Day - Ahead Load Probabilistic Forecasting

Hauser, Cyril Andrea
Howe, Alessandro John
Lacarpia, Luigi

June, 2024

Abstract

Load forecasting is becoming an increasingly important task to be faced. Indeed, accurate load forecasting ensures there is enough electric power supply to meet demand at any given time, thereby maintaining the balance and stability of the power grid. It can help utilities avoid the extra costs associated with producing too much or too little electricity. The goal of this study is to develop a day-ahead prediction over the time of one year. We first focus on the dataset explanation and preprocessing techniques, then we move to the model itself and finally we focus on some conformal prediction techniques. Eventually, we look at the results and draw our conclusions.

Contents

1 Dataset Description and Data Exploration **4**
1.1 The Dataset . . . 4
1.2 Exploratory Data Analysis . . . 4
1.2.1 Load Patterns Over Years . . . 5
1.2.2 Variation Across Hours Within the Same Month . . . 5
1.2.3 Profile for Specific Days . . . 6
1.2.4 Autocorrelation and Partial Autocorrelation . . . 8
1.2.5 Normality . . . 8
1.3 Conclusions . . . 9

2 Data Preprocessing **9**
2.1 Removal of Day in Leap Years . . . 9
2.2 Feature Selection . . . 9
2.3 Target Transformation . . . 11
2.4 PCA . . . 11
2.5 De-Trending and De-Seasonalisation . . . 12
2.5.1 One Hot Encoding . . . 12
2.5.2 Fourier analysis of seasonality . . . 15
2.6 About **CONST**__ variables . . . 17
2.6.1 Cosine Transformation Approach . . . 17
2.6.2 One Hot Encoding Approach . . . 17
2.7 Scaling Techniques . . . 18
2.8 Conclusions . . . 19

3 Neural Network Models **19**
3.1 Dense neural networks . . . 19
3.2 Recurrent Neural Networks . . . 19
3.3 Convolutional Neural Network . . . 20
3.4 Implemented Structures . . . 21
3.5 Conclusions . . . 23

4 Conformal Predictions **23**
4.1 Formal definition . . . 23
4.2 Conformal Prediction . . . 23
4.3 Conformalized Quantile Regression . . . 23
4.4 Adaptive Conformal Inference . . . 24
4.5 Adaptive strategies based on ACI . . . 25
4.5.1 Naive startegy . . . 25
4.5.2 Online Expert Aggregation on ACI, AgACI . . . 25
4.6 Practical remarks . . . 26
4.7 Conclusions . . . 28

5 Results and Discussion **28**
5.1 The holidays issue . . . 29
5.1.1 Rescaling factor . . . 30
5.1.2 Holiday as a weekend . . . 31
5.1.3 Holiday modification . . . 32
5.1.4 Further improvements . . . 34
5.2 The neural network's choice . . . 35
5.2.1 Forward Neural Network . . . 35
5.2.2 Recurrent Neural Network . . . 36

5.2.3 Mixed Neural Network 36
5.2.4 CNN-Mixed Neural Network 37
5.2.5 **CONST_** Variables modification and De-seasonalization Choice 38
5.2.6 Target Transformation and data scaler 38
5.2.7 Holiday processing 39
5.3 Conformal predictions 40
5.4 Our final model 41

6 References **43**

7 Appendix **44**

1 Dataset Description and Data Exploration

1.1 The Dataset

The dataset consists of 35,040 rows and 29 columns and contains data collected from January 2, 2014, to December 31, 2017. It is structured into four main distinct categories of variables, each identifiable through a unique prefix.

- **IDX__**: Indexing Variables - These are variables used for indexing purposes.
- **CONST__**: Time Structure Variables - These variables consider time structures such as months, days, etc.
- **FUTU__**: Weather Forecasting Variables - These variables refer to future weather forecasting quantities.
- **TARG__**: Target Variable - This is the target variable in the analysis.

Table 1 describes the meaning of the main features (and target).

Variable	**Description**
IDX__global	Row index
IDX__step	Index referring to the number of days from the first observation
IDX__sub_step	Index referring to the hour of observation
TARG__NetLoad	Target variable
CONST__DoW	Index referring to day of the week
CONST__DoY	Index referring to day of the year
Hour	Index referring to the hour of the day
CONST__Holiday	Boolean variable referring to a vacation/business day (0 for business)
Date	The current date
FUTU__Variables	Weather forecasting variables including temperature, precipitation, cloud coverage, solar radiation, and wind speed mean and standard deviation

Table 1: Variables description

1.2 Exploratory Data Analysis

In this section, we examine the main patterns in the data.

1.2.1 Load Patterns Over Years

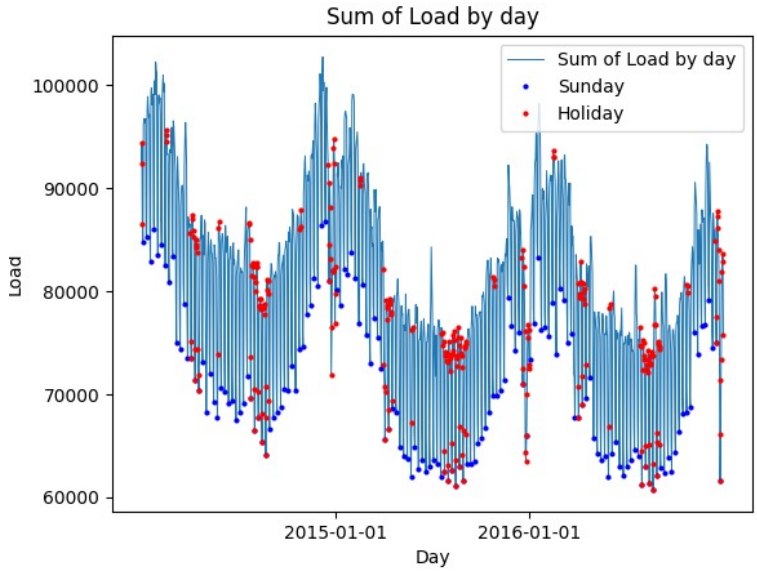
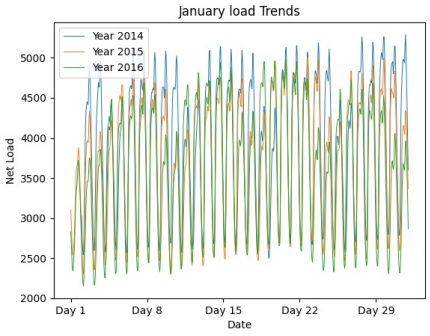


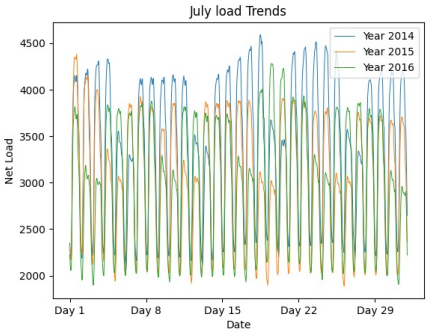
Figure 1: Sum of load throughout the period

In Figure 1 we can see the cumulative daily load over time. It reveals a clear seasonal pattern with high values during the winter and low values during the summer. The reason behind this is the increase in use of heating systems during colder months. The red and blue dots refer to particular days of interest. The first ones reveal that holidays don't exhibit a clear pattern, whereas from the second ones it seems that Sundays consistently show a lower load. We can also notice that the variability does not remain constant throughout the analyzed period.

1.2.2 Variation Across Hours Within the Same Month



(a) Hourly Load during January 2014



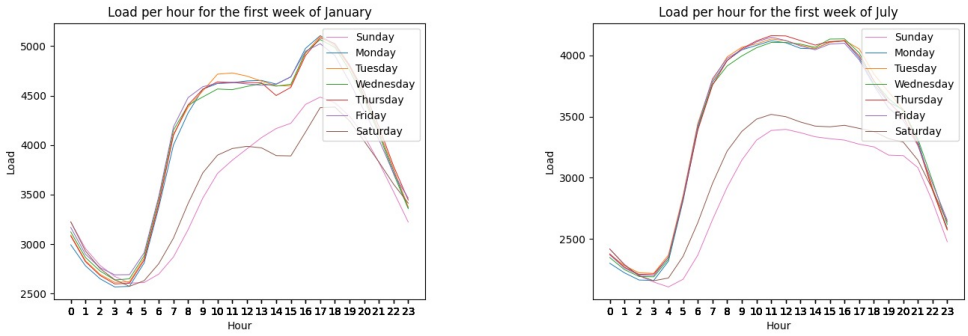
(b) Hourly Load during July 2014

The figures above show the target variable across hours within the same month across different years. We can observe that the patterns are similar, however not identical - slight variations exist.

A daily behavior emerges, with two peaks - the first in the late morning and the second in the late afternoon. A decline during the night is evident. Additionally, as we observed in the previous plot, the load significantly decreases during the weekends.

Section 2.5.2 is dedicated to a more thorough analysis of the subject matter.

1.2.3 Profile for Specific Days



(a) Daily Load profile for the first week of January 2014

(b) Daily Load profile for the first week of July 2014

Figures 3a and 3b provide a closer examination of the target variable for a specific day during the initial week of both a hot month - July (2014) - and a cold month - January (2014). First of all the plot highlights the difference of the load than during weekends and weekdays. In particular if the latter curves are nearly indistinguishable, we can see some difference between Saturdays and Sundays. Secondly, we point out that the curve shapes are slightly different between these months. This is due to an interaction of daily and yearly periodicity.

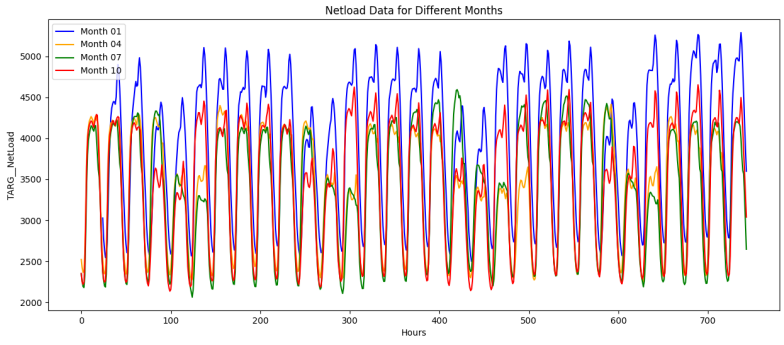


Figure 4: Day-ahead loads through different months of the year 2015

Figure 4 compares the shapes of market loads during months from different seasons evenly distributed throughout 2015: January; April; July; October. The figure displays a strong variability even if there exists a typical repetitive pattern. Nevertheless, magnitude and extensions are different across the seasons: January manifests the more pronounced peaks as well as the highest prices on average.

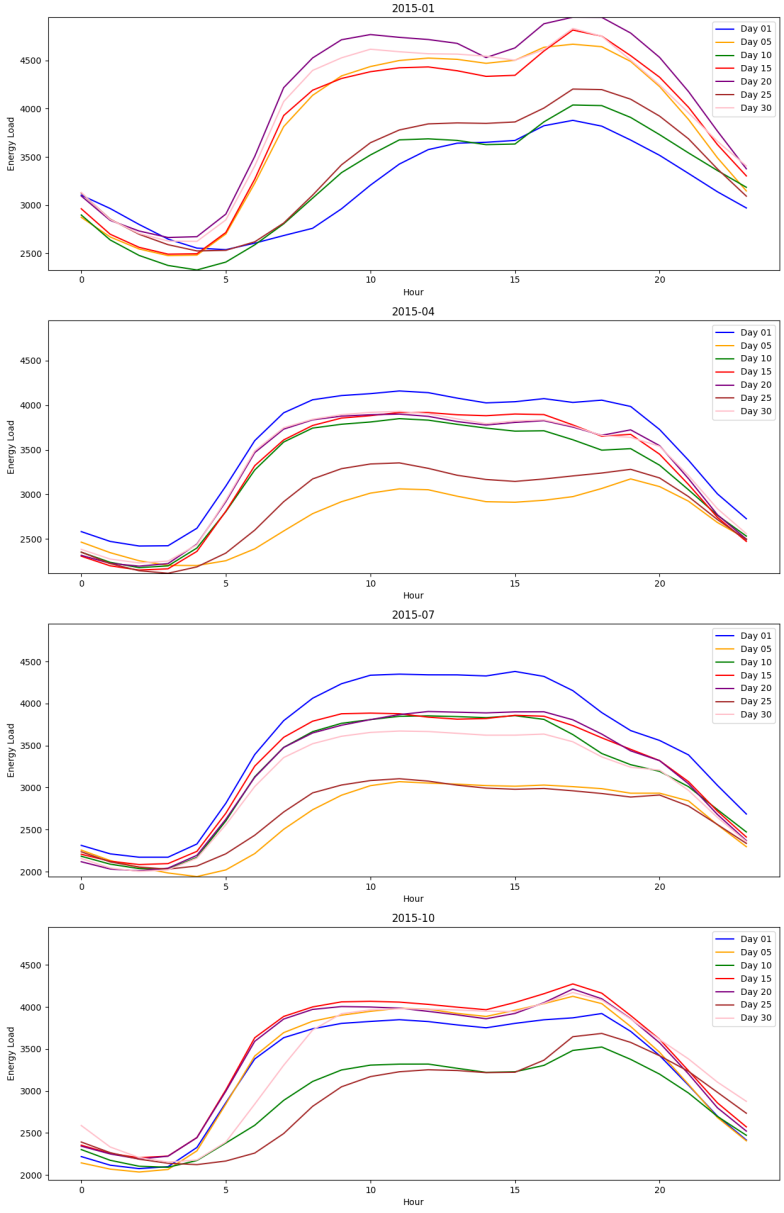


Figure 5: Day-ahead hourly loads through different days of a month of the year 2015: January, April, July, October

Daily prices patterns are even more evident within Figure 5 that reports evenly spaced days within

the same month. Moreover, the specific shape of nonworking days is shown. See for example January 1st with reference to January 30th.

