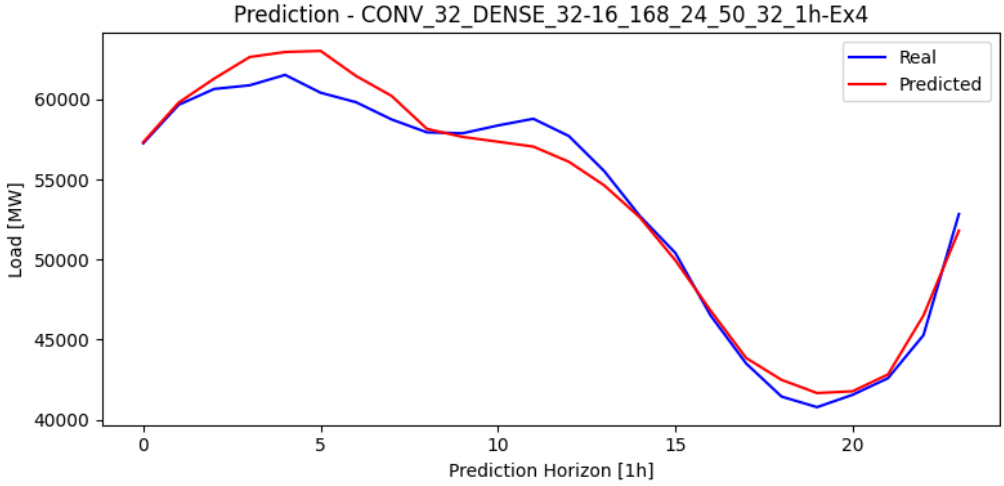

Figure 48: CNN Example 4

5.6 Recurrent Neural Network

5.6.1 MAPE Results

The Simple RNN network manages to get an average MAPE of **2.95**. However, the average results for individual timestamps vary. From the chart below, we can see that the MAPE increases with each point.

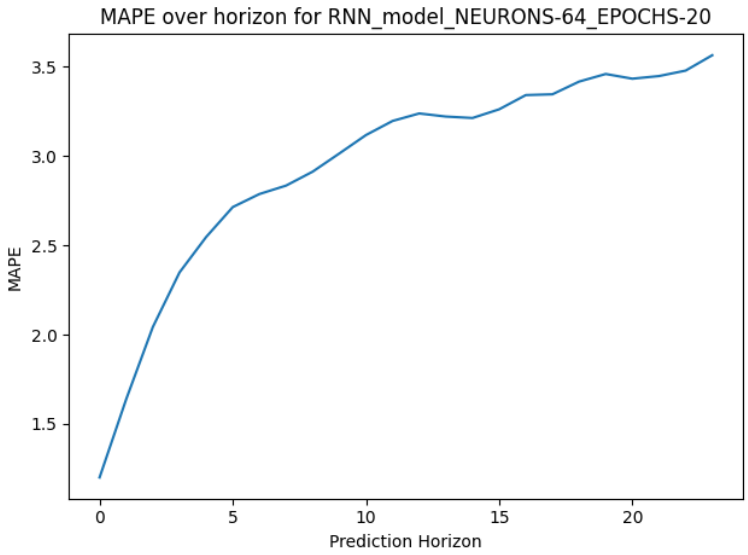

Figure 49: Average MAPE for time stamp in the prediction horizon

5.6.2 Predictions

The plots below compare real values to the predictions made by the SimpleRNN model. The examples shown are only a few randomly selected days, since the test data span over two years.

