information

MDPI

Article

Forecasting Highly Volatile Time Series: An Approach Based on Encoder–Only Transformers

Adrian-Valentin Boicea [1,*] and Mihai-Stelian Munteanu [2]

[1] Electrical Power Systems Department, National University of Science and Technology Politehnica Bucharest, 060042 Bucharest, Romania
[2] Department of Electrotechnics and Measurements, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania
* Correspondence: valentin.boicea@upb.ro

Abstract

High-precision time-series forecasting allows companies to better allocate resources, improve their competitiveness, and increase revenues. In most real-world cases, however, time series are highly volatile and cannot be used for forecasting together with classical statistical methods, which usually yield errors of around 30% or even more. Thus, the goal of this work is to present an approach to obtaining day-ahead forecasts of electricity consumption based on such volatile time series, along with data preprocessing for volatility attenuation. For a thorough understanding, predictions were computed using various methods based on either Artificial Intelligence or purely statistical algorithms. The architectures based on the Transformer were optimized through Brute Force, while the N-BEATS architecture was optimized with Brute Force and OPTUNA because of the highly stochastic nature of the time series. The best method was based on an Encoder-only Transformer, which resulted in an approximate prediction error of 11.63%—far below the error of about 30% usually accepted in current practice. In addition, a procedure was developed to determine the maximum theoretical Pearson Correlation Coefficient between forecast and actual power demand.

Keywords: artificial intelligence; big data; data preprocessing; forecast; Jarque–Bera test; Kolmogorov–Smirnov test; noise; time series; transformer; volatility

1. Introduction

Energy serves as the basis of all processes in the economy. However, growing concerns related to environmental pollution and questions about how to mitigate it have led to the increased adoption of renewable sources, which are highly intermittent [1]. The adoption of renewable sources marked the appearance of prosumers, further complicating the situation, since the majority are fed by photovoltaic (PV) panels and usually batteries. The lack of inertial masses resulting from the use of PV panels threatened to compromise the balance of national power systems and even cause blackouts. To keep the entire power system running under these new conditions, new algorithms and methods had to be developed that could analyze huge quantities of data, called Big Data, in a reasonable amount of time, with the goal of maximizing the system's operational efficiency. Among these applications, those dedicated to decision and control play a very important role. In many situations, decision and control problems are solved through classification and forecasting. Initially, forecasting was carried out using purely statistical algorithms. However, with the emergence of Big

Check for updates

Academic Editor: Francesco Camastra

Received: 26 November 2025
Revised: 5 January 2026
Accepted: 13 January 2026
Published: 23 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Data generated by smart grids or microgrids fed by distributed renewable sources, this became impossible. This is because what makes a grid smart is actually the quantity of data it produces, along with the corresponding processing power. At present, the sampling standard at the national power system level is 15 min, but smart meters can benefit from sample rates of even 1 s [2]. Under these circumstances, high-precision forecasts can no longer be made with statistics alone. Thus, Artificial Intelligence (AI)-based methods began to be adopted and implemented. However, we are still in the beginning stages, and most of these applications are still based exclusively on non-generative AI (also called predictive AI) models, such as neural networks. Then, another need arose: generating new data from existing data. In this way, generative AI also began to be used in power engineering. Its areas of application are mainly data imputation or the generation of text containing various operational procedures and recommendations derived from existing data, and it is less commonly used for forecasting. Thus, the aim of this article is to introduce an Encoder-only Transformer architecture for day-ahead forecasting of power demand from a highly volatile time series. This time series possesses two components: the first one is fixed and is represented by what is being sold directly to the final users, while the second varies randomly and is traded through the stock market. This Transformer will not be used in the classical sense of generative AI, but only for forecasting. It has been chosen for this task because this architecture is very resilient to volatility, as will be demonstrated below. This approach, based on Encoder-only Transformers, introduces two important novelties—issues that have been insufficiently addressed until now but play a major role in predicting highly volatile time series, which are extremely common in the trading sector. The first novelty is represented by the systematic analysis of the noise, its variability, and how it can be attenuated using straightforward procedures, with the goal of efficiently training the Transformer. The second one relates to the performance evaluation of the forecaster based on the theoretical ceiling of the Pearson Correlation Coefficient. The rest of this article is structured as follows: Section 2 contains the most important stages, in chronological order, of adapting the Transformer for time-series prediction. Section 3 presents the dataset used, as well as various statistical tests for preprocessing. Section 4 introduces the results and compares the proposed approach to other relevant forecasting methods. Section 5 is dedicated to discussing the results, as well as presenting the challenges and opportunities of using AI architectures and various digital signal processing algorithms to solve real power engineering problems, while Section 6 presents the conclusions and future research pertaining to foundation models and Large Language Models (LLMs).

2. From Words to Numbers: Applying Transformers to Time-Series Forecasting

Originally developed for sequence-to-sequence modeling in natural language processing, the Transformer architecture employs a Decoder designed for autoregressive text generation. It was introduced by Vaswani et al. in [3], with the aim of facilitating real-time translation between various languages using the classical architecture, containing an Encoder and a Decoder.

The adaptation of the Transformer for time-series forecasting is described in [4]. The authors' improvements to the Transformer's initial configuration were related to the memory bottleneck and weak context modeling:

- The memory bottleneck arises from the nature of time series. The standard self-attention mechanism presents quadratic complexity $O(L^2)$ with respect to the input sequence length L. For a long time series, this quickly becomes infeasible in terms of both memory and computation.

- Weak context modeling is called "locality-agnostics" by the authors. Time series often have strong local dependencies (short-term correlations). Standard self-attention treats all positions equally, which makes it less effective at capturing these local patterns.

The solution to these two problems was the implementation of a convolutional self-attention mechanism that generates queries and keys [4] using causal convolution, which makes context modeling easier to integrate into this mechanism. The second step of this solution involves the development of a LogSparse Transformer, featuring a complexity of $O(L(log\ L)^2)$, aimed at improving the forecasting accuracy, particularly for time series with high sampling rates.

It is important to mention that a high noise level in the time series affects the accuracy of the canonical Transformer's predictions, especially when it surpasses 50% of the entire signal. This is because scalar-product attention treats all input positions equally, so when noise dominates, the attention scores correlate with noise patterns, not with the true temporal relationships. This violates the assumption that temporal proximity or local shape is significant, and the forecast becomes less reliable [4].

In [5], a new architecture is introduced, called the Time Fusion Transformer (TFT), designed for interpretable multi-horizon forecasting. Besides multi-horizon forecasting, handling heterogeneous inputs and interpretability are also very important for time-series analysis. Thus, static features (for example, customer ID or region), known future inputs (holidays or planned prices), and observed historical inputs (like past sales) influence the forecasting precision. Traditional models (Autoregressive Integrated Moving Average—ARIMA; Long Short-Term Memory—LSTM; Vanilla Transformer, as mentioned in [3]) typically fail to provide both high accuracy and interpretability. In order to overcome these disadvantages, the TFT uses temporal processing with attention and LSTM, as well as quantile outputs and other components, as described in [6]. At the same time, the gating mechanisms of the TFT prevent overfitting by letting parts of the network skip unnecessary computations. Variable selection networks provide interpretability by assigning importance weights to features. Static covariate encoders encode time-invariant features (e.g., category, region) and inject this information throughout the model. Temporal processing captures local sequential dependencies through LSTM layers, while multi-head attention focuses on long-term dependencies and relationships over time. Finally, the quantile outputs generate prediction intervals through simultaneous forecasts of different percentiles at each time step [5].

When the noise level in the time series used for prediction with the TFT exceeds 50%, autocorrelation maps are weaker because the noise affects temporal dependencies, reducing the effective information for the LSTM Encoder. As a result, the forecasts will fluctuate around the mean, signifying higher residuals. At the same time, the attention maps will be flattened because the salient time points will be obscured, causing less efficient learning of long-term dependencies. Another phenomenon that appears is gate saturation, which forces the TFT to become less reactive to subtle variations in the signal. Ultimately, quantile widening makes the probabilistic forecasts less reliable. All this can increase the Mean Absolute Percentage Error (MAPE) by approximately 25–40% or even more, depending on the implementation.

As mentioned before, Vanilla Transformers are generally powerful but present quadratic complexity in terms of memory and computation with respect to the input sequence length L, making them impractical for processing very long sequences (e.g., long text, audio, or time series). The Reformer architecture, described in [7], addresses this by introducing two main efficiency innovations that allow Transformers to scale to much longer sequences without loss of modeling power. These are Locality-Sensitive Hashing (LSH) Attention and Reversible Residual Layers. Here, keys and queries are mapped into

buckets using a hash function, according to the principle introduced in [8]. Attention is computed only within the same bucket, drastically reducing memory and computation. Ultimately, the complexity is reduced from $O(L^2)$ to $O(L\ logL)$, where L represents the length of the time series. In general, standard residual connections store all intermediate activations for backpropagation, consuming memory. The Reformer uses reversible layers, where inputs can be recomputed from outputs during backpropagation, saving memory.

When the Reformer is used for time-series forecasting with a noise level higher than 50%, the attention locality becomes unstable, and global context learning degrades faster than for the canonical Transformer. Speed and memory are not influenced by random variations in the signal. However, the forecasting error can be severely affected because the noisy tokens will dominate the local buckets, and the attention weights become less relevant. In the end, this leads to overfitting and, of course, to higher forecasting errors. This degradation in accuracy is in the same range as in the TFT case.

In [9], a new type of Transformer architecture, called Informer, is introduced. The goal is again to address the limitations of the initial architecture when used for text processing. Thus, the main objective was to maintain the modeling power of the self-attention mechanism while reducing the computational and memory burdens, thereby making the prediction more efficient for long output horizons. The most important contributions were related to the following:

- ProbSparse self-attention mechanism: A probabilistic sparse attention mechanism (ProbSparse) is implemented that selects only the "dominant" queries to compute full attention while treating the other queries approximately. This reduces both time and memory complexity of $O(L^2)$, achieving $O(L\ logL)$, where L denotes the time-series length. The method leverages the observation that, in attention distributions, many query–key pairs contribute very little, and thus, focusing on the "important" ones is sufficient.
- Self-attention distilling mechanism (layer pooling): A distillation process is developed between attention layers, where intermediate representations are pooled in order to shrink the computational graph with increasing depth. This helps mitigate memory usage for very long time series.
- Generative-style Decoder (one-shot decoding): Instead of autoregressive decoding (step by step), a "generative decoding" approach is adopted. This outputs the full forecast horizon in one forward pass, significantly speeding up the inference when forecasting long sequences.

After Informer, which focused on efficiency for long sequences, researchers identified two other challenges. Transformer attention is not always the best for capturing periodic or seasonal dependencies in the input sequence.

Furthermore, pure attention may overfit noise and fail to generalize for long-term forecasting.

In time-series forecasting with a noise level greater than 50%, Informer generally performs well. The situation changes dramatically when the signal must not be denoised, like in trading operations. ProbSparse attention ensures robust filtering, but if this is not needed, this mechanism will automatically lead to high forecasting errors. The learning of long-term dependencies is partially preserved, which is a positive aspect since the long-term trend remains detectable. When the noise dominates, long-term modeling becomes severely affected. In conclusion, the overall robustness of Informer can be considered good as long as the noise component does not need to be filtered out. This is why, in this specific situation, where a prediction must be made using a highly volatile time series, Informer is actually not appropriate, because the MAE or MAPE degradation can surpass 40%.

To address some of the issues with Informer, Autoformer was introduced in [10]. This is based on two new concepts: time-series decomposition and a novel autocorrelation mechanism.