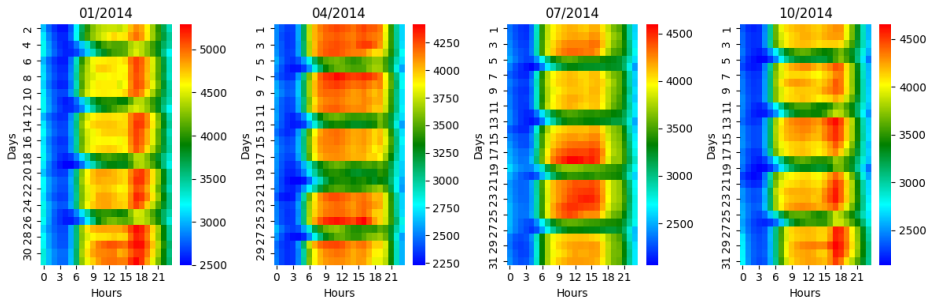
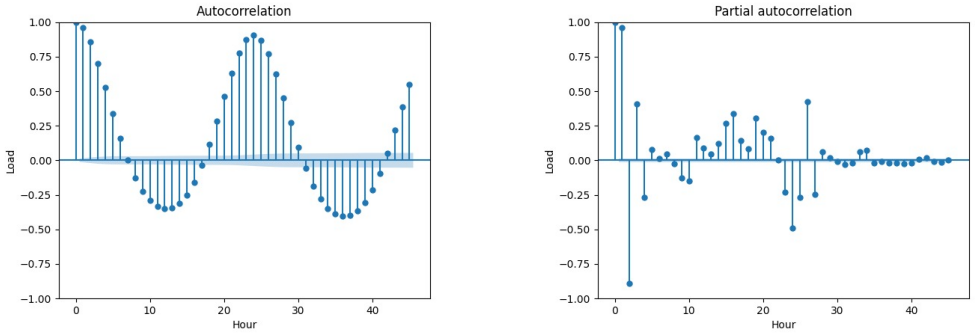


Figure 6: Heat map of day-ahead loads through different months of the year 2017: January, April, July, October

Figure above reports the matrix of hourly prices on the month days. Each figure displays for each month cooler horizontal bands that represent non-working days and higher vertical bands representing peak loads.

1.2.4 Autocorrelation and Partial Autocorrelation



(a) Target autocorrelation

(b) Target partial autocorrelation

Due to these patterns, we expect significant correlation among the observations in the time series. These figures confirm this expectation. Indeed the time series shows strong autocorrelations - particularly evident at lags of 2-3 hours and at a lag of 24 hours.

1.2.5 Normality

At this point, we perform a Shapiro - Wilk test on the target variable (and on some basic transformations used in the literature - see section 2.3 for further information) to assess whether it is possible to assume a gaussian distribution of the variable, which may be useful for further development. We expect that this hypothesis is not met due to the high trend and seasonality of the time series. Table 2 shows that in all cases we have statistical significance to reject the null hypothesis (i.e. gaussian distribution assumption).

Transformation	Shapiro Test p-value
NO transformation	3.007e-49
log transformation	2.6787e-56
asinh transformation	2.9678e-63

Table 2: Shapiro-Wilk's test results

1.3 Conclusions

In this section, we performed exploratory data analysis, which allows us to focus on data patterns and decide how to build our model based on this information. We recognize the need to account for pronounced daily, weekly, and yearly seasonality. Even if we didn't investigate it deeper, another challenge consists of determining the optimal approach in dealing with all the **FUTU**__ variables. In the following section, we discuss the first approaches to address these issues.

2 Data Preprocessing

In this segment we go through the main Data Preprocessing techniques we employed to deal with the criticalities explained in the previous section. We implemented many and eventually we decide not to use several of them - however they are a stepping stone of our work.

2.1 Removal of Day in Leap Years

To maintain the yearly periodicity, we opt to exclude February 29th in every leap year. This is a common approach proposed by literature - see [6] for example. In such way we have the same number of days for each year and it will be helpful to preserve the pattern.

2.2 Feature Selection

Due to the high number of **FUTU**__ variables, it is likely that some may carry along less information than others, making the direct use of these raw variables in model training suboptimal. We are convinced that it is necessary to perform a feature selection at this early stage.

Initially, we decide to aggregate (mean) the standard deviation the different features into a unique variable. This represents the total volatility of the weather conditions for that observation. Even if at first we consider assigning different weights to the different variables, we conclude that it is an unnecessary complication.

Furthermore, we notice that a lot of variables seem to contain similar information. In order to have some quantitative guideline, we compute the correlation among these variables. Correlation is a linear relationship between two variables - the higher its absolute value, the higher the similarity (at least linearly). This analysis is reported in Figure 8. We take notice of predominantly positive relationships among most features. However there is also a smaller subset exhibiting negative correlations.

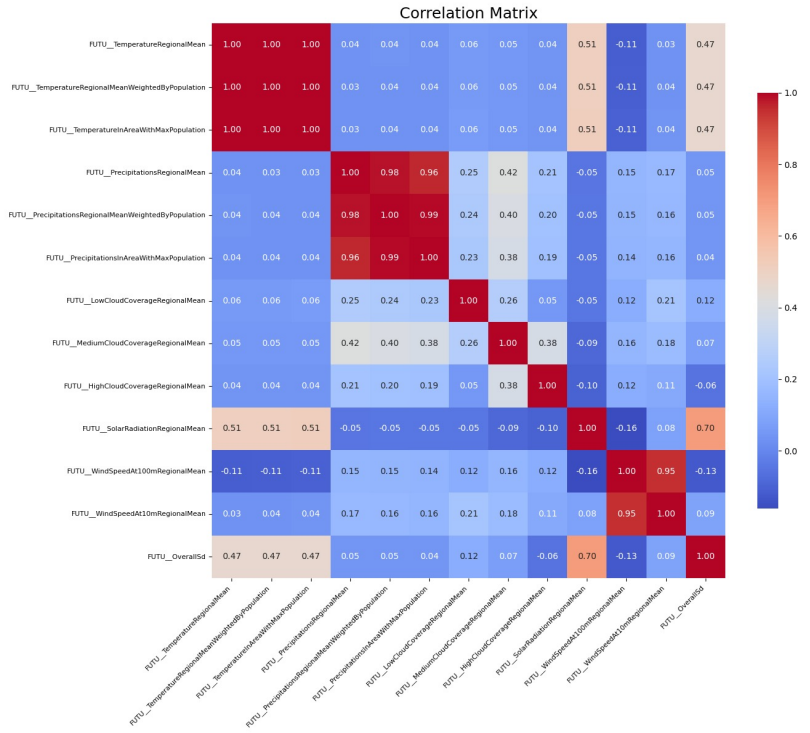


Figure 8: Correlation between **FUTU__** variables

Table 3 shows the correlation of the **FUTU__** and the other variables with the target.

Feature	**Corr. Coeff.**
IDX__step	-0.159299
IDX__sub_step	0.462194
TARG__NetLoad	1.000000
CONST__DoW	0.032518
CONST__DoY	-0.116220
Hour	0.462194
CONST__Holiday	-0.123301
FUTU__TemperatureRegionalMean	-0.040242
FUTU__TemperatureRegionalMeanWeightedByPopulation	-0.037312
FUTU__TemperatureInAreaWithMaxPopulation	-0.033740
FUTU__PrecipitationsRegionalMean	0.039376
FUTU__PrecipitationsRegionalMeanWeightedByPopulation	0.037292
FUTU__PrecipitationsInAreaWithMaxPopulation	0.034315
FUTU__LowCloudCoverageRegionalMean	0.109318
FUTU__MediumCloudCoverageRegionalMean	0.058727
FUTU__HighCloudCoverageRegionalMean	0.012439
FUTU__SolarRadiationRegionalMean	0.334721
FUTU__WindSpeedAt100mRegionalMean	0.083827
FUTU__WindSpeedAt10mRegionalMean	0.194159
FUTU__OverallSd	0.310217
Date	-0.159299

Table 3: Correlation Coefficients Between Features and Target Variable

As [6] suggests, variables contributing significantly to predictive accuracy will show a strong correlation with the target. Consequently, we decide to keep these variables at this phase, particularly noting the significance of `SolarRadiationRegional` information, followed by `OverallSd` (the aggregated standard deviation mentioned above), `WindSpeedAt10mRegionalMean` and `LowCloudCoverageRegionalMean`.

At this point we proceed with variable selection. We initially decide to keep only one feature per pair where the correlation exceeds 0.9 - and in case one of the previously mentioned variables appear we preserve this among the two.

Thanks to this approach we reduce significantly the number of **FUTU**__ variables from 20 to 8. For now, we choose not to implement further dimensionality reduction and we incorporate all selected variables moving forward.

2.3 Target Transformation

In the literature it is standard to model the natural logarithm of the consumption instead of taking the consumption itself. This is a way to manage different load volatilities across different seasons - something we pointed out in Subsection 1.2. We decide that this is possibility which is worth trying.

Another widely employed transformation in the literature [7] is the arcsine hyperbolic sine (asinh) transformation. This method was developed to address situations involving negative prices, such as France on April 6, 2024, where the market experienced a negative price of €0.71/MWh, or in the North Pool market of the Nordic countries on April 7, 2024, marking the first occurrence of negative hourly prices since October 2023. Even if this is not our case, we look into the potential advantages of this transformation, standardizing the values before.

2.4 PCA

At this point, we explore the potential application of Principal Component Analysis (PCA) to the **FUTU**__ variables. The aim of this approach is to find a new reference system where all variables are uncorrelated [4].

We hypothesize that this approach could improve model performance by eliminating redundant information. To achieve this, we first rescale the variables and then apply PCA, replacing the original **FUTU**__ variables with the principal component **PC**__ variables for the model input.

Table 4 shows that the first principal components explain the main part of the variability of the data while the last ones very little.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Explained Variance Ratio	0.26741539	0.24311462	0.12820622	0.10601283	0.08835326	0.06987879	0.06227291	0.03474598
Cumulative Explained Variance	0.26741539	0.51053000	0.63873622	0.74474906	0.83310232	0.90298111	0.96525402	1.00000000

Table 4: Explained Variance Ratios and Cumulative Explained Variances

In our initial attempts we consider to extend PCA beyond the **FUTU**__ variables to all variables. However we abandon this idea due to the risk of discarding valuable information regarding the hour, day of the week, and time-related variables crucial for our model.

Additionally, we consider revisiting variable selection. Although PCA generates uncorrelated variables ordered by their contribution to the original variability, this does not imply that the first principal component (highest variability) is the most significant predictor, neither that components are progressively less impactful. Therefore, by selecting only the first k principal components could be a mistake: we could risk to drop potential significant information of the prediction process. Figure 9 confirms our suspicion: the last components are not necessarily less significant. To address this, we apply LASSO regression to the new **PC**__ features and we exclude the variables corresponding to beta coefficients under a certain threshold in absolute value. Indeed LASSO already detects the most significant variables by weighing the corresponding β.

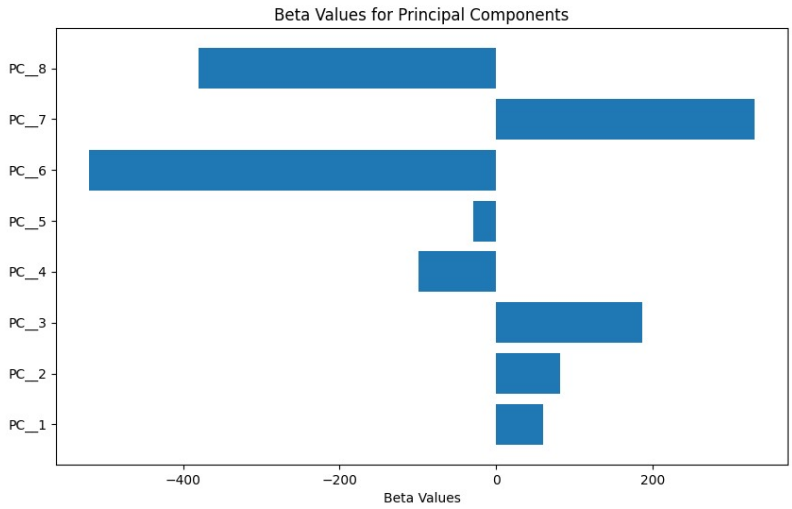


Figure 9: Beta values of LASSO regression on PC variables

A limitation of this approach is that LASSO assumes linear relationships between predictors and the target, and it fails to capture nonlinear associations - which could be detected by more sophisticated models as neural networks. By reducing dimensionality in this manner, we risk losing these complex relationships. Based on these considerations, we conclude that variable selection at this stage is not advisable.

2.5 De-Trending and De-Seasonalisation

As previously discussed, the time series exhibits robust daily, weekly, and yearly periodicities. To address this, we adopt two strategies: One Hot Encoding and Fourier analysis of seasonality. Both methods aim to train a linear model on transformed regressors and after train and test a Neural Network (NN) model on its residuals.