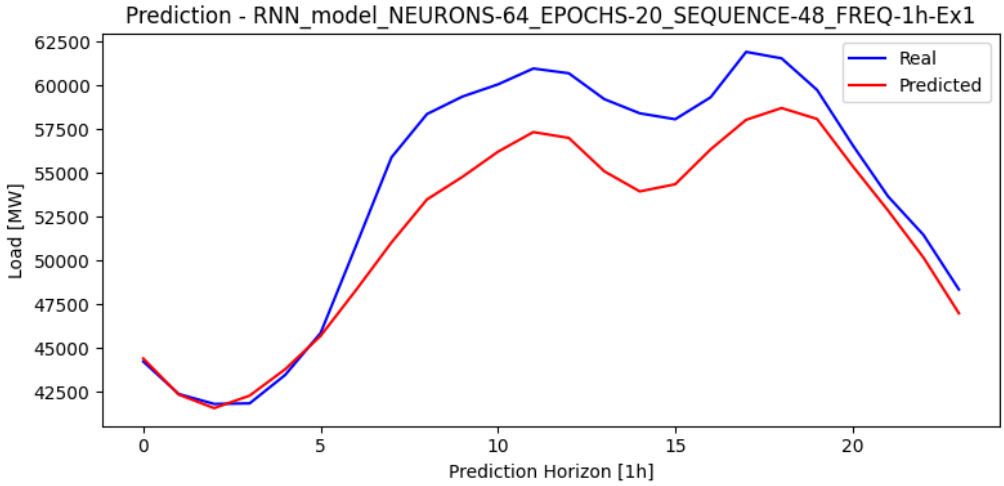

Figure 50: RNN Example 1

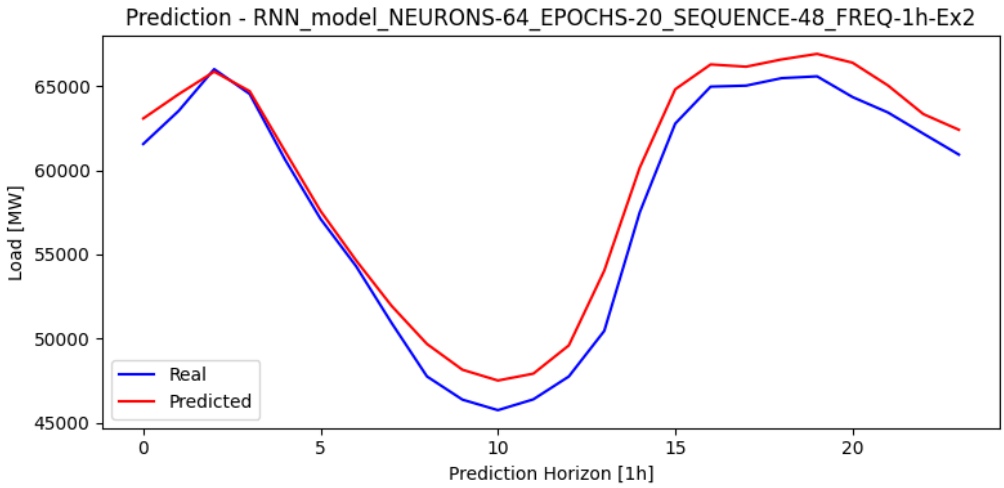

Figure 51: RNN Example 2

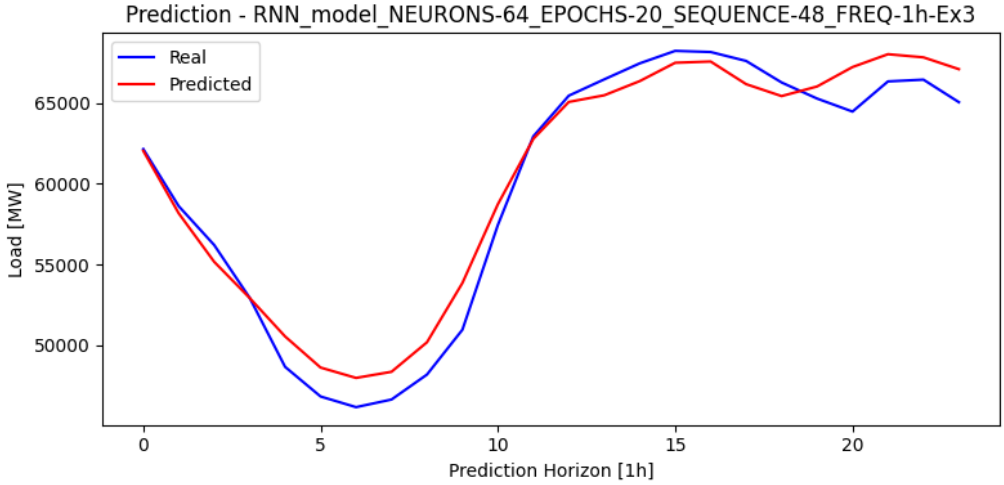

Figure 52: RNN Example 3

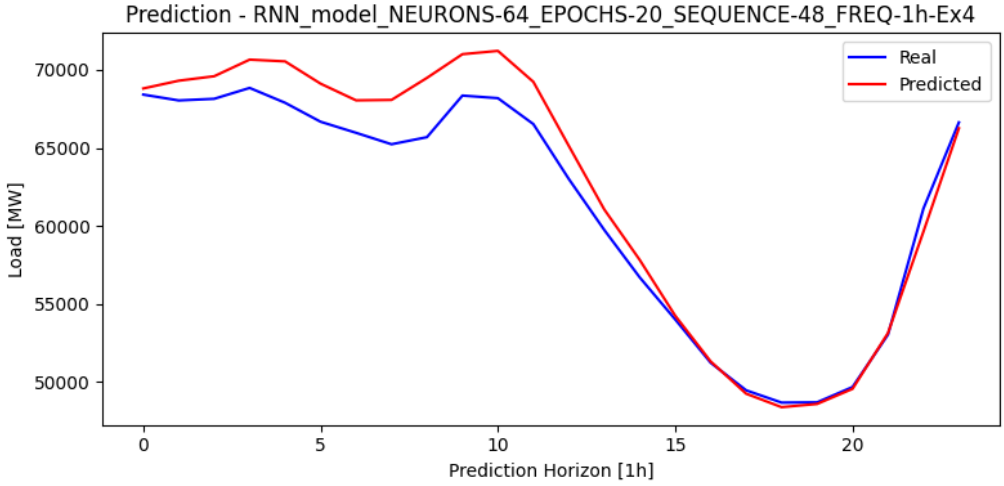

Figure 53: RNN Example 4

5.7 Recurrent Neural Network with LSTM

5.7.1 MAPE Results

LSTM network described earlier has managed to achieve an average MAPE of **3.01**. The average results for individual time stamps in the prediction horizon however, were a bit more varied, and tended to increase with each point. You can see those values on a line chart below.