The series decomposition block splits each input sequence into trend and seasonal parts. One of the most important advantages in this case is that Autoformer will process these components separately from one another, allowing better modeling of smooth long-term trends and periodic seasonal fluctuations. This method reduces noise and improves stability for long forecasting horizons. At the same time, instead of computing attention across all positions, the autocorrelation mechanism focuses on periodic dependencies in the time series so that the model can understand repeating patterns (e.g., daily or weekly cycles in energy data). Another feature of this architecture is progressive decomposition, where the principal and seasonal parts of the signal are refined at each time step, resulting in improved forecasting precision.

When it comes to highly volatile time-series forecasting, although Autoformer performs better than Reformer, the problems that emerge when the random component does not need to be filtered out are very similar to those described above for Informer. This is because Autoformer isolates high-frequency components. Time-series decomposition then extracts the volatile trend. However, this intrinsically leads to high forecasting errors. At the same time, the autocorrelation attention mechanism focuses on structured patterns while ignoring the noise, causing even further degradation of the prediction accuracy.

In [11], a new Transformer architecture is introduced, called Frequency Enhanced Decomposed Transformer (FEDFormer). The challenges related to quadratic complexity in relation to the length of the time series, along with the difficulty in modeling global patterns like trends and seasonality, are once more addressed and solved differently. In this case, the contributions reside in Frequency-Enhanced Decomposition (FED) and in Frequency-Enhanced Attention (FEA). FED decomposes the input into seasonal and long-term parts, utilizing frequency-domain information to enhance the decomposition process, allowing the model to better map the underlying patterns. FEA operates in the frequency domain, enabling the model to concentrate on the most important frequency components. At the same time, it also enhances the ability of this Transformer type to capture long-term dependencies and periodic patterns. By integrating these components, FEDformer reduces the memory cost to linear growth in the sequence length, improving efficiency over conventional Transformers. Empirical results show that FEDformer outperforms leading methods, reducing prediction errors by more than 10% in both multivariate and univariate series, as described in [11].

As mentioned previously, the prediction of highly volatile time series, where the random component contains actionable information and does not need to be removed, is also difficult with FEDFormer. This is because, in this case as well, the high-frequency component is isolated, while the low-frequency component, denoting the long-term trend, is retained. Unfortunately, stochastic fluctuations that do not repeat or align with the dominant frequency components are effectively down-weighted. Thus, FEDformer treats unstructured micro-fluctuations as useless noise. Singular events or sudden volatility are not modeled efficiently, so critical signals are lost, leading to high forecasting errors. Because actionable signals can be present in high-frequency or non-repeating variations, the model is forced to underreact to non-regular variations due to its design, thus delaying trading decisions.

In [12], a Transformer architecture called PatchTST (Patch Time Series Transformer) is introduced. PatchTST incorporates two key innovations: segmentation of the time series into patches and channel independence. This approach retains local semantic information and reduces the quadratic complexity of attention mechanisms, enabling the Transformer

to focus on longer histories. Channel independence refers to the fact that each channel possesses the same weights and representation for all series at the input, thus stimulating efficient learning and reducing the computational burden. Empirical results demonstrate that PatchTST significantly improves long-term forecasting accuracy compared to the best Transformer-based models. Additionally, this architecture performs well in self-supervised pre-training tasks, outperforming supervised training on large datasets. Transfer learning experiments further confirm their effectiveness across different datasets.

Using PatchTST for predicting highly volatile time series in which the high-frequency component contains actionable information is again problematic. This is because the patch representation averages the short-term spikes and micro-fluctuations, reducing their magnitude. On the other hand, channel-independent attention does not recover the information lost in patching, while the Transformer Encoder captures dependencies at the patch level, ignoring fine-grained point-to-point variations. In conclusion, PatchTST tends to smooth the high frequency and micro-fluctuations, making the model slower to react or insensitive to short-term trading opportunities.

After these variations in architecture were introduced, different combinations between neural networks and Transformer models were developed for time-series prediction. For example, hybrid Transformer–CNN (Convolutional Neural Networks) models were implemented, integrating 1D CNNs with Transformers to enhance the model's ability to learn long-term and short-term patterns in multivariate time series, as presented in [13]. In [14], a hybrid CNN–Transformer architecture is proposed, effectively modeling both long-term and short-term variations simultaneously.

The architecture described in [13] performs much better than everything else presented so far. This is because the CNN feature extractor models micro-fluctuations and short-term volatility more efficiently. At the same time, locally correlated spikes that may represent trading signals are preserved. The Transformer Attention Mechanism correlates the micro-fluctuations with the broader context, so even if the unstructured noise is removed, small fluctuations containing actionable information are kept. Given the structure's complexity, the model might become saturated with noise, and in cases where the volatility surpasses 50%, the forecasting errors can become high.

The architecture in [14], compared to that in [13], is theoretically more appropriate for trading operations, given the structure of the model. The CNN and Transformer work in parallel, with the CNN extracting the features and the Transformer capturing the macro-context. Afterwards, a fusion mechanism integrates the two. This is achieved through gating or attention-based fusion that dynamically weights the CNN relative to the Transformer contributions, depending on context. Although it can perform better on highly volatile time series, this architecture also presents the same problem as the previous one: it can quickly become saturated with noise, especially when more than 50% of the time-series variation is attributable to noise.

Even more challenges appear when the time series used in forecasting contains a certain amount of generated energy that is traded through the stock market, while the rest is consumed directly by the final clients. Thus, the goal of this work is to predict the energy demand in such a situation using a Transformer architecture, which usually delivers results with very high precision, given its efficient training and its resilience when handling volatile time series. For a better understanding of how the proposed method compares to the current state of the art, please refer to Appendix A, Table A1. It should be mentioned that to achieve high forecasting precision, the data needs to be analyzed and preprocessed, as described in Section 3.

3. Methodology

As mentioned in Section 1, the proposed forecasting methodology is based on a systematic noise analysis and a model performance evaluation that uses the maximum theoretical limit of the Pearson Correlation Coefficient. The systematic noise analysis explores how noise variability affects the variability of the entire time series, as well as the attenuation of this effect, in order to train the Encoder-only Transformer more efficiently. In fact, this analysis is intrinsically related to the second part of the study, pertaining to the Pearson correlation. Thus, the degree to which the time series can be predicted by the model in the ideal case is first estimated. For this purpose, the proposed approach considers the heteroscedastic or homoscedastic character of the noise. To the authors' knowledge, this combined solution for forecasting highly volatile time series is completely new.

The data used in this work originates from a Romanian energy producer that owns various PV plants across the country. As mentioned in the previous section, its clients utilize a share of this PV-generated electricity as well as electricity delivered by the main grid, while the rest is sold via OPCOM (Operatorul Pieței de Energie Electrică şi Gaze Naturale din România—The Electricity and Natural Gas Market Operator in Romania).

In other words, 100% of the generated electrical energy is sold.

The data provided spans from 1 January 2023 to 31 January 2024 and is sampled at 15-min intervals. The variation in energy demand in this interval is illustrated in Figure 1.

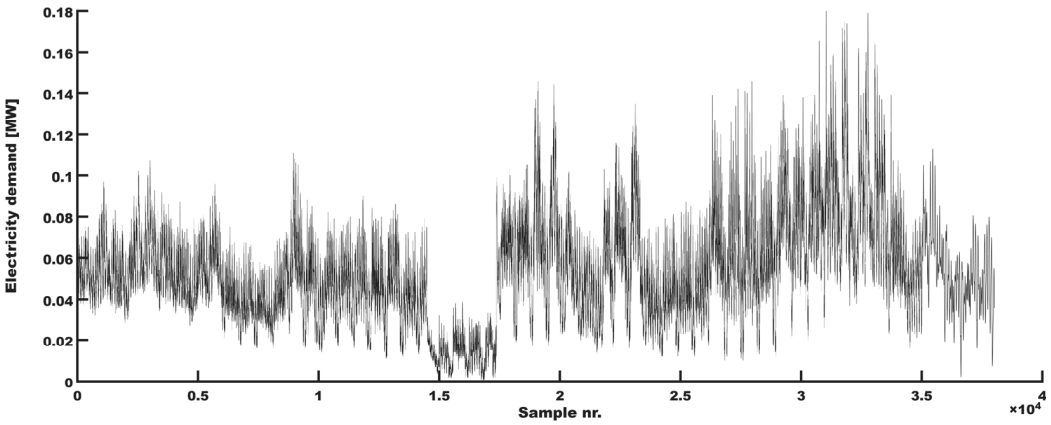


Figure 1. Power consumption between 1 January 2023 and 31 January 2024. The series contains 38,016 values.

The given time interval contains 38,016 samples. Given that this electricity demand includes both the consumption of the final clients and the share being traded through OPCOM, the time series is highly volatile. The forecasting models treat this volatility as noise, so from this point on, it will be referred to as such. Unfortunately, this component cannot be filtered out, as it is an integral part of the signal to be forecast, containing actionable information.

The basic statistical parameters of the series under study have been determined. The average is 0.0523 MW; the standard deviation, 0.0245 MW; the range, 0.1800 MW; the median, 0.05 MW; and the 25%, 50%, and 75% percentiles, 0.037 MW, 0.05 MW, and 0.065 MW.

The average and median are close to one another, which means that there are few outliers in the series, and the signal is centered around a stable level of approximately 0.05 MW. The range, being 3.44 times greater than the average, can be considered significant. Regarding the percentiles, one can observe that 50% of the values are found in the interval [0.037; 0.065] of length 0.028 MW, which is quite narrow. This means that the series has a central stable part and longer "tails" towards the extremes.

Table 1 synthesizes these parameters.

Table 1. Descriptive statistics of the time series under study.

Parameter	Value [MW]
Average	0.0523
Standard Deviation	0.0245
Range	0.1800
Median	0.05
Percentile	25%, 50%, 75%: [0.037 0.05 0.065]

Trend decomposition is performed using Singular Spectrum Analysis (SSA) to estimate the noise level in the signal. This approach is an efficient way to decompose a signal into its distinct parts, such as long-term and periodic (seasonal) trends, along with the residual (noise). SSA is a non-parametric spectral estimation technique that uses Singular Value Decomposition (SVD). In our specific case, similarly to the above-mentioned aspects, the components are the long-term and seasonal trends, and the rest is represented by noise. This method consists of four steps:

- The embedding (converting the initial time series into a multidimensional space by creating lagged copies and generating the trajectory matrix).
- SVD, which decomposes the trajectory matrix to extract the principal components of the time series, which can then be grouped to identify the principal trend, seasonal (oscillatory) components, and residual (noise).
- Grouping or component selection for trend extraction that emphasizes which components to keep, rather than combining them.
- Reconstruction (Hankelization), which ensures that the end result of the SSA will represent a one-dimensional time series.

The most important advantages of this method in trend decomposition are as follows:

- No predefined assumptions: Unlike polynomial or Moving Average filtering, SSA does not require predefined model assumptions.
- Efficient handling of nonlinear trends: It can effectively extract smooth and even nonlinear trends.
- Efficient separation of the trend from noise: By selecting the appropriate singular values, the periodic (seasonal) component can be separated from the trend.

The noise level in the signal is visualized in Figure 2.

The features of this decomposition, in terms of variability for each time-series component, are as follows: long-term trend, 2.1298×10^{-4}; seasonal trend, 1.3098×10^{-5}; noise variability, 3.2849×10^{-4}. More importantly, the contribution of noise variability to the variability of the entire signal is 54.75%.