2.5.1 One Hot Encoding

In this approach we employ a one-hot encoding scheme, treating the categorical variables Hour, Day of the Week, and Month as dummy variables. In this approach we generate 24 new variables for the Hour variable, 7 for the Day of the Week, and 12 for Months. Each realization of these variables is represented as a vector of 0s and 1s, with 1 indicating the observation corresponds to that specific hour/day of the week/month, and 0 otherwise. Furthermore, we introduce a trend variable to account for the passage of time, incrementing an index for each hour. This introduces 24 + 7 + 12 = 43 new variables, replacing the original three. Given our observations during data exploration, we recognize a clear interaction among the three periodicity levels. To address this, we also incorporate interaction variables: an interaction between the Hour and Day of the Week, creating 24 * 7 = 168 new variables, and an interaction between the Day of the Week and Month, adding 7 * 12 = 84 new variables. This results in constructing a set of 295 regressors for the target variable:

$$Y(t) = \beta_0 + \beta_1 t + \sum_{i=1}^{12} \beta_2^i M_i(t) + \sum_{i=1}^{7} \beta_3^i W_i(t) + \sum_{i=0}^{23} \beta_4^i H_i(t) +$$

$$\sum_{i=1}^{7}\sum_{j=1}^{24}\beta_5^{ij}W_i(t)H_j(t) + \sum_{i=1}^{12}\sum_{j=1}^{7}\beta_6^{ij}M_i(t)W_j(t)$$

where $M_i(t)$, $W_i(t)$, $H_i(t)$ are the i^{th} month, day of the week and hour dummy variable of the observation.

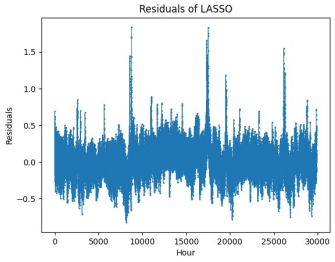
Introducing such a high number of new variables suggests that traditional Ordinary Least Squares (OLS) linear models may not perform optimally. To address this, we propose three alternative approaches:

- **Lasso Regression**: We perform LASSO regression on the new variables and cross-validate the λ parameter
- **Elastic Net**: We conduct Elastic Net regression, cross-validating the λ_1 and λ_2 parameter
- **Classic Linear Regression (OLS)**: We execute a standard linear regression.

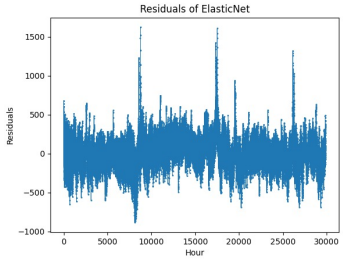
Table 5 and the following Figures show the results by performing the regressions directly on the target variable.

Model	R-squared	Shapiro Test p-value
OLS	0.924	3.2617e-57
Elastic Net	0.923	1.002e-55
LASSO	0.923	9.6371e-57

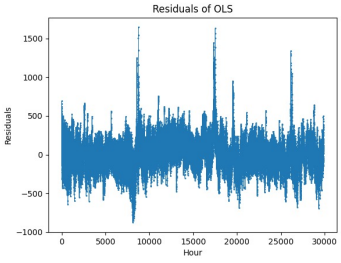
Table 5: Model Results



(a) Residuals LASSO

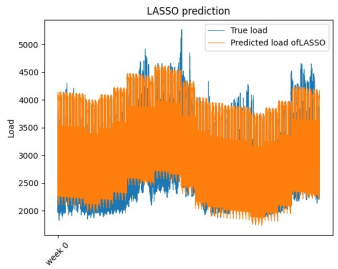


(b) Residuals Elastic Net

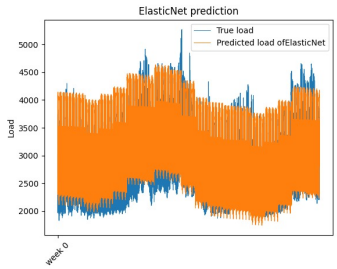


(c) Residuals OLS

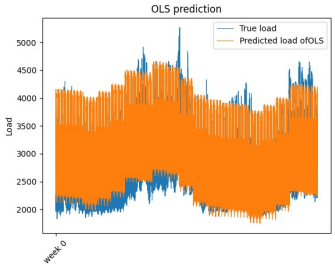
Figure 10: Residuals of different models



(a) Prediction LASSO

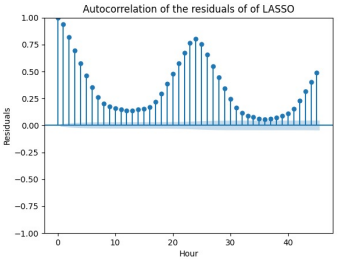


(b) Prediction Elastic Net

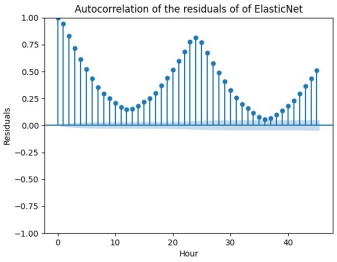


(c) Prediction OLS

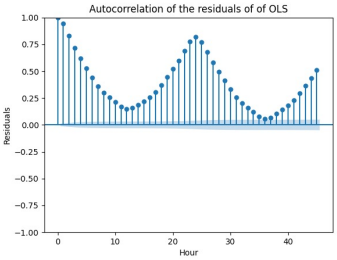
Figure 11: Predictions of different models



(a) Autocorrelation LASSO

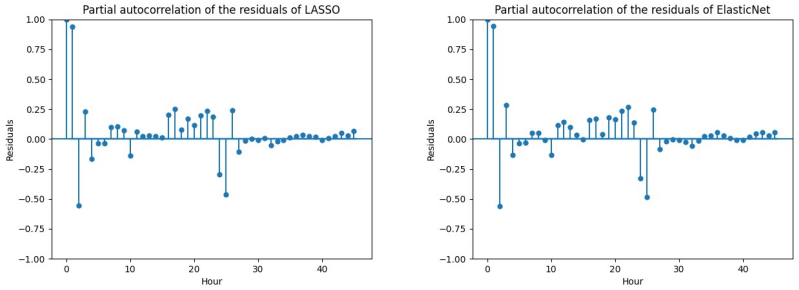


(b) Autocorrelation Elastic Net

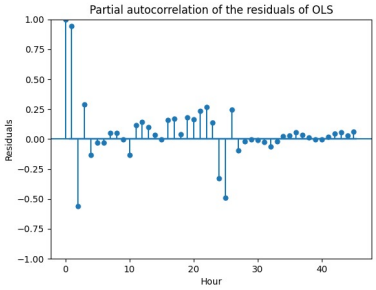


(c) Autocorrelation OLS

Figure 12: Autocorrelation between residuals of different models



(a) Partial Autocorrelation LASSO (b) Partial Autocorrelation Elastic Net



(c) Partial Autocorrelation OLS

Figure 13: Partial autocorrelation between residuals of different models

Initially, we observe that all three methodologies give similar R^2 values. In particular, Ordinary Least Squares (OLS) shows the highest value. This is expected, since it performs a regression on all the variables with no particular constraint. It is crucial to note, however, that a higher R^2 value does not automatically indicate a superior fit; the model's complexity can lead to overfitting. Given these considerations, we opt to advance with the Elastic Net approach because of its adaptability and its higher generality.

Additionally, we can identify from the plot that each model generates heteroscedastic residuals. The Shapiro - Wilk test's extremely low p-value - we have high statistical significance to reject the null hypothesis - reveals these residuals are not normally distributed. These results also hold true in the case of standard transformations applied to the target variable. This finding precludes assuming a specific underlying distribution for the neural network's new target. This is something we need to take into account for our further development.

We can also notice that the residuals show strong autocorrelation - even though not as evident as the original target variable. This indicates that addressing periodicity remains a critical issue.

2.5.2 Fourier analysis of seasonality

Not completely satisfied by the previous, we find this approach in the literature.

The concept originates from signal processing, where a signal is transformed from the time domain into the frequency domain, and the most crucial frequencies are utilized to capture various levels of seasonality. To achieve this, first of all we introduce a trend variable as we did in the previous case.

Secondly, we compute the Fast Fourier Transform (FFT) of the target variable to decompose the time series in sine or cosine waves and identify the most significant frequencies and their corresponding amplitudes - which represent the primary seasonal cycles present in the data.

$$s(t) = c_0 + \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right) \tag{1}$$

Fourier series with trend term

- t is time
- c_0 is the trend term
- P is the base period of a seasonal feature — the period of the sine/cosine pair with the largest period
- n, the index in the series, is a period demultiplier (frequency multiplier)
- $a(n)$ and $b(n)$ are parameters defined in training phase

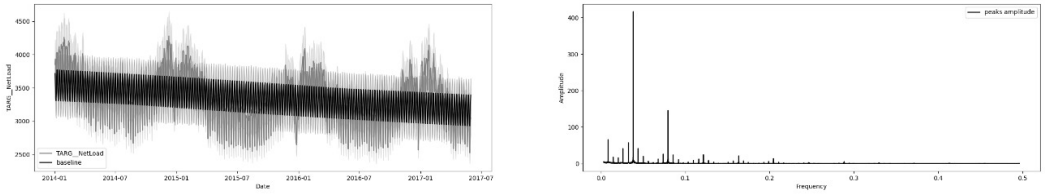


Figure 14: Baseline without annual component

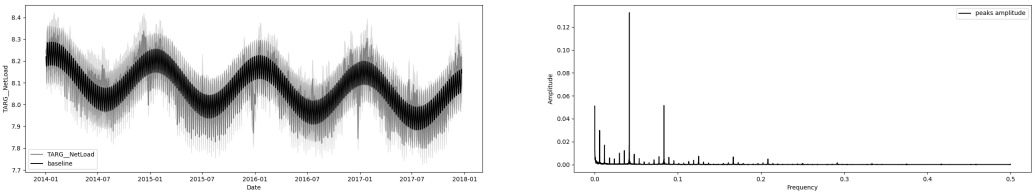
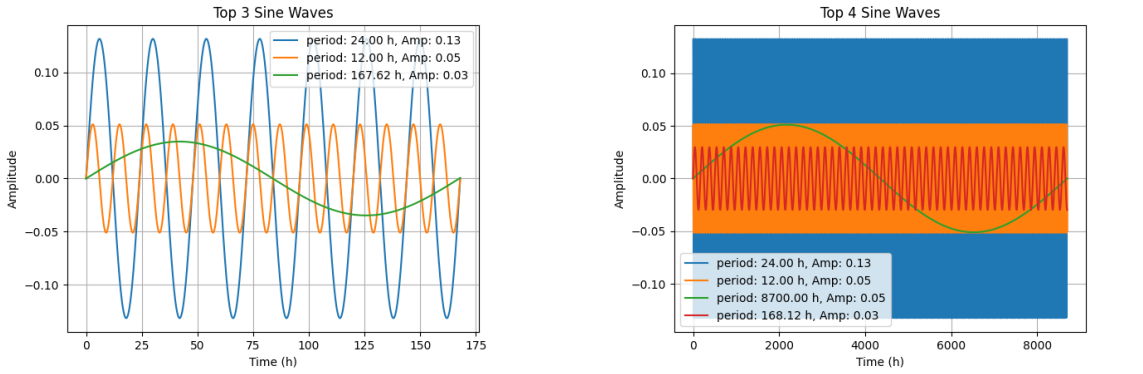


Figure 15: Baseline with annual component



(a) 300 hours max period

(b) 10000 hours max period

Figure 16: Fourier Baseline Decomposition

We observe how by considering only the 3 highest frequencies and restricting the search for seasonality to a maximum period of 300 hours a clear 12 hour sub-daily season emerges along with a daily and weekly periodicity (Figure 16a). By enlarging our search space to a period of maximum 10^4 hours also an additional yearly term surfaces (Figure 16b). Yearly and daily seasons confirm fluctuations of load request between winter-summer and day-night while the twelve hour season characterizes typical morning and evening peaks in residential areas.

In this scenario, trend and seasonality components just derived become the new regressors for our linear model.

Given the reduced number of regressors compared to the total number of samples, employing sophisticated regression models is not necessary. As a consequence we fit a standard linear model and let the residuals between true load and baseline of such model en-capture all the more complex relationships within the data. Further processing is clearly required and will be applied in the forthcoming sections.

2.6 About CONST__ variables

At this point, we are nearly ready to introduce the Neural Network. Before this, we need to address how to handle the **CONST__** variables present in our dataset. As initially observed, these variables are formatted in a manner that does not convey the periodicity we wish our model to recognize.

We propose two approaches.

2.6.1 Cosine Transformation Approach

Our first strategy involves transforming the variables using a periodic function, specifically the cosine wave function. This transformation produces four periodic variables, each representing a different aspect of time:

$$\textbf{CONST\_\_Hour}(i) = \cos\left(\frac{2\pi H(i)}{24}\right) \tag{2}$$

$$\textbf{CONST\_\_DoW}(i) = \cos\left(\frac{2\pi D_o W(i)}{7}\right) \tag{3}$$

$$\textbf{CONST\_\_DoY}(i) = \cos\left(\frac{2\pi D_o Y(i)}{365}\right) \tag{4}$$

$$\textbf{CONST\_\_Month}(i) = \cos\left(\frac{2\pi M(i)}{12}\right) \tag{5}$$

where H(i), DoW(i), DoY(i), M(i) denote respectively the hour, the day of the week, the day of the year and the month of the ı-th observation. We keep **CONST__Holiday** in the original format. By applying this transformation, we aim to make the model perceive, for instance, 0 p.m close to 11 p.m., or December close to January by reducing the distance between these points in the transformed space.

2.6.2 One Hot Encoding Approach

Alternatively, we consider reapplying a re-elaboration of the One Hot Encoding approach described before. We apply One Hot Encoding to the Hour and Month variables. However, we opt to treat the day of the week in a different manner. We notice from the data exploration that the load is very similar during weekdays, while it changes significantly during the weekends. For this reason, we decide to create a new dummy variable, **DUMMY__Weekend** computed as the sum of dummies relative to Sunday and Saturday. Dummies relative to weekdays are then discarded. The intuition behind this idea is to allow the Neural Network to interpret the value of 1 as a significant "shock" amidst the zeros.

However, a potential drawback of this approach is the increase in the number of variables introduced into the model and subsequent risk of the known "Dimensionality Curse".