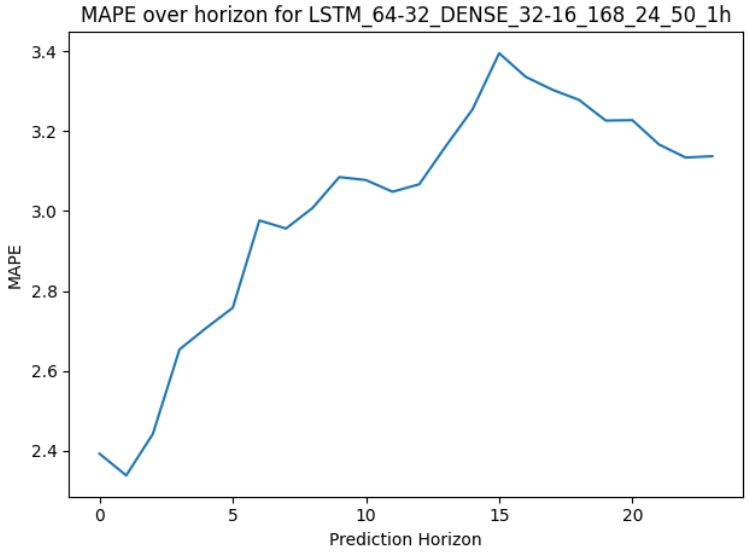

Figure 54: Average MAPE for time stamp in prediction horizon

5.7.2 Predictions

Since test sets spans over two years worth of data, we decided to only showcase a few randomly selected days for the sake of readability. Below you can find plots comparing real values to the predictions made by our LSTM.