Theoretical research in deep learning shows that machine learning models, particularly neural networks, tend to perform better when the input data are normalized and exhibit distributions close to Normality. Proper normalization can improve numerical stability and accelerate convergence during training. This behavior has been consistently reported across foundational works in the field [15–17]. Thus, this signal has to be tested to determine whether the data follows this statistical distribution or not. To this end, the Kolmogorov–Smirnov (KS) and Jarque–Bera (JB) tests were carried out [18,19].

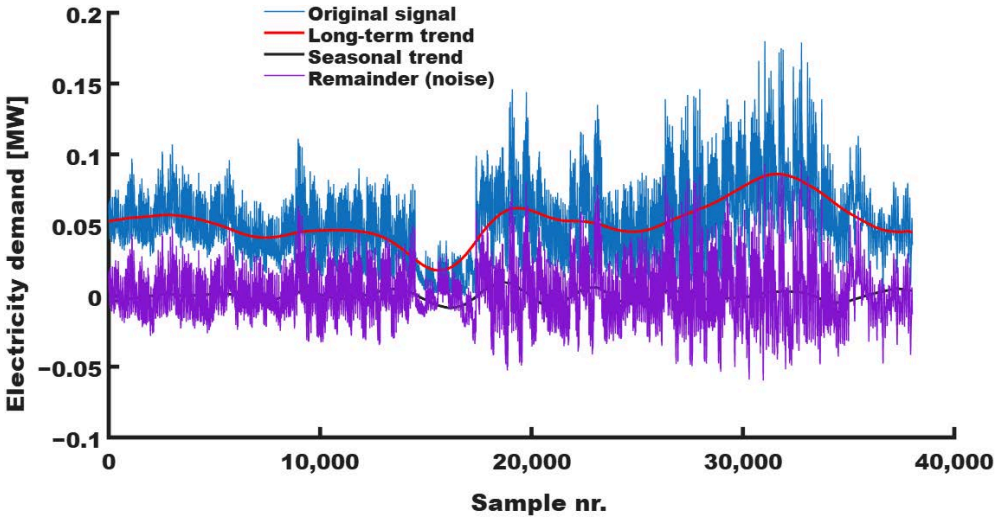


Figure 2. Trend decomposition of the original signal. The decomposition indicates that approximately 54.75% of the total variance is attributable to noise.

The KS test operates with the empirical cumulative distribution function (ECDF) of the time series and with the cumulative distribution function (CDF) of the desired theoretical distribution—in our case, the Normal Distribution. Whether the tested data follows the desired distribution or not is determined by comparing the two functions. Given the time series $X = \{x_1, x_2, \ldots, x_n\}$, the ECDF is defined as in (1):

$$F_e(x) = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathbf{1}(x) \tag{1}$$

where

n—the number of observations in the time series;

$\mathbf{1}(x)$—a characteristic function that can take the values of either 0 or 1.

For a Normal Distribution $N(\mu, \sigma^2)$, the theoretical CDF is given by (2):

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \int_{-\infty}^{x} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) dt \tag{2}$$

where

μ—the mean of the Normal Distribution;

σ^2—the variance.

If the population parameters μ and σ^2 are unknown, they are estimated from the sample as in (3) and (4):

$$\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i \tag{3}$$

$$\sigma^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (X_i - \mu)^2 \tag{4}$$

On the other hand, δ_n represents the greatest absolute deviation between $F_e(x)$ and the theoretical function $F(x)$, taking into account the whole time series, as in (5):

$$\delta_n = max\ |F_e(x) - F(x)| \tag{5}$$

The critical value for the KS test depends on the significance level α and on the size of the time series. This threshold α is usually set to 0.05. Thus, if δ_n is small enough and the

probability value is high, there is no strong evidence against Normality; when δ_n is large and the probability value is low, the time series is not normally distributed.

The JB test assesses Normality based on the third and fourth moments of the distribution, that is, Skewness and Kurtosis, respectively. If a dataset is normally distributed, its Skewness should be 0, and its Kurtosis should be 3. Given the same time series and mean and variance as in (3) and (4), the Skewness is defined as in (6):

$$S = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{X_i - \mu}{\sigma} \right)^3 \tag{6}$$

where

S—the Skewness;
n—the number of data points in the sequence;
X_i—the values in the time series;
μ—the mean of the Normal Distribution;
σ—the standard deviation.

The Kurtosis measures the "tails" of the distribution. Its formula can be seen in (7):

$$K = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{X_i - \mu}{\sigma} \right)^4 - 3 \tag{7}$$

where

K—the Kurtosis;
n—the number of observations in the time series;
X_i—the values in the time series;
μ—the mean of the Normal Distribution;
σ—the standard deviation.

For a Normal Distribution, $K = 0$ (because the Normal Distribution has a theoretical Kurtosis of 3).

The JB statistic is calculated in (8):

$$JB = \frac{n}{6} \cdot \left(S^2 + \frac{K^2}{4} \right) \tag{8}$$

where

S—the sample Skewness;
K—the Kurtosis of the distribution;
n—the sample size (number of observations in the time series).

If the time series follows a Normal Distribution, the JB statistic follows a Chi-square distribution with 2 degrees of freedom, according to (9) [19]:

$$JB \in \chi^2(2) \tag{9}$$

The probability is computed in (10):

$$p = 1 - F_{\chi^2}(JB, 2) \tag{10}$$

where $F_{\chi^2}(JB, 2)$ is the CDF of the Chi-square distribution with 2 degrees of freedom.

If the probability value is greater than the desired threshold, which is usually set at 0.05, the time series is considered to follow a Normal Distribution, and vice versa.

Practically, the KS test checks for deviations in the entire cumulative distribution function, while JB focuses on deviations in Skewness and Kurtosis. The JB test is more

relevant when the Normality deviations are due to these two specific moments (Skewness and Kurtosis), while KS is more general in character.

In our specific case, both tests rejected the hypothesis that the data are normally distributed. For a visualization of the data, refer to Figure 3.

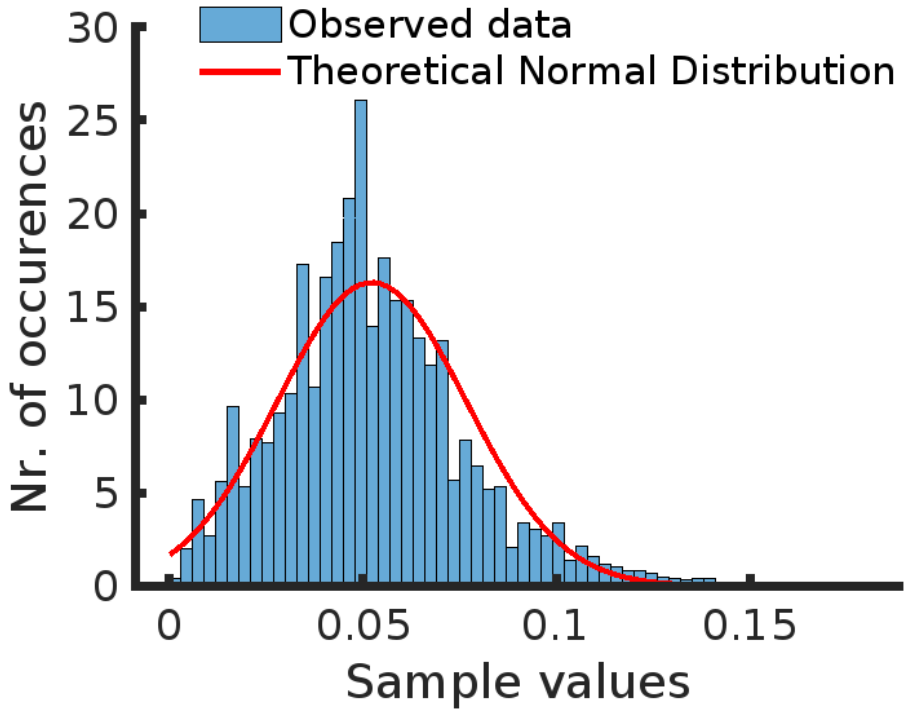


Figure 3. Comparison between empirical data distribution and its Theoretical Normal Distribution. The empirical data distribution does not fit well with the Normal Distribution.

The power of the tests was 1. The *p*-value of KS was 0, whereas the *p*-value of JB was 0.001. The Kurtosis in this case was 4.2525, which is greater than 3, meaning that the time series is very noisy.

In Figure 3, one can also observe that the histogram does not fit very well to the curve of the Theoretical Normal Distribution. This was obtained using the mean and the standard deviation of the entire signal, 0.0523 and 0.0245, respectively. The fact that both statistical tests rejected the Normal Distribution hypothesis indicates that the noise variability makes a large contribution to the variability of the whole time series. Given this, a method must be used to attenuate the noise. In this case, the Moving Average was chosen. This method is very simple to implement and usually delivers good results because the errors introduced to the newly averaged time series permit, theoretically, good forecasting precision.

The formula for the Moving Average is presented in (11):

$$MA_t = \frac{1}{N} \cdot \sum_{i=0}^{N-1} x_{t-i} \tag{11}$$

where

x_t—the value of the time series at time t;

N—the window size (number of data points considered for the average).

In other words, summation is performed over the past N observations.

For this specific situation, the chosen window was 24 h, taking into account that the signal was sampled every 15 min.

In MATLAB®, online version R2025b, this type of averaging is obtained through zero padding for the first few values, and then the actual values in the series are used. The newly obtained vector has exactly the same length as the initial one and the same trend components. Only the noise will be attenuated.

The results of this operation are shown in Figure 4.

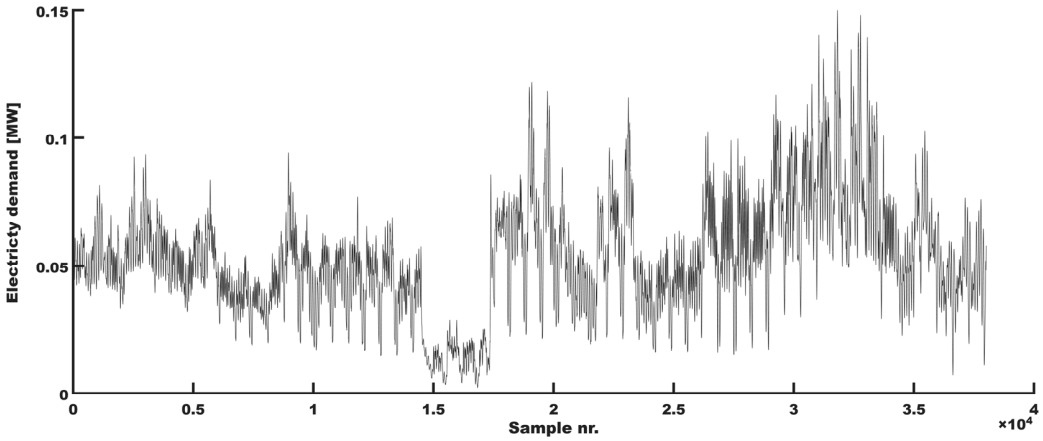


Figure 4. Time series after noise attenuation with the Moving Average. The "tails" of the distribution are somewhat shorter than in Figure 1.

Visually, Figure 4 does not significantly differ from Figure 1. It is also relevant to observe the comparison between the newly obtained empirical distribution and the Theoretical Normal Distribution after averaging (see Figure 5).