2.7 Scaling Techniques

Eventually before feeding the neural network we need to decide which technique to employ to scale the variables. Literature frequently proposes several common approaches:

- **Standard Scaler**: The Standard Scaler removes the mean and scales the data to unit variance.

$$x_i' = \frac{x_i - \mu}{\sigma}$$

  where μ is the mean of the feature and σ is the standard deviation.

- **MinMaxScaler**: The MinMaxScaler transforms features by scaling each feature to the range between 0 and 1 without modifying relations between data. The transformation is given by:

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

  where $\min(x)$ and $\max(x)$ are the minimum and maximum values of the feature.

- **RobustScaler**: The RobustScaler scales features using statistics that are robust to outliers. This scaler removes the median and scales the data according to the interquartile range (IQR). The transformation is given by:

$$x_i' = \frac{x_i - \text{median}(x)}{\text{IQR}(x)}$$

  where $\text{IQR}(x)$ is the interquartile range of the feature.

- **QuantileTransformer**: The QuantileTransformer transforms the features to follow a normal distribution. It uses quantiles to map the data to a different distribution. For a feature x_i, it is transformed by:

$$x_i' = F^{-1}(G(x_i))$$

  where G is the cumulative distribution function (CDF) of the data and F^{-1} is the inverse CDF of the normal distribution.

In the upcoming sections we will discuss the most appropriate scaling methods for our needs. At this stage, we implement the mentioned options.

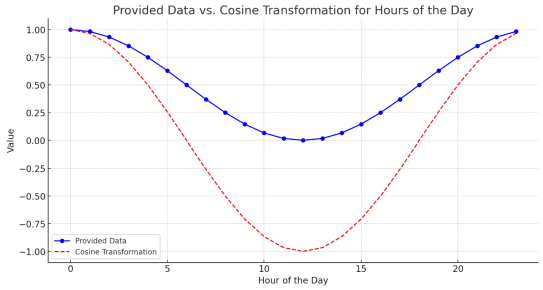


Figure 17: Cosine transformation of Hour variable vs Scaled data provided to the model

2.8 Conclusions

In this section we proposed different methods for the Data Preprocessing problem. It is evident that a wide range of possibilities is available to us. As previously discussed, certain choices have been identified as more suitable for our objectives, whereas others have been excluded from the beginning. However, the options that remain are model-specific, necessitating a decision related to the particular scenario. Moreover challenges related to periodicity persist and require attention. In the following section we move into the Neural Network structures and try to find the best strategy to address these issues.

3 Neural Network Models

Artificial neural networks have lately become increasingly used in forecast problems because of their ability to learn complex nonlinear relationships that are difficult to model with other statistical techniques.

The structure can be described as multiple sets of processors called "neurons" communicating with each other, weighing data, and processing it with the help of (generally) nonlinear functions. Neural Networks train by adjusting their weights in order to minimize a certain loss function that describes how much the output differs from the actual solution to the problem being modeled.

Criticalities can arise during this phase since the high learning capabilities of these models can lead them to memorize data contained in training sets, leading to overfitting.

Hence, during this study, we start from less complex architectures and then expand them with more advanced components, all the while paying attention not to oversize the model.

We make use of the flexibility and simplicity of the Keras and TensorFlow frameworks to efficiently implement such model architectures.

3.1 Dense neural networks

The simplest type of network we implement is a Fully Connected Feed-Forward Neural Network (DNN). In this type of network, data flows from the input layer to the output without looping back. Each node in a layer is connected to every other node in the next layer but doesn't interact with nodes from the same layer. The output is compared to the true value during the training phase, and the error gets minimized by changing the weights in such a way that the loss function's gradient dependent on such weights remains negative.

Such structures tend to have difficulties handling time series due to the feed-forward nature of the network not allowing them to maintain information from previous inputs over long sequences. Furthermore, if implemented with a high number of hidden layers, they add a lot of parameters (weights) to the structure, drastically increasing computation time.

3.2 Recurrent Neural Networks

Recurrent neural networks generally outperform DNNs in handling sequential data due to their ability to hold memory.

The particular type of neuron implemented in our studies is of the Long Short-Term Memory type (LSTM), which we find to be more suitable for our purposes. These complex types of recurrent neural networks allow for the capturing of long and short-term dependencies through four key components:

- **Forget Gate**: After getting the output of the previous state, h_{t-1}, the Forget gate helps to take decisions about what must be removed from the h_{t-1} state, thus keeping only relevant information. It is surrounded by a sigmoid function which helps to squash the input between [0,1].

- **Input Gate**: In the input gate, new data from the present input is added to the present cell state, scaled by how much data is added. The sigmoid layer decides which values to be updated and the tanh layer creates a vector for new candidates to be added to the present cell state.
- **Output Gate**: The input gets multiplied with tanh to squash the values between (-1,1) and then multiply them with the output of the sigmoid function so that the output gets filtered and is able to get propagated onto the cell further into the same layer and onto the next layer if present.

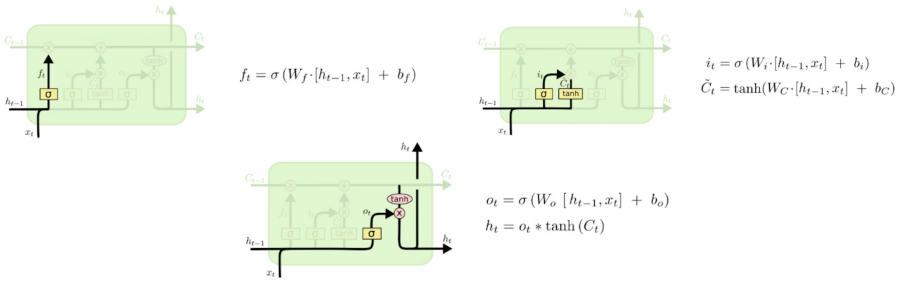


Figure 18: Forget Gate (left), Input Gate (right), Output Gate (center)

In our models each LSTM cell handles time series data for multiple features at a specific time step and propagates its hidden state h_t to the next cell on the same layer that uses it in combination with feature data on the next time step to compute the following hidden state h_{t+1}.

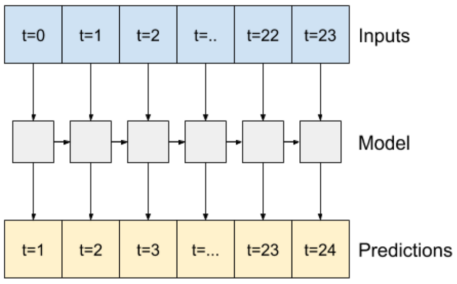


Figure 19: LSTM layer returning sequences

Another approach would be to feed each cell with the complete time series instead of only one time step extracting an output only from the last component of the last layer , but for computational purposes and to fully make use of the LSTM cell potential we don't make use of this technique.

3.3 Convolutional Neural Network

We make use of Conv1D layers to extract rich features from past load time series. A set of filters slides over the input sequence applying dot products between the filter weights and the input extracting the corresponding set of features.

The kernel size represents the window size on witch the filter applies the product while the number of filters represents the number of features extracted.

Padding is applied in order not to reduce the time series length by adding zeros at the start of the series and at the end of the series. By using multiple filters, each filter learns to detect different patterns or features in the input sequence.

In our models we will further process the output from the Conv1D layer through LSTM layers.

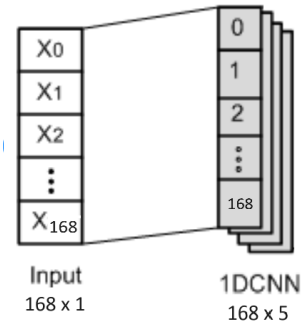


Figure 20: Convolution process

3.4 Implemented Structures

We consider four main structures in our study: DNN, RNN, MIXED, CNNMIXED.

Data contained in the training and validation set is split into time windows of 8 days (192 hours).In this way models are fed with data relative to the past 7 days and forecasts or time information up to the next date. The precise input features depends on the model.

Outputs of such structures depend instead on the type of prediction selected. We consider either 24 x 1 hourly point prediction using MSE as a loss funtion, 24 x 11 quantile hourly prediction using the pinball loss funtion, 24 x 4 Jhonson's SU probability distribution parameters estimate using log likelyhood loss function.

The DNN structure consists of a simple dense network composed of two hidden layers containing 512 neurons each equipped with a soft plus activation function. The model takes as input a flat vector where past load, next day weather forecasts, next day holiday and time information are stored.

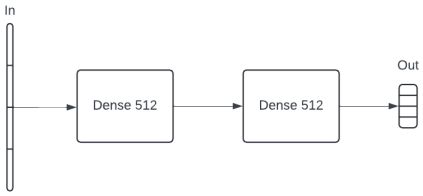


Figure 21: DNN

The RNN structure consists of 168 LSTM cells stored in a single hidden layer. We decide not to use future weather forecasts in this structure and just take all the available information in the past 168 hours. In this way each cell receives as input multiple features relative to one time step and is able to learn patterns over time lags in order to make a good prediction. The cell uses tanh as an activation function and a dropout of 0.1 between cells is selected in order not to overfit the model on training. Return sequences has been set to false.

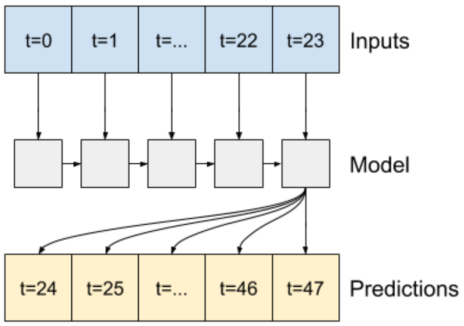


Figure 22: Example RNN

Combining LSTM and Dense modules we create the MIXED model. Past load values as well as Holiday information is processed trough 3 LSTM layers with size 168 and dropout 0.1. Last LSTM layer returns sequences of hidden states. The hidden state h_t is then averaged over all features before getting concatenated with the output generated from 4 dense layers of size 24 taking as input values relative to the next day date info and weather forecast. The then 192 long vector gets processed in a final dense layer with 24 neurons witch elaborates the output.

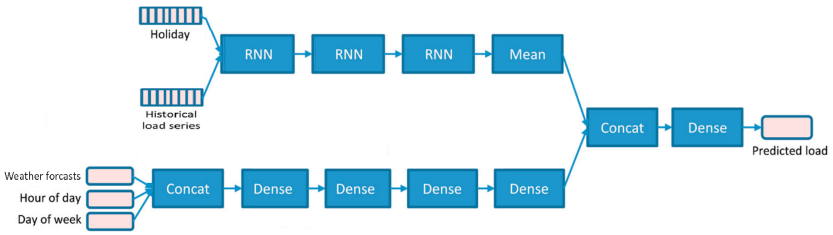


Figure 23: MIXED

By just adding to the previous model two convolutional layers with 5 filters having kernel size 3 before the LSTM cells originates our final model CNNMIXED, able to capture more complex characteristics from past loads. All other structural features remain unchanged from the previous model.

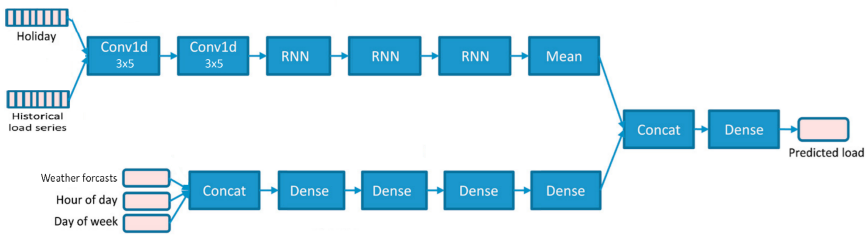


Figure 24: MIXEDCNN

3.5 Conclusions

In this section we looked into our neural networks proposals. Nonetheless, to proceed with probabilistic forecasting, we decide to explore some widely employed methods in the literature. The following section aims to present these techniques.

4 Conformal Predictions

The most common way of obtaining probabilistic forecasts is by generating prediction intervals, which provide an admissible range of values for future observations with a given confidence level. The width of the prediction interval is thus governed by the confidence interval but also by the performance of the underlying prediction algorithm. Conformal prediction is a promising framework to overcome both issues. It's a general procedure to build predictive intervals for any predictive model that achieves nominal marginal coverage with the only assumption that the data are exchangeable.

4.1 Formal definition

Suppose we have n training samples $(x_i, y_i) \in R^d, i \in [1, n]$ that are realizations of random variables $(X_1, Y_1), \ldots, (X_n, Y_n)$, and that we aim at predicting a new observation y_{n+1} at x_{n+1}. Given a confidence level $\alpha \in [0, 1]$ fixed by the user, the aim is to build a predictive interval C_α such that the **coverage** of the PI i.e. the probability that the interval contains the actual value of the predicted variable:

$$P\left(Y_{n+1} \in C_\alpha(X_{n+1})\right) \geq 1 - \alpha \tag{6}$$

Where the interval is:

- **valid** if the interval satisfies equation 6
- **efficient** when it is as small as possible

Thus, probabilistic forecasts should yeald PIs that are as narrow as possible, while ensuring the designed confidence level.

4.2 Conformal Prediction

Conformal Prediction is a probabilistic forecasting technique that constructs valid PIs in finite samples without making any distributional assumptions besides being **exchangeable**. This means that the information provided by the observations is independent of the order in which the observations are presented. In particular, in our analysis we refer to the inductive conformal prediction that requires the training data to be split into two disjoint sets. Given the previous training set $(X_1, Y_1), \ldots, (X_n, Y_n)$, we split it into two subsets: the proper training set I_1 and the calibration set I_2. We use the conformity score obtained from I_2 to quantify the uncertainty in future predictions, given the model fitted using I_1. Conformal Prediction provides this PI centred on the predicted value:

$$\hat{C}_\alpha(x_{n+1}) := [\hat{\mu}(x_{n+1}) \pm \hat{Q}_{1-\alpha}(S_{I_2})] \tag{7}$$

Where in regression it's usually used the absolute value of the residuals $s_i = |\hat{\mu}(x_i) - \mu_i|, i \in I_2$. Finally, to define the size of the interval is computed the (1-α)-th quantile of these scores.