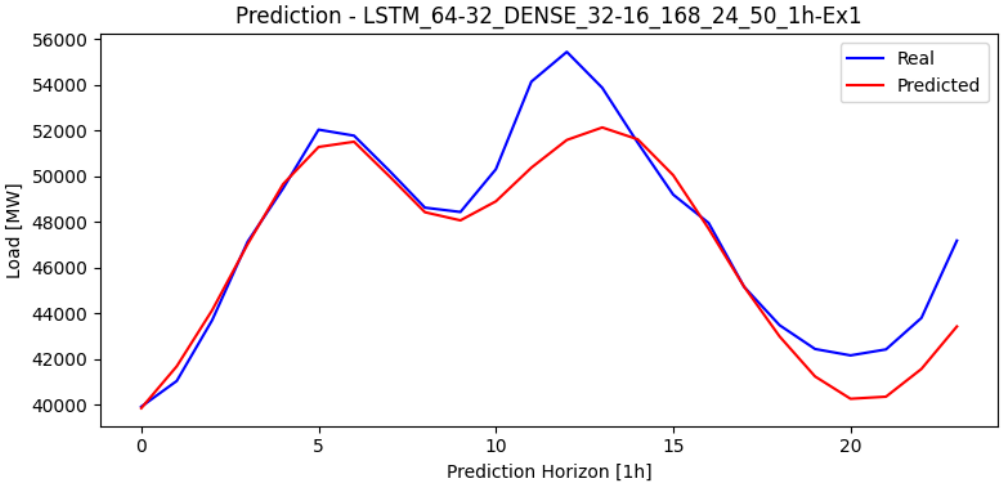

Figure 55: LSTM Example 1

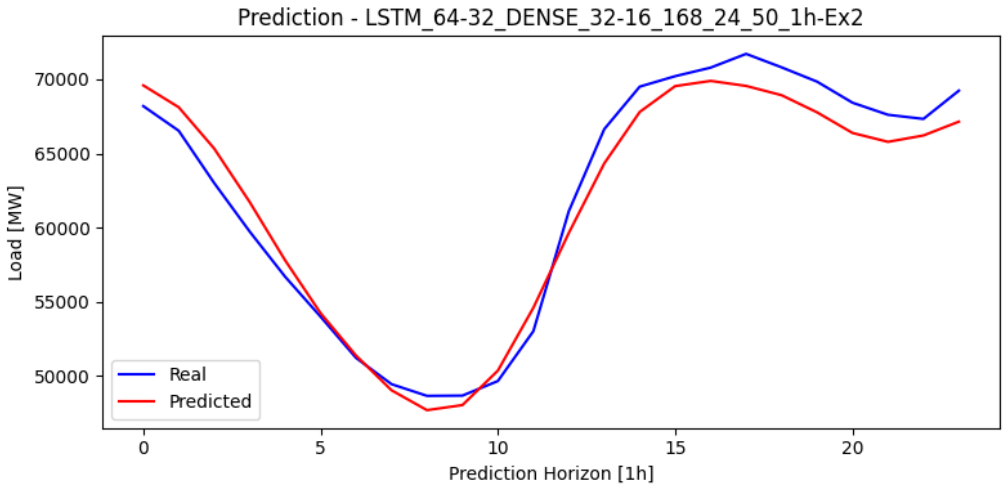

Figure 56: LSTM Example 2

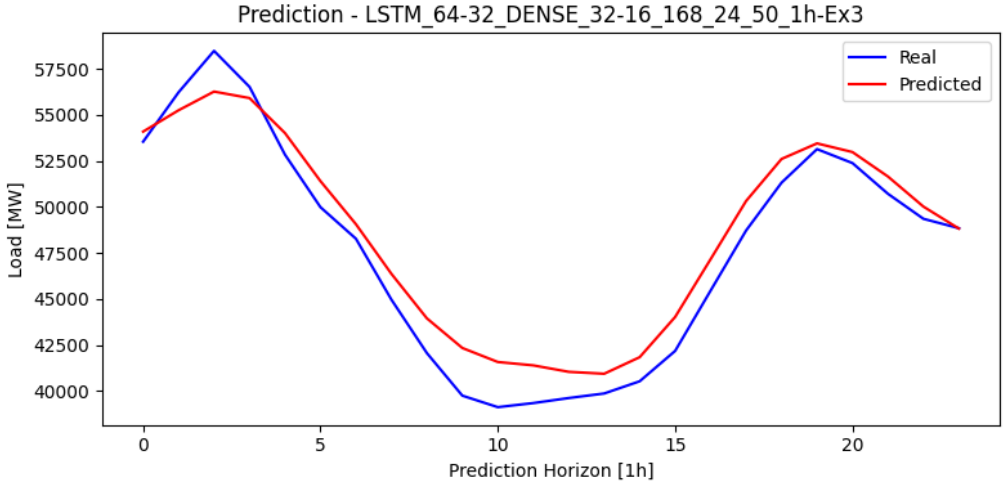

Figure 57: LSTM Example 3

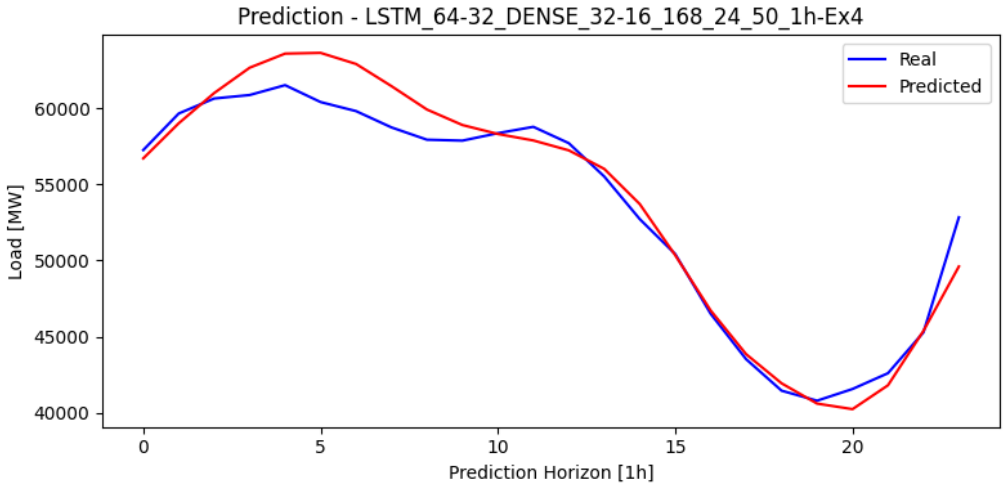

Figure 58: LSTM Example 4

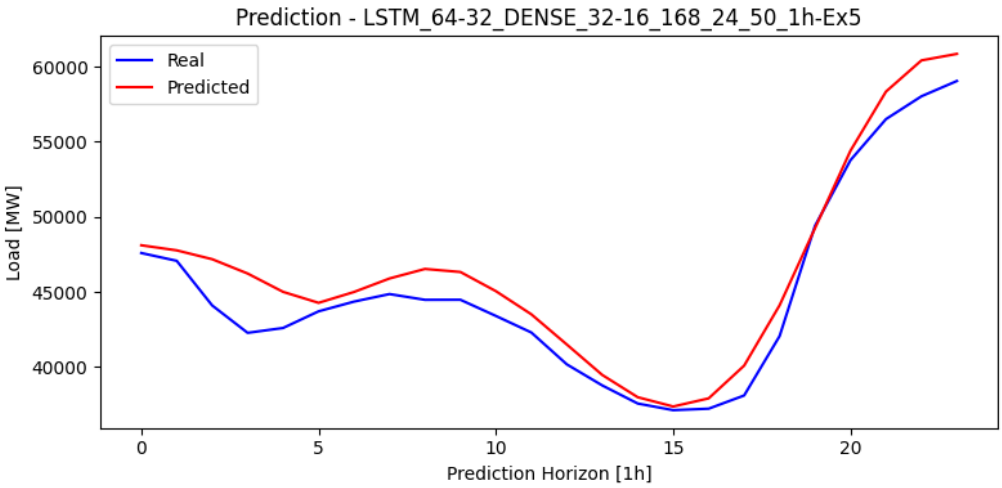

Figure 59: LSTM Example 5

5.8 CNN-LSTM Hybrid

5.8.1 MAPE Results