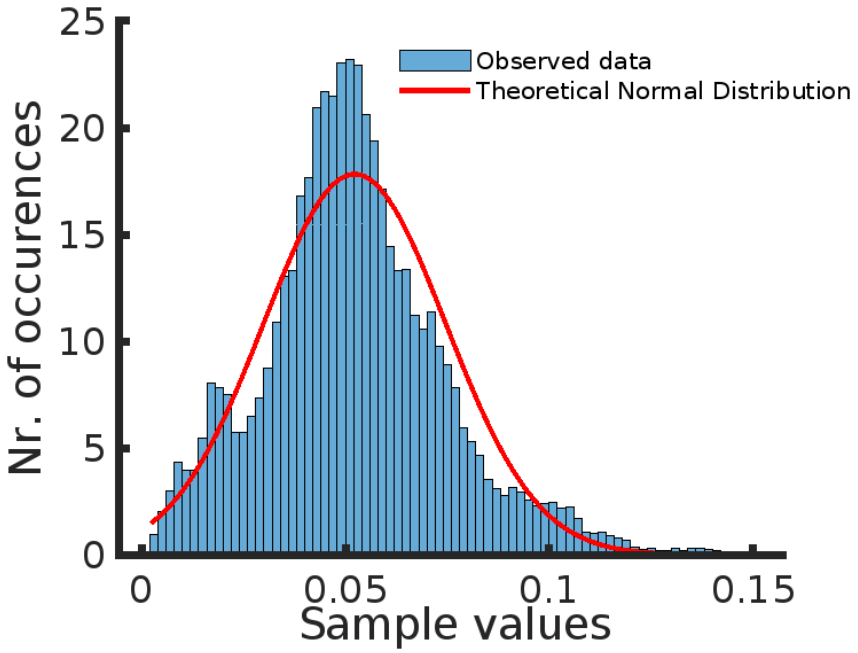


Figure 5. Comparison between empirical data distribution and its Theoretical Normal Distribution after applying the Moving Average. The empirical data distribution fits better with the Normal Distribution.

Once again, both of the above-mentioned tests demonstrate that the data do not follow a Normal Distribution. The power of the tests and the *p*-values were identical to those in the previous case. In spite of this, as depicted in Figure 5, the Theoretical Normal Distribution fits better with the histogram this time, with a mean of 0.0523 MW and a standard deviation of 0.0224 MW. The Kurtosis here is 3.8255, meaning that the noise level is lower than in the previous situation. In addition, the standard deviation is slightly lower in this case than in the previous one, which serves as additional proof, besides the visual results and the Kurtosis, that the noise has been successfully attenuated. The contribution of noise variability to the variability of the entire signal is now 45.79%.

Because there are a few important differences between the averaged and non-averaged signals, the error introduced by the Moving Average will not be too high. As mentioned above, this will help the Transformer to provide a precise forecast, as will be seen in the next section. Subsequently, all the forecasting models were trained on the averaged signal, which is closer to the Normal Distribution, and the results were compared to the original, non-averaged signal.

4. Results and Comparison to Other Methods

Initially, Transformers were implemented for natural language processing applications, and as mentioned in Section 2, they have been efficiently adapted for time-series prediction, given their versatility in learning long-term patterns that frequently appear in time series. The classical Transformer is actually a sequence-to-sequence (seq2seq) neural network architecture, introduced in [3], designed to model dependencies in sequential data without recurrence or convolution. The key concepts used in the Transformer for time-series prediction are the Encoder, the self-attention mechanism, and the Decoder. Before entering the Encoder, each input element is represented as a vector and has a positional encoding added to it (so that the model can process the order). The data is passed through stacked layers, each containing:

- Multi-Head Self-Attention, which lets each position attend to all other positions in the sequence;
- A Feedforward Network that applies nonlinear transformations to enrich the representations;
- Residual Connections and a Normalization Layer that stabilize and speed up the training.

The results will be a set of contextualized hidden representations for the entire input sequence, where each value in the time series has an ordered position. These representations are then sent either directly to the Decoder or to the prediction head, depending on the adopted architecture.

Because Transformers do not possess a sense of order, positional encodings are added to input embeddings to provide a sense of chronology.

On the other hand, the role of the Decoder is to generate the prediction stepwise. This is conditioned simultaneously by:

- The Encoder output embeddings (the context is taken from the input sequence);
- The previously generated outputs (autoregressive generation).

In general, a Decoder contains a Feedforward Network with a Normalization Layer, a Multi-Head Causal Attention Mechanism, and a Cross-Attention Mechanism.

Thus, within the Decoder, the output sequence generated so far is processed, and further, a mask is used so that the model cannot "cheat" by looking at future outputs (this is very important in autoregressive tasks).

When the Encoder processes the input sequence (say, past time-series values), it produces a set of hidden states (contextual representations). Each hidden state corresponds to one time step in the input and is enriched with information from the whole sequence, thanks to the well-known self-attention mechanism described in (12) and introduced by Vaswani et al. in [3]:

$$Attention(Qr, Ky, Va) = softmax\left(\frac{Qr \cdot Ky^T}{\sqrt{dim_k}}\right) \cdot Va \quad (12)$$

where

Qr—queries that come from the Decoder's current hidden state (the state of what is being generated);

Ky^T—the transposed matrix of the keys;

Va—values;

dim_k—the key vector dimension.

When the Decoder is generating the output sequence (e.g., the future values), it will use the previous values generated so far and those values in the input that are relevant to the forecast. In other words, the Decoder applies the attention mechanism over the Encoder outputs [3], where a combination of weights reflects which input time steps are most relevant to predicting the current output.

The Encoder takes the input sequence (e.g., words in NLP (natural language processing), or past time-series values, as in this case) and produces contextualized representations of each time step.

In contrast to Long Short-Term Memory (LSTM) or Recurrent Neural Networks (RNNs), Transformers do not perform sequential processing. This allows for parallel computation and better long-range modeling.

For forecasting, many models use just the Encoder (e.g., Neural Basis Expansion Analysis for Time Series—N-BEATS) or Encoder–Decoder structures (as in NLP).

In this particular case, the architecture used contains just the Encoder. This kind of architecture was adopted for two reasons. First, the normal Transformer, containing both an Encoder and a Decoder, becomes saturated with noise and does not perform well on this noisy series, since it becomes prone to overfitting. Second, less complex architectures, like those containing just a Decoder, are unable to efficiently model the input sequence. Usually, the Decoder works using the previous output sequence. If there is no Encoder, the Decoder will lack the global context of the input series, and its latent embeddings will not be modeled. As a result, the complex features of the noisy time series will not be captured, and the forecast will not be accurate. In terms of data partitioning, the above series of 38,016 samples corresponds to 396 days. Using sliding windows, 2-day contexts and 1-day-ahead forecasts were generated. Thus, the model effectively used 395 days of context (the last day in the series can never represent an input for prediction) and 394 days for forecasts (the first 2 days in the series can never serve as forecasts because they lack previous context). As will be seen below, no validation set was used. This decision was made because of the high volatility of the series and based on the multiple tests that were carried out. The results thus obtained are then compared with the Vanilla Transformer, a Transformer without an Encoder, N-BEATS, and Prophet and Holt–Winters (Exponential Smoothing), as well as other algorithms, as shown in Table 2. All of these methods were chosen for comparison because of their high resilience in forecasting volatile time series.

In Table 2, the proposed method's results are in bold because it achieved the best performance among the compared methods.

Table 2. Obtained results and comparisons to other methods.

Method	MAPE [%]	MAE [MW]	MSE [MW]	RMSE [MW]	R^2 [%]	Pearson's Correlation Coefficient	Directional Accuracy [%]
ARIMA	20.5	0.0175	0.0004	0.0207	3.2110	n/a	50.26
SARIMA	29.92	0.0256	0.0010	0.0322	2.4482	n/a	55.79
Holt–Winters	28.46	0.0231	0.0006	0.0253	2.9932	0.528	57.89
Prophet	12.58	0.0102	0.0001	0.0130	7.8835	0.39	76.84
LSTM (optimized with Brute Force)	28.73	0.0245	0.0009	0.0307	2.9981	n/a	52.63
CNN (optimized with Brute Force)	32.57	0.0276	0.0011	0.0331	1.9950	n/a	63.16
GRU (optimized with Brute Force)	26.77	0.0234	0.0008	0.0292	2.0112	n/a	57.89
SVR (optimized with Brute Force)	16.49	0.0138	0.0003	0.0165	0.7610	0.1576	68.42
Random Forest (optimized with Brute Force)	28.68	0.0243	0.0009	0.0293	0.210	n/a	50.53
XGBoost (optimized with Brute Force)	29.86	0.0252	0.0009	0.0293	0.214	n/a	48.42
DeepAR (optimized with Brute Force)	23.85	0.0186	0.0006	0.0244	1.112	n/a	44.21
N-BEATS (optimized with Brute Force)	26.54	0.022	0.0008	0.0287	1.9677	n/a	61.05
N-BEATS (optimized with Optuna)	35.99	0.03	0.0014	0.0386	0.5512	n/a	46.32
Vanilla Transformer (optimized with Brute Force)	21.37	0.0182	0.0004	0.0216	5.9102	n/a	50.42
Decoder-only Transformer (optimized with Brute Force)	33.8100	0.0289	0.0010	0.0314	2.4251	n/a	50.51
Encoder-only Transformer (optimized with Brute Force)	**11.63**	**0.0099**	**0.0001**	**0.0130**	**35.6479**	**0.53**	**77.89**

The AI architectures were trained with ADAM, and the MAE (Mean Absolute Error) was minimized. The MAE is calculated as in (14). The results were compared using MAE, MAPE, MSE (Mean Squared Error), RMSE (Root Mean Square Error), R^2, Pearson Correlation Coefficient, and Directional Accuracy (DA), defined in (13) through (19).

$$MAPE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \cdot 100 \tag{13}$$

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{14}$$

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{15}$$

$$RMSE = \left[\frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right]^{\frac{1}{2}} \tag{16}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2} \tag{17}$$

$$DA = \frac{1}{n-1} \cdot \sum_{t=2}^{n} Indic[(y_t - y_{t-1}) \cdot (\hat{y}_t - \hat{y}_{t-1}) > 0] \tag{18}$$

In (13) through (18),
n—the total number of observations;
y_i—the actual (true) value;

$\hat{y}_i$—the forecasted value;
$\overline{y}_i$—the average of the real values;
Indic—an indicator function that equals 1 if the expression is true;
y_t—the real (actual) value at time t;
y_{t-1}—the real (actual) value at time $t-1$;
$\hat{y}_t$—the predicted value at time t;
$\hat{y}_{t-1}$—the predicted value at time $t-1$.

The Pearson Correlation Coefficient (Pearson's r) is a statistical parameter that represents the linear interdependency between two variables.

This coefficient takes values in the interval $[-1; +1]$:

- $r = -1$ means that the two variables are negatively correlated.
- $r = 0$ means that the variables are not correlated (though other types of nonlinear relationships may still exist).
- $r = +1$ means that the two variables are positively correlated.

The formula for the Pearson Correlation Coefficient for two variables A and B is described in (19):

$$r_{X,Y} = \frac{Cov(A,B)}{\sigma_A \sigma_B} \tag{19}$$

where

$Cov(A,B)$—the covariance of A and B;
σ_A—the standard deviation of A;
σ_B—the standard deviation of B.

The expanded form of (19) can be seen in (20).

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - m_a)\cdot(b_i - m_b)}{\sqrt{\sum_{i=1}^{n}(a_i - m_a)^2}\cdot\sqrt{\sum_{i=1}^{n}(b_i - m_b)^2}} \tag{20}$$

where

m_a—the average value of A;
m_b—the average value of B.