4.3 Conformalized Quantile Regression

In application domains such as energy analytics, time series often exhibit strong seasonal behaviour (as showed previously), since the variance in the observations relate to the cyclic nature of the data: when the variability is lower one can make more confident predictions with narrower PIs. We tackle

the challenge of constructing adaptive efficient and valid PIs for time series data by combining the previous conformal prediction and the strengths of quantile regression, generating PIs that adapt to the local volatility in the time series.

PIs can be directly estimated from the 2 quantile functions estimated via the Quantile Regression (QR) and computed from the training set I_1. The confidence level (1-α) of the PI is the difference between such two quantile levels $\alpha_{\text{lo}} = \alpha/2$ and $\alpha_{\text{hi}} = 1 - \alpha/2$. The corresponding estimated PI thus becomes:

$$\hat{C}_\alpha(x) = [\hat{q}_{\alpha_{\text{lo}}}(x), \hat{q}_{\alpha_{\text{hi}}}(x)] \tag{8}$$

Unlike the Pi in Equation (7), the width of the PI in Equation (8) depends on each specific data point x and can vary significantly from point to point adapting to heteroscedasticity in the data. Nevertheless, the coverage of the PI in Eq. (8) is not guaranteed to match the designed confidence level (1-α).

Conformalized Quantile Regression (CQR) is a probabilistic forecasting method that combines CP and QR to construct **valid** PIs for heteroscedastic data: CQR inherits the advantages of both QR and CP: the properties of QR allow the method to adapt to the local variability in the data and the use of CP guarantees valid coverage. Similarly to CP, in CQR we assume the samples to be exchangeable and we split the training data into a proper training set I_1 and a calibration set I_2. In order to conformalize the Pis we use as conformity scores $e_i = \max\{\hat{q}_{\alpha_{\text{lo}}}(x_i) - y_i, y_i - \hat{q}_{\alpha_{\text{hi}}}(x_i)\}, \quad i \in I_2$, which quantifies the error made by the PI of the previous QR algorithm in Eq. (8). As we did before, to define the size of the interval is computed the (1-α)-th quantile of these scores:

$$\hat{C}_\alpha(x) = [\hat{q}_{\alpha_{\text{lo}}}(x) - Q_{1-\alpha}(E_{I_2}), \hat{q}_{\alpha_{\text{hi}}}(x) - Q_{1-\alpha}(E_{I_2})] \tag{9}$$

Computationally, in order to check the **validity**, we asked the proportion of the 24 hourly actual values outside the predicted interval to be $\leq \alpha$.

4.4 Adaptive Conformal Inference

The cornerstone of the previous theory is the exchangeability assumption of the data. However, this assumption is not met in time series forecasting problems: applying it to electricity/load price forecasting on various data, the **validity** varied greatly depending on the markets. One case that breaks the exchangeability assumption is the distribution shift, e.g. where the test data is shifted with respect to the training data. Accounting for an undefined number of shifts on the joint distribution, Gibbs and Candés proposed Adaptive Conformal Inference (ACI). The idea underlying this algorithm is to update online the quantile level used by a recursive scheme depending on an hyper-parameter γ, called **learning rate**. Given any data distribution, it's even possible to prove an asymptotic validity result.

The aim is to give predictive intervals for I_2 subsequent observations sequentially: at any prediction step t the previous response values up to $t-1$ have been revealed and are used to construct the predicted interval. In order to improve adaptation when the data is highly shifted, an effective confidence level α_t updated recursively is used instead of the target level α. Set $\alpha_1 = \alpha$, and for $t \geq 1$:

$$\begin{cases} \hat{C}_{\alpha_t}(x_t) = [\hat{\mu}(x_t) \pm \hat{Q}_{1-\alpha_t}(S_{I_2})] \\ \alpha_{t+1} = \alpha_t + \gamma \left(\alpha - 1\{y_t \notin \hat{C}_{\alpha_t}(x_t)\}\right) \end{cases} \tag{10}$$

For $\gamma \geq 0$. If ACI does not cover at time t, then $\alpha_{t+1} \leq \alpha_t$, and the size of the predictive interval, given by the $(1-\alpha_t)$-th quantile of the same conformity scores of Eq. (7), increases. Conversely, when it covers. Nevertheless, nothing prevents $\alpha_t \leq 0$ or $\alpha_t \geq 1$. In case $\alpha_t \geq 1$ by convention $\hat{C}_{\alpha_t}(.) = \{\hat{\mu}(.)\}$ i.e. $\hat{Q}_{1-\alpha_t} = 0$. Since α is small (around 0.1) this situation is rare. The other case can happen frequently for some λ, giving $\hat{C}_{\alpha_t}(.) = R$ i.e. $\hat{Q}_{1-\alpha_t} = +\infty$. In this situation, it could be

possible to impute the length of infinite intervals by twice the overall maximum of the residuals. We disregard this situation because the code is designed to forecast 365 days only when the alpha values are not excessively low. Otherwise, an extensive training set I_1 is required.

Since it works online and updates the quantile levels according to the previous error, ACI could cope with a model that has not correctly caught the temporal evolution, thus being a perfect candidate for CP for time series with general dependency. The choice of the parameter λ impacts the behavior of C: while the method always satisfies the asymptotic validity property, i.e., $\frac{1}{T}\sum_{t=1}^{T} 1\{y_t \notin \hat{C}_{\alpha_t}(x_t)\} \xrightarrow[T\to\infty]{a.s.} \alpha$ Directly from the fact that $\frac{1}{T}\sum_{t=1}^{T} 1\{y_t \notin \hat{C}_{\alpha_t}(x_t)\} - \alpha \leq \frac{2}{\gamma T}$ (Gibbs and Candès), the use of larger γ values gives an insight to the length of resulting intervals. Regrettably, the code is not supported to handle small or negative values for α_t. Through an asymptotic analysis of ACI's behaviour for simple time series distribution it can be even proved that ACI deteriorates efficiency in an exchangeable case.

4.5 Adaptive strategies based on ACI

To prevent the critical choice of γ an ideal solution is an adaptive strategy with a time dependent γ. In particular, we implemented two strategies based on running ACI for $K \in N$ values $(\gamma_k)_{k\leq K}$, chosen a priori by the user: the only additional cost is the computation of K different quantiles.

4.5.1 Naive startegy

The first strategy uses at each step the γ that achieved the best efficiency in the past while ensuring validity. The user needs to select even a warm-up period $T_\omega \leq T_1 - 1$, where T_1 represents the time length of the calibration set I_2. For the first T_ω steps, the algorithm selects the first value for the learning rate λ_1. For the following ones, we select

$$\begin{cases} k_{t+1} = \arg\min_{k\in A_t} \left\{ \frac{1}{t}\sum_{s=1}^{t} \text{length}(\hat{C}_{\alpha_{s,k}}(x_s)) \right\}, & \text{if } A_t = \{k \in [1,K] \mid \frac{1}{t}\sum_{s=1}^{t} 1\{y_s \in \hat{C}_{s,k}(x_s)\} \geq 1-\alpha\} \\ k^*_{t+1} = \arg\min_{k\in[1,K]} \left\{ \left| 1-\alpha - \frac{1}{t}\sum_{s=1}^{t} I\{y_s \in \hat{C}_{\alpha_{s,k}}(x_s)\} \right| \right\}, & \text{if } A_t = \emptyset \end{cases}$$

4.5.2 Online Expert Aggregation on ACI, AgACI

Now we introduce AgACI, a parameter-free method that uses online expert aggregation, avoid choosing γ and achieving good results in terms of validity and efficiency. Each expert of the aggregation is ACI with parameter γ_k. The algorithm performs two independent aggregations of the K-ACI intervals, one for each bound, obtaining $\tilde{C}_t(\cdot) = [\tilde{b}^l_t(\cdot), \tilde{b}^u_t(\cdot)]$. The algorithm computes an optimal weighted mean of the experts, where the weights $w^l_{t,k}$ and $w^u_{t,k}$ assigned to expert k depend on all depend on all experts performances at time steps 1,..,t. In particular, we choose the pinball loss ρ_β with $\beta = \frac{\alpha}{2}$ and $1-\frac{\alpha}{2}$ respectively. These losses are plugged in the aggregation rule Φ. For our algorithm, in particular, we choose the Bernstein Online Aggregation relying on R package OPERA. We use the gradient trick in the simulations. Here is the algorithm:

1. $\beta^l = \frac{\alpha}{2} \quad \beta^u = 1 - \frac{\alpha}{2}$
2. **For** $t \in [T_0+1, T_0+T_1]$, **do**
3. Set $\tilde{C}_t(x_t) = [\tilde{b}^l_t(x_t), \tilde{b}^u_t(x_t)]$
4. for $k \in [1,K]$, do

5. Compute $\hat{b}^{(\cdot)}_{t,k}(x_t)$ using ACI with γ_k
6. end for
7. for $k \in [1, K]$, do
8. $\omega^{(\cdot)}_{t,k} = \Phi\left(\rho^{(\cdot)}_{\beta}(y_s, \hat{b}^{(\cdot)}_{s,l}(x_s)), s \in [T_0 + 1, t], l \in [1, K]\right)$
9. end for
10. Define $\tilde{b}^{(\cdot)}_{t+1}(x_t) = \frac{\sum_{k=1}^{K} \omega^{(\cdot)}_{t,k} \hat{b}^{(\cdot)}_{t,k}(x_t)}{\sum_{k=1}^{K} \omega^{(\cdot)}_{t,k}}$
11. **end for**

Even in this prediction we disregard the possibility of infinite intervals since the code is neither able to work with values of $\alpha \overset{\geq}{\Rightarrow} 0$

4.6 Practical remarks

In order to avoid the mentioned criticalities and to empathize the improvements in the results, we consider the following settings:

PF_method	"JSU"
model_class	"DNN"
num_vali_samples	100
idx_start_train	2016/4/1
idx_start_oos_preds	2017/12/31
num_cali_samples, I_1 (days)	636
I_2 (days)	4
transformation_target_variable	"log"
optuna_m	"grid_search"
pred_horiz	24

Firstly, we apply the algorithm of conformal predictions and conformalized quantile regression.

Average Delta Coverage	3.148
Average Pinball Score	18.500
Average Winkler Score	929.475

Table 6: CP

Average Delta Coverage	2.069
Average Pinball Score	13.054
Average Winkler Score	739.751

Table 7: CQR

In line with our expectations, the results (see Figure 25) indicate that, unlike CP, CQR exhibits both valid coverage and local variability in the data.

Subsequently, we employ the first algorithm based on Adaptive Conformal Inference. By increasing the learning rate γ, ACI can expand the interval's size faster when cover is not achieved, thereby improving **validity**, as illustrated in Figure 25. Considering both $\gamma = 0.001$ and $\gamma = 0.002$:

	ACI
learning rate	0.001

Average Delta Coverage	3.322
Average Pinball Score	14.165
Average Winkler Score	834.585

	ACI
learning rate	0.002

Average Delta Coverage	3.077
Average Pinball Score	11.435
Average Winkler Score	681.879

Despite this, the more sophisticate Naive algorithm

	NACI
learning rates	0.001, 0.002
warm-up period (T_w)	2 (days)

Average Delta Coverage	1.204
Average Pinball Score	13.231
Average Winkler Score	742.956

suggests to alternate among to the 2 values -Table 8- for the learning rates even after only 1 step (there are only 2 remaining days apart from the warm-up period). This algorithm results in selecting a γ that achieved good results in the past. Consequently, it's slightly more likely to fail to cover in future steps.

α	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
best_γ (day 2)	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
A_t (day 3)	[]	[]	[]	[]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[]	[]
best_γ (day 3)	0.001	0.001	0.001	0.001	0.002	0.002	0.002	0.002	0.001	0.001
A_t (day 4)	[]	[]	[]	[]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[]	[]
best_γ (day 4)	0.001	0.001	0.001	0.001	0.002	0.002	0.002	0.002	0.001	0.001

Table 8: Best γ, NACI

Finally, the Online Expert Aggregation on ACI still recommends a combination of the two learning rates after the warm-up period. This approach is advantageous as the aggregation function enhances the coverage, reducing the dependence on the selection of γ.