Unfortunately our predictions for CNN-LSTM turned out to be wrong, since it only scored **2.93** MAPE. It is a better score than our pure LSTM model, but still not enough to outperform a simple CNN architecture. Plot for MAPE over horizon is below.

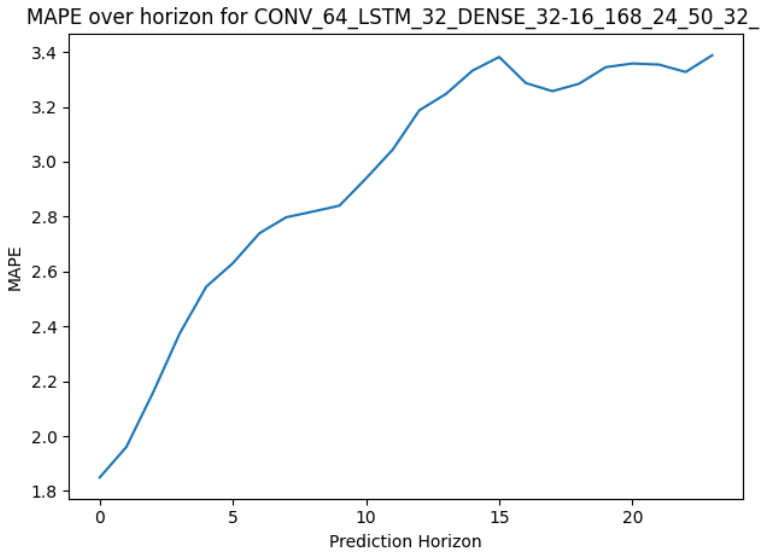

Figure 60: Average MAPE for time stamp in prediction horizon

5.8.2 Predictions

Like for other architectures, we randomly selected a few examples of prediction made by this model.