In our case, if the model works perfectly, meaning that it can forecast the entire useful signal (that is, principal and seasonal trends together), the maximum Pearson Correlation Coefficient that can be achieved will be approximately 67.27% (see (27)).

A Breusch–Pagan F (BP–F) statistical test was carried out to determine whether the noise is additive or multiplicative. The test revealed the latter, which means that the noise in the model strongly varies with the level of the signal. Higher or lower predicted values are associated with systematically larger or smaller noise values, so the noise is heteroscedastic rather than uniform. The values obtained from this test were *LM stat* = 1551.7376, *LM p-val* = 0, *F stat* = 1623.5474, and *F p-val* = 0. The first value, denoting the Lagrange Multiplier (*LM*) statistic, measures how strongly the noise variance depends on the useful signal. A high value, as in this case, indicates heteroscedasticity. The second value, denoting the probability of observing the LM statistic under the null hypothesis of homoscedasticity, is 0, so this hypothesis is rejected. The third value represents an alternative version of the LM statistic, based on the F-distribution. It also tests the heteroscedasticity, and a high value confirms the multiplicative character of the noise. Finally, the fourth value denotes the probability of observing the F-statistic under the null hypothesis of homoscedasticity. Since the value is 0, this hypothesis is again rejected.

Thus, for heteroscedastic noise, taking a signal y consisting of a useful trend and noise, as in (21), one can write the corresponding correlation between the entire signal and its predicted values, as in (22).

$$y = s \cdot (1 + n) \tag{21}$$

where

y—the overall signal;
s—the useful signal;
n—the noise.

$$r_{\hat{y},y} = \frac{Cov(\hat{y}, y)}{\sqrt{Var(\hat{y}) \cdot Var(y)}} \tag{22}$$

where

y—the original signal;
$\hat{y}$—the forecasted signal.

Assuming that s and n have a covariance of 0, the covariance between the useful signal and the original signal when the forecasting is perfect can be written as in (23):

$$\begin{aligned} Cov(s, y) = Cov(\hat{y}, y) = Cov(s, s + s \cdot n) = Cov(s, s) + Cov(s, s \cdot n) \\ = Var(s) + 0 = Var(s) \end{aligned} \tag{23}$$

where

$Cov(s, s + s \cdot n)$—the covariance between the useful signal without noise and the useful signal with noise;
$Var(s)$—the variance of the useful signal.

Given the initial supposition, one can further describe the forecast and the original signal as in (24) and (25).

$$Var(\hat{y}) = Var(s) \tag{24}$$

$$Var(y) = Var[s \cdot (1 + n)] = Var(s) + Var(n) \cdot E\left[s^2\right] \tag{25}$$

In both (24) and (25),
$Var(\hat{y})$—the variance of the forecasted signal;
$Var(s)$—the variance of the initial useful signal;
$Var(y)$—the variance of the original signal;
$E\left[s^2\right]$—the expected squared value of the useful signal.

Finally, the correlation between $\hat{y}$ and y can be written as in (26):

$$\begin{aligned} r_{\hat{y},y} = \frac{Cov(s,y)}{\sqrt{Var(s) \cdot Var(y)}} = \frac{Var(s)}{\sqrt{Var(s) \cdot Var(y)}} = \sqrt{\frac{Var(s)}{Var(y)}} \\ = \sqrt{\frac{Var(s)}{Var(s) + Var(n) \cdot E[s^2]}} \\ = \sqrt{\frac{Var(s) + Var(n) \cdot E[s^2] - Var(n) \cdot E[s^2]}{Var(s) + Var(n) \cdot E[s^2]}} = \\ = \sqrt{1 - \frac{Var(n) \cdot E[s^2]}{Var(s) + Var(n) \cdot E[s^2]}} \end{aligned} \tag{26}$$

where

$r_{\hat{y},y}$—the Pearson Correlation Coefficient between the forecasted signal $\hat{y}$ and the original signal y;
$Var(s)$—the variance of the initial useful signal;
$Var(n)$—the variance of the noise;
$E\left[s^2\right]$—the expected squared value of s.

In (26), $\frac{Var(n)\cdot E[s^2]}{Var(s)+Var(n)\cdot E[s^2]}$ is actually the contribution of the noise variance to the variance of the entire signal (for multiplicative noise) and, thus, the maximum value for the Pearson Correlation Coefficient for the non-averaged signal can be defined as in (27):

$$r_{\hat{y},y} = \sqrt{1 - 0.5475} = 0.672681 \cong 67.27\% \quad (27)$$

In order to validate the result in (24), the covariance between the useful signal and noise was computed and found to be 0. The heteroscedastic/multiplicative character of the noise can also be confirmed by visually inspecting Figure 2. For an overview of the entire methodology used, refer to Figure 6.

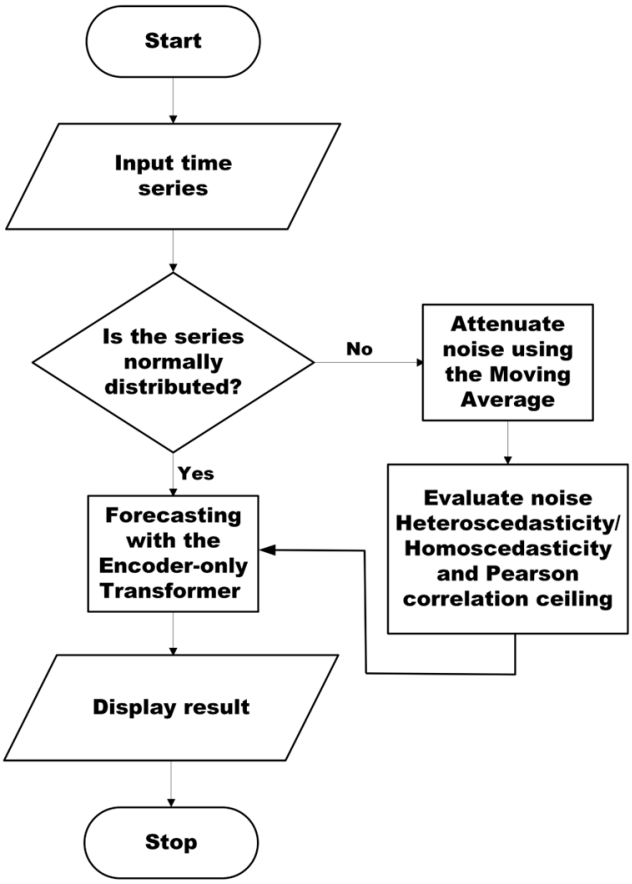


Figure 6. Flowchart of the overall forecasting methodology. This comprises the main stages of data preprocessing for forecasting.

In terms of the actual forecast when using the Transformer or N-BEATS architecture, in this particular case, the model learns from a 2-day window (that is, 192 time steps in the past) and predicts the next day (meaning the next 96 time steps). All (100%) of the data available for training was used for training (no validation set was used for any AI architecture) because the series is very noisy, and the dependencies on which the forecast is based are very hard to determine otherwise. As mentioned previously, the metrics presented in Table 2 are the MAPE, MAE, MSE, RMSE, and R2, but more important are the Pearson Correlation Coefficient and the DA. For some of the methods, the predicted values had near-zero variance, making the Pearson correlation numerically unstable; thus, the metric has been omitted from the table. As far as the DA is concerned, a value of less than 50% means that the model provides an inverse or constant prediction, approximately 50% means that the model performs randomly, between 50 and 65% signifies moderate accuracy, and between 65% and 80% means high precision. Thus, the Encoder-only Transformer

performs the best, with 77.89%. All these algorithms were chosen for comparison because they are very resilient when it comes to highly volatile time-series forecasting [20–23]. For example, Holt–Winters builds a running estimate of the baseline demand, its trend, and its daily cycle. It then projects this structure forward to predict the next day.

The day-ahead forecast obtained with the Holt–Winters algorithm is depicted in Figure 7.

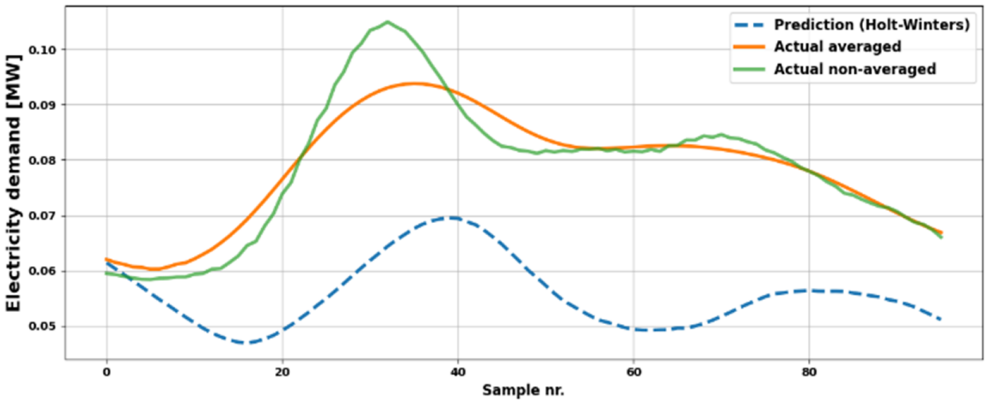


Figure 7. Day-ahead forecast obtained with the Holt–Winters algorithm. The correlation is good, but the forecast lacks accuracy.

On the other hand, in the case of the Prophet algorithm, the weekly seasonality is enabled, meaning that the algorithm will learn a 7-day cycle, with differences across weekdays. Daily seasonality is enabled as well, so a 24 h cycle will also be learned, capturing intra-day fluctuations that are useful for forecasting the electricity demand. At the same time, the trend component is included, meaning that the growth in the long-term part of the data is considered piecewise linear. Further, no holiday effects are specified, so no explicit holiday impact is modeled. The day-ahead forecast obtained with the Prophet algorithm is depicted in Figure 8.

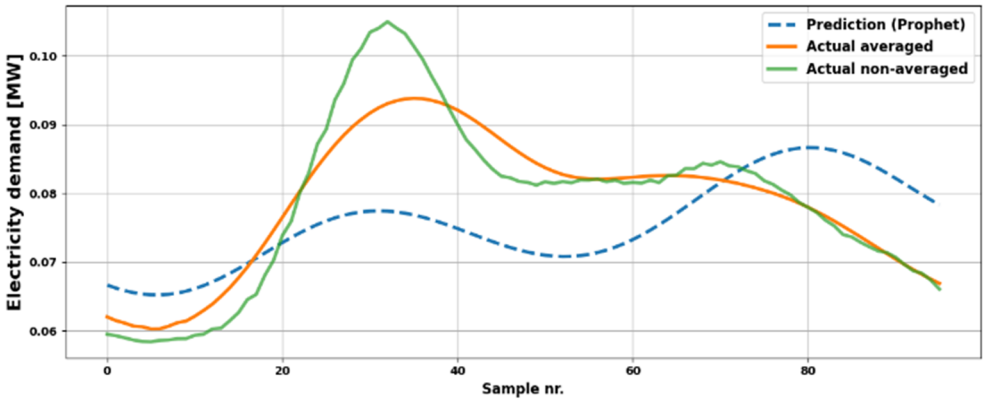


Figure 8. Day-ahead forecast obtained with the Prophet algorithm. The correlation is good, but the forecast lacks accuracy.

N-BEATS is a data-driven, block-based neural forecasting model that takes a window of recent history and predicts multiple future steps, learning both trend and seasonality automatically, without explicit decomposition. This makes it faster than Transformer architectures.