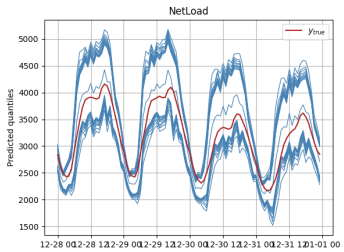
	AgACI
learning rates	0.001, 0.002

α	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
ω_l(day 1)	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]
ω_u(day 1)	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]
ω_l(day 2)	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]
ω_u(day 2)	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.5]
ω_l(day 3)	[0.5, 0.5]	[0.5, 0.5]	[1.0, 0.0]	[0.5, 0.5]	[0.9, 0.1]	[0.5, 0.5]	[0.5, 0.5]	[1.0, 0.0]	[0.5, 0.5]	[0.5, 0.5]
ω_u(day 3)	[0.5, 0.5]	[0.5, 0.5]	[0.0, 1.0]	[0.5, 0.5]	[0.0, 1.0]	[0.5, 0.5]	[0.5, 0.5]	[0.0, 1.0]	[0.5, 0.5]	[0.5, 0.5]
ω_l(day 4)	[0.5, 0.5]	[1.0, 0.0]	[0.9, 0.1]	[1.0, 0.0]	[0.1, 0.9]	[1.0, 0.0]	[1.0, 0.0]	[0.8, 0.2]	[0.5, 0.5]	[0.5, 0.5]
ω_u(day 4)	[0.5, 0.5]	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]	[0.1, 0.9]	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]	[0.5, 0.5]	[0.5, 0.5]

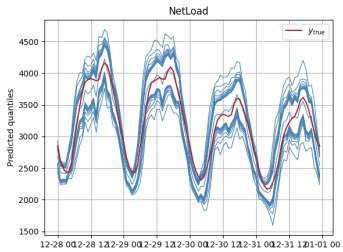
The scores obtained are the following:

Average Delta Coverage	1.352
Average Pinball Score	17.477
Average Winkler Score	939.676

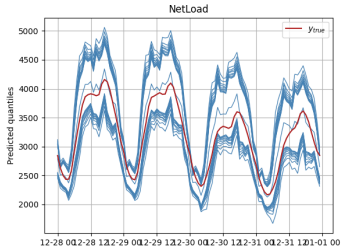
Table 9: AgACI



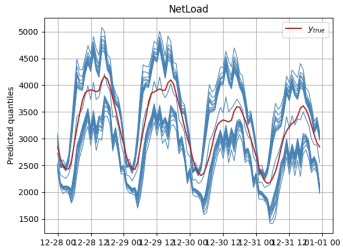
(a) CP



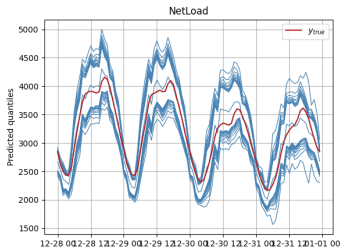
(b) CQR



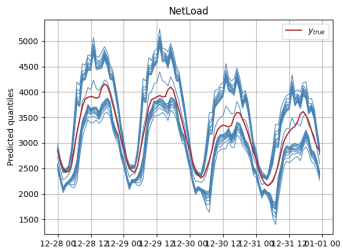
(c) ACI $\gamma = 0.001$



(d) ACI $\gamma = 0.002$



(e) NACI



(f) AgACI

Figure 25: Load Predictions 2017/12/27 − > 2017/12/31

4.7 Conclusions

In this segment, we looked into some methodologies commonly used in the literature to generate probabilistic confidence intervals. Throughout the discussion, we have presented a multitude of techniques. Now it is time to shift our focus towards evaluating these methods based on their performances and outcomes.

5 Results and Discussion

In this section we draw our conclusions and present the best model in the prediction process. Our reference index, as indicated in the competition rules, will be Delta Coverage. However we decide to report also other indexes, useful in drawing other conclusions. We report the most used probabilistic and point scores used in the literature.

In particular we will use Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Symmetric Mean Absolute Percentage Error (sMAPE) to assess the quality

of the point forecast and Pinball's score, Winkler's score and Delta Coverage to assess the quality of the probabilistic forecast.

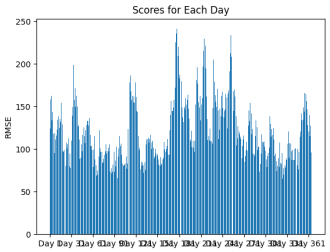
5.1 The holidays issue

Before proceeding to discuss the results, we present another challenge we faced. Our analysis reveals that the models we propose show uniform scores across the year, except for certain specific days. These days coincide with holidays or days preceding or following them. For the moment holidays are treated as a **CONST**__ variable. Let us consider for example our hybrid model with convolutional layer with hyper-parameters and preprocessing techniques in Table 19.

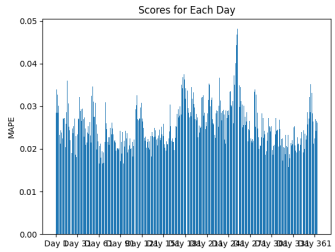
Parameter	**Value**	**Parameter**	**Value**
Dropout	0.6	Recurrent Dropout	0.6
LSTM Size	168	Number of LSTM Layers	1
Learning Rate	0.001	LSTM Activation Function	ReLU
Dense Activation Function	ReLU	Number of Encoder (CNN) Layers	1
Filter Size	5	Kernel Size	3
Number of Final Dense Layers	1	Dense Size	512
Last layer after dense sequence Size	24	Batch Size	32
Target Transformation	Log Transformation	Residual Analysis	No
PCA	No	**CONST**__ Transformation	One Hot Encoding

Table 10: Hyper-parameters for the Hybrid Model with Convolutional Layer

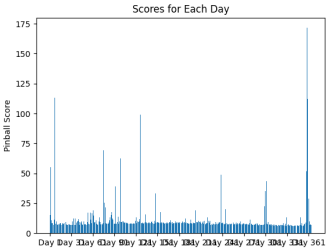
We perform a prediction on a year horizon - 2017. The figures below show two point scores - RMSE and sMAPE - and two probabilistic scores - Pinball's score and Winkler's score -grouped by day.



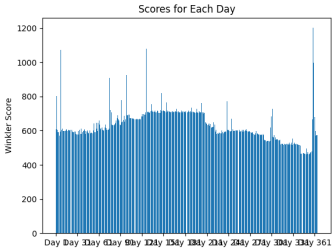
(a) Daily RMSE



(b) Daily sMAPE



(c) Daily Pinball's score



(d) Daily Winkler's score

Figure 26: Model's daily scores with no holiday modification

We can see that the scores worsen drastically in specific days. In order to solve the issue, as [8] proposes, we decide to model the holiday days as Sundays. Despite not being explicitly evident from our initial data exploration, holiday periods typically show lower values compared to other weekdays. In particular we focus on the the Christmas week period - from December 23 to December 31 2017 to present the analysis.

We perform point prediction with our hybrid model before any modification. Table 12 and figure 27 report the point forecast results.

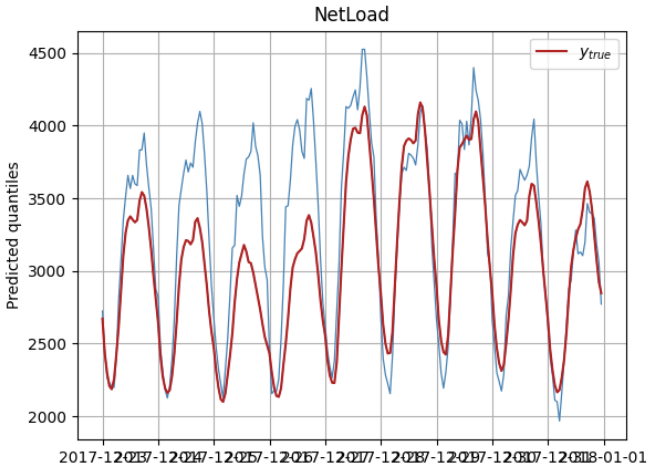


Figure 27: Prediction before holiday modification

Metric	Value
MSE Average	137492.1945
MAE Average	277.3967
RMSE Average	370.7993
sMAPE Average	0.0857

Table 11: Performance Metrics before holiday modification

We decide approach the problem in two ways.

5.1.1 Rescaling factor

Given the Neural Network's limited capability to distinguish these special days because of their rare appearance, we decide to initially ignore them to then apply a post-processing scaling adjustment after an initial prediction has been computed. After determining witch day of the week the holiday falls into we select all same days across previous month and compute the average load requested in such days. Subsequently we apply the same process for Sundays of the prior month and divide such quantity for the value computed before obtaining a ratio that will be applied to the holiday prediction. In this way if the holiday occurs on a Sunday, the adjustment will be very small, with the rescaling factor very close to one. Conversely, if the holiday falls on a weekday the associated rescaling factor will be smaller. Moreover, by considering only values up to 30 days prior, we also take into account the season effect. In a second instance we decide to rescale only the hours starting from approximately the minimum value of the target during the day (around 4 a.m.) in orther to smoothen the prediction curve.

We perform point prediction with our hybrid model on the period after such modification. Table 12 and figure 28 report the point forecast results.

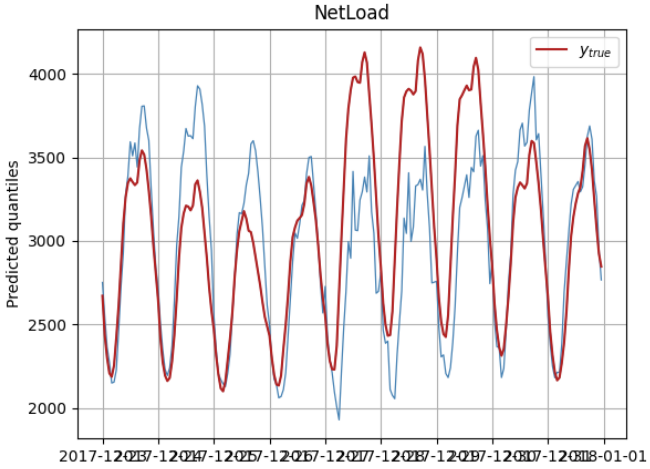


Figure 28: Prediction with rescaling factor

Metric	Value
MSE Average	155797.5447
MAE Average	293.2599
RMSE Average	394.7119
sMAPE Average	0.0936

Table 12: Performance Metrics before holiday modification

We notice that in this way the prediction during Christmas and on December 26 improves significantly. However, even if it is not our case of day-ahead prediction, we do realize a limitation this method is subjet to: indeed prediction can be adjusted only up until the Sunday after the day for which true load is known due to the information needed for the adjustment. Furthermore, we notice how the prediction worsens during December 27, 28, 29. This is due to the fact that the calendar refers to these days as holiday days; however, the consumption is not as low as the "real" holiday days. This globally worsens the prediction, as the point scores indicate.

5.1.2 Holiday as a weekend

We use this approach when treating the dummy option for the **CONST**__ variables. With a similar rationale as before, we decide to set the holidays as weekends as section 2.6.2 describes. Also in this case, we are aware of the fact that probably the model will underperform on days which are considered "fake" holidays. The point scores can be found in Table 13 and the point prediction is illustrated in Figure 29.

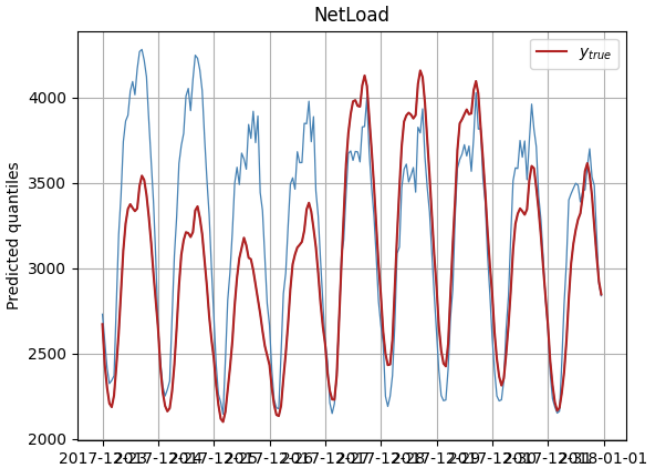


Figure 29: Prediction with rescaling factor

Metric	Value
MSE Average	164764.9001
MAE Average	324.1973
RMSE Average	405.9124
sMAPE Average	0.10035

Table 13: Performance Metrics before holiday modification

We notice how this approach applies weekend loads on holidays effectively underestimating loads during "fake" holidays and overastimating it in "true" holidays.

5.1.3 Holiday modification

As we highlighted above, the rescaling factor seems to be an optimal approach in case of the "true" calendar holidays, whereas the standard model seems to perform better in case of "fake" holidays. To address the issue of distinguishing between these two sets of dates, we adopt a historical approach.

First of all, we retrieve from the dataset all the holiday days over the years. We observe that the dataset marks a significant number of days as holidays, including many that are not traditionally recognized as such, particularly in August and December.

We focus our analysis on a specific year of the training set, 2016. For each month, we plot the maximum net load for each day considered as a holiday. Figure 30 shows this pattern during December 2016.

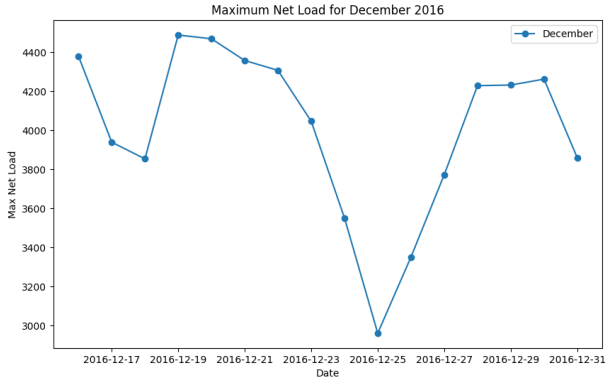


Figure 30: Maximum Load per day during December 2016 holidays

We can observe a very low value corresponding to Christmas and the days in this period, as confirmed by the previous plots. Instead we see high loads values during the mid-week and on the 28th, 29th, 30th of the month. These days, despite being labeled as holidays, show evidence of activity.

We propose to modify the variable **CONST_Holiday** by excluding the days with high load values.

In order to do this, we compute the mean of the maximum load per day accross the month for working days and we set a threshold by subtracting the standard deviation from its mean. We set all the **CONST_Holiday** variables of the days where the maximum load exceeds this threshold equal to 0. In such way we are partitioning the holiday set in a subset of days characterized by low activity levels and other days marked as holidays but showing higher activity.

Eventually, we reperform the test using the rescaling factor on the previous period. Figure 31 and table 14 show our results.