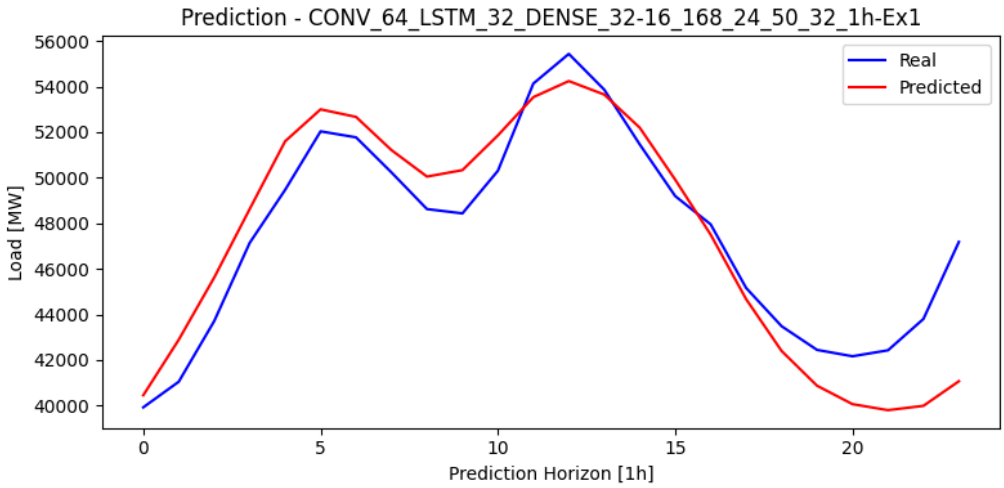

Figure 61: CNN-LSTM Example 1

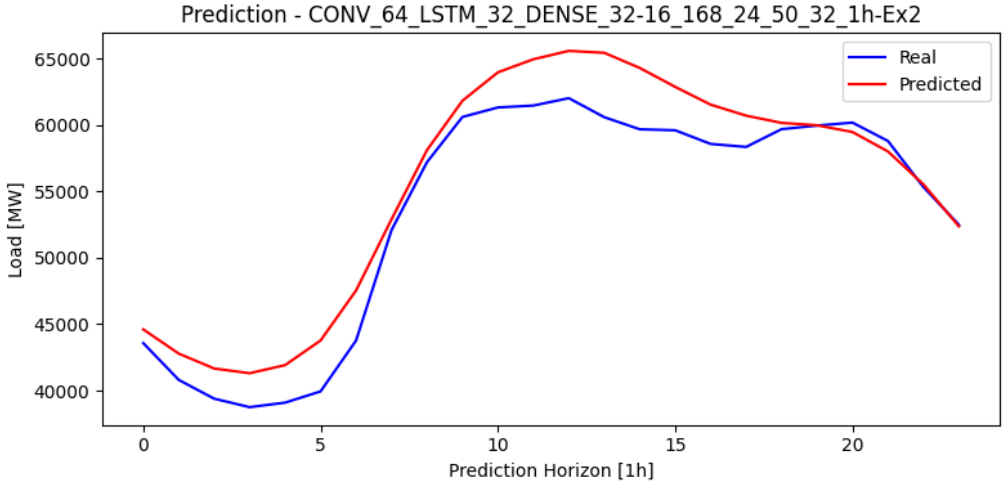

Figure 62: CNN-LSTM Example 2

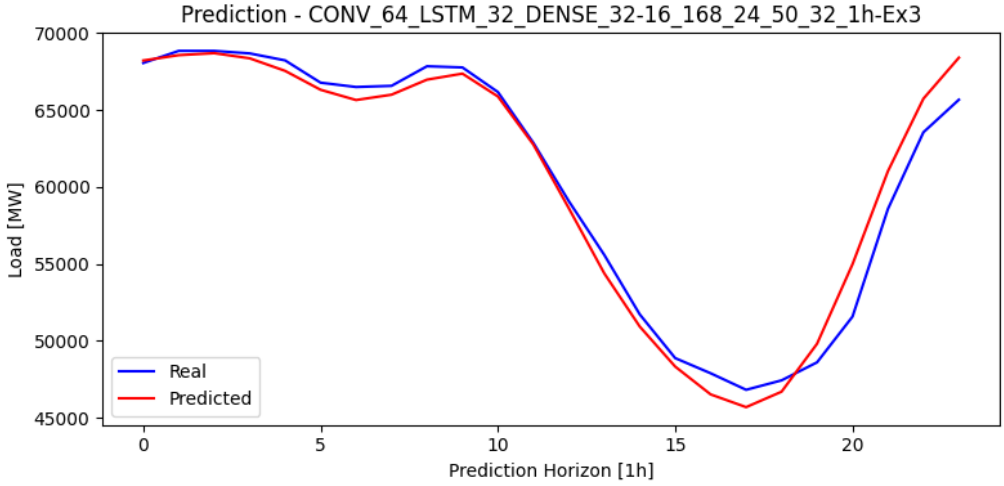

Figure 63: CNN-LSTM Example 3

5.9 Sequence-To-Sequence Encoder-Decoder Network

5.9.1 MAPE Results

For 24-hour forecasting, the Sequence-To-Sequence Encoder-Decoder Network's MAPE was **2.60%** for the best starting hour, with MAPE scores for each starting hour shown in the table below:

Starting hour	**MAPE**
0	2.98%
1	2.94%
2	2.87%
3	2.87%
4	2.79%
5	2.67%
6	2.60%
7	2.62%
8	2.64%
9	2.73%
10	2.70%
11	2.70%
12	2.74%
13	2.75%
14	2.81%
15	2.93%
16	3.01%
17	3.04%
18	3.04%
19	3.03%
20	3.04%
21	3.00%
22	3.02%
23	3.01%
Mean	**2.86%**

The Sequence-to-sequence model's results are slightly better than the standard LSTM-based RNN's results.

The mean MAPE scores for each forecasted hour in a 24-hour forecast are shown on the chart below:

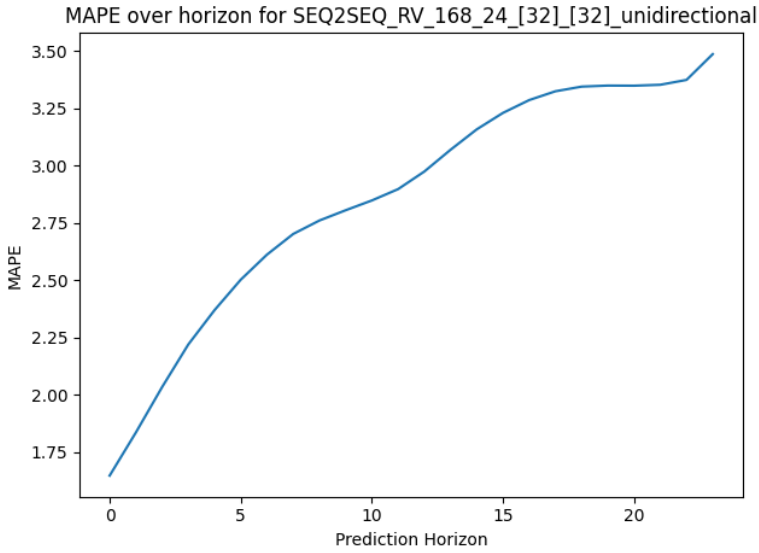

Figure 64: Average MAPE for time stamp in prediction horizon

5.9.2 Predictions

Plots showcasing the Sequence-To-Sequence Encoder-Decoder Network's predictions for a few selected 24-hour ranges are shown below:

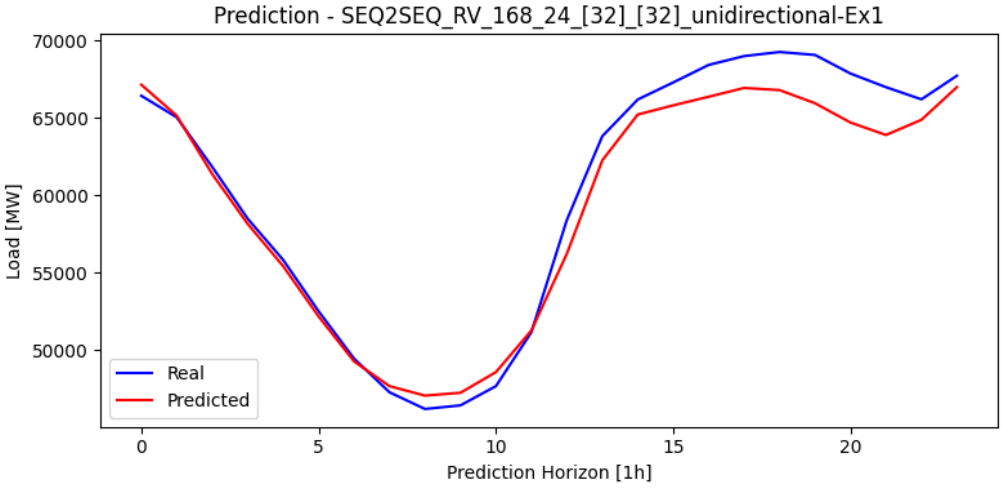

Figure 65: Sequence-to-sequence Example 1

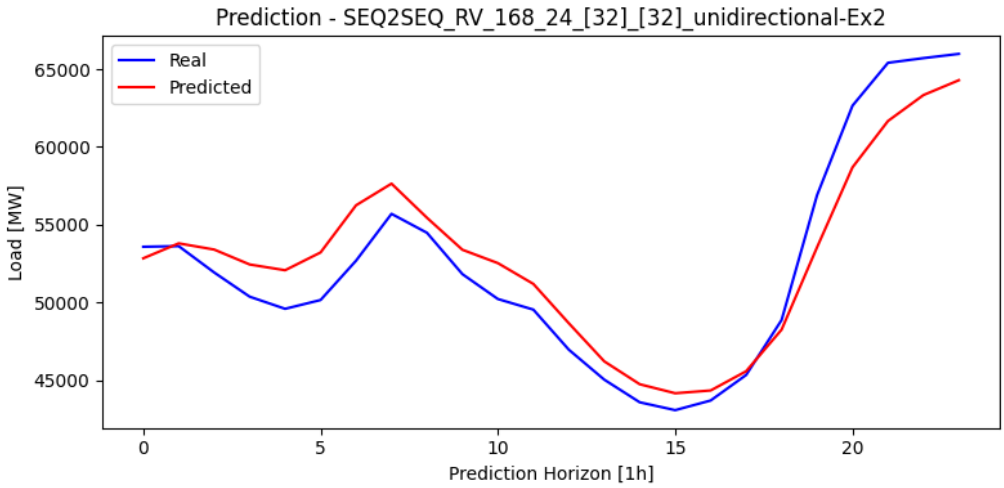

Figure 66: Sequence-to-sequence Example 2

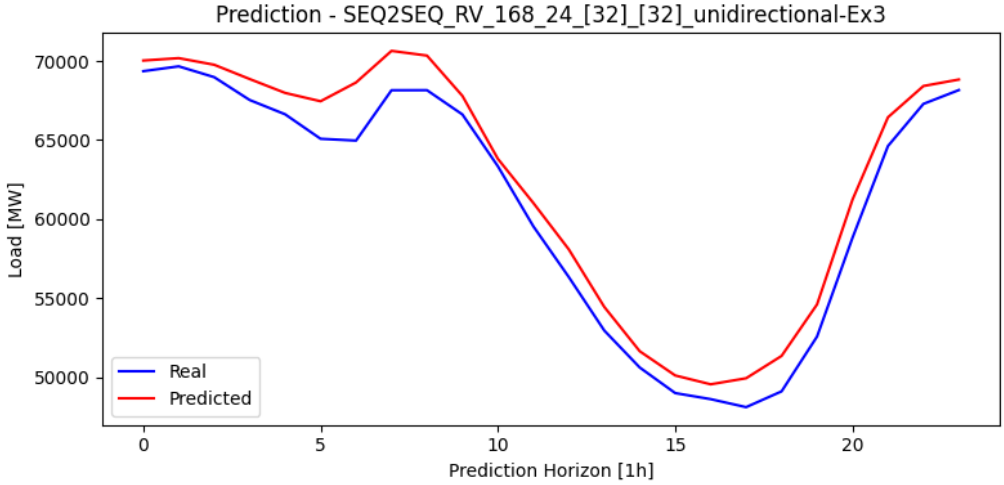

Figure 67: Sequence-to-sequence Example 3

6 Conclusions

To sum up this project we have gathered the average MAPE of all models and put it on a bar chart depicted below:

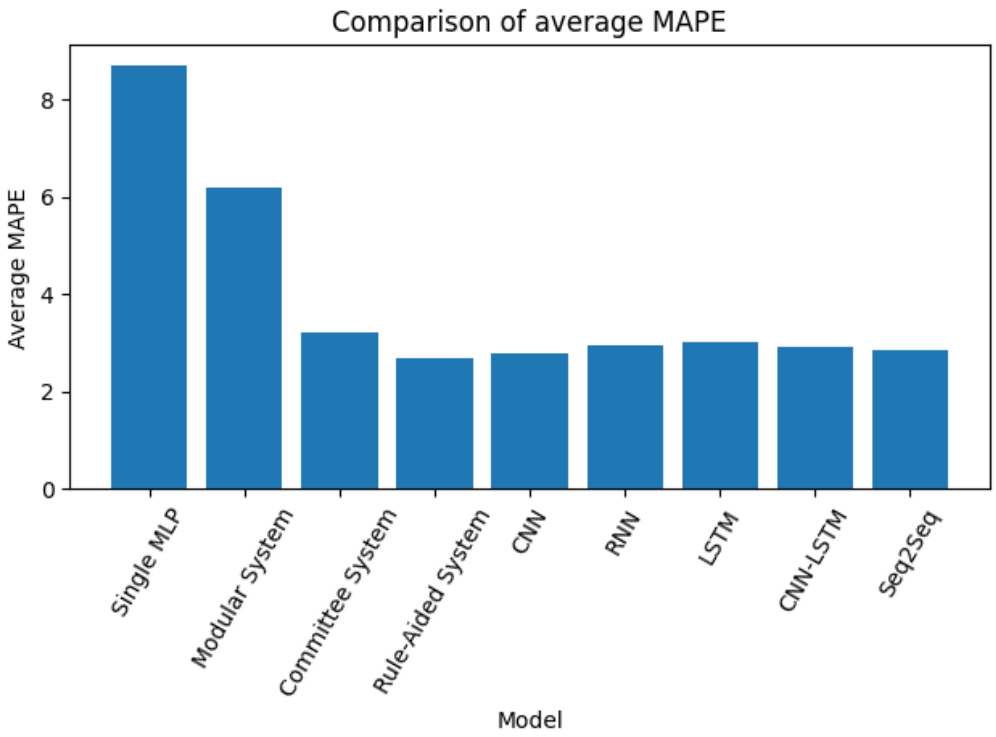

Figure 68: All models' comparison

The analysis of those results led to the following conclusions:

- All models achieved single-digit MAPE values, which is the same order of magnitude as those that were reached by systems described in the paper we were using as our base.
- Models achieve better results for the first few data points in the prediction horizon, which is to be expected due to them having more "fresh" data to work with.
- Embeded systems of MLPs based on those used in the original paper (Module System, Committee System, Rule-Aided System) did deliver significantly better results compared to a single Multi-Layered Perceptron.
- LSTM's ability to preserve historical data turned out to be of little importance, as it was outperformed by less complex models.
- More advanced systems like Seq2Seq or CNN-LSTM Hybrid did not yield as satisfying results as expected (they were better than LSTMs, but still outperformed by pure CNNs). Probable cause for lack of improvement is the simple and cyclic nature of our data, which makes even simple models able to yield accurate predictions.

7 Work input

1. **Andre Baron** - Modular System
2. **Rinchyen Bayarsaikhan** - Committee System, bar charts for data analysis
3. **Kacper Dłubała** - Rule-Aided System, Sequence-To-Sequence, data loading and cleaning, holiday analysis
4. **Filip Duda** - Recurrent Neural Network

5. **Michał Dworniczak** - Long-Short Term Memory, Convolutional Neural Network, CNN-LSTM Hybrid, data analysis
6. **Mateusz Działowski** - Multi-Layered Perceptron