The context length in Table 3 was set to 192 steps because greater numbers risk saturating the architecture with noise, in which case it will stop learning and succumb to overfitting. This value and the prediction length are fixed, and they are not determined by Brute Force. The rest of the hyperparameters are obtained through optimization. Thus,

lower numbers are preferable for the given context. In other words, the context length is twice as long as the prediction length. The hidden size is the number of neurons in the hidden layers of each block. Further, one has two N-BEATS blocks. Each block predicts a component of the forecast and passes residuals forward.

Table 3. N-BEATS hyperparameters optimized with Brute Force.

Hyperparameter	Value	Role
Context length	192 steps	The model looks back 192 steps to predict the future 96 steps (fixed value)
Prediction length	96 steps	The required result (fixed value)
Hidden size	64	The number of neurons in each dense layer within a block
Number of blocks	2	The number of neural networks the model uses
Number of layers	2	The number of fully connected layers within a block
Dropout	0.1	10% of the neurons are disabled during training to prevent overfitting
Batch size	32	The number of samples processed simultaneously
Learning rate	1×10^{-4}	The model updates its weights with 0.01% of the gradient within each optimization step during training

The input hyperparameters of N-BEATS when optimized with Brute Force (also called Grid Search) are shown in Table 3.

Theoretically, more blocks can refine the model and capture subtler patterns. A lower number can be trained faster, but this does not always deliver good results. In this case, just two blocks were used, given the high volatility of the data, in order to avoid noise saturation of the architecture. This is again the main reason why the depth of the Feedforward Network (number of layers in each block) is set to 2, and not to a higher value. The dropout is set to 0.1, meaning that 10% of the neurons are deactivated during training, in order to prevent overfitting. In this context, the batch size specifies how many training examples are fed into the model at once.

The best results were obtained for a Transformer without a Decoder whose hyperparameters were optimized with Brute Force. Compared to Optuna, Brute Force is more thorough, taking into account all possible combinations of hyperparameters with the goal of minimizing the error function. This is the best strategy to follow when the time series is less predictable, as in this case.

The first step of this optimization algorithm is to define a hyperparameter grid, with finite candidate values for

- Context length (L_c);
- Embedding dimension (*dmodel*);
- Number of attention heads (h);
- Number of Encoder layers (N);
- Hidden size of the feedforward layer (*dff*);
- Learning rate (η);
- Batch size (B).

In the second step, itertools.product generates all possible combinations of hyperparameters.

In the third step, for each configuration θ, the following operations take place:

- A dataset is created with context length L_c = 192 and prediction length L_p = 96;
- A Transformer Encoder model is trained for 10 epochs using the Adam optimizer;

- The forecast is evaluated on the unaveraged test set using the MAE and MAPE (as in (26) and (27)).

Finally, in the fourth step, the best-performing configuration θ* with the hyperparameters yielding the smallest MAE (computed on all 96-step forecast windows, except for the last one, which is initially excluded from the training) is stored. Then, the model chosen in this way is tested on this final 96-step window. The results of this comparison are presented in Figures 9 and 10.

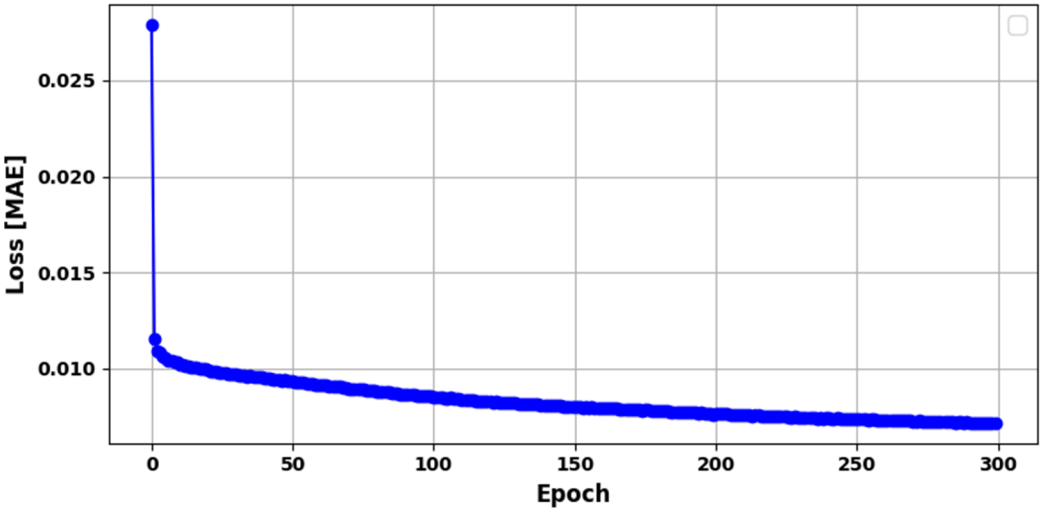


Figure 9. MAE training loss of the Encoder-only Transformer with optimized hyperparameters. The training of the Transformer is very robust, in spite of the noise.

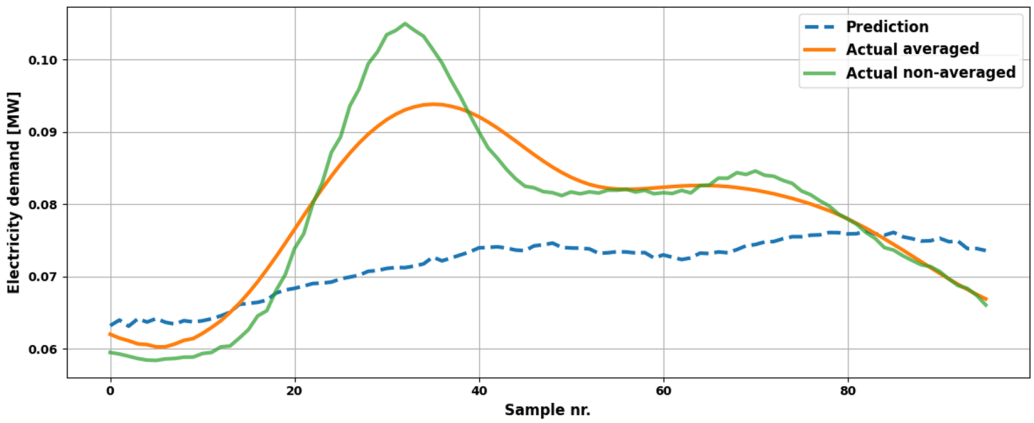


Figure 10. Day-ahead forecast obtained with the Encoder-only Transformer. The observed correlation between the forecast and non-averaged actual signal reaches roughly 78.78% of the theoretical ceiling of 67.27%, illustrating the gap imposed by inherent noise.

The mathematical formulation can be seen in (28)–(32).
The hyperparameter space is shown in (28):

$$\theta = \left(L_c, d_{model}, h, N, d_{ff}, \eta, B\right) \tag{28}$$

Each hyperparameter is chosen from a finite set: Θ_{L_c}, $\Theta_{d_{model}}$, Θ_h, Θ_N, $\Theta_{d_{ff}}$, Θ_η, Θ_B.
The total number of combinations can be seen in (29):

$$\Theta = \Theta_{L_c} \cdot \Theta_{d_{model}} \cdot \Theta_h \cdot \Theta_N \cdot \Theta_{d_{ff}} \cdot \Theta_\eta \cdot \Theta_B \tag{29}$$

where

Θ—the total number of hyperparameter combinations;

Θ_{L_c}—a fixed hyperparameter pertaining to context length (it possesses just one value, 192);

$\Theta_{d_{model}}$—the finite set of hyperparameter combinations, pertaining to embedding;

Θ_h—the finite set of hyperparameter combinations, pertaining to attention head number;

Θ_N—the finite set of hyperparameter combinations, pertaining to the number of Encoder or Decoder layers;

$\Theta_{d_{ff}}$—the finite set of hyperparameter combinations, pertaining to the hidden size of the feedforward layer;

Θ_η—finite set of hyperparameter combinations, pertaining to possible learning values

Θ_B—the finite set of hyperparameter combinations, pertaining to the batch size.

For each configuration θ, the model produces forecasts $\hat{y}_t(\theta)$ over the prediction horizon $t = 1,\ldots, L_p$.

Thus, the MAE is described in this case as in (30):

$$MAE(\theta) = \frac{1}{L_p} \cdot \sum_{t=1}^{t=L_p} |y_t - \hat{y}_t(\theta)| \tag{30}$$

$$MAPE(\theta) = \frac{100}{L_p} \cdot \sum_{t=1}^{t=L_p} \left| \frac{y_t - \hat{y}_t(\theta)}{y_t} \right| \tag{31}$$

In both (30) and (31),

t—the time step;

L_p—the length of prediction;

y_t—the real unaveraged load at horizon t;

$\hat{y}_t(\theta)$—the forecasted load at time step t.

The best hyperparameter set is the one minimizing MAE, according to (32):

$$\theta^* = arg \min_{\theta \in \Theta} MAE(\theta) \tag{32}$$

The values of the optimal hyperparameters for the Encoder-only Transformer obtained with Brute Force are described in Table 4.

Table 4. Encoder-only Transformer hyperparameters optimized with Brute Force.

Hyperparameter	Value	Role
Context length	192 steps	The model observes the past 192 steps to predict the next 96 steps (fixed value)
Prediction length	96 steps	The required result (fixed value)
Embed dimension	16	Each scalar input value is transformed into a 16-dimensional embedding vector before entering the self-attention layers
Number of heads	8	The multi-head attention mechanism splits the embedding into 8 parallel subspaces, modeling various temporal features
Number of layers	3	The number of stacked Transformer Encoder blocks
Dropout	0.1	A total of 10% of the neurons are disabled during training to prevent overfitting
Batch size	32	The model processes 32 sequences (samples) per optimization step before updating its weights
Learning rate	0.001	At each training step, the model updates its weights by 0.1% of the computed gradient

Brute Force and the Encoder-only Transformer, as seen in Tables 2–4, perform much better than OPTUNA and N-BEATS in terms of MAPE and Pearson's Correlation Coefficient. This is in spite of OPTUNA's ability to carry out Bayesian optimization of the hyperparameters. The primary reason for this lies in the characteristics of the time series. Brute Force is more thorough than OPTUNA and, in this case, delivers more accurate results because it scans the entire solution space in any circumstance and, in the end, finds the right combination of hyperparameters.

The training of the Encoder-only Transformer with optimized hyperparameters is illustrated in Figure 9, and the forecast is shown in Figure 10. Thus, the training of the Encoder-only Transformer is very robust despite the noise level, which is still rather high, even after applying the Moving Average. As a result, the MAE drops continuously from 0.028 MW to approximately 0.005 MW after 300 epochs. On the other hand, as shown in Figure 10, the day-ahead forecast yielded an MAPE of 11.63% and a Pearson's Correlation Coefficient of 53% compared to the original, non-averaged signal.

5. Discussion

The results obtained in this work demonstrate the challenges and opportunities inherent in forecasting highly volatile time series—as is usually the case, for example, when it comes to energy generation from renewable sources. As mentioned above, in the present dataset, approximately 55% of the signal represents random variations, leaving only about 45% as the useful signal. In order to observe how well the statistical distribution of the time series fits the Normal Distribution, two tests related to digital signal processing—namely, the Kolmogorov–Smirnov and Jarque–Bera tests—were performed, which revealed that the noise attenuated with the Moving Average better fits the desired Theoretical Normal Distribution. On the other hand, standard forecasting metrics, like the Pearson Correlation Coefficient, are sensitive to highly volatile content. Indeed, for several models—including N-BEATS (optimized with Brute Force and Optuna), Vanilla Transformer (optimized with Brute Force), and the Decoder-Only Transformer (optimized with Brute Force)—the forecast exhibits negligible variation, rendering the Pearson Correlation Coefficient non-computable. This highlights a critical limitation of conventional performance metrics when applied to very noisy time series.