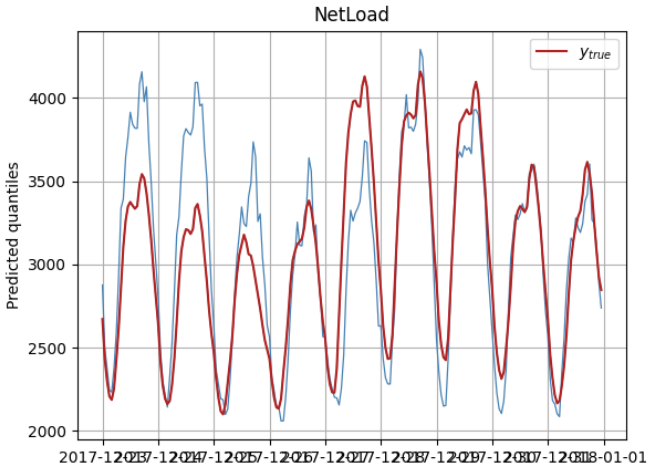


Figure 31: Prediction with rescaling factor with holiday modification

Metric	Value
MSE Average	94887.7107
MAE Average	224.9665
RMSE Average	308.0384
sMAPE Average	0.0723

Table 14: Performance Metrics with holiday modification

We can notice that the point scores improve. Moreover, the model with this modification performs better during actual calendar holidays, and in the meantime improves in capturing the peak load values missed by the second case study.

5.1.4 Further improvements

In the context of the cosine transformation for **CONST__** variables, we propose introducing a new feature, **PAST__Distance_To_Holiday**, that replaces the old **CONST__** holiday. This new variable quantifies the remaining time (in hours) until the first hour of the next scheduled holiday and is integrated into the neural network model as a historical input.By computing the newly added variable for days considered holidays after the manipulation developed in the previous section we present our findings in figure 32 and table 15.

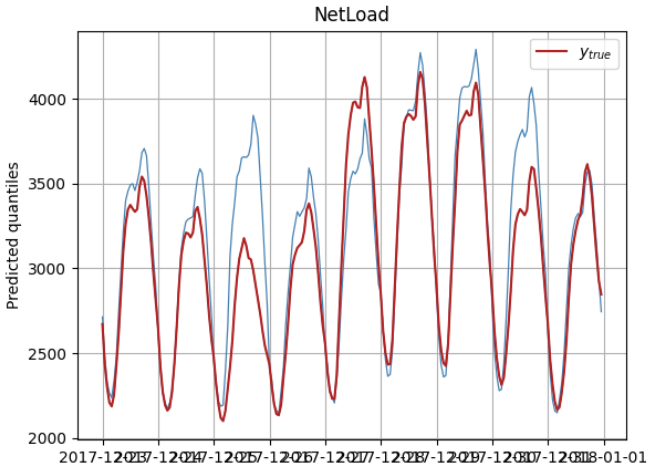


Figure 32: Prediction with variable and holiday modification

Metric	Value
MSE Average	66172.5548
MAE Average	177.4989
RMSE Average	257.24026
sMAPE Average	0.0551

Table 15: Performance Metrics with holiday modification

We can notice that during Christmas itself the prediction still remains above the true value. However if we introduce the rescaling factor, this would lower other predictions that the Neural Network already matches well. In the next sections we will discuss which approach to employ for the issue.

5.2 The neural network's choice

Due to time constraints, it is unfeasible to test all the proposed combinations described above. Therefore, we decide to follow a logical reasoning path, comparing different results and excluding some models.

We start by comparing the four neural network models we proposed. In order to have a fair comparison, we decide to fix the hyperparameters in advanced and use, for all the models, the following preprocessing techniques:

- MinMaxScaler preprocessing
- Introduction of the holiday distance variable without redefining holidays
- Log transformation of target variable
- Cosine transformation of **CONST**__ variables
- Hourly next day point prediction with weekly lookback window
- ACI conformal prediction with 214 calibration samples (01/06/2016-31/12/2017)

With these preprocessing techniques fixed, we investigate how our models perform under various combinations of feature reduction and de-seasonalization techniques.

5.2.1 Forward Neural Network

We start by applying the DNN model fixing with the following hyper-paramters:

Hyperparameter	Value
Hidden Size	512
Number of Hidden Layers	2
Learning Rate (lr)	0.001
Activation Function	Softplus

Table 16: DNN Model Hyperparameters

We test the model on the year 2017; figure 33 shows the results.

	NO PCA NO RESIDUALS	PCA NO RESIDUALS	NO PCA FOURIER	PCA FOURIER	NO PCA GLM	PCA GLM
Delta Coverage	0,3288	0,2085	0,2755	0,3190	1,2456	1,0300
Pinball Score	12,3150	12,4630	13,4700	13,4300	12,1525	11,1170
Winkler Score	615,8400	633,7800	699,7000	687,0200	684,9100	622,1500
MSE	22215,0000	23421,0000	28418,0000	29047,0000	16939,0000	14149,0000
MAE	102,7000	106,9700	127,3000	124,3000	93,5200	84,8500
RMSE	149,0500	153,0410	168,6000	170,4000	130,1500	118,9500
sMAPE	0,0315	0,0330	0,0390	0,0380	0,0290	0,0270

Figure 33: DNN results

Firstly, we notice that the model performs better when applying GLM-One-Hot-Encoding for de-seasonalization. Instead, applying a Fourier de-seasonalisation seems to result in worse performance. It is not clear if PCA feature reduction is beneficial, as the scores remain similar. Additionally, we observe that the Average Delta Coverage is highly variable and does not necessarily correspond to the best scores in other metrics.

5.2.2 Recurrent Neural Network

We fix the following hyper-parameters for our recurrent neural network model as follows:

Parameter	Value	Parameter	Value
Dropout	0.1	Recurrent Dropout	0.1
LSTM Size	168	Number of Hidden Layers	1
Learning Rate (lr)	0.0001	Activation Function	Tanh

Table 17: RNN Model Hyperparameters

By performing the prediction on the horizon, we report our results in figure 34.

	NO PCA NO RESIDUALS	PCA NO RESIDUALS	NO PCA FOURIER	PCA FOURIER	NO PCA GLM	PCA GLM
Delta Coverage	0,3329	1,2222	0,2694	0,4459	1,7710	0,3346
Pinball Score	10,8775	17,3459	12,4282	21,8720	10,8190	17,3185
Winkler Score	594,2889	974,3144	702,4735	1240,1673	627,6462	940,2888
MSE	18356,8949	54818,2023	23940,3609	86625,2283	15757,7155	45379,4514
MAE	95,1946	183,0535	114,0869	231,6558	85,6834	157,1435
RMSE	135,4876	234,1329	154,7267	294,3216	125,5297	213,0245
sMAPE	0,0292	0,0567	0,0356	0,0744	0,0261	0,0480

Figure 34: LSTM results

We notice that the scores in the worst scenarios are generally higher compared to the DNN model. However, in the best scenarios, the scores seem to outperform the DNN model. Similarly to before, the combination of PCA and Fourier de-seasonalization does not appear to be benefiacial. Moreover, in this case we notice that in general PCA worsens the situation.

5.2.3 Mixed Neural Network

For this model, we fix the hyperparameters in table 18.

Parameter	Value	Parameter	Value
Dropout	0.1	Recurrent Dropout	0.2
LSTM Size	168	Number of LSTM Layers	3
Learning Rate (lr)	0.001	LSTM Activation	ReLU
Dense Activation	Sigmoid	Number of Encoder Layers	2
Filter Size	10	Kernel Size	10
Number of Dense Layers	4	Dense Size	24
Dense Final Size 2	24	Number of Final Dense Layers	1
Final Dense Size	512		

Table 18: Mixed Model Hyperparameters

We perform the test; figure 35 shows the obtained results.

	NO PCA NO RESIDUALS	PCA NO RESIDUALS	NO PCA FOURIER	PCA FOURIER	NO PCA GLM	PCA GLM
Delta Coverage	0,34374	0,3672714	1,3255614	0,3301842	0,8189376	1,3566
Pinball Score	10,749984	10,58964	12,044466	14,572128	8,845134	8,461512
Winkler Score	538,84509	557,2507044	695,586144	771,317472	501,67323	490,94028
MSE	15306,59797	15099,96933	19498,23524	27928,44742	10579,95459	9857,409234
MAE	74,97	78,344874	98,006598	110,375322	70,702728	67,448724
RMSE	124,950816	124,104624	141,025506	168,781032	103,8819	100,272324
sMAPE	0,0231132	0,024072	0,030192	0,0341598	0,021726	0,0207774

Figure 35: Mixed model

We notice that the aggregate scores improve compared to the initial two neural network models (see figures 33 and 34). Despite the general improvement, there appears to be no significant enhancement in Delta Coverage, which remains unstable. However the other probabilistic and point indices demonstrate progress - for instance RMSE decreases from a peak of 118 in the optimal scenario for DNN to 100 in the current optimal scenario.

5.2.4 CNN-Mixed Neural Network

We set the same hyper-parameters of the mixed model for our final model proposal (see table 18) and with "relu" activation function for the convolutional neurons. We perform over the given year the test with the usual data preprocessing techniques described above. Results are reported in figure 36.

	NO PCA NO RESIDUALS	PCA NO RESIDUALS	NO PCA FOURIER	PCA FOURIER	NO PCA GLM	PCA GLM
Delta Coverage	1,0317	0,4934	0,7729	0,2187	1,1487	1,2582
Pinball Score	9,6267	10,8569	12,6299	11,5331	8,1270	7,8344
Winkler Score	547,3696	541,4842	714,7913	650,1885	470,7482	461,9136
MSE	12565,4688	15894,9381	22681,2585	19167,5055	9295,4902	8812,5204
MAE	73,8856	76,0706	103,6573	96,1956	64,2746	64,4145
RMSE	112,0958	126,0751	150,6030	138,4468	96,4131	93,8750
sMAPE	0,0229	0,0233	0,0324	0,0297	0,0197	0,0198

Figure 36: CNN Mixed model

It can be observed how the overall indices generally outperform those of the previous models. Particularly notable are the very good results achieved with the combination of GLM and PCA. Specifically, in these two latter scenarios we obtain an average sMAPE below 2%. However, the Delta Coverage index tends to worsen when low point prediction scores are achieved.

This behavior appears to be common across all models tested. Models that perform well in point predictions seem to be less outlier-robust, hence causing conformal prediction to fail critically when encountering them. Less specific point predictors tend instead to have higher average sMAPE but show improvement in handling outliers causing for an overall improvement on the delta score. Balancing accuracy and robustness seems to be key.

Given the overall superior performance of the CNN-mixed model compared to the others, we decide to continue with this neural network model.More specifically we focus on the model running on GLM residuals since it yielded the best point scores.

5.2.5 CONST__ Variables modification and De-seasonalization Choice

First, we investigate how to treat the **CONST__** variables. We conduct a test on the year 2017 using the same preprocessing techniques and hyper-parameters in table 18 while considering next days information as dummy variables instead of periodic. Figure 37 exhibits the results.

	NO PCA GLM	PCA GLM
Delta Coverage	0,8162	0,18209
Pinball Score	8,5849	8,9365
Winkler Score	482,2554	493,6477
MSE	10322,2209	11230,9163
MAE	67,2868	69,103
RMSE	101,5983	105,976
sMAPE	0,02094	0,02122

Figure 37: CNN Mixed GLM model with dummy

By comparison with the last two columns in Figure 36 we note how a slight worsening in point prediction causes for a general improvement in delta scores. The model now seems to be more balanced. We emphasize such result by directly comparing models running on PCA feature reduction with the two different CONST handling approaches:

	PCA PERIODIC	PCA DUMMY
Delta Coverage	0,33105	0,27905
Pinball Score	14,3255	14,8115
Winkler Score	704,3258	727,7229
MSE	24996,8892	26321,4648
MAE	84,7691	87,1634
RMSE	158,104	162,2389
sMAPE	0,0266	0,02719

Figure 38: GLM-CNNMIXED for periodic and dummy approach

Again, a slight worsening on point predictions brings to an improvement to delta coverage. Dummy variables and PCA on forecasts will be used in following tests.

5.2.6 Target Transformation and data scaler

Now, we compare various target transformation techniques. The results are shown below.

	NO TARGET TRANSF	LOG TRANSF	ASINH STAND
Delta Coverage	0,4108	0,27905	0,53907
Pinball Score	15,0609	14,8115	14,7319
Winkler Score	753,6665	727,7229	742,3164
MSE	28720,4346	26321,4648	27636,777
MAE	90,7996	87,1634	91,9304
RMSE	169,471	162,2389	166,2431
sMAPE	0,0285	0,02719	0,02871

Figure 39: CNN Mixed with different target transformations results

In this case, nearly all the scores agree on applying a log transformation to the target variable, although they are similar. Therefore, we opt to maintain this approach.
Additionally, we explore which data normalization method better suits the problem by comparing metric scores in figure 40.

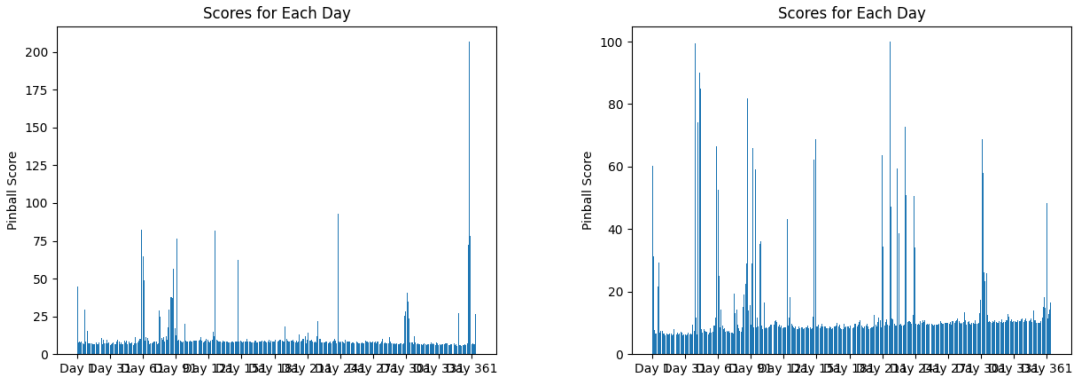
	STANDARD SCALER	MINMAX SCALER	ROBUST STANDARD SCALER	QUANTILE TRANSFORMER
Delta Coverage	1,0412	0,27905	0,3641	0,77693
Pinball Score	14,4648	14,8115	13,9789	14,6821
Winkler Score	737,42	727,7229	720,7042	750,6753
MSE	27500,2204	26321,4648	25443,505	28385,6848
MAE	93,4895	87,1634	88,9333	95,233
RMSE	165,8319	162,2389	159,51	168,4805
sMAPE	0,02917	0,02719	0,0278	0,02969

Figure 40: CNN Mixed with different data scalers results

Globally, the minmax scaler and robust standard scaler exhibit better scores. We opt for the former option.