To address this limitation, we introduced a procedure to estimate the maximum theoretical Pearson Correlation Coefficient between the forecast and the actual power demand based on the noise character of the series (additive or multiplicative, heteroscedastic or homoscedastic). This procedure provides a robust benchmark for model evaluation, allowing practitioners to distinguish between data inconsistency and genuine forecasting deficiencies. When the theoretical performance ceiling is known, the model results can be better interpreted, and more informed decisions can be made, even when most standard metrics fail.

The comparison of the models underlines the practical benefits of the current approach. The Encoder-only Transformer, applied to the noise-attenuated time series, consistently yielded strong results in terms of both Pearson Correlation and MAPE relative to the original signal. This demonstrates that noise reduction, when combined with an appropriate model architecture, significantly enhances predictive performance in volatile environments. Conversely, models that fail to produce variable forecasts underscore the importance of aligning evaluation metrics with data characteristics: without sufficient forecast variability, conventional correlation measures provide little actionable insight.

Methodologically, these results stress the need for customized evaluation frameworks in forecasting. Traditional metrics may misrepresent model performance in highly volatile environments, leading to potentially misguided operational decisions.

Finally, the study opens several avenues for future research. First, extending the proposed Pearson ceiling methodology to multivariate time series could enable more comprehensive evaluations across interconnected systems. Second, integrating probabilistic forecasting approaches may further capture uncertainty in very noisy data, providing richer insights for real-time decisions pertaining to energy consumption or generation. Lastly, exploring adaptive noise reduction techniques in conjunction with Transformer-based architectures could yield additional improvements in prediction accuracy for smart grids, among other applications.

Further, a sensitivity analysis was conducted to evaluate the robustness of the model. The first parameter that was studied was the variation in MAPE, depending on optimal or nonoptimal hyperparameters and the number of training epochs. The results can be seen in Figure 11.

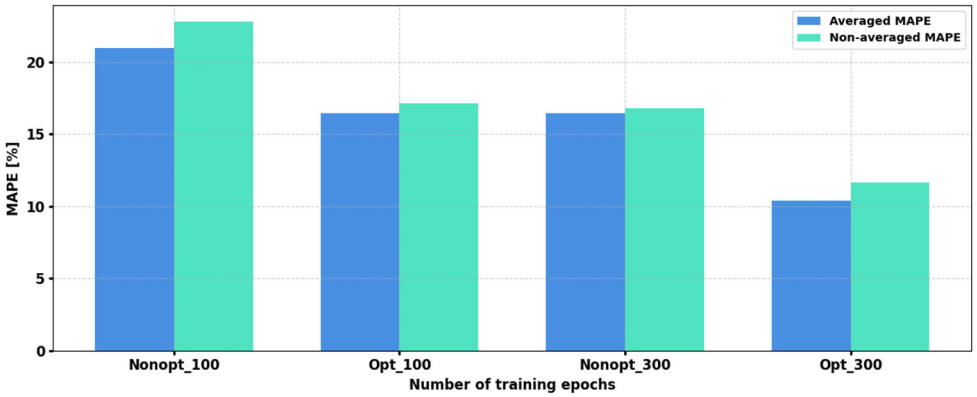


Figure 11. MAPE variation in relation to the optimization of hyperparameters and the number of training epochs. Hyperparameter optimization was carried out with Brute Force.

As one can see in Figure 11, the best results are obtained with 300 training epochs and hyperparameters optimized wtih Brute Force. The averaged MAPE refers to the comparison of the forecast to the actual signal for which noise has been attenuated. When comparing the forecast with the non-averaged signal contaminated with noise, the errors are slightly higher, which is expected since the Transformer was trained on the averaged signal. This is the first limitation of the proposed method. Actually, the same behavior can also be observed for the variation in RMSE in Figure 12. The same types of considerations also apply in this case.

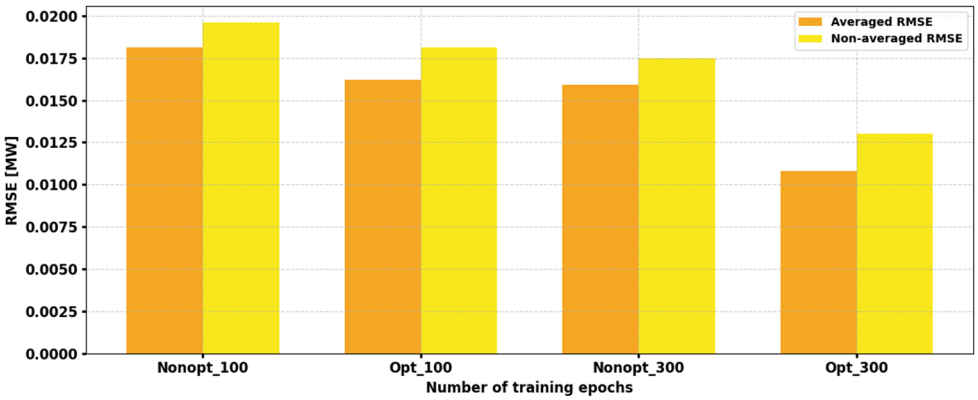


Figure 12. RMSE variation in relation to the optimization of hyperparameters and the number of training epochs. Hyperparameter optimization was carried out with Brute Force.

On the other hand, the variation in the Pearson Correlation Coefficient with respect to the number of training epochs and hyperparameter optimization through Brute Force can be seen in Figure 13.

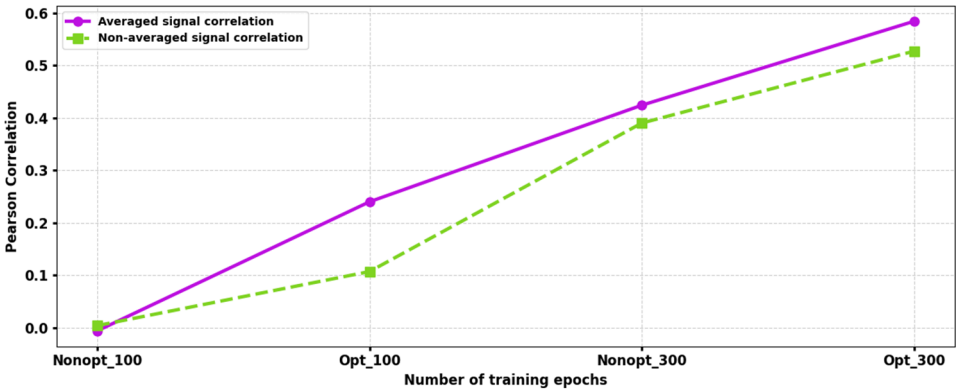


Figure 13. Variation in the Pearson Correlation Coefficient in relation to the optimization of hyperparameters and the number of training epochs. Hyperparameter optimization was carried out with Brute Force.

Again, when comparing the variation in the Pearson Correlation Coefficient between the forecast and the series contaminated with noise, the coefficient is consistently lower than that obtained when one compares the forecast with the attenuated-noise series. The reason is the same as before—namely, that the Encoder-only Transformer was trained on the averaged signal. Although the difference caused by this approach is not large, there is a danger that if the noise level increases, this approach might become unreliable.

The superiority of the proposed solution compared to the baseline methods, as illustrated in Table 2, can be further proven through the Diebold–Mariano (DM) test. If one observes Table A2, in Appendix A, the DM statistics are consistently negative, meaning that the Transformer always performs better than the other methods. Moreover, the fact that the p-value is always 0, or very close to 0, indicates that the Transformer's superior performance is not random but is statistically demonstrated. In this case, p must be lower than 0.05. Thus, the validity of the proposed method is verified.

Another limitation of this approach is that the model must be trained daily in order to guarantee the accuracy of the day-ahead forecast.

Another important aspect is that all the concepts presented in this work, relating to both data preprocessing techniques and forecasting models, can be easily integrated as modules into a foundation model dedicated to helping transmission system operators (TSOs) or distribution system operators (DSOs) make the right decisions in real time regarding efficient power system operation under normal or abnormal conditions.

6. Conclusions

This work presents a framework for the day-ahead forecasting of highly volatile time series of electrical energy consumption and trading.

As stated in the Introduction, this approach introduces two important novelties. The first is represented by the systematic analysis of the noise, how its variability affects the variability of the entire time series, and how this can be mitigated. The second novelty, which is even more important, lies in computing the theoretical ceiling of the Pearson Correlation Coefficient and taking this into account when evaluating the forecasting architecture's performance.

The proposed forecasting framework demonstrates that reliable predictions can still be made even when most of the variation in the data is caused by random fluctuations.

As mentioned previously, this has direct implications for forecasting energy generated from renewables, which is usually destined to feed either end clients or various clusters of consumers.

The procedure for estimating the maximum theoretical limit of Pearson's Correlation Coefficient provides a new interpretative tool for assessing model performance under noisy conditions. Ultimately, this prediction framework is expected to help city planners and engineers alike make better decisions based on uncertain, real-world data.

In addition, various forecasting methods were implemented, based on either pure statistics or architectures that have become the foundation for many Generative Artificial Intelligence models. The best results were obtained using an Encoder-only Transformer whose hyperparameters were optimized with Brute Force. Quite paradoxically, N-BEATS optimized with Optuna failed to deliver the best results, despite Optuna's ability to carry out Bayesian optimization and N-BEATS's faster convergence compared with the Transformer, even on noisy time series. Given that the time series does not fit well to a Normal Distribution, a strategy was necessary to compute all possible combinations of hyperparameters in order to obtain the best results possible. In addition, the procedure developed to determine the maximum theoretical limit for the Pearson Correlation Coefficient was based on defining the noise character, which, in real situations, can be either multiplicative (and always heteroscedastic) or additive (which may be heteroscedastic or homoscedastic).

When Brute Force was used to optimize the hyperparameters of the Encoder-only Transformer, a very good MAPE and a good correlation coefficient were obtained—quite close to the maximum possible theoretical limit. Less complicated forecasting architectures were used in order to avoid their saturation with noise and potential overfitting. As a result, 78.78% of the theoretical limit obtained by an ideal forecasting architecture was actually achieved in practice. This value is valid for correlation. For the MAPE, as mentioned in the abstract, a 30% prediction error is considered reasonable and is valid for a time series with a low level of volatility [24,25]. Hence, it can be concluded that, for a highly volatile time series, the current model delivers interesting results in terms of both correlation and prediction error. Future work will be devoted to improving this percentage, as well as the necessary time to train the model, since this technology is highly relevant for energy efficiency in smart grid and smart city contexts [26]. As mentioned in the previous section, another important aspect for further investigation is the integration of the presented approaches and new data preprocessing techniques [27], along with forecasting architectures as modules, into foundation models that use LLMs as communication interfaces.

Author Contributions: Conceptualization, A.-V.B.; methodology, A.-V.B.; software, A.-V.B.; validation, A.-V.B. and M.-S.M.; formal analysis, A.-V.B. and M.-S.M.; investigation, A.-V.B.; resources, A.-V.B.; data curation, A.-V.B.; writing—original draft preparation, A.-V.B.; writing—review, M.-S.M.; visualization, M.-S.M.; supervision, M.-S.M. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by the National University of Science and Technology POLITEHNICA Bucharest, Romania, through the Program PubArt.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is unavailable due to the privacy and confidential market strategies of the power producer.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Within Appendix A, a summary table is presented pertaining to noise handling in time-series prediction using the AI architectures introduced in Section 2. The main goal is to provide readers with a better understanding of how the proposed method compares with the current state of the art.

Table A1. Summary table pertaining to noise handling in time-series forecasting for trading.

Model	Key Innovation	Application Domain	Datasets	Noise Handling	Reported Performance (MAPE/MAE/MSE Trends)
LogSparse Transformer [4] (Li et al., 2019)	Sparse attention for long dependencies	Electricity consumption, solar energy production, and traffic	Real-world and synthetic	Moderate	10–20% lower error across standard benchmarks with respect to ARIMA, ETS, TRMF, DeepAR, and DeepState
Temporal Fusion Transformer (TFT) [5]	Combines LSTM Encoder–Decoder with multi-head attention and gating mechanisms for interpretable multi-horizon forecasting	Energy demand, traffic, retail sales, and financial data	Real-world time series relating to electricity, traffic, volatility, retail	Moderate: robust to moderate noise due to gating, variable selection, and static covariate encoders	Outperformed ARIMA, ETS, TRMF, DeepAR, DSSM, etc. (MAE is typically 3–26% lower across the datasets)
Reformer [7]	Introduces LSH (Locality-Sensitive Hashing) attention and reversible layers to reduce memory and computation from $O(L^2)$ to $O(L\ log\ L)$	Energy demand, traffic, retail sales, and financial data	Character-level language modeling, image generation, and long-range associations between integers	Moderate: sparse attention reduces overfitting to noise by focusing on top-K relevant tokens	Demonstrated similar or slightly worse forecasting accuracy than Vanilla Transformer with much lower memory footprint; exact error metrics depend on dataset and fine-tuning.
Informer [9]	Introduces ProbSparse attention to reduce attention computation to $O(L\ log\ L)$ and self-attention distillation for long-sequence forecasting	Energy, traffic, weather, and finance	ETTh1, ETTh2, ETTm1, weather, electricity consumption	High forecasting errors due to volatile component filtering	Outperformed LogTrans, Reformer, LSTM, DeepAr, ARIMA, and Prophet baselines; typically improved MSE/MAE by 10–25%, depending on the dataset
Autoformer [10]	Introduces series decomposition into trend and seasonal components and auto-correlation-based attention for long-term forecasting	Energy, traffic, weather, finance, medical sector	ETT, electricity, exchange, traffic, weather, ILI	High forecasting errors due to volatile component filtering	Outperformed LogTrans, LSTM, Informer, LSTNet, Reformer, and TCN in MSE/MAE by 15–30%, depending on the dataset and horizon

Table A1. *Cont.*

Model	Key Innovation	Application Domain	Datasets	Noise Handling	Reported Performance (MAPE/MAE/MSE Trends)
FEDformer [11]	Combines series decomposition with frequency-domain attention to capture both trend and periodic components for long-term forecasting	Energy, traffic, weather, climate, finance	ETTm2, exchange, traffic weather, ILI	High forecasting errors due to volatile component filtering	Outperformed Autoformer, Informer, LogTrans, and Reformer; typically improved MSE/MAE by 20–35%, depending on the dataset and forecasting horizon
PatchTST [12]	Partitions the time series into local temporal windows for attention (inspired by Vision Transformers); uses channel-independent attention for multivariate forecasting	Energy, traffic, weather, finance, and industrial	Traffic, electricity, ETTm1, ETTm2, ETTh1, ETTh2, weather, ILI	High forecasting errors due to further reduction in the magnitude of the micro-variations	Achieved state-of-the-art performance with a MAPE of approx. 10–15% on standard benchmarks, outperforming Autoformer, FEDformer, Informer, D-Linear, Pyraformer, and LogTrans
Hybrid Transformer–CNN [13]	Combines Transformer self-attention with CNN-based local feature extraction to capture both global dependencies and local patterns	Especially financial markets	Multivariate, containing 1 or 5 features	Good since it reduces the short-term impact of the noise, but the model is sometimes prone to noise saturation, given the architecture complexity	Outperforms the Transformer, 1D-CNN, LSTM, CNN-LSTM, and ARIMA by about 15% in MSE
CNN–Transformer Hybrid [14]	Integrates CNN layers to capture short-term local dependencies with Transformer attention for long-term dependencies in financial time series	Financial markets, stock prices, and multivariate financial indicators	S&P stock market prices	Good: CNN layers smooth high-frequency noise, but the model is sometimes prone to noise saturation, given the architecture complexity	Outperforms ARIMA, EMA, and DeepAR by approx. 1–15% in MAPE
Proposed method	**Attenuates the noise, trains the Encoder-only Transformer on the attenuated noise time series, and compares the forecast with the actual demand**	**Specially designed for trading energy generated by renewable sources, containing more than 50% noise**	**Custom, coming from a real energy provider**	**Excellent: the architecture is adapted to avoid noise saturation**	**Achieves 11.63% in MAPE, which is very low, since less than 50% of the signal can be used in forecasting**

In Table A1, the proposed method is written in bold because it exhibits the best performance when compared to all other architectures.

Table A2. Diebold–Mariano (DM) test results relating to the performance comparison between the Encoder-only Transformer and other baseline models.

Compared to	DM Statistic	*p*-Value
ARIMA	−8.198	0
SARIMA	−8.8327	0
Holt–Winters	−12.1633	0
Prophet	−1.98	0.0375
LSTM	−8.5888	0
CNN	−9.6485	0
GRU	−8.7675	0
SVR	−5.7557	0
Random Forest	−9.1845	0
XGBoost	−10.0693	0
DeepAR	−5.1903	0
N-BEATS (optimized with Brute Force)	−8.9512	0
N-BEATS (optimized with Optuna)	−10.9868	0
Vanilla Transformer (optimized with Brute Force)	−6.2687	0
Decoder-only Transformer (optimized with Brute Force)	−12.9976	0

References

1. Rodrigues Dos Reis, J.; Tabora, J.M.; de Lima, M.C.; Monteiro, F.P.; Monteiro, S.C.d.A.; Bezerra, U.H.; Tostes, M.E.d.L. Medium and Long Term Energy Forecasting Methods: A Literature Review. *IEEE Access* **2025**, *13*, 29305–29326. [CrossRef]
2. Sănduleac, M.; Ciornei, I.; Toma, L.; Plămnescu, R.; Dumitrescu, A.-M.; Albu, M.M. High Reporting Rate Smart Metering Data for Enhanced Grid Monitoring and Services for Energy Communities. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4039–4048. [CrossRef]
3. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.
4. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; Yan, X. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5243–5253. Available online: http://papers.nips.cc/paper/8766-enhancing-the-locality-and-breaking-the-memory-bottleneck-of-transformer-on-time-series-forecasting.pdf (accessed on 9 November 2025).
5. Lim, B.; Arik, S.Ö.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. *arXiv* **2020**, arXiv:1912.09363. Available online: https://arxiv.org/abs/1912.09363 (accessed on 9 November 2025).
6. Rühmann, S.; Leible, S.; Lewandowski, T. Interpretable Bike-Sharing Activity Prediction with a Temporal Fusion Transformer to Unveil Influential Factors: A Case Study in Hamburg, Germany. *Sustainability* **2024**, *16*, 3230. [CrossRef]
7. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The Efficient Transformer. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 26–30 April 2020. Available online: https://iclr.cc/virtual_2020/poster_rkgNKkHtvB.html (accessed on 9 November 2025).
8. Caetano, R.; Oliveira, J.M.; Ramos, P. Transformer-Based Models for Probabilistic Time Series Forecasting with Explanatory Variables. *Mathematics* **2025**, *13*, 814. [CrossRef]
9. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11106–11115. Available online: https://ojs.aaai.org/index.php/AAAI/article/view/17325 (accessed on 9 November 2025).
10. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430. Available online: https://proceedings.neurips.cc/paper_files/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html (accessed on 9 November 2025).
11. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency Enhanced Decomposed Transformer for Long-Term Series Forecasting. In Proceedings of the 39th International Conference on Machine Learning (ICML), Baltimore, MD, USA, 17–23 July 2022; Volume 162, pp. 27268–27286. Available online: https://proceedings.mlr.press/v162/zhou22g.html (accessed on 9 November 2025).

12. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is Worth 64 Words: Long-Term Forecasting with Transformers. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023. Available online: https://openreview.net/forum?id=Jbdc0vTOcol (accessed on 9 November 2025).
13. El Zaar, A.; Mansouri, A.; Benaya, N.; Bakir, T.; El Allati, A. Hybrid Transformer-CNN Architecture for Multivariate Time Series Forecasting: Integrating Attention Mechanisms with Convolutional Feature Extraction. *J. Intell. Inf. Syst.* **2025**, *63*, 1233–1264. [CrossRef]
14. Tu, T. Bridging Short- and Long-Term Dependencies: A CNN-Transformer Hybrid for Financial Time Series Forecasting. *arXiv* **2025**, arXiv:2504.19309. Available online: https://arxiv.org/abs/2504.19309 (accessed on 9 November 2025).
15. LeCun, Y.; Bottou, L.; Orr, G.B.; Müller, K.-R. Efficient BackProp. In *Neural Networks: Tricks of the Trade*, 2nd ed.; Montavon, G., Orr, G.B., Müller, K.-R., Eds.; Springer: Berlin, Germany, 2012; pp. 9–48.
16. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
17. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 448–456.
18. Kolmogorov, A. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari* **1933**, *4*, 83–91.
19. Bera, A.K.; Jarque, C.M. Efficient tests for normality, homoscedasticity and serial independence of regression residuals: Monte Carlo evidence. *Econ. Lett.* **1981**, *7*, 313–318. [CrossRef]
20. Gelper, S.; Fried, R.; Croux, C. Robust Forecasting with Exponential and Holt–Winters Smoothing. *J. Forecast.* **2010**, *29*, 285–300. [CrossRef]
21. Croux, C.; Gelper, S.; Fried, R. Computational Aspects of Robust Holt–Winters Smoothing Based on M-Estimation. *Appl. Math.* **2008**, *53*, 163–176. [CrossRef]
22. Ma, P.; Ren, J.; Sun, G.; Zhao, H.; Jia, X.; Yan, Y.; Zabalza, J. Multiscale Superpixelwise Prophet Model for Noise-Robust Feature Extraction in Hyperspectral Images. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–12. [CrossRef]
23. Apte, M.; Haribhakta, Y.V. Advancing Financial Forecasting: A Comparative Analysis of Neural Forecasting Models N-HiTS and N-BEATS. In *Information Systems for Intelligent Systems (ISBM 2024)*; Iglesias, A., Shin, J., Patel, B., Joshi, A., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2025; Volume 1255.
24. Veit, A.; Goebel, C.; Tidke, R.; Doblander, C.; Jacobsen, H.-A. Household Electricity Demand Forecasting—Benchmarking State-of-the-Art Methods. *arXiv* **2014**, arXiv:1404.0200. Available online: https://arxiv.org/abs/1404.0200 (accessed on 9 November 2025).
25. Ma, X.; Liu, Z. Application of a Novel Time-Delayed Polynomial Grey Model to Predict the Natural Gas Consumption in China. *J. Comput. Appl. Math.* **2017**, *324*, 17–24. [CrossRef]
26. Gracias, J.S.; Parnell, G.S.; Specking, E.; Pohl, E.A.; Buchanan, R. Smart Cities—A Structured Literature Review. *Smart Cities* **2023**, *6*, 1719–1743. [CrossRef]
27. Letchford, A.; Gao, J.; Zheng, L. Optimizing the moving average. In Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.