5.2.7 Holiday processing

As noted in section 5.2.4 models with good point prediction seem to generate high delta coverage due to their higher sensitivity to outliers. We understand that in our framework outliers are inherently tied to holidays hence we decide to process predictions in such days with the ratio developed in section 5.1.1. Furthermore the holidays set is restricted via the technique introduced in section 5.1.3. Pinball score graphs shows how the model after holiday handling does not fail critically as much as in the vanilla option case. We decide to keep this feature in the model.



(a) Pinball score per day -before holiday handling (b) Pinball score per day -after holiday handling

5.3 Conformal predictions

As of now the conformal prediction method was set to CP.

Most of the other methods described in section 4 are now explored with the CNNMIXED-PCA-GLM point predictor with both 250 and 80 maximum epochs. Results are shown in Table.

	CP, 250 EP	ACI, lr=0.001, 250 EP	NACI, lr=[0,0005, 0,001], warmupperiod = 80, 250 EP	CP, 80 EP	ACI, lr=0.001, 80 EP	NACI, lr=[0,0005, 0,001], warmupperiod = 80, 80 EP
Delta Coverage	0,736	0,62747	1,10073	0,5222	0,4023	1,0834
Pinball Score	17,4825	14,8079	16,4978	17,41024	14,0334	17,575
Winkler Score	877,7411	918,3081	900,375	885,8792	957,937	963,4275
MSE	39302,4792	35919,106	33562,0572	37356,6889	35001,7434	38598,3198
MAE	106,7424	93,582	93,281	99,9157	96,878	100,3299
RMSE	198,2485	189,5233	183,1995	193,2788	187,0875	196,46455
sMAPE	0,0336	0,03056	0,0306	0,0328	0,0316	0,0328

Figure 42: CNN Mixed model

As one can note, ACI prediction is the most performing out of the techniques tested with CP coming in second and NACI last. However, a critical factor is the number of dates required for accurate quantile computation: the ACI algorithm could even fail to provide a result with a learning rate of 0.001. At the same time, as specified above, decreasing the learning rate is not advisable since the algorithm may become unable to expand the interval's size when coverage is not achieved. Trivially, similar considerations could apply to NACI and AgACI. Due to the computational complexity of handling 365 dates, AgACI results are not provided: it's necessary to consider minuscule values for the learning rates that exacerbate the prediction.

Regarding the number of epochs we decide to keep the model from overfitting and getting more sensible to outliers by cutting the maximum number down from 200 to 160 (Figure 43). This sacrifices average point performance but benefits our target score.

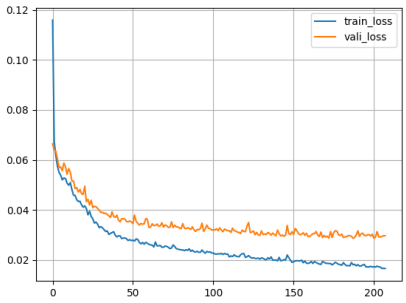


Figure 43: Loss as a function of the number of epochs

5.4 Our final model

Our final model, CNNMIXED, combines benefits from Convolutional Neural Networks (CNNs), Dense Neural Networks (DNNs) and Recurrent Neural Networks using Long Short-Term Memory (LSTM) cells. The model forecasts future loads using various input data processed in multiple stages following the final structure illustrated in section 3.4.

Firstly historical load data is log-transformed for stability. A Generalized Linear Model (GLM) is applied to these transformed loads, and the resulting residuals are used as inputs. Future features are decomposed via Principal Component Analysis (PCA) and next-day time information is encoded with dummy variables.

The point prediction produced by the model then finally undergoes a Conformal Prediction process (ACI) where quantiles are extracted.

Our results show how the model successfully captures rich information from the time series and how it makes use of it to determine complex relationships between them and the target variable: Future Load. Results for day ahead prediction over one year time span outperform all the other models we have considered on a consistent basis in terms of Delta Coverage.

We run the model on the time span between 2017 and 2019 recalibrating the model twice: once at starting date and once after one year predictions. First year's predictions are then used to calibrate our conformal prediction method, at last complete probabilistic forecast for the year 2018 is then yielded. The model fits batches of 32-198 hour time windows on training data starting from 2014 (988-198 hour windows). Past training data is kept while predictions move forward. Validation is performed over the last 100-198 hour windows approaching 2017.

Specific parameters for the network obtained by trial and error are described in table below

Parameter	**Value**	**Parameter**	**Value**
Dropout	0.1	Recurrent Dropout	0.2
LSTM Size	168	Number of LSTM Layers	1
Learning Rate	0.001	LSTM Activation Function	ReLU
Dense Activation Function	Sigmoid	Number of Encoder (CNN) Layers	1
Filter Size	5	Kernel Size	3
Number of Final Dense Layers	1	Dense Size	512
Last layer after dense sequence Size	24	Target Transformation	Log Transformation

Table 19: Hyper-parameters for the final model

Figure 44 and figure 45 show respectively the scores and the prediction plot over the horizon. Figure 46 shows the plot of the scores over the given days. We also present the mean of the probabilistic scores over the 24 hours.

	FINAL MODEL
Delta Coverage	0,16756
Pinball Score	16,5041
Winkler Score	875,1536
MSE	38396,2635
MAE	105,6121
RMSE	195,9496
sMAPE	0,0332

Figure 44: Final scores of our model

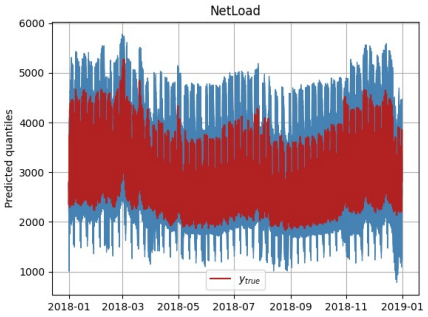
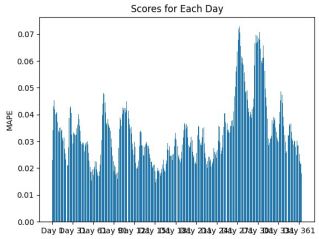
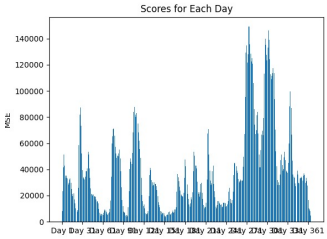


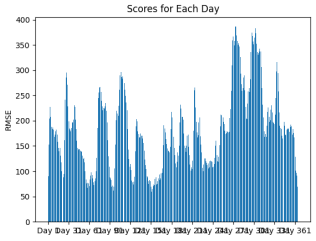
Figure 45: Final prediction of our model



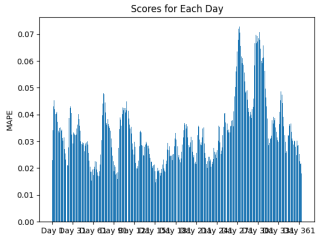
(a) MAE per day final model



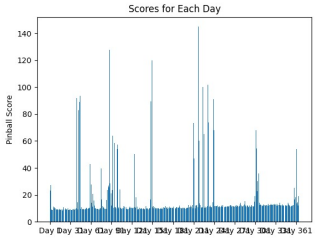
(b) MSE per day final model



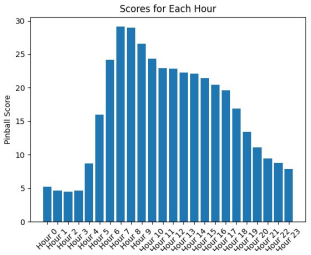
(c) RMSE by day of our final model



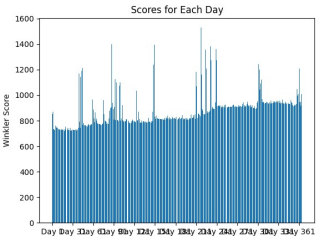
(d) sMAPE by day of our final model



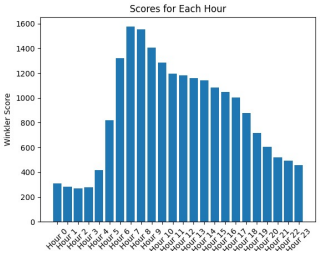
(e) Pinball score by day final model



(f) Pinball score by hour final model



(g) Winkler score by day final model



(h) Winkler score by hour final model

Figure 46: Final results of our model

6 References

[1] Baviera, R., & Messuti, G. (2023). Daily middle-term probabilistic forecasting of power consumption in North-East England.

[2] Baviera, R., & Azzone, M. (2020). Neural Network Middle-Term Probabilistic Forecasting of Daily Power Consumption.

[3] Kubler, R. (2024). Neural Networks For Periodic Functions.

[4] Hardle, W., & Simar, L. (2014). Applied Multivariate Statistical Analysis.

[5] Jensen, V., Bianchi, F., & Anfinsen, S. (2022). Ensemble Conformalized Quantile Regression for Probabilistic Time Series Forecasting.

[6] Abumohsen, M., Owda, A., & Owda, M. (2023). Electrical Load Forecasting Using LSTM, GRU, and RNN Algorithms

[7] Schneider, S., & Schneider, S. (2010). Power Spot Price Models with negative Prices

[8] Ziel, F. (2018). Modeling public holidays in load forecasting: a German case study.

[9] Zaffran M. et al, Adaptive Conformal Predictions for Time Series, PMLR22

[10] Matteucci M. et al, Bayesian deep learning based method for probabilistic forecast of day-ahead electricity prices

[11] Ahsha N. Tribble Norman (2003). The relationship between weather variables and electricity demand to improve short-term load forecasting

7 Appendix

In this final section, we provide guidelines on the structure of the code and how to execute the various implementations described earlier. The code was developed using PyCharm, enabling the execution with a single run. Several customization options are embedded throughout the code.

- **Data Exploration Plots**: These plots are automatically generated upon runtime.
- **Core Execution in** `main_recalibration`:
  - **Json Files** (Line 49): Choose the Json files for Preprocessing and Calibration options or Neural Network model parameters. Modifications are allowed.
  - **Data Preprocessing**: In the Json file choose the Data Preprocessing techinques to use.
    - **preprocess**: Indicates the preprocessing technique used. Values are "MinMaxScaler" ,"RobustScaler" ,"StandarScaler", "QuantileTransformer".
    - **analysis**: Specifies the type of deseasonalization performed. Values can be "Glm" (Generalized Linear Model) or "Fourier" (Fourier Transform).
    - **per_min**: Minimum period used for Fourier transform.
    - **per_max**: Maximum period used for Fourier transform.
    - **freq**: Specifies the top highest frequencies chosen.
    - **pca**: A boolean value indicating whether PCA (Principal Component Analysis) is applied.
    - **periodic**: A boolean value indicating whether periodic transformation of time variables is considered. If set to "false", a dummy variable approach is used.
    - **holiday**: Specifies the treatment of holidays. "Ratio" for ratio method, "Dummy" to consider them as weekends (Not compatible with periodic = true).
  - **Hyperparameters for MIXED**: In the Json file choose hyperparameters to use. Ambigous notation is explained below
    - **dense size**: Specifies the size of all dense layers in the lower part of the structure exept for the last .
    - **dense 2 final size**: Specifies the size of the last dense layer in the lower part of the structure.
    - **final dense size**: Specifies the size of the dense layers at the end of the structure.
    - **n encoder layers**: Specifies the number of convolutional layers.
  - **Neural Network Parameters** (Line 56): Select either "load_tuned" or "optuna_tuner" for importing NN parameters or performing cross-validation with optuna.
  - **Holiday re-definition** (Line 63): Specify whether to re-define the holidays.
  - **Target Variable Transformation** (Line 66): Specify the desired transformation ("None", "log", "asinh") and whether to standardize variables before transformation (available only in the "asinh" case).
  - **De-seasonalization**
    - **Fourier analysis Implementation**: Located in `data_utils`.
    - **One Hot Encoding Implementation** (Line 89 in `GLM`): Choose "LASSO", "Elastic-Net", or "OLS".
  - **PCA Transformation**:
    - **PCA variables** (Line 17 in `VariableSelection`): Set `flag` to "FUTU" (suggested) to perform PCA on **FUTU**__ variables, "ALL" to perform PCA on all variables.

    - **Dimensionality Reduction** (Line 17 in `VariableSelection`): Use `LASSO_selection` to reduce dimensions based on principal components. Set the β threshold (line 99).
- **Recalibration**: In `PrTSF_Recalib_tools` (line 812) choose the recalibration frequency (days)
- **Conformal Predictions** (Line 70):
    - Enable by setting `exec_CP` to True.
    - Set the number of calibration samples (Line 73)
    - Specify the type of conformal prediction (Line 82).
    - For AGACI implementation Python connects to R trough a plugin. On line 412 of `conformal_prediction` file directory has to be specified.
- Choose whether to print other probabilistic or point scores (Lines 85 and 88).